

The Big MAT Book:

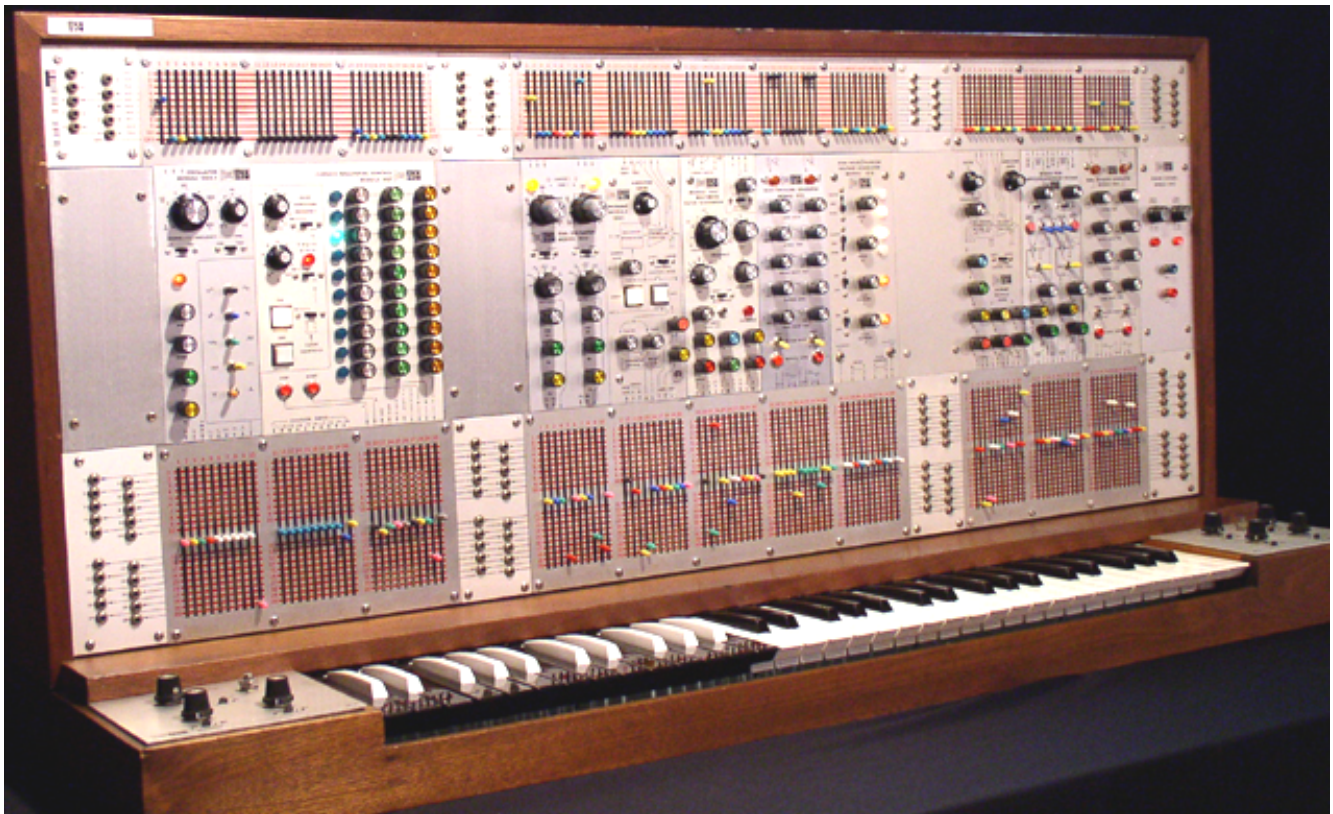
Courseware for Audio & Multimedia Engineering

Volume 1: Multimedia Engineering

Stephen Travis Pope

Graduate Program in Media Arts & Technology (MAT) &
Dept. of Music

University of California, Santa Barbara (UCSB)



MEDIA ARTS & TECHNOLOGY PROGRAM

CREATE



Front cover: ARP 2500 synthesizer

Inside flap: EMS Synthi 100 synthesizer

Copyright © 2014. Stephen Travis Pope. Some rights reserved. *The Big MAT Book* is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 License.



The Big MAT Book: Courseware for Audio & Multimedia Engineering

Overview

Volume 1: Multimedia Engineering

Volume 2: Audio Software

Volume 3: Audio Hardware

Table of Contents

Volume 1: Multimedia Engineering

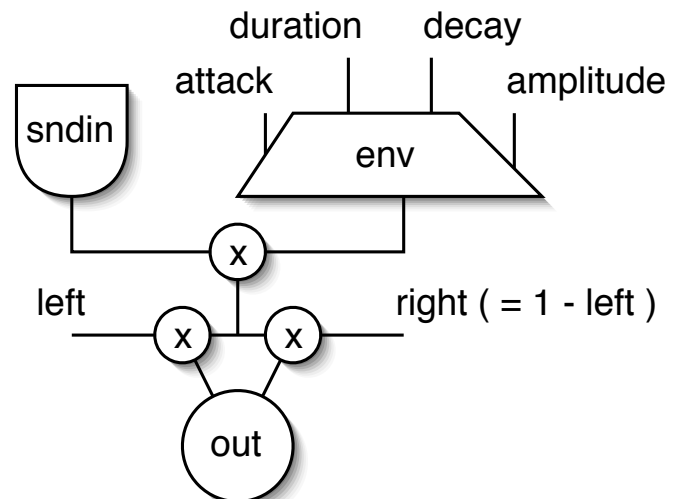
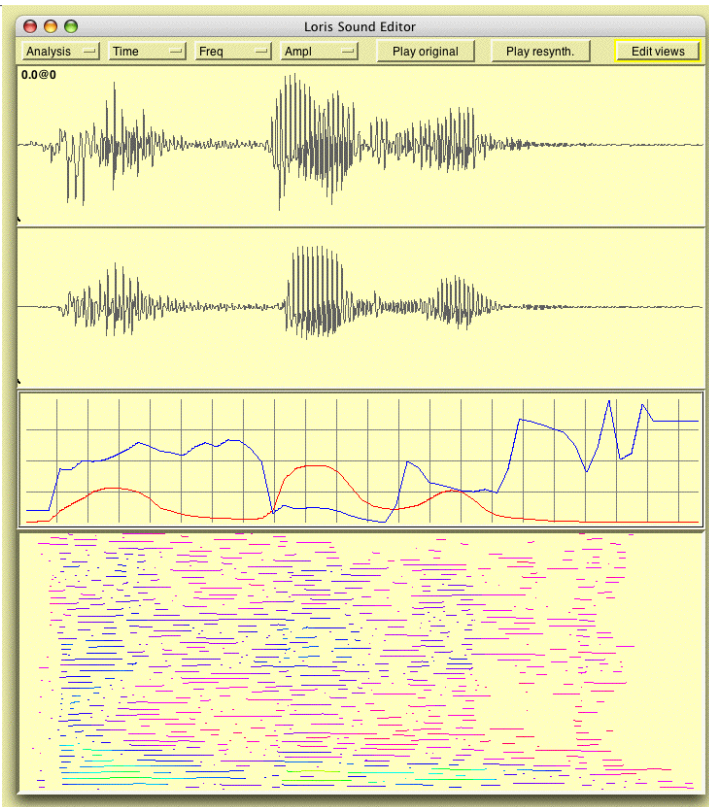
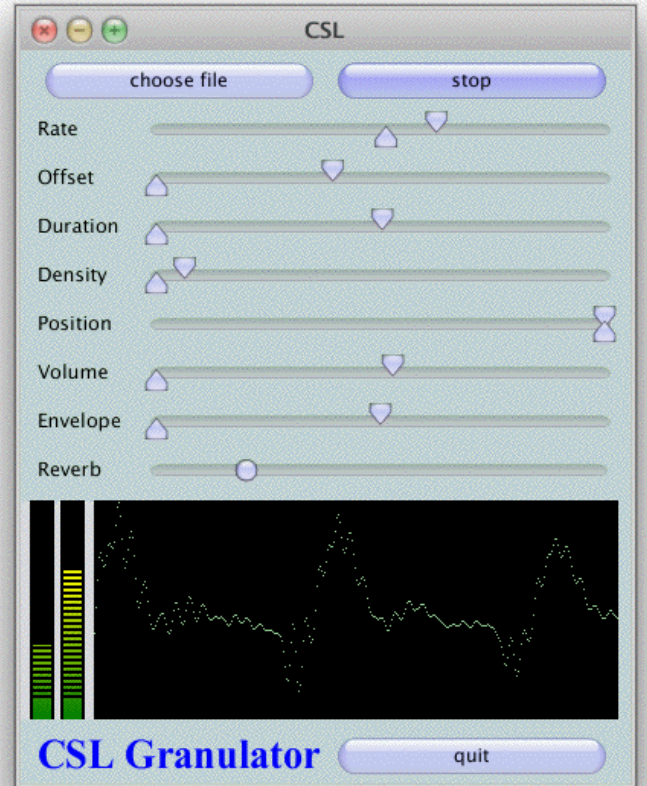
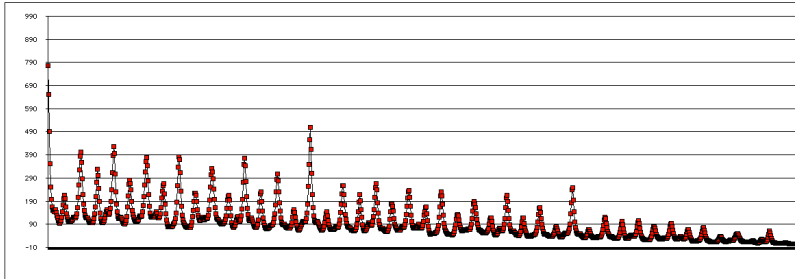
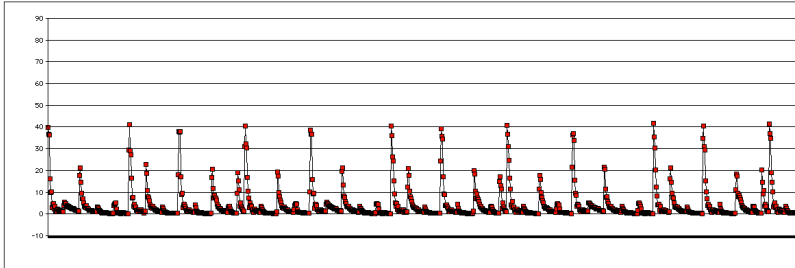
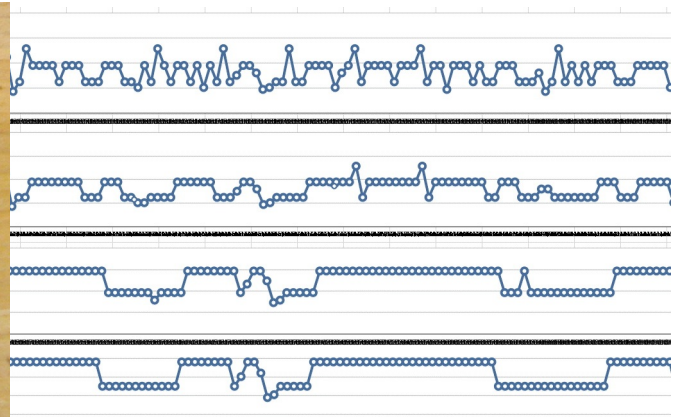
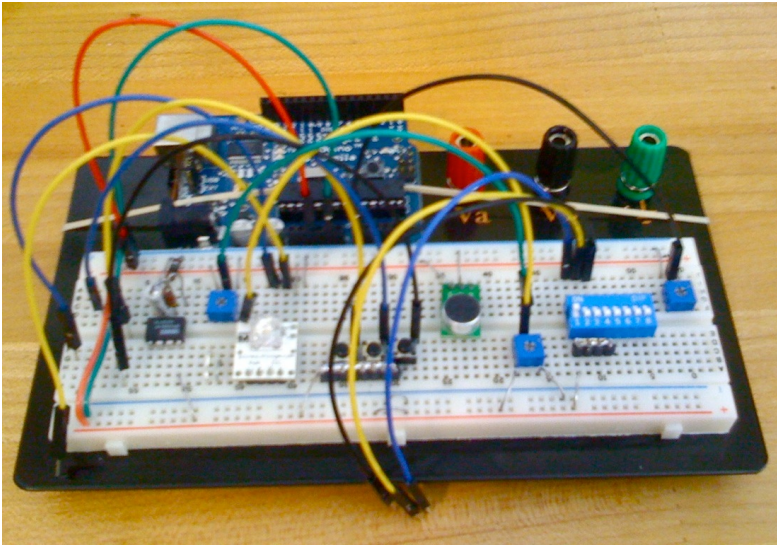
- 1. Survey of Media Engineering & Technology1
- 2. Media Signal Processing115
- 3. Computing with Media Data.....183
- 4. Sensors and Interfaces for Media Art293

Volume 2: Audio Software

- 5. Sound IO and Streaming APIs.....345
- 6. The Spectral Domain: Filter and the FFT393
- 7. Spatial/Surround Sound and Reverb.....423
- 8. Sound Synthesis Techniques.....475
- 9. Synthesis Control and Streaming.....507
- 10. Audio Analysis and Music Information Retrieval.....543

Volume 3: Audio Hardware

- 11. Audiophile Engineering593
- 12. Recording Studio Design and Engineering667



Introduction to the Series “Courseware for Audio & Multimedia Engineering”

Multimedia engineering is a broad and complex topic. It is also one of the fastest-growing and most valuable fields of research and development within electronic technology. The book before you is an anthology of curriculum materials developed over the space of 12 years at the University of California, Santa Barbara for students in UCSB’s *Graduate Program in Media Arts and Technology*.

The *BigMATBook* consists of the presentation slides for twelve ten-week courses, amounting to over 600 hours of presentation time. For each of the twelve courses, the presentation slides are accompanied by the tables of contents of the course readers, and an overview of the example code archives. These resources are available from the HeavenEverywhere web site; see

<http://HeavenEverywhere.com/TheBigMATBook>

The multimedia engineering courses included here cover theory and practice, hardware and software, visual and audio media, and arts as well as entertainment applications. Some of the courses (the first two chapters) are required of all MAT graduate students, and thus must target less-technical and also non-audio-centric students. The bulk of this material, though, consists of elective courses that have somewhat higher-level prerequisites and assume basic knowledge of acoustics and some (minimal) programming experience in mainstream programming languages.

The *BigMATBook* courses borrow liberally from R&D publications by my friends and colleagues, especially Roger Dannenberg, Julius O. Smith, D. Gareth Loy, F. R. Moore, Perry Cook, Adrian Freed, George Tzanetakis, Ross Bencina and Dan Overholt. I want also to express my deepest thanks to my MAT and Music Dept. colleagues JoAnn Kuchera-Morin, Curtis Roads, Clarence Barlow, Matthew Wright, and Matthew Turk, and to the many students who helped these courses evolve, either as course participants or teaching assistants (you know who you are).

Stephen Travis Pope (stephen@HeavenEverywhere.com)

Santa Barbara, California—September, 2009 (updated January, 2014)

Introduction to Volume 1: “Multimedia Engineering”

Volume 1 of *TheBigMATBook* consists of four courses on medium-agnostic multimedia engineering: a theory survey course, a math and digital signal processing course, a practical computer science course, and one concentrating on sensor electronics and microcontrollers in the arts.

The first chapter consists of the course “Survey of Media Technology and Engineering”—a broad survey/reading course that covers topics as diverse as computer architecture and haptics. This leaves us with a broad range of both theoretical and practical issues related to multimedia perception, communication and information theory, processor and computer design, representation of signals and events, networking hardware, signal processing, and applications.

The second chapter covers the basics of digital signal representation, synthesis, analysis and processing; it is dedicated to introducing students to the multimedia digital signal processing methods. We explore a range of topics from theoretical principals to practical considerations in multimedia (speech, audio, still images, and video) signal representation, synthesis, analysis and processing. The course exercises consist of reading and homework tasks where students explore the concepts introduced in the lectures through concrete software applications using the MATLAB programming language and custom-developed SSUM framework. The premise of the course is that the multimedia expert of the future will need to have a firm grasp of the mathematical foundations of data representation and to understand (and even be able to extend) the algorithms used on common signal synthesis/processing/analysis applications.

The third chapter is a hands-on graduate-level software development course related to media-rich applications and tools. The course reader consists of a mix of computer science and software engineering theory, and praxis-oriented how-to programming tutorials on media data processing and modern software development. The lecturers concentrate on the concrete skills required for development of multimedia applications and tools in mainstream programming languages using the standard off-the-shelf software development libraries.

In the final chapter—sensors and interfaces—we explore the use of multimedia sensor technologies and embedded microcontroller systems for interactive environments/installations and responsive artwork/performance systems. We start with an introduction to the theories of space in art and human-computer interaction (HCI), and proceed to an in-depth analysis of current art-HCI technologies. The course introduces the principles and operation of many sensor techniques and discuss applications in gestural human-computer interfaces, new musical instruments, the visual and spatial arts, engineering, science, and other areas of interest. It covers the design of computer interface systems including analog/digital electronics and human factors/interaction styles. Output modalities include interactive visuals, music/sound, and motion control/robotics.

MAT 200C: Survey of Media Technology and Engineering, Winter Quarter 2008

MAT 200C *Survey of Media Technology and Engineering* is a broad survey/reading course that covers topics as diverse as computer architecture and haptics. The goal is to cover all those areas related to multimedia application development that are not covered in the other MAT core courses. As such, this leaves us with a broad range of both theoretical and practical issues related to multimedia perception, communication and information theory, processor and computer design, representation of signals and events, networking hardware, signal processing, and applications.

Course Outline

0. Introduction & review: math, EE, CS, psychology, etc.
1. Some topics from electrical and computer engineering
2. Media data, signals, and symbols
3. Data models, representations, and formats
4. Digital manipulation of signals and symbols
5. Media I/O devices
6. Applications in arts, entertainment, and education

Format

- 2 * 2 hours/week lectures (Mon/Wed)
- 1 * 2 hours/week lab (Fri)
- Homework programs, quizzes, development project, final presentation

Materials

Readings book and presentation slides (both at the UCSB book store)

Requirements

- 1-2 hours required reading per week: see the reader
- Short (20 minute) topical presentation with advanced readings
- Short (8-10 pages) paper: survey or summary of advanced readings
- Software project: implementation of a program related to the readings

Instructor

Stephen T. Pope (stp@mat.ucsb.edu)

Meeting time and place

MWF 11:00 AM - 12:50 PM; Music 2215

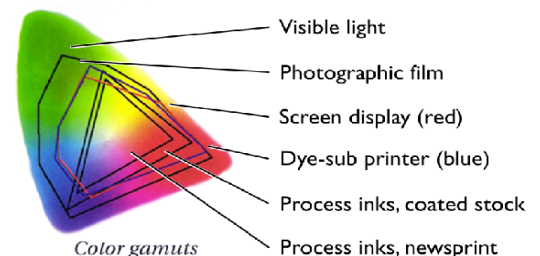
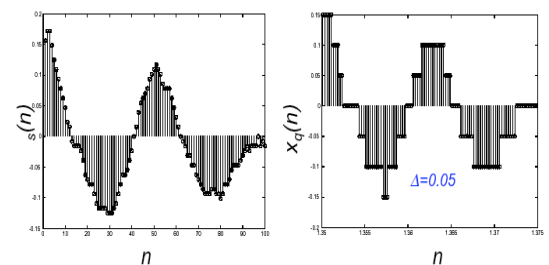
Electronic Resources

Course Web Site

<http://www.create.ucsb.edu/200C>

Email Mailing List

<http://mat.ucsb.edu/mailman/listinfo/200C> to join



MAT 200C Reader Contents

0. Introduction & Review

Course introduction and overview
Reviews of topics from Math, EE, CS, Psychology, etc.

1. Some Topics from Electrical and Computer Engineering

Computer design: system and processor architecture
Hierarchical memory and the storage pyramid
High-performance computing
I/O and Networking
Advanced software architecture

Readings

Wikipedia Entries for *Computer*, *Computer Hardware*, *CPU*, *CPU Design*, *Microarchitecture*, *Super-computer*, *Operating System*, and *IEEE 802.11*.
M. Murdocca and V. Heuring. “Principles of Computer Architecture” (slides from book)
AMD, Inc. “AMD K6 Processor Multimedia Technology”
M. Altmann. “Hardware Acceleration for Window Systems.” (<http://davis.wpi.edu/~matt>)
Wikipedia Entries for *USB*, *FireWire* and *MIME*
L. Alldrin. “Firewire and USB” in Pro Audio Review 12/99
Bluetooth and Wireless Networking, *JAES* 11/2002
G. Robertson and D. McAuley. “Sample Rate Sync over an ATM Network” Proc 1997 ICMC

See also

C. Hand. “CSYS1001 Computer Architecture Notes” (www: De Montfort University)
J. Pawasauskas. “MMX Technology.” (<http://davis.wpi.edu/~matt/courses/cs563/talks/powwie/p3/mmx.html>)
S. Ball and R. Probin. “An Overview of AltiVec on the PowerPC (Lightsoft)”
H. Nguyen and L. John. “Exploiting SIMD Parallelism in DSP and Multimedia Algorithms using the AltiVec Technology (www: UT)”

Software

Compilers, interpreters, scripting languages, operating systems, APIs

2. Media data, signals, and symbols

Overview of Information theory
Time and dimension in media data signals
Data quantization and conversion
Media data: events and streams

Readings

Wikipedia Entries for *Information Theory*, *Data Compression*, *Codec*, *Audio Compression*, *Image Compression*, *Video Compression*, *Comparison of Audio Codecs*, and *Comparison of Video Codecs and Container Formats*.
C. Shannon and W. Weaver. “Recent Contributions to the Mathematical Theory of Communication.”
Wikipedia Entries for *Auditory System* and *Visual System*
G. Garnett. “Music, Signals, and Representations” in “Rep. of Musical Signals”
R. Dannenberg. “Music Representation Issues, Techniques, and Systems” CMJ 17:3
G. Wiggins et al. “A Framework for the Evaluation of Music Rep. Systems” CMJ 17:3

See also

S. T. Pope. “Musical Object Representation” in “Musical Signal Processing” (SZ)

Coding, encryption and compression readings

Software

CSound, SuperCollider schedulers and score languages

SoundHack (format conversion)

GraphicConvertor

3. Data models, representations, and formats; data storage and transfer

Digital representation of sound, image, and media stream

Sound and image data formats

Audio/visual rendering: models of objects, space and light

Real-time and isochronous transfer of media data over networks

Readings

Formats & Languages

Wikipedia entries for *Audio file format*, *Image file formats*, *Format war*, and *Comparison of high-definition optical disc formats*

F. Harris. "Discrete Techniques" Lecture Slides.

M. Altmann. "About NURBS." (www)

J. Goubeaux. "Color Theory and Digital Color Management." MAT 200C Course paper and presentation slides.

S. Dunn. "Digital Color." (<http://davis.wpi.edu/~matt/courses/color>)

MPEG 2, 4 and 7

F. Pereira. "New Trends in Video and Image Coding." (www)

Background on MPEG TV Compression (www)

MPEG-1 Data Structures (www)

N. Day. "MPEG-7." XML-Journal

MPEG-Industry.com. "What is MPEG-7?" (www)

See also

[S. T. Pope. "Music Composition and Editing by Computer" in "Music Processing" (AR)]

<http://vivaldi.ece.ucsb.edu/PDF/multimedia.pdf>, <http://vivaldi.ece.ucsb.edu/Netra>

RealAudio/LiquidAudio descriptions (www)

http://ils.unc.edu/dempsey/other_stuff/rfc2045.shtml

Software

MP3/RealAudio/LiquidAudio encoders/players

PBM Tools

4. Digital processing of signals and symbols

Data processing in the time, spectral, spatial, and other domains

Sound analysis/synthesis techniques

Image and video processing and animation

Readings

J. O. Smith. "Physical Modeling Synthesis Update" CMJ 20:2

G. Kendall. "A 3-D Sound Primer" CMJ 19:4

V. Galloway and S. Wilkinson. "Surrounded by Sound" in *Electronic Musician* 12/99

N. Stewart. "Choosing Imaging Sensors for Visible Light." (www)

Photocourse. "The Digital Image and Image Sensor." (www)

POV-Ray Beginning Tutorial (www.povray.org)

S. Martin. "Animation Tricks." (<http://davis.wpi.edu/~matt>)

See also

DASP refs. MAT240A-F

Persistence of Vision Ray Tracer -- <http://www.povray.org/documents>

OpenGL tutorials

Supercollider refs

Software

Soundhack (analysis/resynthesis), pvoc

Persistence of Vision Ray Tracer (POV-Ray)

MSP/Jitter

5. Media I/O devices

Perception, psychology, and media

Issues of communication theory

Media data capture and performance

Transducers for signals, control variables, and events

Readings

Wikipedia entries for *Human interface device*, *Computer keyboard*, *Computer mouse*, *Game controller*, *Speech recognition*, and *Motion capture*

J. Pressing. "Cybernetic Issues in Interactive Performance Systems" *CMJ 14:1*

M. Kölsch, M. Turk. "Keyboards without Keyboards: A Survey of Virtual Keyboards" *Proc 2002 SIMS Workshop*, UCSB

See also

<http://www.cs.monash.edu.au/~dcron/glove>

Ergonomic I/O devices such as <http://www.tifaq.com>

Software

MIDI sequencer SW

SuperCollider + MIDI + VRML

6. Applications of media technology in arts, entertainment, and education

Composition, arrangement, production, data management

Interactive hardware/software tools/instruments/installations

Media content and distribution/dissemination

New (digital/distributed/interactive) media

Virtual environments, telepresence, and remote collaboration

Media data in education

Readings

Wikipedia entries for *Scientific visualization*, *Hypertext*, *HTML editor*, *Graphic art software*, *Digital audio workstation*, *Collaborative software*, *Edutainment*, and *MMORPG*

TOC from baddesigns.com

TOC from B. Schneiderman's 1993 *Encyclopedia of Virtual Environments* (HITL @ UW)

G. Wang, A. Misra and P. R. Cook. "Building Collaborative Graphical Interfaces in the Audicle." *Proc NIME 2006*

See also

S. T. Pope. "Music Composition and Editing by Computer" in *Music Processing* (AR)

E. Brandt and R. Dannenberg. "Low-latency Music Software" *Proc 1998 ICMC*

E. Frecon and M. Stenius. "DIVE: A scaleable network architecture for distributed virtual environments" (<http://www.sics.se/dive/dive.html>)

J. Young and I. Fujinaga. "Piano Master Classes via the Internet" *Proc 1999 ICMC*

R. Rowe and E. Singer. "Two Highly Integrated Music and Graphics Performance Systems" *Proc. 1997 ICMC*


D. Keislar et al. "A Content-Aware Sound Browser" *Proc 1999 ICMC*

Beatnik Audio Engine Description (Beatnik, Inc.)

MAT 200C Code Archive

- Lab images, sounds,
- and utilities
- Reference readings
- Student projects

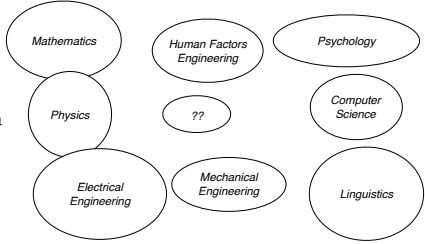
▼	Lab	85 MB
▶	ImageTester	4.5 MB
▶	pbmplus	1.5 MB
▶	Presentations	12.4 MB
▶	Ras_file_Info	131.5 KB
▶	SampleImages	4 MB
▶	snd-utils	84.6 KB
▼	SoundExamples	75.4 MB
▶	CelestialTerrestrialCommuters	7.8 MB
▶	testsamples	40 MB
	FallenAngels_8bit_mu-law.snd	3.5 MB
	FallenAngels_8bit.aiff	3.5 MB
	FallenAngels.aiff	7 MB
	MrDC_8bit_mu-law.snd	3.3 MB
	MrDC_8bit.aiff	3.3 MB
	MrDC.aiff	6.7 MB
▼	Venice_photo	26.8 MB
	scan-16.tiff	656.3 KB
	scan-256.tiff	1.1 MB
	scan-gs.tiff	1.1 MB
	scan.tif	20.3 MB
	scan2.jpg	162.9 KB
	scan2.tiff	3.3 MB
	LabNotes.txt	7.8 KB
	lsf.java	2 KB
	samp_quant.sc	3.6 KB
▼	Refs	166.1 MB
▶	CCS130	14.4 MB
▶	ComOrg	1.6 MB
▶	LispMachines	150 MB
▶	POCA	2.5 MB
▶	UML-BOOST	6.5 MB
▼	Students	332.9 MB
▶	200C-2008-Students	165 MB
▶	MAT200C_2008 Folder	15 MB
▶	Students-2000	6.8 MB
▶	Students-2001	22.3 MB
▶	Students-2002	22.9 MB
▶	Students-2004	70.4 MB
▶	Students-2005	30.1 MB



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C: Survey of Multimedia Technology and Engineering

Stephen T. Pope
MAT/UC Santa Barbara
stp@mat.ucsb.edu
Winter Quarter, 2008



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

1

MAT 200C Course Overview

- Multimedia Technology and Engineering
 - 0. Introduction and review
 - 1. Computer engineering: hardware and software
 - 2. Information, media data, signals, and symbols
 - 3. Data models, representations, and formats
 - 4. Digital manipulation of media data
 - 5. Media I/O devices and HCI
 - 6. Applications in arts, entertainment, education
 - 7. In-class presentations

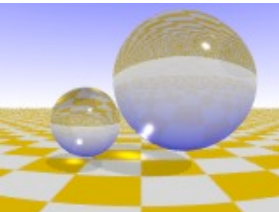
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

2

Lecture 1 Outline

- Logistics
- Course Materials
- Grading/Projects
- Overview/Outline
- Review
 - Math, Physics, Psychology, EE, CS, etc.



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

3

Logistics

- Meeting time
 - Lectures: Mon/Wed 11:00 - 12:50, Music 2215
 - Lab/Studio: Fri 11:00 - 12:50
- Office hours
 - STP: Mon/Wed 10:00 - 11:00 AM and by appt.
 - TA (Ami Amar): by appt.
- Course format
 - Readings, lectures, discussion, demos
 - Quizzes, papers, presentations, projects

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

4

Format

- Four components:
 - Lectures, reading, quizzes
 - Application presentations, discussion
 - Design studio, homeworks
 - Coding lab, projects

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

5

Course Materials

- Readings book
 - Surveys, papers, case studies, etc.
- Presentation slides
 - Every slide presented in the lectures
- Additional readings & web research
 - Current state-of-the-art, commercial developments, artistic applications
- Software to be used
 - Sound/image synthesis, processing, etc.

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

6

Grading/Projects

- Class participation
- Quizzes on readings (may happen any time)
- Written paper
 - 8-10 pages
- Software project
 - 300-800 line program in C, C++, or Java
- In-class presentation (15-25 min)

Important Skill Areas

- Skills to be developed
 - Research/writing
 - Preparation/presentation
 - Technical/programming

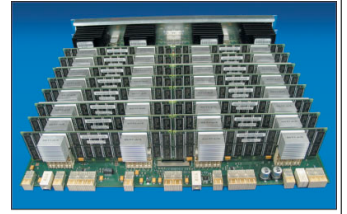


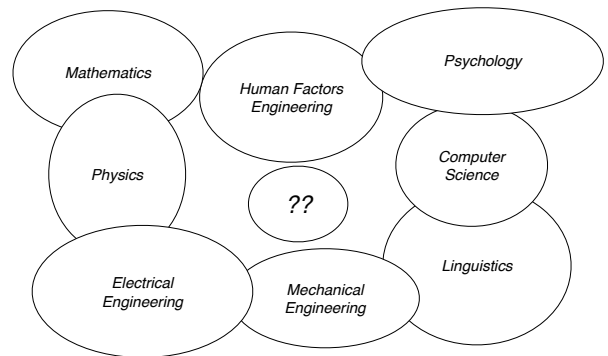
Figure 3
Blue Gene/L node card.

Why MAT 200C?

- Assumed: we all use digital technology for multimedia applications with computers, cell phones, cameras, music instruments, content distribution systems, etc.
- So: what are the core theoretical and technology issues in these applications?



What is MAT 200C?



Course Overview/Outline

- Multimedia Technology and Engineering
 - 0. Introduction and Review (1 week)
 - 1. Computer engineering: hardware and information (1.5)
 - 2. Media data, signals, and symbols (1)
 - 3. Data models, representations, and formats (1)
 - 4. Digital manipulation of media signals (1.5)
 - 5. Haptics, Media I/O devices and HCI (1)
 - 6. Applications in arts, entertainment and education (1)
 - 7. In-class presentations (1.5)

Outline: Topic 1

- Topics from electrical engineering and computer science:
 - Computer design
 - System architecture and configuration
 - Memory and the storage pyramid
 - Processor architecture and instruction sets
 - High-performance media computing
 - Networks for media data : USB & FireWire
 - Application Architecture

Outline: Topic 2

- Information theory and its applications
 - Encoding, encryption, error-handling, data hiding, compression
- Media data, signals, and symbols
 - Time and dimension in media data
 - Kinds of signals and symbols
 - Data quantization and conversion
 - Media data: events and streams
- Representations, storage formats, and interchange formats/protocols

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

13

Outline: Topic 3

- Data models, representations, and formats; data storage and transfer
 - Representation of sound, image
 - Sound and image data formats
 - Audio/visual rendering: models of objects, space and light
 - Storage and retrieval of media data: media databases
 - Real-time and isochronous transfer of media data over networks

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

14

Outline: Topic 4

- Digital manipulation of signals and symbols
 - Data processing in the time, spectral, spatial, and other domains
- Basic models and synthesis techniques
 - Sound analysis/synthesis techniques
 - Control and interaction
 - Image/scene description and rendering
 - Video generation and animation

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

15

Outline: Topic 5

- Media I/O devices
 - Perception, psychology, and media
 - Issues of communication theory
 - Media data capture and performance
 - Transducers for signals, control variables, and events
 - Keyboard ergonomics
 - I/O Interface survey

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

16

Outline: Topic 6

- Applications of media technology in arts, entertainment, and education
 - Composition, arrangement, production, data management
 - Interactive hardware/software tools, instruments, and installations
 - Media content and distribution/dissemination
 - New (digital/distributed/interactive) media
 - Virtual environments, telepresence, and remote collaboration
 - Media data in education

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

17

Student Projects

The collage displays various student projects. On the left, a terminal window shows code for a media control system. In the center, there's a 3D rendering of a scene with a car and buildings. To the right, a 2D image shows a person's silhouette. At the bottom right, a photograph shows a physical installation with a small car and a screen.

MAT 200C

18

MEDIA ARTS & TECHNOLOGY PROGRAM

Preview
Review




Image by STP,
generated with
POV RayTracer

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

19

Review: Mathematics in 15 minutes

- Basic math skills
 - Arithmetic
 - Algebra
 - Geometry
 - Trigonometry
 - Calculus
 - Notations, Greek alphabet

Review: Mathematics

- Arithmetic
 - Numbers, scalars and vector magnitudes, $3 + 4 = 7$
units, representations and radices $0xFF$
 - Expressions involving numbers $\ln(x) + \ln(y) = z$
 - Basic operations $e^{i\theta}$
 - Exponentials and logarithms $2 + 3i$
 - Imaginary and complex numbers

Example: Numerical Code in C++

```
if (freqDyn) {
    this->pullInput(freqPort, outputBuffer); // if we have a dynamic freq.
    freq = freqPort->mBuffer->monoBuffer(0); // pull its buffer (this calls its nextBuffer method)
    freqC = *freq++; // get the pointer to its buffer
} else { // grab the first value
    freqC = freqPort->mValue; // otherwise
    // get the port's constant value
}

for (unsigned i = 0; i < outputBuffer.mNumFrames; i++) { // the sample loop
    *buffer++ = sin(mPhase); // compute and store sine value
    mPhase += (freqC * rateRecip); // increment phase by possibly dynamic frequency
    if (freqDyn) // if the freq. is dynamic
        freqC = *freq++; // update it from the control input's buffer pointer
}

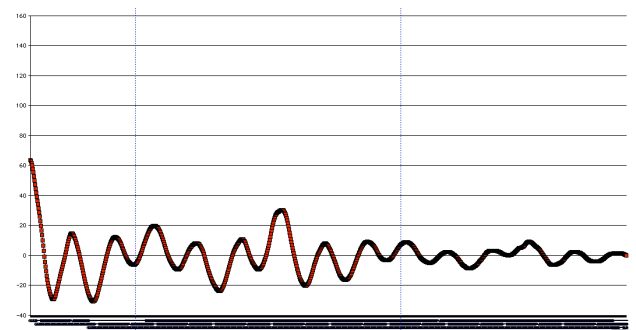
while (mPhase >= CSL_TWOPHI) // wraparound after 2pi radians
    mPhase -= CSL_TWOPHI;
```

CSL Sine oscillator example

Review: Mathematics

- Algebra
 - Values and variables, naming, assignment, $x = 3$
and calculation
 - Functions of 1 or more variables $y = 3x + 2$
 - Normal polynomial functions $y = a + bx + cx^2$
 - Bound and unbound variables
 - Units of measure (solving for the units)
 - Graphing of functions in 2- and 3-D

Function of 1 Variable



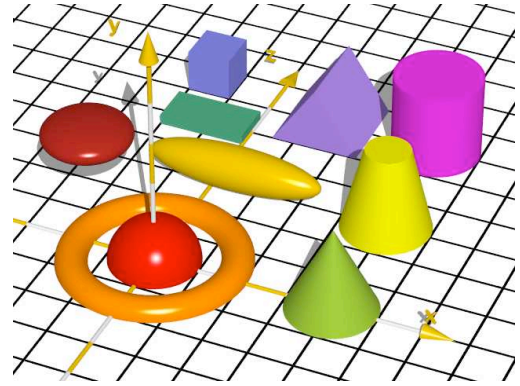
- Y vs. X or Y vs. T (e.g., auto-correlation of speech)

Review: Mathematics

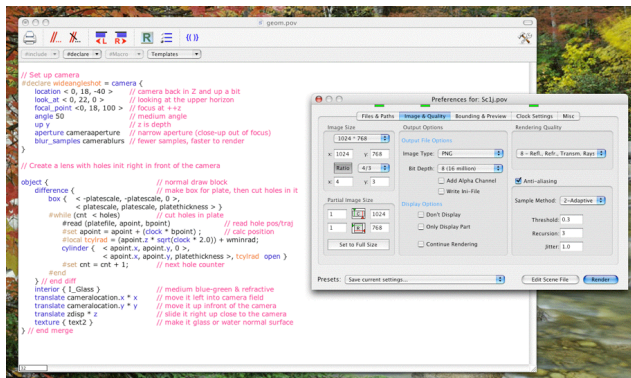
- Geometry
 - Cartesian and polar coordinate spaces
 - Common polygons
 - Dimensionality
 - Motion
- Trigonometry
 - The unit circle (as a function)
 - Sine/cosine as functions
 - Radial vectors and velocity
 - The exponential formulation

$$\begin{aligned} x &= 3 \\ y &= 2 \\ r &= 2 \\ \theta &= 60 \end{aligned}$$

Geometrical Primitives in POV-Ray

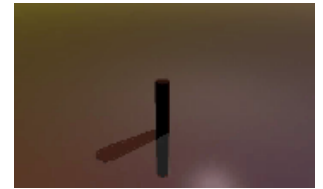


Example: POV-Ray Code



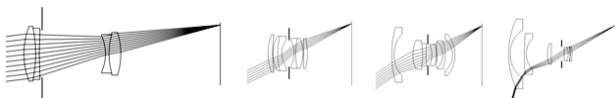
Review : Mathematics

- Calculus
 - Derivatives and rates of change
 - Example: position, velocity, acceleration
 - Integrals and accumulation
- Notations
- Other topics?
 - Analysis
 - Topology
 - Higher Algebra

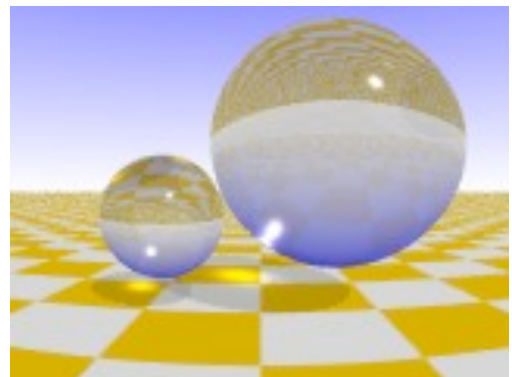


Review: Physics for MAT

- Optics and Acoustics
 - Waves and radiation
 - Frequency, dispersion speed, and wavelength
 - Spectra and filters (a vs. t and a vs. f)
 - Boundaries: reflection, absorption, and diffusion
 - (more later)



Optical Reflection and Refraction



MEDIA ARTS & TECHNOLOGY PROGRAM MAT 200C

MEDIA ARTS & TECHNOLOGY PROGRAM **MAT 200C**

Circuits and Systems

Figure 3. Practical power amplifier circuit.

Fig. 3. Practical power amplifier circuit.

MEDIA ARTS & TECHNOLOGY PROGRAM **MAT 200C**

Example: SW Architecture

The diagram illustrates the high-level view of the Blue Gene/L system software architecture. It is organized into several interconnected components:

- Console front end:** Contains the MMCS library, which connects to a Database.
- Database:** A central data store connected to the Console front end and the Scheduler.
- Scheduler:** Contains the MMCS library and connects to the Database.
- Midplane monitoring and control system (MMCS):** A central component that interfaces with the Control-FPGA and the Control Ethernet.
- Control-FPGA:** A hardware component that manages the system, connected to the MMCS and the Control Ethernet.
- Control Ethernet:** A network fabric that connects the MMCS, Control-FPGA, and the various compute nodes.
- File servers:** External storage units connected to the system via a Functional Ethernet.
- Compute Nodes:** The system is divided into two main sections, each connected to the Control Ethernet via a JTAG interface:
 - Top Section (psst 0):** Contains an I/O node 0 (Linux/CIOD), a Compute node 0 (CNK/Compute kernel/User applications), and a Compute node 63 (CNK/User applications). These nodes are interconnected via Torus and Collective communication links.
 - Bottom Section (psst 1,023):** Contains an I/O node 1,023 (Linux/CIOD), a Compute node 0 (CNK/Compute kernel/User applications), and a Compute node 63 (CNK/User applications). These nodes are also interconnected via Torus and Collective communication links.

The diagram shows a complex network of connections between these components, highlighting the distributed nature of the Blue Gene/L architecture.

Figure 2

High-level view of the Blue Gene/L system software architecture.

© MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200

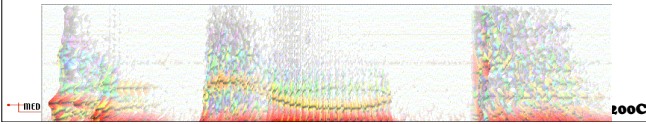
High-level view of the Blue Gene/L system software architecture.

MEDIA ARTS & TECHNOLOGY PROGRAM **MAT 200C**

MEDIA ARTS & TECHNOLOGY PROGRAM **MAT 200C**

Transforms and Representations

- The Fourier formulation (and spectra)
 - DFT (discrete)
 - STFT (short-time)
 - FFT (fast)
- Convolution
 - Time-domain vs. spectral-domain operations
- The Z transform
 - General-case of DFT



37

Processing of Digital Signals

- Recording and reproduction (transduction, rendering)
- Analysis for model derivation, parameter estimation, feature extraction, data reduction
- Model-based synthesis
- Perceptual processing
- Signal semantics

CSL
Max/MSP
SSUM
OpenGL
POV-Ray

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

38

Looking Forward: Kinds of Signals

- Sound pressure waves (+ analogs)
- Spatial signals (decorrelation, etc.)
- Light on film or retina
- “Control” of some parameter
- Clock, event trigger
- “Symbol” as abstraction

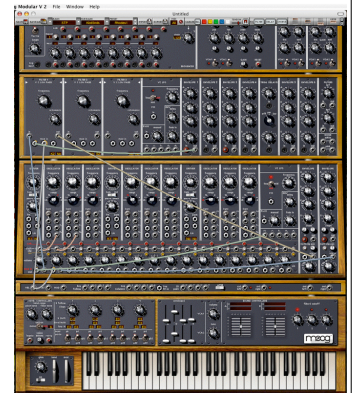
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

39

Audio Fundamentals

- Sound and Acoustics
- Analysis/Synthesis techniques and sound models
 - Additive
 - Subtractive
 - “Non-linear”
- The frequency domain and spectra



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

40

Digital Images and Video

- Optics, sight and images
- Space, light, texture and color
- Representation of still images
- Representation of dynamic images
- Color models and representations
- Scene/world description and rendering

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

41

Object Models for Rendering

- Objects
 - Solids, surfaces, volumes
- Color
- Light sources
- Space
- The viewer and perspective
- Inter-object effects

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

42

What's Next

- Readings for computer engineering
 - Computer design
 - Processor architecture
 - Storage and memory hierarchy
- Readings for information theory

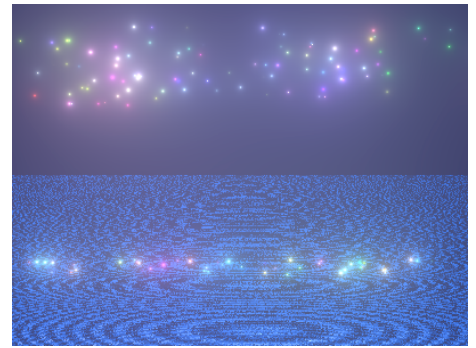
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

43

MEDIA ARTS & TECHNOLOGY PROGRAM

End of Review



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

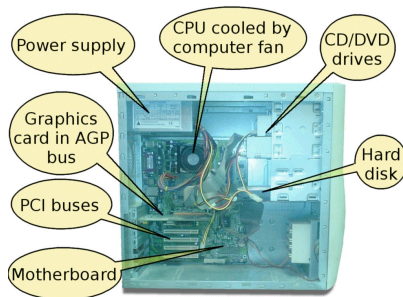
44

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C: Survey of Media Technology

Topic 1: Elements of Electrical Engineering and Computer Science

Stephen T. Pope
MAT/UC Santa Barbara
stp@mat.ucsb.edu
Winter Quarter, 2008



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

45

Topic 1: Elements of Electrical Engineering and Computer Science

- Computer architecture
- Instruction set and processor design
- High-performance computing
- Programming and program translation
- IO, media networks and time
- SW & application architecture

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

46

Topic 1 Readings: Computers

- Wikipedia Entries for *Computer*, *Computer Hardware*, *CPU*, *CPU Design*, *Microarchitecture*, *Operating System*, and *Supercomputer*.
- M. Murdocca and V. Heuring. "Principles of Computer Architecture" (slides from book)
- AMD, Inc. "AMD K6 Processor Multimedia Technology"
- See also
 - C. Hand. "CSYS1001 Computer Architecture Notes" (www: De Montfort University)
 - J. Pawasauskas. "MMX Technology." (<http://davis.wpi.edu/~matt/courses/cs563/talks/powwie/p3/mmx.html>)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

47

Readings: Compression & Networks

- Wikipedia on *USB*, *FireWire*, *MIME* and *IEEE 802.11*
- Bluetooth and Wireless Networking, *JAES 11/2002*
- Background on MPEG TV Compression (www)
- MPEG-1 Data Structures (www)
- L. Alldrin. "Firewire and USB" *Pro Audio Review*
- G. Robertson and D. McAuley. "Sample Rate Synchronization over an ATM Network" *Proc 1997 ICMC*

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

48

Extra References

- Computer design
 - New processors and configurations
- Software architecture
 - Compilers, interpreters and assemblers
- Information theory
 - Coding, transmission, compression, encryption
- Applications

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

49

Overview of Computer Architecture

- Computer and processor design
 - HW/SW systems
 - CPU architecture
 - Processor and memory issues
 - Bus and I/O architecture
- Software/hardware translation
 - Assembly languages
 - Development: compilers and tools
- Hardware for multimedia
 - Instruction set and architecture support

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

50

Computer Architecture

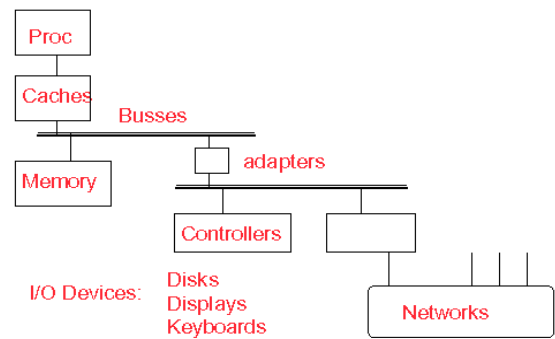
- General organization
 - Processors and memory configuration
 - Busses and interconnection
 - Processing, storage, and I/O
 - Configurations: server, workstation, PDA, ...
- CPUs and microsequencers
 - Instruction-set processor
 - Instruction/data formats
 - Register set

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

51

Computer System Components



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

52

Computer Organization

- Components
 - Processing, control
 - Storage
 - I/O (media, networking)
 - Busses (interconnection)
- Models: bus, von Neumann, switch, ...
- Hierarchical machine model
- Hierarchical memory model

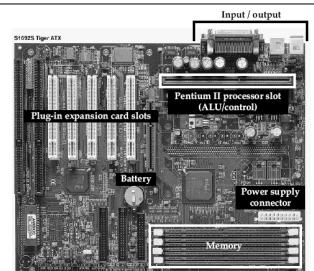
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

53

Computer System “Motherboard”

- CPU(s) (possibly on daughter-boards)
- Cache (on-chip and L2/3)
- Main RAM (socketed)
- Memory/CPU bus controller
- I/O bus controller(s) (std interface)
- Network interface(s): LAN, lo-speed
- Graphics interface, VRAM (?)
- Sound I/O (?)



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

54

I/O and Interfaces

- Storage systems – disk (RA), tape (SA)
- LAN/WAN (different?)
- Serial I/O (USB, FireWire)
- Special user I/O - Mouse, frame buffer
- Multimedia I/O - Sound, MPEG HW
- DMA Controllers
 - Take over the RAM bus to do block transfers for buffered I/O (disk, net, DAC, etc.)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

55

Bus(s)es

- Parallel interconnection between subsystems
- Internal CPU buses
 - Separate operand and cache busses?
- Cache Bus between CPU and Cache
 - Widest, custom
- Memory Bus & RAM speed/size
 - Wide, standardized
- General-purpose I/O Bus
 - SCSI, PCI (backplane, plug-in cards)
- Serial network busses

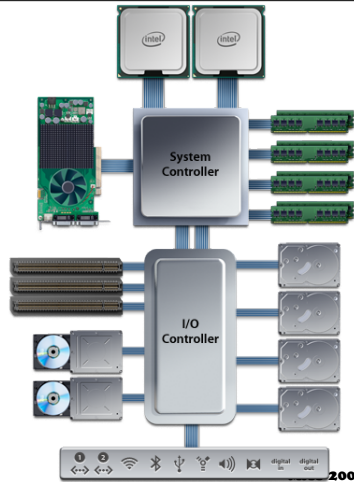
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

56

Computer Components

- Example: modern PC



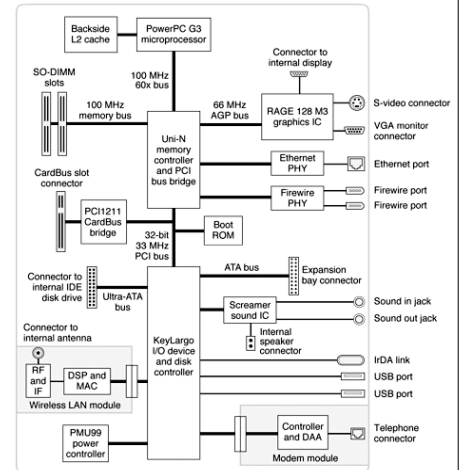
MEDIA ARTS & TECHNOLOGY PROGRAM

200C

57

Computer Components

- Example: Apple PowerBook G3 (2000)

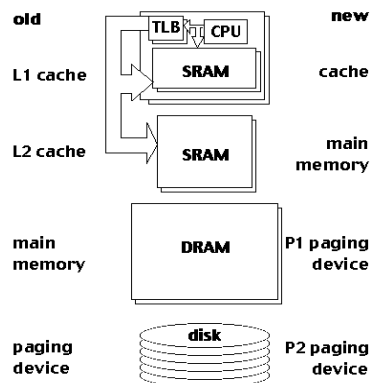


MEDIA ARTS & TECHNOLOGY PROGRAM

58

Storage

- The Storage Hierarchy
- Nearly constant speed * volume product over many orders of magnitude



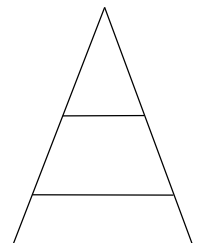
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

59

The Storage Pyramid

Medium	log(size) 10 ^x bytes	log(speed) 10 ^x sec
• Registers	3	-8
• L1 Cache	5	-7
• L2 Cache	6	-6
• SDRAM	8	-5
• Disk	11	-2
• Tape	15	1
• Tape/DVD juke-box		



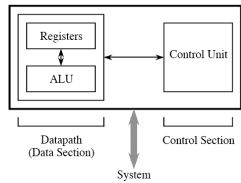
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

60

CPU Organization

- Storage Registers (8-1024)
 - General, floating-point
 - Program counter, status register, stack pointer, interrupt vector
- 1 or more ALUs (arithmetic/logic unit)
- 1 or more FPUs (floating-point math unit)
- Bus Interfaces
 - Cache, memory, graphics, storage, I/O
- Cache/VM logic (TLB)
- Microsequencer, control store (writeable?)

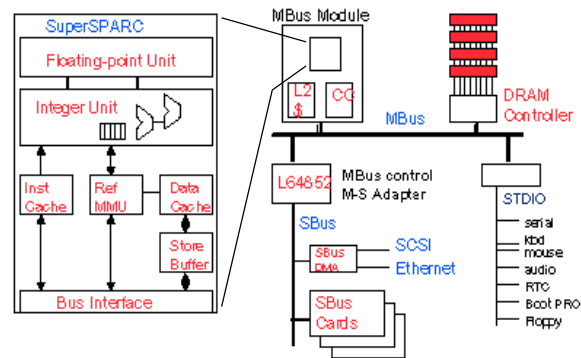


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

61

Sun SPARCstation-20 (1994)

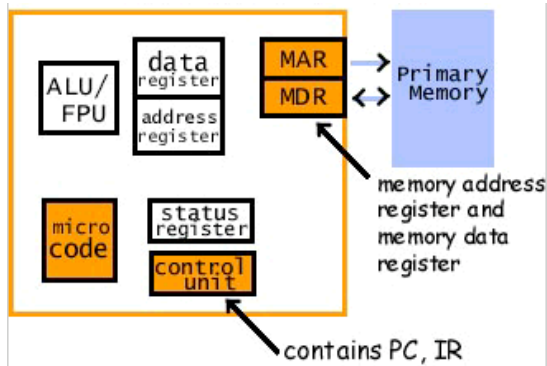


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

62

CPU Data Management



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

63

Control Registers in the CPU

- Program counter (address of current instruction)
- Status register (describes last result: zero, negative, etc.)
- Interrupt control (current priority)
- Special memory address/data registers
- Stack pointer(s)
- Memory management (cache pages)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

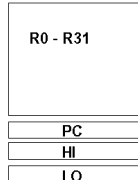
64

MIPS R3000 Registers and Operations

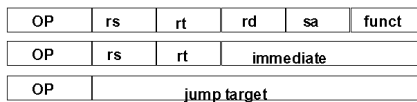
Instruction Categories

- Load/Store
- Computational
- Jump and Branch
- Floating Point
 - coprocessor
- Memory Management
- Special

Registers



3 Instruction Formats: all 32 bits wide

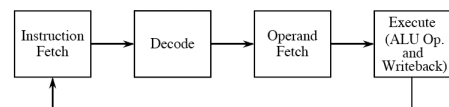


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

65

Instruction Processing



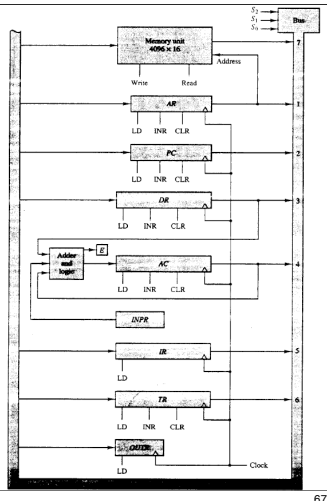
- Fetch instruction(s)
- Parse operand and addressing mode
- Load operands
- Perform operation
- Store result & status flags
- Chose next instruction to execute

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

66

Lowest-level View of a CPU: Register- transfer



67

Instruction Set (see readings)

- Move/copy data (load/store)
 - MOV, b := a
- Logical (ops, shift/rotate)
 - XOR, BIT(x), CMP
- Arithmetic (integer)
 - SEZ, CCC
- Status registers
 - SEZ, CCC
- Control flow (jump)
 - JMP, JSR, RTS, BGT
- Traps, I/O, Interrupts
 - HALT, IOT, RTI

MAT 200C

68

Instruction Set Example

- The ARC ISA is a subset of the SPARC ISA.

	Mnemonic	Meaning
Memory	ld	Load a register from memory
	st	Store a register into memory
Logic	sethi	Load the 22 most significant bits of a register
	andcc	Bitwise logical AND
	orcc	Bitwise logical OR
	orncc	Bitwise logical NOR
Arithmetic	srl	Shift right (logical)
	addcc	Add
	call	Call subroutine
Control	jmp	Jump and link (return from subroutine call)
	be	Branch if equal
	bneg	Branch if negative
	bcs	Branch on carry
	bvs	Branch on overflow
	ba	Branch always

Principles of Computer Architecture by M. Murodova and V. Heuring

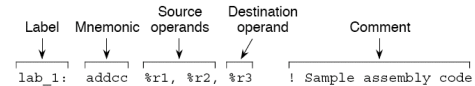
© 1999 M. Murodova and V. Heuring

MAT 200C

69

Assembly Language (finally)

- Example: ADD dst := dst + src
 - PDP-11 = 06SSDD8 = 0.110.sss.sss.ddd.ddd
 - Set N, Z status bits, V on overflow, C on carry
 - Register to register ADD R1, R2
 - Constant to register ADD 20, R0
 - Register to memory ADD R1, XXX
 - Memory to memory ADD@ #17750, XXX
- Assembler code format and directives



MAT 200C

70

Assembly Language Program

```
! This programs adds two numbers
.begin
.org 2048
progl: ld    [x], %r1      ! Load x into %r1
      ld    [y], %r2      ! Load y into %r2
      addcc %r1, %r2, %r3  ! %r3 ← %r1 + %r2
      st    %r3, [z]      ! Store %r3 into z
      jmp  %r15 + 4, %r0  ! Return

x:    15
y:    9
z:    0
.end
```

MAT 200C

71

A More Complex Assembly Language Program

```
! This program sums LENGTH numbers
! Register usage:
!   %r1 - Length of array a
!   %r2 - Starting address of array a
!   %r3 - The partial sum
!   %r4 - Pointer into array a
!   %r5 - Holds an element of a

.begin
.org 2048
! Start program at 2048
a_start .equ 3000
! Address of array a

ld [length], %r1 ! %r1 ← length of array a
ld [address], %r2 ! %r2 ← address of a
andcc %r3, %r0, %r3 ! %r3 ← 0
loop: andcc %r1, %r1, %r0 ! Test # remaining elements
      be done ! Finished when length=0
      addcc %r1, -4, %r1 ! Decrement array length
      addcc %r1, %r2, %r4 ! Address of next element
      ld %r4, %r5 ! %r5 ← Memory[%r4]
      addcc %r3, %r5, %r3 ! Sum new element into r3
      ba loop ! Repeat loop

done: jmp %r15 + 4, %r0 ! Return to calling routine

length: 20 ! 5 numbers (20 bytes) in a
address: a_start

.org a_start
a: 25 ! length/4 values follow
   -10
   33
   -5
   7

.end ! Stop assembling
```

72

CPU Characterization

- Data formats
- Register set
- Instruction/address formats
- Instruction set
- Memory map (lo, hi)
- Instruction processing cycle
- Data paths, I/O, timing
- Assembler format, directives, macros

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

73

CISC Addressing Modes

- Register: Argument is in register
- Absolute: Argument is at location XXX
- Relative: Argument is at location PC - XXX
- Indirect: Register points to argument
- Indexed: Register A points to base, and register B to the offset
- And more
 - Pre-/post-increment/decrement
 - Byte/word displaced

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

74

Memory Addressing Modes

- Example: to do $C = A + B$, a 3-address operation
 - In 1 operation (CISC)
 - Operands may be registers, memory locations...
 - Execution time varies; may involve up to 6 memory references
 - In 4 operations (RISC)
 - Load A
 - Load B
 - Add (2 registers or top 2 items on stack)
 - Store C
 - Operands are assumed to be registers, stack, or register windows, leads to easily predictable and faster execution

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

75

CISC vs. RISC

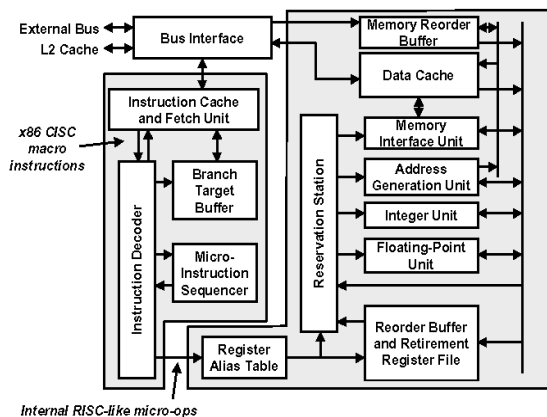
- CISC: Complex Instruction Set Computer
 - i86, MC68000, DEC PDP-11/VAX, IBM-360
 - Many addressing modes
 - Compilers/optimizers are complex
 - Some operations take many clock cycles
- RISC: Reduced Instruction Set Computer
 - i860, SPARC, PowerPC, MIPS, Alpha
 - Use explicit load/store instructions (pgms are longer)
 - Most instructions take 1 clock cycle
 - Compilers/optimizers and silicon are simpler
- RISC has a 2X performance advantage independent of technology (R2000 vs VAX8700, nVAX vs Alpha, i486 vs i860)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

76

CISC: Intel PentiumPro



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

77

HP(T)C



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

78

High-Performance Computing

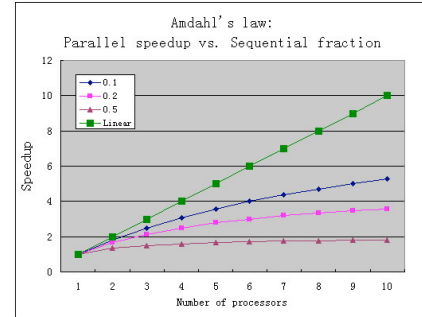
- How to speed up computational tasks
 - Better algorithms (if possible)
 - Faster CPU (esp. FPU)
 - Parallel computation (if appropriate)
 - Application-specific ICs (ASIC)
 - Are there other options?
- Is your program compute-bound or IO-bound?
 - Examples: sound synthesis vs. image rendering

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

79

Amdahl's (not Moore's) Law



- System architecture and algorithms determine *sequential factor* or *sequential fraction*.

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

80

Parallel Machines

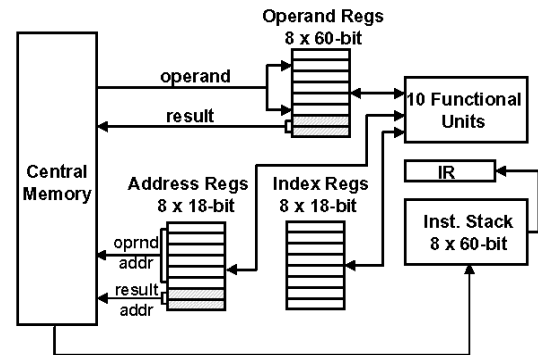
- SIMD: Single instruction, multiple data
 - Vector CPUs, tightly coupled, shared memory
 - Good for many scientific applications and signal processing
 - Programmed in parallel or vector languages
- MIMD: Multiple instruction, multiple data
 - Clusters, server farms, loosely coupled, I/O communication protocols
 - Good for dynamic applications and service-oriented architecture
 - Programmed as distributed services

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

81

SIMD: CDC 6600 (1970s)



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

82

MIMD: Beowulf Cluster (200X)



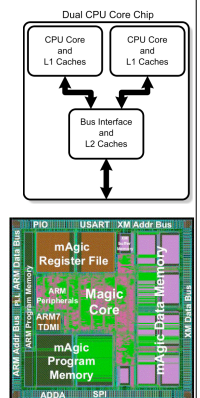
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

83

SMP MIMD

- Challenge: the limits of Moore's law
- Symmetric multi-processing (SMP) and multi-core processors
- Benefits: linear MIPS scaling
- Problem: software/OS support
- Current SOTA: 8-core SPARC or 4-core Intel-based systems
- A-SMP
 - Amtel VLIW f-p DSP + ARM7
 - IBM/Sony Cell: PowerPC + 8 SPEs



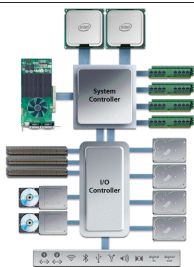
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

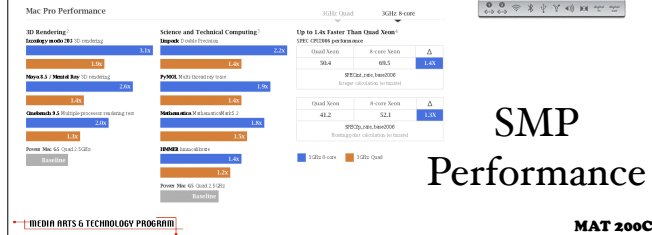
84

SMP Architecture

- ...not all that different
- Sequential factor $\ll 1$



SMP Performance



85

SuperComputer Design

- SIMD/MIMD processor design
 - ALU/FPU/I/O processors
 - Interconnection topology (figure)
 - # of layers and protocols
- Balance of processing and I/O
 - Interprocessor communication
 - Mass storage I/O
- Minimizing signal path length
- Coping with the heat

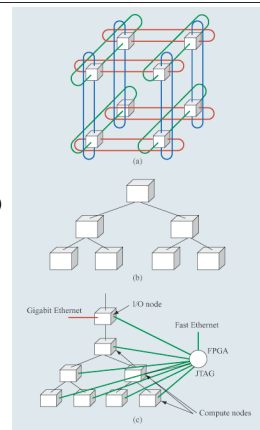
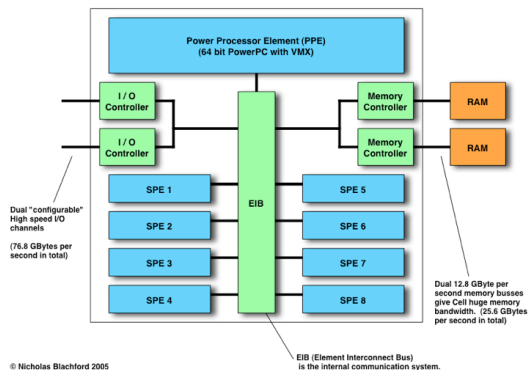


Figure 4
(a) Three-dimensional torus. (b) Global collect tree. (c) Genie control system network and Gigabit Ethernet.

86

High-end Processors (PS₃)

Cell Processor Architecture



© Nicholas Blachford 2005

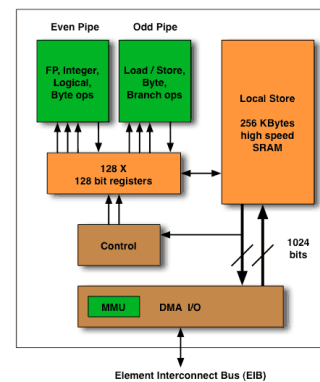
MAT 200C

87

Sony PS₃ Cell CPU

Cell SPE Architecture

Each SPE is an independent vector CPU capable of 32 GFLOPs or 32 GOPs (32 bit @ 4GHz).



© Nicholas Blachford 2005

88

Super-Computer History

Year	Supercomputer	Peak speed	Location
1942	Atanasoff-Berry Computer (ABC)	30 OPS	Iowa State University, Ames, Iowa, USA
	TRE Health Robinson	200 OPS	Blatchley Park
1944	Flowers Colossus	5 KOPS	Post Office Research Station, Dollis Hill
1946	Lipson ENIAC	100 KOPS	Abertown Proving Ground, Maryland, USA
	(after 1946 modifications)		
1954	IBM NORC	67 KOPS	U.S. Naval Proving Ground, Dahlgren, Virginia, USA
1956	MIT TX-0	83 KOPS	Massachusetts Inst. of Technology, Lexington, Massachusetts, USA
1958	IBM ANFSQ-7	400 KOPS	25 U.S. Air Force sites across the continental USA and 1 site in Canada (52 computers)
1960	UNIVAC LARC	500 MFLOPS	Lawrence Livermore National Laboratory, California, USA
1961	IBM 7030 "Stretch"	1.2 MFLOPS	Los Alamos National Laboratory, New Mexico, USA
1964	CDC 6600	3 MFLOPS	Lawrence Livermore National Laboratory, California, USA
1969	CDC 7600	36 MFLOPS	
1974	CDC STAR-100	100 MFLOPS	
1975	Burnough ILLIAC IV	150 MFLOPS	NASA Ames Research Center, California, USA
1976	Cray-1	250 MFLOPS	Los Alamos National Laboratory, New Mexico, USA (80+ sold worldwide)
1981	CDC Cyber 205	400 MFLOPS	(numerous sites worldwide)
1983	Cray X-MP/4	941 MFLOPS	Los Alamos National Laboratory, Lawrence Livermore National Laboratory, Battelle, Boeing
1984	M-13	2.4 GFLOPS	Scientific Research Institute of Computer Complexes, Moscow, USSR
1985	Cray-2B	3.9 GFLOPS	Lawrence Livermore National Laboratory, California, USA
1989	ETIA-D-GM	10.3 GFLOPS	Florida State University, Florida, USA
1990	NEC SX-3/44R	23.2 GFLOPS	NEC Fuchu Plant, Fuchu, Japan
1993	Thinking Machines CM-5/1024	65.5 GFLOPS	Los Alamos National Laboratory, National Security Agency
	Fujitsu Numerical Wind Tunnel	124.50 GFLOPS	National Aerospace Laboratory, Tokyo, Japan
	Intel Paragon XP/S 140	143.40 GFLOPS	Sandia National Laboratories, New Mexico, USA
1994	Fujitsu Numerical Wind Tunnel	170.40 GFLOPS	National Aerospace Laboratory, Tokyo, Japan
1996	Hitch SR2201/1024	20.4 GFLOPS	University of Tokyo, Tsukuba, Japan
	Hitch/Tsukuba CP-PACS/2048	368.2 GFLOPS	Center for Computational Physics, University of Tsukuba, Tsukuba, Japan
1997	Intel ASCI Red/1502	1.338 TFLOPS	Sandia National Laboratories, New Mexico, USA
1999	Intel ASCI Red/1802	2.3796 TFLOPS	
2000	IBM ASCI White	7.268 TFLOPS	Lawrence Livermore National Laboratory, California, USA
	NEC Earth Simulator	35.86 TFLOPS	Earth Simulator Center, Tsukuba, Japan
2002	GRAPE-6	64 TFLOPS	National Astronomical Observatory/University of Tokyo, Japan
2004	SGI Project Columbia	42.7 TFLOPS	NASA Advanced Supercomputing Facility at NASA Ames Research Center, California, USA
2004	IBM Blue Gene/L	70.72 TFLOPS	U.S. Department of Energy/IBM, USA
2005		136.8 TFLOPS	U.S. Department of Energy/U.S. National Nuclear Security Administration, 286.6 TFLOPS
			Lawrence Livermore National Laboratory, California, USA

89

SuperComputing SOTA (11/06)

Rank	Site/Country/Year	Name/Computer/Processors	Manufacturer Rmax/Rpeak (GFlops)
1	Lawrence Livermore National Laboratory United States / 2005	BlueGene/L eServer Blue Gene Solution / 131072 (Power)	IBM 280600/367000
2	Sandia National Laboratories United States / 2006	Red Storm Cray XT3 / 26544 (Opteron)	Cray 101400/127411
3	IBM Thomas J. Watson Research Center United States / 2005	BGW eServer Blue Gene Solution / 40960 (Power)	IBM 91280/114688
4	Lawrence Livermore National Laboratory United States / 2005	ASC Purple eServer pSeries p5 575 1.9 GHz / 12208 (Power)	IBM 75760/92761
5	Barcelona Supercomputing Center Spain / 2006	ManHearth BladeCenter J521, 2.3 GHz, Myrinet / 10240 (Power)	IBM 62630/94208
6	Sandia National Laboratories United States / 2006	Thunderbird Dell PowerEdge 1850 3.6 GHz, Infiniband / 9024 (Xeon)	Dell 53000/64972.8
7	Commissariat à l'Energie Atomique France / 2006	Tera-10 Novascale 5160, 1.6 GHz, Quadrics / 9968 (Itanium 2)	Bull SA 52840/63795.2
8	NASA Ames Research Center United States / 2004	Columbia SGI Altix 3700 1.5 GHz, Voltaire Infiniband / 10160 (Itanium 2)	SGI 51870/60960
9	GSFC Center, Tokyo Institute of Technology Japan / 2006	TSUBAME Grid Cluster Sun Fire X6400 Cluster, 2.425 GHz, ClearSpeed accelerators, Infiniband / 11088 (Opteron)	NEC and Sun 47380/82124.8
10	Oak Ridge National Laboratory United States / 2006	Jaguar Cray XT3 / 10424 (Opteron)	Cray 43480/54204.8

90

Highlights from the TOP Ten - 11/07

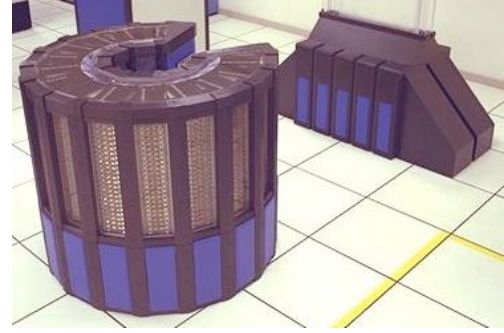
- 5 new systems and 1 substantially upgraded system with 5 of these changes placing at the top 5 positions.
- New #1 system is an enlarged version of the previous #1 - the DOE's IBM BlueGene/L system, 478.2 TFlop/s compared to 280.6 TFlop/s 6 months ago.
- #2 is a new installation of the same type of IBM system, a BlueGene/P installed in Germany, 167.3 TFlop/s.
- #3 system is new, in NM, built by SGI and based on the Altix ICE 8200 model. It was measured at 126.9 TFlop/s.
- At #4, the Computational Research Laboratories, a subsidiary of Tata in Pune (India), HP Cluster Platform 3000 BL460c system, 117.9 TFlop/s. This is the first system outside the US, Europe, or Japan to ever enter the TOP5.
- #5 system is also an HP CP 3000 BL460c system, 102.8 TFlop/s.
- The last new system in the TOP10 - #9 - is a Cray XT4 system at DOE's LBNL, 85.4 TFlop/s.

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

91

Cray SuperComputer (late-80s)



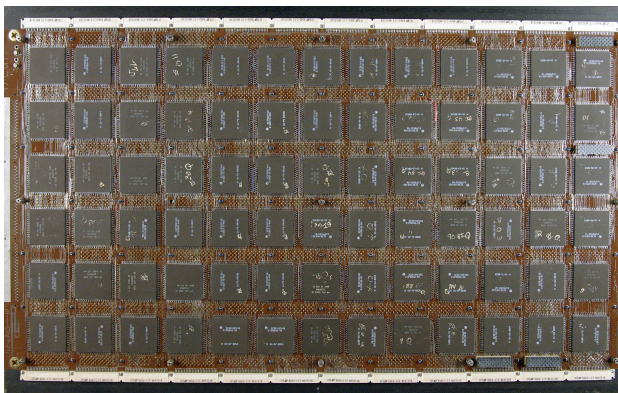
Cylindrical layout helps minimize signal path length while easing (water) cooling

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

92

Cray YMP (1992) CPU Board



MEDIA ARTS & TECHNOLOGY PROGRAM

Vector CPU + shared memory

MAT 200C

93

IBM BlueGene/L Architecture

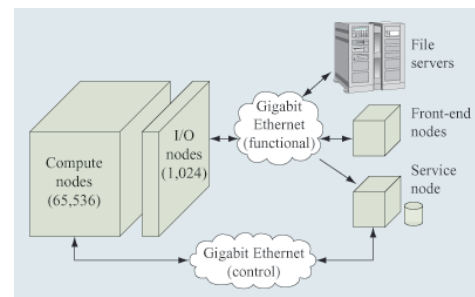


Figure 1

High-level architectural view of a complete Blue Gene/L system.

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

94

IBM BlueGene/L

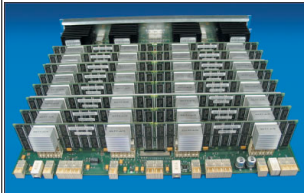


Figure 3
Blue Gene/L node card.

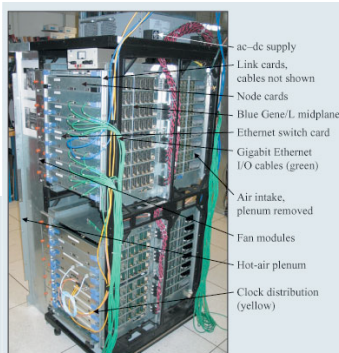


Figure 4

Blue Gene/L compute rack.

MEDIA ARTS & TECHNOLOGY PROGRAM

95

IBM BlueGene/L

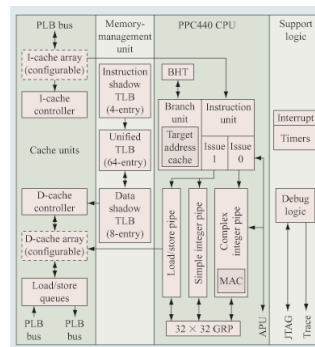


Figure 6

PowerPC 440 core.

MEDIA ARTS & TECHNOLOGY PROGRAM

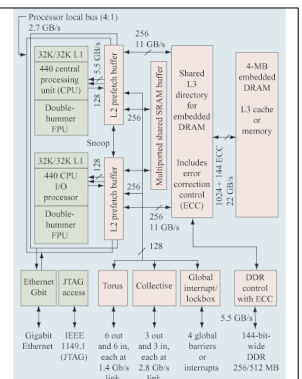


Figure 5

Blue Gene/L compute (BLC) chip architecture. Green shading indicates off-the-shelf cores. ©2002 IEEE. Reprinted with permission from G. Almási et al., "Cellular Supercomputing with System-on-a-Chip," *Digest of Technical Papers*, 2002. IEEE International Solid-State Circuits Conference.

MAT 200C

96

IBM BlueGene/L Software

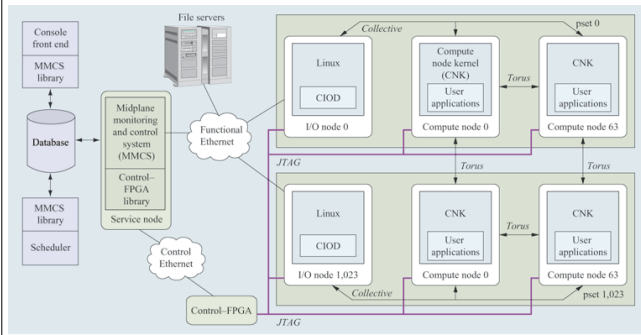


Figure 2

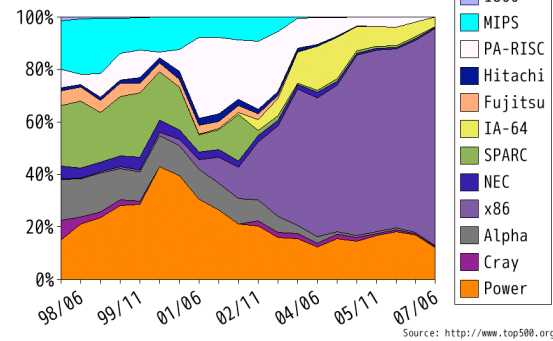
High-level view of the Blue Gene/L system software architecture.

MAT 200C

97

HPTC Processors

Processor Family Share of Top500

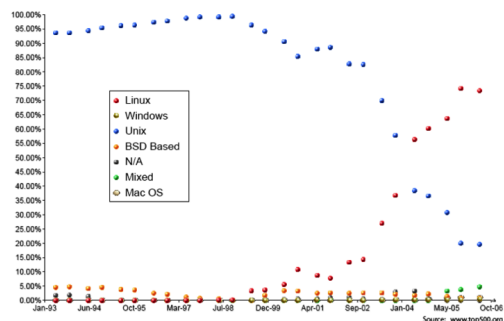
Source: <http://www.top500.org>

MAT 200C

98

Operating Systems for HPTC

Operating Systems Used On Top500 Supercomputers

Source: www.top500.org

MAT 200C

99

Review: Hardware

- Computer systems
 - System configuration and architecture
 - Processors, memory, storage, I/O, busses
 - Instruction-set processors
 - Processor architecture: CISC & RISC
- High-performance (technical, multimedia) computing

MAT 200C

100

Exercises, Demos

- Computer design
 - Standard system configuration
 - Advanced systems for gaming, multimedia
 - New processors and configurations
- Software architecture
 - Compilers, interpreters and assemblers
 - Organization of large software systems

MAT 200C

101

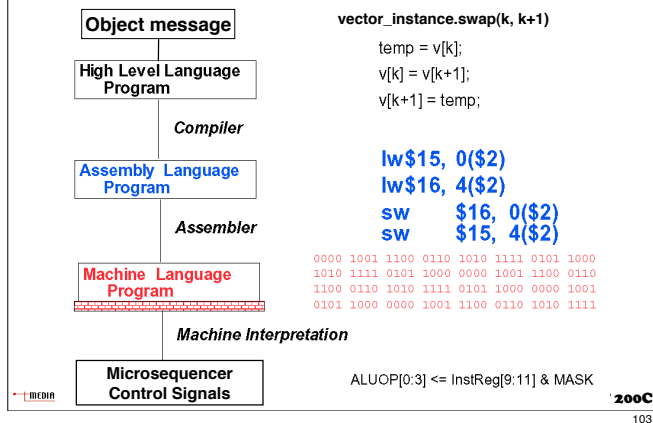
Interpretation of Programs



MAT 200C

102

Programs and Translation



103

Compiler, Interpreter

- Pre-process (includes, macros)
- Parse (read, tokenize) input stream
- Generate compact and expressive internal representation of code
- Optimize in the abstract
- Generate target machine code
- Optimize the native code
- Link in external libraries
- Typical flow: **cpp, cc1, as, ld**

MAT 200C

MAT 200C

104

Example: Compilation Steps

```
make
# call gcc compiler (input = .c file)
gcc -O3 -v -I../include -c -o LPanal.o LPanal.c
# call preprocessor/compiler (output .s file)
/usr/libexec/gcc/cc1 -quiet -v -I../include -fPIC
-quiet LPanal.c -O3 -o /var/tmp/ccqxcIfZ.s
# call assembler (input .s, output .o)
as -arch i386 -o LPanal.o /var/tmp/ccqxcIfZ.s
# call gcc loader (input .o files and libs)
gcc LPanal.o LOptions.o ../lib/libtsp.a -s -lm
-o LPanalyzer
```

MAT 200C

MAT 200C

105

Yet Another Programming Language Comparison (G. van Rossum, 1999)

	Teachability	Ease of use	Expressiveness	Expert usability
C/C++	hard	hard*	good	very good
Java	hard	hard	good	very good
JavaScript	easy	so-so	so-so	poor
Lisp	hard*	hard	good	very good
Logo	easy	easy	okay	poor
LogoMation	easy	easy	okay	poor
Perl	hard	hard	very good	very good
Python	easy	easy	good	very good*
Scheme	so-so	so-so	good	good
Smalltalk	so-so*	easy	good*	good*
Tcl	easy	so-so	so-so	so-so
Visual Basic	so-so	so-so	so-so	so-so

MAT 200C

MAT 200C

106

Linker, Link Editor, Loader

- To get from assembled routines to runnable apps
- Manage method args and return values (symbol table, entry points)
- Set up stack and memory map
- Connect to external libraries - at link time (static) or at start-up time (dynamic)
- Object files (.o, .obj) and executable files (a.out, .exe) standard formats
- Run-time: app loader, dynamic linker

MAT 200C

MAT 200C

107

Multi-stage Compilers, Virtual Machines

- Java, Smalltalk, LISP, etc.
- Compile source into platform-independent “virtual” machine language
- Run-time system (“virtual machine”) translates VM code to platform-specific (native) machine code
- Many options and trade-offs in VM language design and VM translator implementation
- See M&H reading

MAT 200C

MAT 200C

108

MEDIA ARTS & TECHNOLOGY PROGRAM MAT 200C

111

MEDIA ARTS & TECHNOLOGY PROGRAM **MAT 200C** 112

MEDIA ARTS & TECHNOLOGY PROGRAM **MAT 200C** 113

MEDIA ARTS & TECHNOLOGY PROGRAM **MAT 200C** 114

Multimedia Hardware

- Frame buffer interface
 - Drawing, rendering hardware
 - Special handling of pixel planes
- Graphics Cards
 - Fast frame buffer (pixel RAM)
 - Even faster (multi-port) texture RAM
 - Rendering pipeline: prune/translate/clip, draw, texture
- OpenGL/OSG example code

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

115

Multimedia Hardware

- Sound Cards
 - DMA-buffered DAC/ADC
 - Filters and balanced analog I/O
 - Serial digital I/O
 - Multi-format
 - MIDI driver, I/O, buffer, clock
 - Synthesizer
 - Wave-table, FM, GeneralMIDI
 - DSP for effects
 - Rate conversion, spatialization, MP3

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

116

MEDIA ARTS & TECHNOLOGY PROGRAM

Networks for MM



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

117

Media Networks and Protocols

- Local-Area Networks for Media
 - USB
 - FireWire
- Wide-Area Networks for Media
 - All the pretty protocols
 - Networking issues
 - Distributed time

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

118

History

- Serial ports and devices
 - Low speed
 - RS232, RS422
 - Many HW interfaces
- Parallel devices
 - Disk, Lab, Inter-CPU busses
 - SCSI, IDE, ATA
 - GPIB (IEEE-488)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

119

I/O in the Past

- Lo-speed (medium-distance) I/O (serial)
 - ASCII terminals
 - Kbd/mouse
 - Printers
 - Modems
- Hi-speed (short-distance) I/O (parallel)
 - Disks
 - LANs
 - Scanners

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

120

I/O Connection Speeds

- Serial port: up to 115 kbits/s (.115 Mbits/s)
- Standard parallel port: 115 kBytes/s (.115 MBytes/s)
- USB 1.0: 12 Mbits/s (1.5 MBytes/s)
- IDE: 3.3-16.7 MBytes/s
- SCSI-1: 5 MBytes/s
- SCSI-2 (Fast SCSI, Fast Narrow SCSI): 10 MBytes/s
- Fast Wide SCSI (SCSI-3, Wide SCSI): 20 MBytes/s
- UltraIDE: 33 MBytes/s
- Wide Ultra SCSI (Fast Wide 20): 40 MBytes/s
- USB 2.0: 60 Mbytes/s (480 Mbits/s)
- Ultra3 SCSI: 80 MBytes/s
- FireWire (current): 400-800 Mbits/s (25 - 100 MBytes/s)
- Wide Ultra3 SCSI: 160 MBytes/s
- FC-AL Fiber Channel: 100-400 MBytes/s
- FireWire 1600 (1394b): 1600 Mbits/s (200 MBytes/s)
- FireWire 3200 (1394b): 3200 Mbits/s (400 MBytes/s)
- iSCSI: 10000 Mbits/s

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

121

LANs for Media Data

- USB
 - Low-/medium-speed
 - Input devices
 - Low-speed peripherals
- FireWire
 - High-speed
 - Cameras
 - Disks

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

122

Universal Serial Bus (Intel et al.)

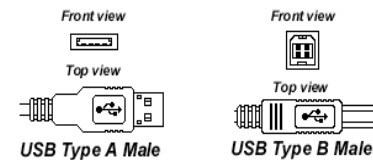
- 1.5 or 12 Mb/sec (or faster, up to 480 Mb/sec)
- 4-wire plugs (2 signal, 2 power)
- +5 V power, 500 mA per port
- Up to 127 devices, star topology with multiple repeater/hubs
- Up to 5 m cable between devices
- Auto Discovery
- Three types of transfer: block, continuous, event

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

123

USB Interconnect



Pin	Label	Function
1	+5 Volts	Power
2	Data -	Signal -
3	Data +	Signal +
4	Ground	Ground

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

124

Hot-plug (and un-plug)

- No termination resistors (as in SCSI)
- No fixed device ID (as in SCSI)
- Each device introduces itself to the LAN upon connection (using some control protocol = auto-discovery)
- Bus reset causes universal device re-introduction and ID re-assignment

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

125

USB Applications

- Keyboards/mice
- Gaming controllers, sensor interfaces
- Low-speed I/O (floppies)
- Printers
- MIDI interfaces
- Stereo DA/AD audio converters
- Stereo speakers, microphones
- Hi-speed: disks, n-channel audio

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

126

FireWire (Apple)

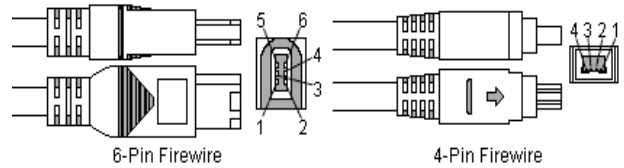
- = IEEE 1394 = Sony i.LINK
- 400/800/1600 Mb/sec
- 6-wire plugs, (2 signal, 2 control, 2 power)
- Up to 15 W per device
- Up to 64 devices, multi-master
- Up to 16 consecutive cable hops of 4.5 m

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

127

FireWire Interconnect

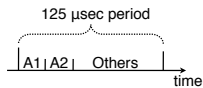


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

128

FireWire Communication



- Block Isochronous and Asynchronous transfer
 - 125 µsec basic block rate
 - Up to 64 “channels”
 - Isochronous senders get time at the start of each block
- Function Control Protocol (FCP)
- Connection Management Protocol (CMP)
- 64-bit memory-mapped addressing
 - BusID (10) NodeID (6) Address (48)
- Backplane and wired environments

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

129

FireWire Applications

- Hard disks
- Scanners, printers
- Digital video in full size, full motion
 - 720 by 480 pixels at 30 frames per second
- HDTV
- N-channel audio (up to 168 channels with 3 RME FireFace 800s)
- Yamaha mLAN digital audio
- LANs
- Shared memory back-plane (clusters)

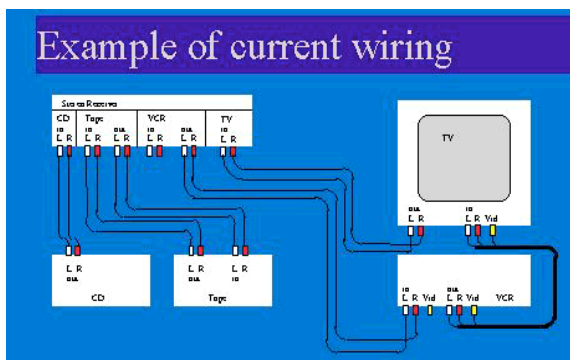
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

130

FireWire Example: Before

Example of current wiring



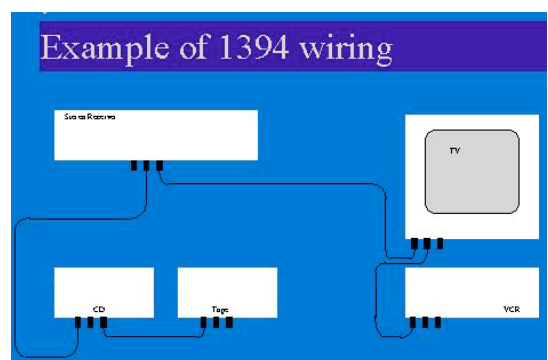
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

131

FireWire Example: After

Example of 1394 wiring



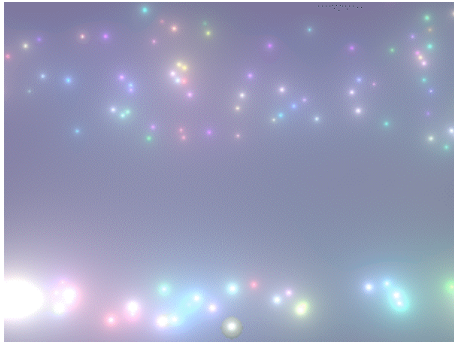
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

132

MEDIA ARTS & TECHNOLOGY PROGRAM

IP Networking

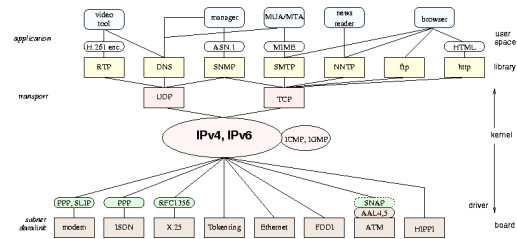


MAT 200C

133

Networking Review

- OSI model: 7 Layers
- Hardware data links at bottom
- Transport & application-layers at top



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

134

Multimedia Networks over Ethernet/Internet

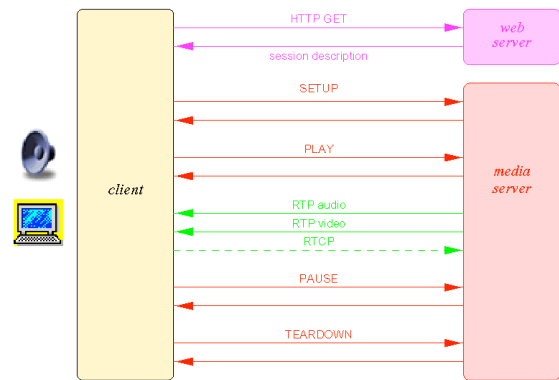
- First large-scale experiments: MultiG, MAGIC
- RTP: Real-Time Transport Protocol (IETF RFC 1889)
- RSVP: Resource ReSerVation Protocol (RFC 2205, 2379, etc.)
- RTSP: Real Time Streaming Protocol (RFC 2326) (RealAudio) (HTTP as RPC for Inet)
- Multicast (RFC 1301)
- HyTime

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

135

RTSP Activity



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

136

IP-Multicast Networks

- MBONE Backbone
 - IETF "audiocast" experiments
- Member, producer, consumer
- Transport state:
 - Heartbeat
 - Retention
 - Window
- Two control objects
 - Network service access point
 - Transport service access point
- IP Multicast Initiative (IPMI)

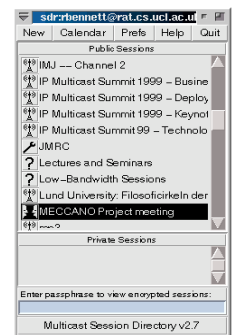
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

137

MBONE Tools (Linux)

- mrouted -- multicast routing daemon
- sd -- Session Directory
- vat -- Visual Audio Tool
- vic -- Video Conference Tool
- wb -- White Board Tool
- nv -- Network Video Tool
- isc -- Integrated Session Controller
- NeVoT -- Network Voice Tool
- Steve Deering's Multicast API (<http://www.kohala.com>)



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

138

MBONE Applications

- Web-casting of IETF meetings
- Web-conferencing
- Distance education
- MP3 streaming
- See live.com

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

139

MBONE Applications

Compare
with
iChatAV

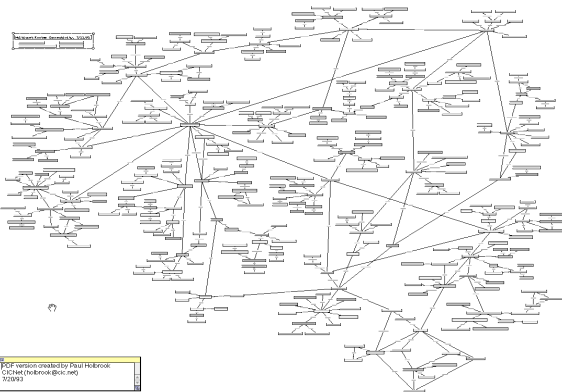


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

140

MBONE Map (1993) World

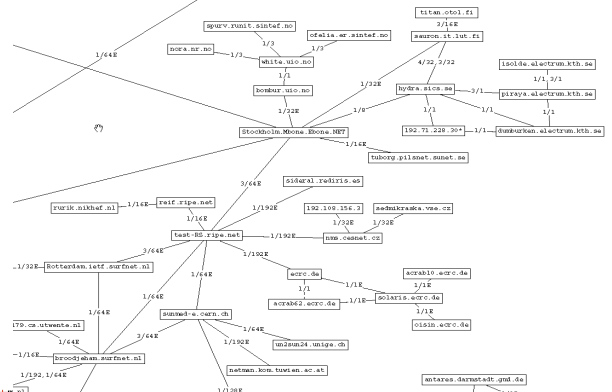


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

141

Zoomed-in (Northern Europe)



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

142

Wireless Networks

- Variables:
 - Range, rate, protocol
- Wide-area: satellite networks, packet-ham
- Metropolitan-area: cell phone, Ricochet
- Domestic/site-area: 801.11*
 - 11b: 11 Mb/s, 11g: 54 Mb/s, 11n: 540 Mb/s
- Short-range (personal-area)
 - BlueTooth, 1 Mb/s

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

143

Email and MIME

- RFC 822 Email (Aug, 1982)
- MIM-Extensions to RFC 822
 - (1) textual message bodies (and header information, addresses) in character sets other than US-ASCII,
 - (2) extensible set of different formats for non-textual messages,
 - (3) multi-part message bodies
- Defaults: image (JFIF), audio (8m8k), video (MPEG), and application (octet-stream, PostScript) types
- Composite, mixed, multi-part messages

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

144

Standard MIME Types

- **audio** – 32kadtcm basic, vnd.qcelp (more)
- **image** – cgm, g3fax, gif, ief, jpeg, naplps, png, tiff, vnd.dwg, vnd.dxf, vnd.fpx, vnd.net-fpx, vnd.svf, vnd.xiff
- **video** – mpeg, quicktime, vnd.motorola.video, vnd.motorola.videop, vnd.vivo
- **multipart** – alternative, appledouble, byteranges, digest, encrypted, form-data, header-set, mixed, parallel, related, report, signed, voice-message

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

145

Other Issues

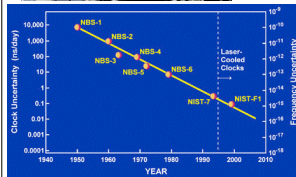
- QoS over IP
 - OS and network issues
- Voice-over Internet
- Teleconferencing/video over Internet
- XML Media Types
- Web References
- ...and many more

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

146

Time & Clocks



MAT 200C

147

Distributed Clocks

- Speed of light – finite
- Network latency – asymmetrical
- Exact distributed clocks – impossible
- There are several partial solutions
- Clock signals (UTC) are distributed via radio, satellites, telephone, and other media); receivers can be attached to LANs

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

148

Absolute Clock Distribution

Reference Clock Drivers



From top:

- Austron 2100A GPS Receiver with LORAN-C assist
- Austron 2000 LORAN-C Receiver
- Spectracom 8170 WWVB Receiver
- Hewlett Packard 5061A Cesium Beam Standard

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

149

Absolute (Calendar) Time

- Universal Coordinated Time (UTC)
- International Atomic Time (TAI)
- (These differ by an inserted leap second every 18 months.)
- Solutions:
 - Network Time Protocol (NTP)
 - Digital Time Synchronization Protocol (DTSS)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C


150

- Determine the time offset of the server clock relative to the client clock
 - Send a request; receive a time-stamp
 - Guess at latency
 - Assume symmetrical latency
 - Assume zero latency jitter
- Large, hierarchical system (230 primary and 100,000 secondary servers in 1998)
- NTP is included in modern OSs
- Some kernels also support accurate clock patches (Precision Timekeeping Kernels)

[illegible]

Review

- Networks and protocols
- Serial and parallel busses
- Connection speeds and bandwidths
- USB & FireWire
- IP and protocols
- Email and MIME
- Distributed time
- Lots of external references
- Project potential

 **MAT 200C**

154

Application
Architecture




MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

155

Application Architecture

- SW Architecture and Design Patterns
- Applications and Architecture
- Application Frameworks
- GUIs and Design Patterns
- Architecture of Multimedia Applications

 MINISTRY OF EDUCATION AND HIGHER EDUCATION
STATE OF PALESTINE

MAT 200C

156

SW Architecture and Design Patterns

- Software Architecture
 - Components, Libraries, and Frameworks
- Application Models
- Design Patterns
 - Repeated configurations of objects
 - Structural archetypes
 - Observer
 - Visitor
 - Aspect adaptor
 - Model/view/controller

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

157

Applications and Architecture

- Input/Processing/Output Model
- Screen/Query/Report Model
- Model/View/Controller Model
- 2-/3-Tier Architecture
- EJB Architecture
- Web Services

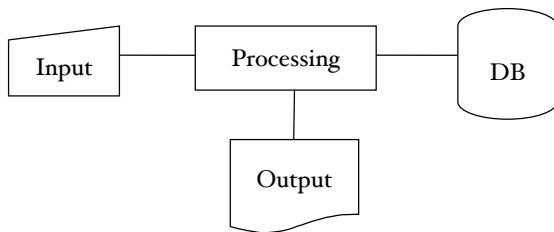
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

158

Application Architecture

- The old way



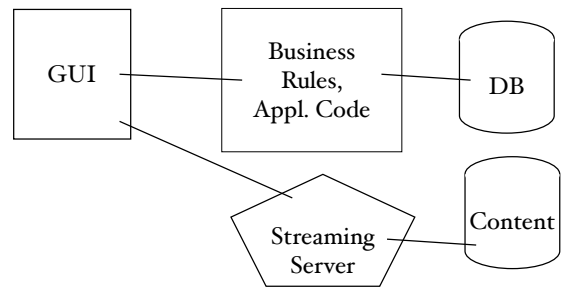
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

159

1990s Architecture

- 2- or 3-Tiered Models

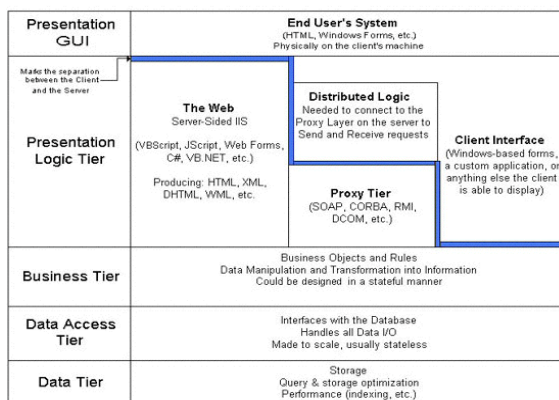


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

160

Modern N-Tiered Model



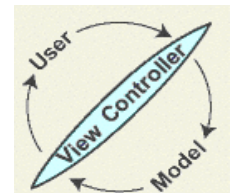
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

161

Application Frameworks

- Depend on Architectural Model
- E.g.,
 - Screen Painters
 - Report Generators
- MVC
- Widget Sets



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

162

GUIs and Design Patterns

- Model/View/Controller (see below)
- Application object and Application Model
- What's a widget?
- GUI Painters (see below)
 - Widget layout
 - Visual properties
 - Hook-up to model objects (or application aspects)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

163

Architecture of MM Applications

- Simple Media Player/recorder
- Authoring Tool
- MM-enhanced Content Delivery (course)
- MM Interaction (game)
- Immersive UI (VE)
- Others...

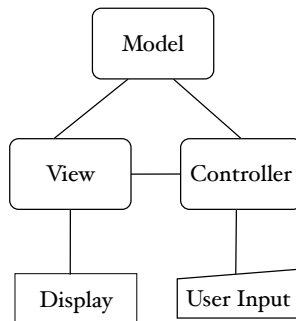
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

164

Traditional MVC (1980)

- The Model / View / Controller Design Pattern
 - Model represents domain-specific objects or simulation
 - View queries model for state to render on display
 - Controller receives events and sends messages to Model and (opt.) View
 - Model has some change broadcast mechanism (dependency, observer) so Views get updated transparently
 - Views and Controllers can be designed separately from one another and from model (reuse)
 - View/Controller classes can be parameterized (pluggable MVC)



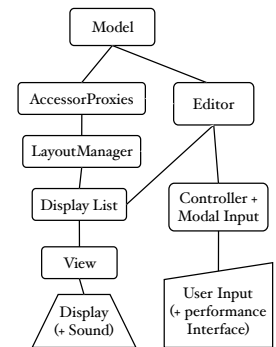
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

165

"Navigator" MVC (1990)

- Extension to Pluggable MVC
 - View renders a display list, which a LayoutManager creates using model access proxies (protocol converters, value holders)
 - Editor object for control mapping, interaction state, and proxy mgmt
 - Controller supports flexible "performance interfaces" or "vehicles"
 - Model may be remote (always accessed via proxies)
 - LayoutManagers, (multiple) Views, and (multimodal) Controllers are pluggable and reusable



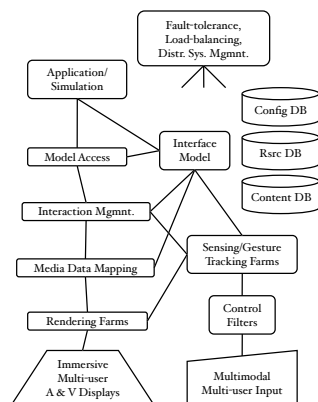
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

166

DSCP MVC (2003)

- "Back-end" application models are CNSI/MAT science/numerical/simulation apps (that need to be "wrapped")
- Presentation/interaction via Sphere, DMA, wide-area streaming
- Multimodal multi-user control and sensing (CV, feature-extraction/analysis farms)
- Interface model = output data mappings (like LayoutManagers for spatial interfaces) + sensing/tracking policies (input states)
- Framework supports wide-area distribution or replication of any or all components
- Infrastructure managed by distributed processing tools (app. start-up/shut-down)
- Based on design patterns and description languages for applications and (spatial, media-rich) interfaces
- Uses OO databases for configurations, resources (mgmt. system), and media content (renderers)



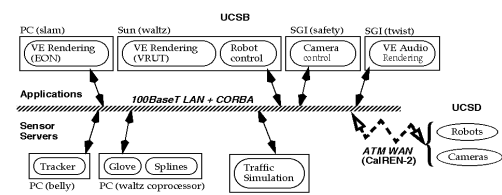
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

167

ATON MVC (2001)

- Sensing and control of remote robots and interactive simulation
- Rendering into multi-user immersive virtual environment
- Standard interaction framework and protocol for multimodal input via trackers, cameras, etc.
- Heterogeneous system including traffic simulation, gesture mapping, multi-user multimodal rendering, both wireless and wide-area networks, performance and fault monitoring, controlled start-up and shut-down



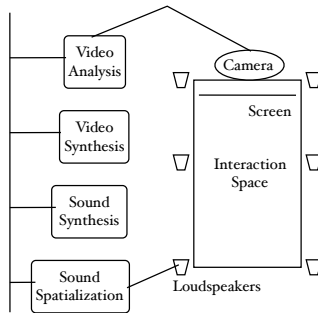
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

168

SeSpSp MVC (2002)

- DSCP Application: sensing, computation, (multiple) presentation
- Camera-based multi-user sensing (aware space)
- Computer vision SW for object model of grouping among attendees
- Distributed real-time network protocols for synchronized multi-camera projection and 6-channel surround sound

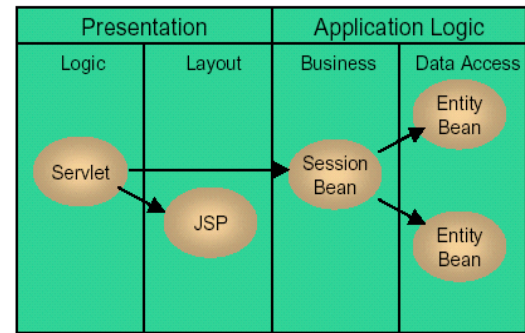


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

169

J2EE “MVC”



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

170

GUIDEs (Screen Painters)

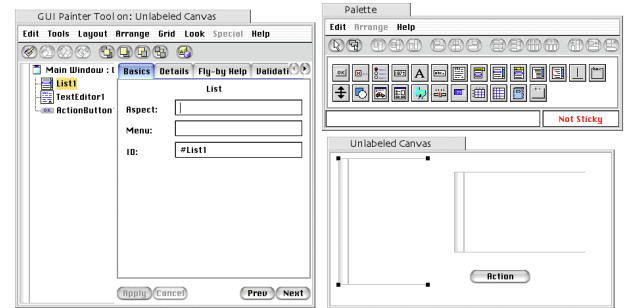
- Interface Builder
- VisualStudio
- JUCE Jucer
- Qt QtDesigner
- WxWidgets WxDesigner
- Mostly all the same (like IDEs)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

171

VisualWorks GUI Painter List Properties

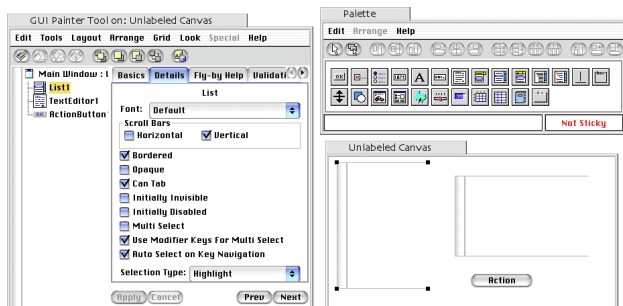


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

172

List Details

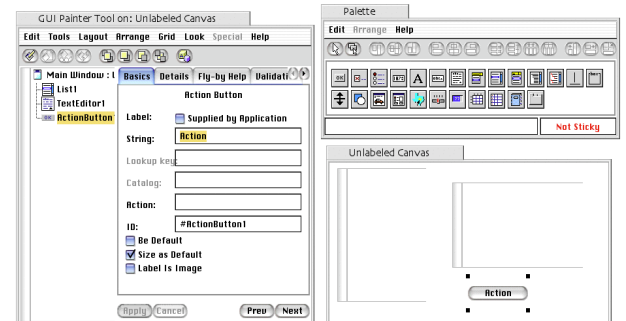


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

173

Button Properties

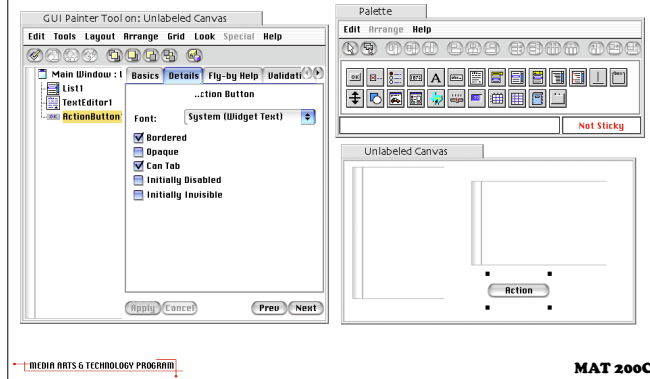


MEDIA ARTS & TECHNOLOGY PROGRAM

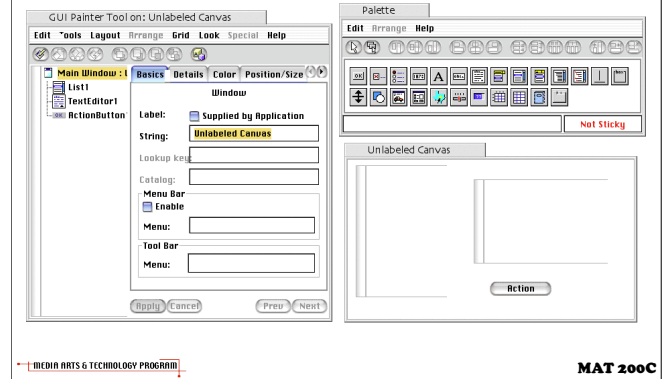
MAT 200C

174

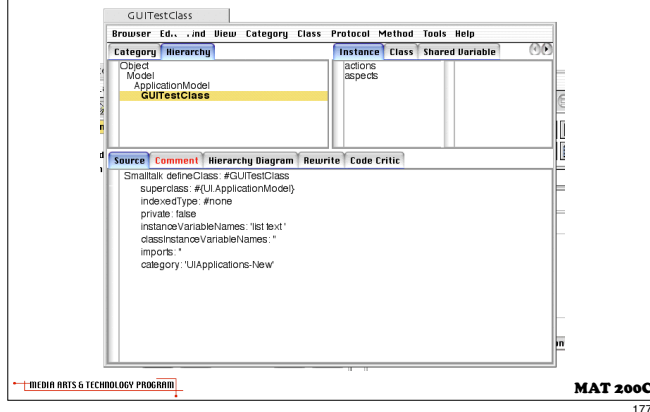
Button Details



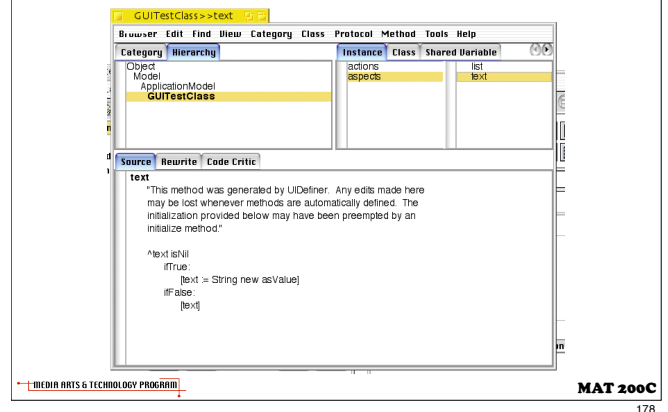
Canvas Properties



Generated Class Definition



Generated Accessor Method



Architecture of Distributed Applications

- Client/server architecture
- N-tier applications
- Web-based delivery
- Distribution via Proxies
 - CORBA, RMI, etc.
- Remote Databases
- Thin Clients

Web-based Applications

- CGI scripts
- Database back-ends
- Page generation
- Session data management
- Security and authorization
- Interface to back-end logic/data

Web Page Generation

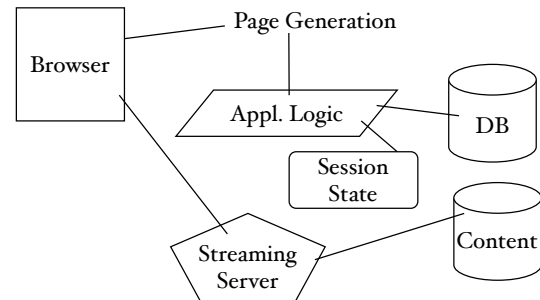
- A browser sends a request to a server; the server may:
 - Deliver the contents of a static file
 - Use a page template that includes keys to data drawn from a database
 - Run a script or program to populate a template
 - Run a script or program to generate the HTML altogether
- There is a variety of different architectures to achieve this (template engines, servlets, ...)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

181

Web Appl. Delivery

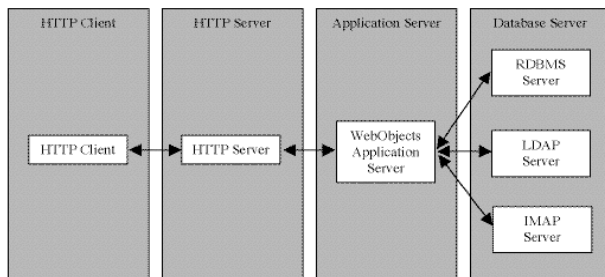


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

182

Web Application Delivery

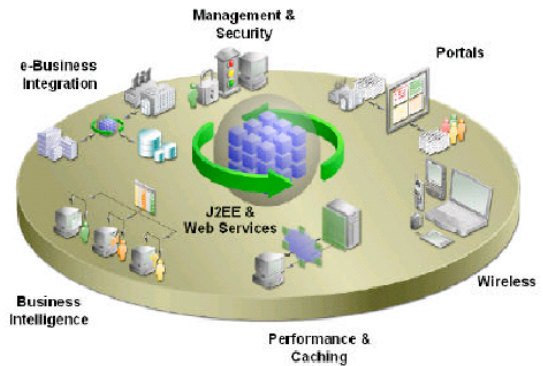


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

183

Oracle Appl. Server



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

184

Appl. Server Evolution (Sun)

Application Server Evolution

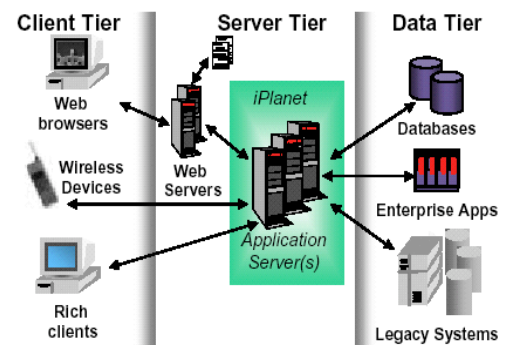
Web Server	CGI APIs	JavaScript™/ Visual Basic	Application Servers
Static Text and Images	Dynamic Pages with Database Content	Business Process Support for Small Communities	Business Critical Applications for Extended Enterprises
Limiting	CGI – Slow APIs – Difficult	Difficult to Scale	Reliable, Scalable, and Fast

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

185

Modern 3-Tiered Architecture

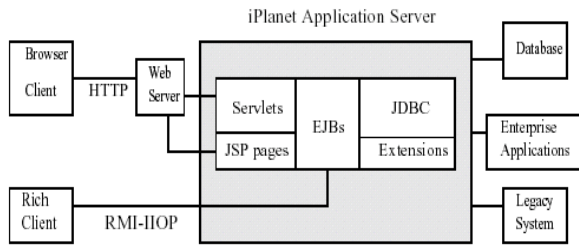


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

186

iPlanet Appl. Server



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

187

Other/Future Architectures

- WAP and very thin clients
- P2P networks
- DSCP-like Architectures
- Rabiner/Moore Graph (WAN-based + thin clients)
- Immersive Uis (as the UI rather than as an application)
- Invisible Uis (augmented reality)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

188

Review

- SW Architecture and Design Patterns
- Applications and Architecture
- Application Frameworks
- GUIs and Design Patterns
- Architecture of Multimedia Applications

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

189

Summary: Topic 1

- Some topics from ECE
 - Computer Architecture
 - Processors and instruction sets
 - System architecture
 - Extensions for Multimedia
 - High-performance computing
 - Networks for media
 - Software architecture

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

190

What's Next?

- Information theory
 - Basics and definitions
 - Applications to multimedia
 - Encoding
 - Compression
 - encryption
 - Data-hiding
 - Error-handling
- Media Signals and Symbols

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

191

What's Next? (Topic 2 Readings)

- Information theory and its applications
- Media data, signals, and symbols
 - Time and dimension in media data
 - Kinds of signals and symbols
 - Data quantization and conversion
 - Media data: events and streams
- Representations, storage formats, and interchange formats
- Protocols

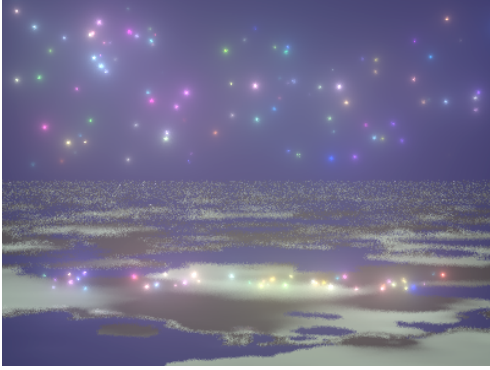
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

192

MEDIA ARTS & TECHNOLOGY PROGRAM

End of Topic 1



MEDIA ARTS & TECHNOLOGY PROGRAM

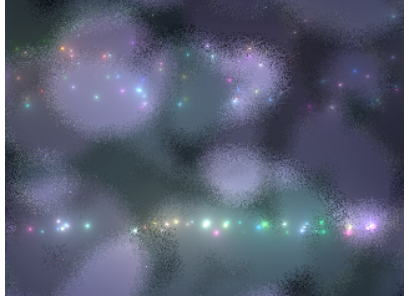
MAT 200C

193

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C: Survey of Media Technology

Topic 2: Media Data, Signals, and Symbols



Stephen T. Pope
MAT/UC Santa Barbara
stp@mat.ucsb.edu
Winter Quarter, 2008

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

194

Topic 2: Media Data, Signals, and Symbols

- Part 1: Information Theory
- Part 2: Definitions, theory
 - Signals, symbols, transducers
 - Types of signals
 - Time, events and clocks
- Part 3: Practical issues
 - Time in Data
 - Representations and Models

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

195

Part 1 Readings

- Wikipedia Entries for *Information Theory*, *Data Compression*, *Codec*, *Audio Compression*, *Image Compression*, *Video Compression*, *Comparison of Audio Codecs*, and *Comparison of Video Codecs and Container Formats*.
- W. Weaver. "Recent Contributions to the Mathematical Theory of Communication."

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

196

Part 2 Readings

- G. Garnett. "Music, Signals, and Representations" in *Representations of Musical Signals*
- R. Dannenberg. "Music Representation Issues, Techniques, and Systems" *CMJ* 17:3
- G. Wiggins et al. "A Framework for the Evaluation of Music Representation Systems" *CMJ* 17:3

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

197

Extra References

- Communication theory
- Perception
- Representations and formats
- Models of sound and music
- Computer graphics representations and languages

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

198

Information Theory

- History
- Topics: signals, symbols and information
- Aspects of signals (later)
- Units of information
- Coding of information
 - Error detection and correction
 - Encryption, compression, embedding

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

199

Aspects of Information

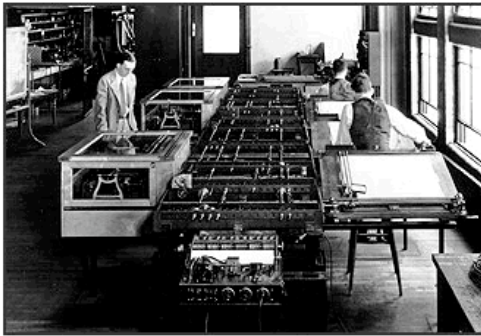
- Technical: accuracy of symbol transmission
- Semantic: symbol/meaning map
- Effectiveness: of received meaning

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

200

Differential Analyser



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

201

History of Information Theory

- Claude Shannon
 - 1938: "Application of Symbolic Logic to Relay Circuits" (Vannevar Bush and the "Differential Analyser" – Boolean algebra)
 - "It just happened that no one else was familiar with both fields at the same time."
 - 1948 "A Mathematical Theory of Communication" Bell System Technical Journal
 - Online at <http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html>

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

202

Information Theory

- The crux of information theory is the realization that the information content of a message stream is directly connected to the probability of appearance of each possible message, and that we can take advantage, on average, of any non-uniformity of these probabilities.
- (From <http://ece.wpi.edu/infoeng/textbook/main.html>)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

203

Information System

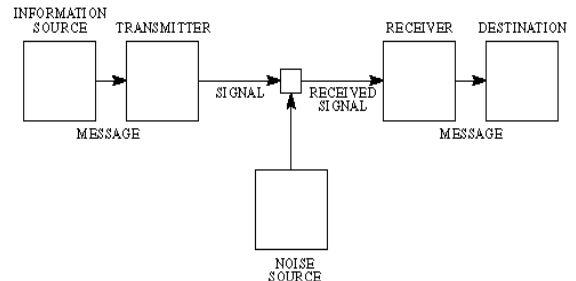


Fig. 1—Schematic diagram of a general communication system.

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

204

Error Correction

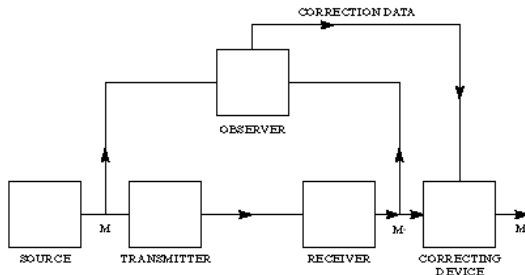


Fig. 8—Schematic diagram of a correction system.

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

205

Signals (first pass)

- Fidelity of transmission
 - Real vs. reproduced signals
- The impact of transducers
 - Impact on perception and on information content
- Mapping signals onto information
- “Entropy” in signal sources
 - Information density

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

206

Information

- Units: bits
- Data rate vs. information rate
 - Coding and redundancy
- Tangent: Ontology
 - (1) *Die Welt ist alles, was der Fall ist.*
 - (1.1) *Die Welt ist die Gesamtheit der Tatsachen, nicht der Dinge.*
 - Ludwig Wittgenstein, Tractatus Logico-Philosophicus

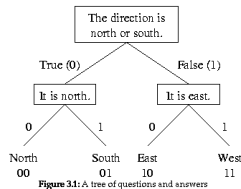


Figure 3.1: A tree of questions and answers

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

207

Information Translation and Coding

- Codings
 - Thought as Speech
 - Speech as ASCII
- Representations: should be unique, standardized, technology-appropriate, lossless
- These are language-design issues (lots to come later)
- Design of codes and alphabets

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

208

Coding Examples

- Example: Integer-as-bits
 - 1s-complement little-endian --
 $749_{10} = 1011101101_2 = 2ED_{16} = 1355_8$
 - 2s-complement -- $749 = (1)1011101101$
 - BCD -- $749 = 0111\ 0100\ 1001$
 - 32-bit floating-point -- $749 = 10000001000011101010000000000000$
 - And that's just a small number!
- Finite- vs. infinite-precision numbers
- Scale and resolution (sci. notation)
- Transcendental data (π , e)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

209

Standard Codings

- Text
 - ASCII, EBCDIC, UniCode, ...
- Numbers
 - IEEE floating-point
 - Extended numbers
- Sound
- Images

00 NUL	10 DLE	20 SP	30 0	40 @	50 P	60 `	70 p
01 SOH	11 DC1	21 !	31 1	41 A	51 Q	61 a	71 q
02 STX	12 DC2	22 "	32 2	42 B	52 R	62 b	72 r
03 ETX	13 DC3	23 #	33 3	43 C	53 S	63 c	73 s
04 EOT	14 DC4	24 \$	34 4	44 D	54 T	64 d	74 t
05 ENQ	15 NAK	25 %	35 5	45 E	55 U	65 e	75 u
06 ACK	16 SYN	26 &	36 6	46 F	56 V	66 f	76 v
07 BEL	17 ETB	27 '	37 7	47 G	57 W	67 g	77 w
08 BS	18 CAN	28 (38 8	48 H	58 X	68 h	78 x
09 HT	19 EM	29)	39 9	49 I	59 Y	69 i	79 y
0A LF	1A SUB	2A *	3A :	4A J	5A Z	6A j	7A z
0B VT	1B ESC	2B +	3B ;	4B K	5B [6B k	7B {
0C FF	1C FS	2C ,	3C <	4C L	5C \	6C l	7C }
0D CR	1D GS	2D -	3D =	4D M	5D]	6D m	7D ~
0E SO	1E RS	2E .	3E >	4E N	5E ^	6E n	7E `
0F SI	1F US	2F /	3F ?	4F O	5F _	6F o	7F DEL

NUL	Null	FF	Form feed	CAN	Cancel
SOH	Start of heading	CR	Carriage return	EM	End of medium
STX	Start of text	SO	Shift out	SUB	Substitute
ETX	End of text	SI	Shift in	ESC	Escape
EOT	End of transmission	DLE	Data link escape	FS	File separator
ENQ	Enquiry	DC1	Device control 1	GS	Group separator
ACK	Acknowledge	DC2	Device control 2	RS	Record separator
BEL	Bell	DC3	Device control 3	US	Unit separator
BS	Backspace	DC4	Device control 4	SP	Space
HT	Horizontal tab	NAK	Negative acknowledge	DEL	Delete
LF	Line feed	SYN	Synchronous idle		
VT	Vertical tab	ETB	End of transmission block		

ASCII Character Set

MAT 200C

210

- Apply insights into the characteristics of different kinds of information, codings and channels
- Design of efficient representations and encodings
- Codings that support error detection, correction
- Compression
- Encryption
- Information embedding (error handling, data hiding, watermarking)

- How do you encode a signal to ease error handling (in transmission/storage)?
- What do you need to take into account?
- 1: Characterize the system's error sources
- 2: Design an encoding that enables you to detect certain classes of errors
- 3: Add redundancy/repetition as necessary to recover from detected errors of the given kinds

- Error characterization: what kinds of errors are expected in a system? (describe via theory or measurement)
 - (e.g.,) LAN: short bursty drop-outs (collisions)
 - DVD-ROM: large blocks across many tracks (scratches)
- Error detection: how can you be very sure whether you got a message right ?
 - LAN: use hand-shake with seq # and check-sum
 - DVD-ROM: aggressive statistical checksums on blocks
- Error recovery: how do you get it then?
 - LAN: re-send packet, hand-shake again
 - DVD-ROM: look somewhere else on the disc
 - (distributed redundant data)

- Formats and protocol design
 - Analysis of expected error allows the design of minimal effective redundancy
 - Trade-off between data rate and redundancy
 - impact and error recovery
 - Coding vs. compression artifacts
- Pretty well-solved
 - (for existing media)
 - Parity-, polarity-based
 - Polynomial-based, etc.

Data Bits								ECC Bits		
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₀	E ₁	E ₂
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₃	E ₄	E ₅
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₆	E ₇	E ₈
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₉	E ₁₀	E ₁₁
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₁₂	E ₁₃	E ₁₄
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₁₅	E ₁₆	E ₁₇
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₁₈	E ₁₉	E ₂₀
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₂₁	E ₂₂	E ₂₃
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₂₄	E ₂₅	E ₂₆
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₂₇	E ₂₈	E ₂₉
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₃₀	E ₃₁	E ₃₂
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₃₃	E ₃₄	E ₃₅
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₃₆	E ₃₇	E ₃₈
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₃₉	E ₄₀	E ₄₁
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₄₂	E ₄₃	E ₄₄
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₄₅	E ₄₆	E ₄₇
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₄₈	E ₄₉	E ₅₀
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₅₁	E ₅₂	E ₅₃
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₅₄	E ₅₅	E ₅₆
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₅₇	E ₅₈	E ₅₉
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₆₀	E ₆₁	E ₆₂
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₆₃	E ₆₄	E ₆₅
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₆₆	E ₆₇	E ₆₈
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₆₉	E ₇₀	E ₇₁
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₇₂	E ₇₃	E ₇₄
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₇₅	E ₇₆	E ₇₇
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₇₈	E ₇₉	E ₈₀
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₈₁	E ₈₂	E ₈₃
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₈₄	E ₈₅	E ₈₆
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₈₇	E ₈₈	E ₈₉
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	E ₉₀	E ₉₁	E ₉₂

Transmitted	Transmitted	Transmitted	Received	Received	Was there
Character	Information	Parity	Information	Parity	an Error
H	1001000	0	1000000	0	Yes
e	1100101	0	1100101	0	No
l	1101100	0	1101100	0	No
p	1110000	1	1110000	1	No

Table 3.1: Even Parity Example

- Parity-checking (parity = enforce even or odd # of 1's in binary words by added p-bits)
- Checksums (transmit sum of words in a block)
- Hierarchical or cyclical coding (data as underspecified polynomial of a certain family)
- Many others

Encryption

- Encryption: make a signal look like noise to most observers, but easily recoverable to some
- Simple: substitution tables

ABCDEFGHIJKLMNOPQRSTUVWXYZ
actqgwrzdevfbhinsymujxplok

Using this key, the plaintext:

THE SECURITY OF THE RSA ENCODING SCHEME RELIES ON THE
FACT THAT NOBODY HAS BEEN ABLE TO DISCOVER HOW TO TAKE
CUBE ROOTS MOD N WITHOUT KNOWING NS FACTORS

becomes the ciphertext:

UZG MGTJYDUO IW UZG YMA GHTIQDHR MTZGBG YGFDGM IH UZG
WATU UZAU HICIQO ZAM CQGH ACFG UI QDMTIXGY ZIP UI UAVG
TJCG YIIUM BIQ H PDUZIJU VHIPDHR HM WATUIYM

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

217

Advanced Encryption

- Multi-stage variable substitution encoding
- Use number theory: modulus congruences, remainders, primes and roots
- Often based on factorization of very large numbers
- Multi-part public/private keys
- No known “perfect” method, but many that are quite complex (hard to crack in the useful lifetime of the information)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

218

Steganography

- Data-embedding, data-hiding
- Can be used for error recovery
- Example application: add a “watermark” to a signal
- Watermark must be:
 - Indistinguishable (well-hidden)
 - Incontrovertable (un-removeable)
 - Robust across encoding or numerical operations

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

219

Data-hiding Example



Before

Watermark

After

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

220

Compression

- What assumptions can you make about the information and its coding?
 - Repetitions?
 - Noise?
- Does the reconstructed version have to be numerically or “perceptually” identical?
- Examples (lossless)
 - B/W scanned graphics: run-length-encoding
 - ASCII text: Huffman codes

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

221

Run-Length Encoding

- Detect repeating pixels on each row; translate them to a pixel count and value pair
- Does very well for large areas of solid color
- Does not do well for stipple patterns or scanned images
- Used in BMP, TIFF, and PICT file formats
- Can be extended to longer patterns and 2-D runs (rectangle runs)
 - Original: AAAAAAAAAABBBBCDE
 - Encoded: -8A-3BCDE

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

222

Huffman Coding (ZIP, StuffIt, gzip, ...)

- Make a histogram of vocabulary usage
- Replace frequent input strings with n-bit codes (2n special terms in vocabulary)
- Can be very effective (e.g., almost any natural text, programs, machine code, DBMS)
- Fully reversible (lossless)
- Used in the GIF/TIFF file formats (12-bit codes = 4096 coded patterns)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

223

Sound Compression

- Old rule:
 - Effective / real-time / artifact-free : pick any 2
- Linear/exponential: μ -law (~ A-law)
 - Non-linear quantization, look-up tables
 - 4:1 -- OK for speech only
- Statistical: Huffman
 - Generally ineffective for sound (try it!)

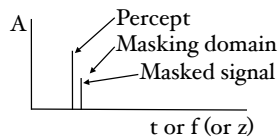
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

224

Lossy Compression Models

- LPC model (subtractive synthesis)
 - CELP, ADPCM, MPEG
 - Very effective (100:1), complex, OK for speech
- Masking model
 - MPEG, MP3
 - Effective, complex, has artifacts



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

225

Perceptual Coding for Compression

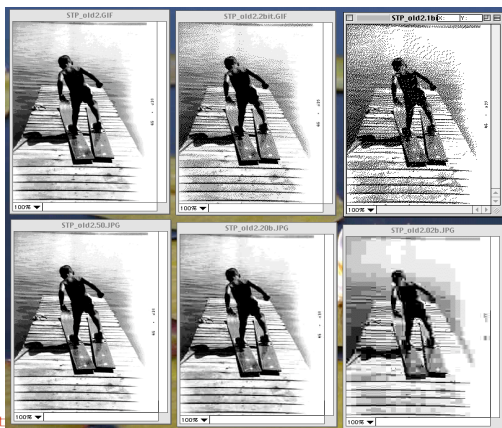
- MP3
 - Represent signal in spectral domain (DCT, like FFT)
 - Use freq/ampl-dependent masking threshold model to prune spectral data
 - Use linear algebra to compress resulting sparse matrices
- JPEG
 - Same except uses 2-D DCT

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

226

Dithering vs. Compression

File sizes
(kB):

GIF 1: 36
GIF 1.2: 14
GIF 2: 18
GIF 2.2: 10
GIF 3: 10

JPEG 1: 18
JPEG 2: 6.6
JPEG 3: 3.3

MEDIA ARTS & TECHNOLOGY PROGRAM

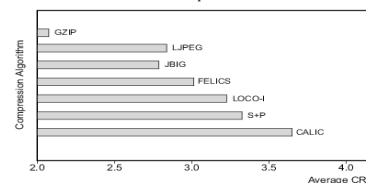
MAT 200C

227

Image Compression

- Lossless statistical methods (Huffman, LZW)
- Lossless image-specific (RLE)
- Lossy image-specific (JPEG/MPEG)
- Really fancy (wavelets, IFS)

Figure 2.3: Comparison among various lossless image compression techniques.



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

228

Other Compression Techniques

- Fancy:
 - Wavelets
 - Chaos-based
- Color table compression and color dithering
- Color-to-grayscale conversion
- Custom techniques for special domains
 - Faxes
 - Video

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

229

Review

- Information theory
 - Theory and history
 - Communication systems and their elements
 - Applications to data and media content
 - Coding
 - Compression
 - Encryption

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

230

Exercises, Demos

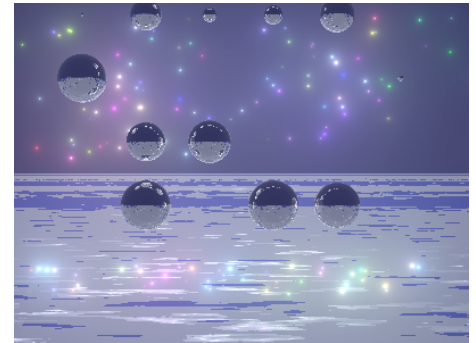
- Information theory
 - Coding, transmission systems
 - Error detection and recovery
- Applications
 - Content encoding
 - Data compression, encryption
 - Statistical and perceptual coding

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

231

Topic 2 Part 2



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

232

Part 2

- Loads of definitions
- Time and dimension in media data signals
- Perception and signal semantics
- Data quantization and conversion
- Media data: events and streams

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

233

Communication and Information

- Communication: transfer or exchange of information via symbols or signals
- Information: fact or impression, data, semantics of a signal or symbol
- Relation to data and knowledge?

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

234

Signal and Symbol

- Signal: detectable physical quantity by which messages or information can be conveyed; object that can convey meaning; has value(s); changes over time
- Symbol, object or act that represents something else by reason of relationship, association, convention, or accidental resemblance; exists or not

** | BEHOLD ARTS & TECHNOLOGY PRESENTS | **
MAT 200C
235

Transducers

- Generate a direct analog between signals in different media (little time-domain effect)
- Translate signals between media (e.g., camera, TV, microphone, loudspeaker, phono pick-up, sensors)
- Define analog medium, units, and reference magnitude
- Are potentially (theoretically) loss-less

** | BEHOLD ARTS & TECHNOLOGY PRESENTS | **
MAT 200C
236

Transmission and Medium

- Transmission: transfer a signal between like transducers via a medium (I:I, I:n, n:m)
- Medium: means of conveyance of a signal; related to the transducers (units, reference)

** | BEHOLD ARTS & TECHNOLOGY PRESENTS | **
MAT 200C
237

Perception

- A combination of several physiological and psychological processes that map signals onto symbols
 - A feature of systems that exhibit consciousness or awareness (cognition)
 - One temporal flow of percepts may be distinguished into several streams

** | BEHOLD ARTS & TECHNOLOGY PRESENTS | **
MAT 200C
238

Information Theory

- Definition of the units of information and meaning (syntax and semantics of data and information)
- Metrics of the density of information
- Relationships between information and media
- Has engineering and philosophy aspects

** | BEHOLD ARTS & TECHNOLOGY PRESENTS | **
MAT 200C
239

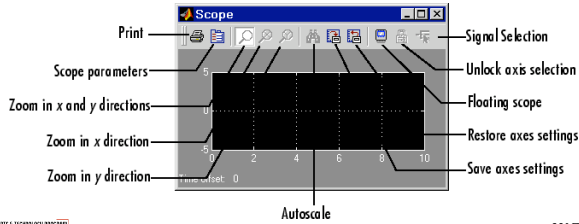
Communication Theory

- Study relationships between communication, information, signals, and symbols
- Provide details of the relationships between representative, representand, and channel/medium
- Has engineering, psychology, and philosophy aspects

** | BEHOLD ARTS & TECHNOLOGY PRESENTS | **
MAT 200C
240

Signals

- Relationship to real-world physical phenomena
- Representation as 1- or n-dimensional functions of time
- Agents that can map signals to symbols



241

Abstract and Concrete Signals

- Some signals have obvious 1:1 relationships to measurable physical quantities (direct signals)
 - Sound pressure
 - Light level
- Other signals need to be derived from direct signals by a process of analysis or feature extraction
 - (Sound/Light) spectrum
 - Tempo

242

Analytical and Parametric Signals

- Real-world: physical medium
- Analytical: implied semantics and mathematical properties
- Parametric: assumes some underlying model, e.g., sound synthesis, lighting, color models; may be derived by analysis or property estimation

243

Transformations

- Information-preserving (loss-less, reversible): Fourier, Wavelet
- Lossy: sampling, quantization, masking-based
- Translations between “domains”
- Feature extraction
- Analysis/synthesis models

244

Symbols and Events

- Symbols represent something; we attach meaning to them
- An event is a set of symbols that are associated in time (i.e., properties that are valid over some span of time)
- Events are typically associated with derived features of physical phenomena

245

Physical vs. Perceptual

- Most direct signals of interest to us map onto perceptual quantities
- There are physical measures of the direct aspects and derived features of signals
- These generally map to perceptual aspects, but the mapping is often non-linear, indirect, or context-dependent

246

Physical vs. Perceptual: Examples

- Frequency and pitch
- Amplitude and loudness
- Brightness and intensity
- Measured and perceived distance
- Others?

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

247

Continuous and Discrete Phenomena

- Physical phenomena and signals tend to be continuous
- Symbols and events tend to be discrete
- One can generally convert back and forth via the processes of (lossy) sampling and interpolation (independent of any quantization)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

248

Clocked and Isochronous Signals

- Signals can be represented as continuous analog values or can be sampled at a fixed rate and represented/transmitted isochronously
- Isochronous signals have an implicit (out-of-band) clock rate
- Derived features can be scheduled and sent at non-uniform clock intervals
- Examples?

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

249

Isochronous vs. Clocked Signals

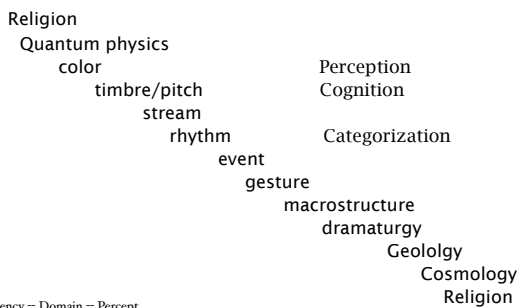
- Isochronous signals are generally simpler to represent and transmit (simple S/H, implicit time [iso-chronos]), but also less efficient (fixed sample-rate)
- Clocked signals are more complicated to represent and transmit (need representation of time and scheduler), but can be much more compact

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

250

Signals/Symbols, Time, and Perception (after Roads)



Period/Frequency – Domain – Percept

Log-sec	-24	-12	-6	-3	-2	-1	0	1	2	3	6	12	24
Log-Hz	24	12	6	3	2	1	0	-1	-2	-3	-6	-12	-24

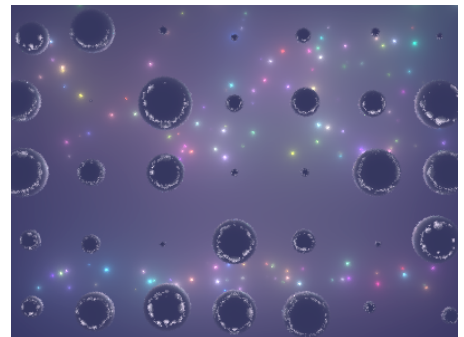
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

251

MEDIA ARTS & TECHNOLOGY PROGRAM

Practical Considerations



(video)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

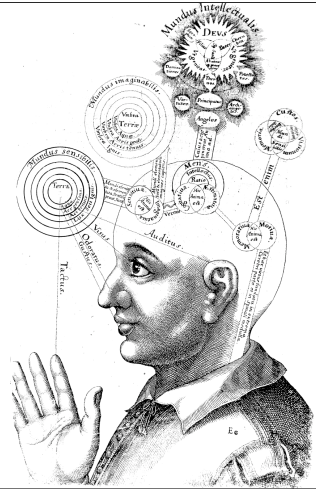
252

MEDIA ARTS & TECHNOLOGY PROGRAM **MAT 200C**

- MEDIA ARTS & TECHNOLOGY PROGRAM **MAT 200C**

MAT 200C

Sensation, Perception and Consciousness



254

Sound Signals

- Sound: air pressure changes in the range 20 Hz - 20 kHz, 0 - 120 dB (μ Bar levels)
- Microphones and loudspeakers are sound/electricity transducers (many kinds with varying properties)
- DACs can sample and quantize continuous analog electrical signals into isochronous binary data streams

© 2000 ASEE & TECHNOLOGY PRESS

MAT 2000

- MEDIA ARTS & TECHNOLOGY PROGRAM **MAT 200C**

MAT 200C

Sound Perception

- The two ears receive different (time-domain) signals
- Some processing occurs per-ear in the inner ear (frequency domain representation)
- The extracted features are then combined in the brain via several different programs
- Psychoacoustics is a large and active field

© 2000 MIT & TECHNOLOGY PROGRAMS

MAT 200C

- WILBUR WATTS & TECHNOLOGY PROGRAM MAT 200C

256

Your Ears

The diagram illustrates the anatomy of the human ear, divided into three main sections: External Ear, Middle Ear, and Inner Ear. The External Ear includes the Auricle and the External acoustic canal. The Middle Ear contains the Auditory ossicles (Hammer, Anvil, and Stirrup), the Tympanic membrane, the Oval window, and the Round window. The Inner Ear consists of the Semicircular canals, the Vestibule, the Cochlea, and the Vestibulocochlear nerve (VIII). Other structures shown include the Facial nerve (VII), the Petrous part of the temporal bone, the Bony labyrinth of the inner ear, the Cochlea, and the Vestibule. The diagram also shows the connection to the pharynx via the Auditory tube.

Fig. 17-20

EXTERNAL EAR

MIDDLE EAR

INNER EAR

Auricle

Auditory ossicles

Semicircular canals

Petrous part of temporal bone

Facial nerve (VII)

Vestibulocochlear nerve (VIII)

External acoustic canal

Tympanic membrane

Oval window

Round window

Vestibule

Auditory tube

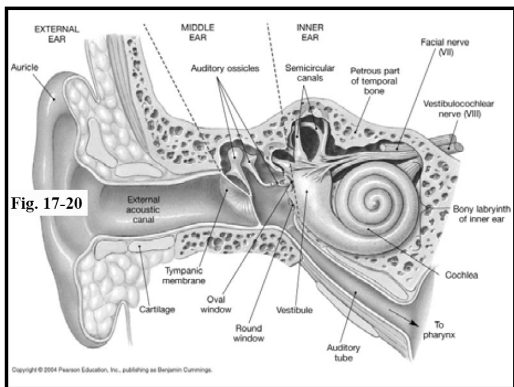
To pharynx

Cochlea

Bony labyrinth of inner ear

Copyright © 2004 Pearson Education, Inc., publishing as Benjamin Cummings.

MAT 200C


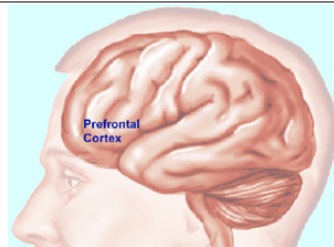
**Fig. 17-20**

Copyright © 2004 Pearson Education, Inc., publishing as Benjamin Cummings

MAT 200C

Sound Cognition

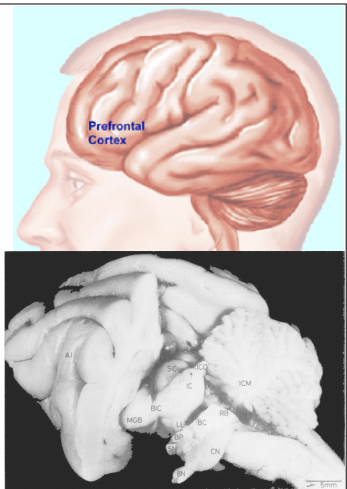
- Several centers activated by sound
- Very low-level
- Multiple high-level



The diagram shows a lateral view of the human brain with the Prefrontal Cortex highlighted in blue. The 3D rendering shows various brain regions labeled with abbreviations: AI, SCG, CG, ICM, BC, IC, MGB, LU, RC, RI, CN, and UN.

BRUNNEN & TECHNOLOGY PRODUKT

- WILSON ARTS & TECHNOLOGY PROGRAM



258

Properties of Sounds

- Amplitude
- Pitch
 - Fundamental frequency
 - Spectral centroid
 - Harmonicity
- Spectrum/Timbre
- Spatiality/"Room"
- Many more direct/derived properties

MAT 200C

MAT 200C

259

Semantics of Sound

- Speech semantics: phoneme extraction, speaker ID, vocal stress ID, etc.
- Environmental sound (effect): spectral and spatial features, root cause analysis
- Musical sound: pitch, rhythm, harmony, timbre
- Other domains?
- The brain switches between programs within a msec or so!

MAT 200C

MAT 200C

260

Sound Data (signals, symbols, and events)

- Mono sampled/quantized data stream
- Macro-level continuous properties, e.g., crescendo, panning
- Events, e.g., words, phonemes, notes, parts of notes
- Macro-events, e.g., chords, phrases
- Streams, "cocktail party effect," visual parallelism

MAT 200C

MAT 200C

261

Visual Perception

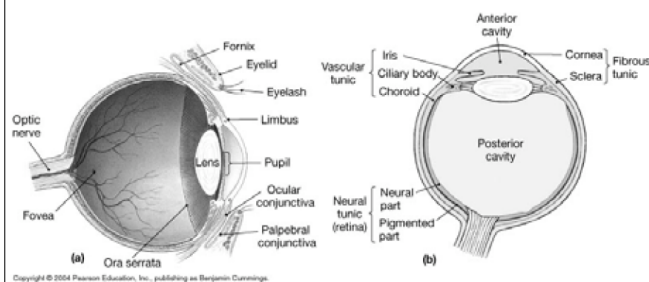
- The two eyes have independent motion and focus
- Brightness and color are perceived separately
- Correlation gives us distance cues
- Surfaces are merged into objects and objects into scenes

MAT 200C

MAT 200C

262

Eye Physiology



Copyright © 2004 Pearson Education, Inc., publishing as Benjamin Cummings.

MAT 200C

MAT 200C

263

The Retina

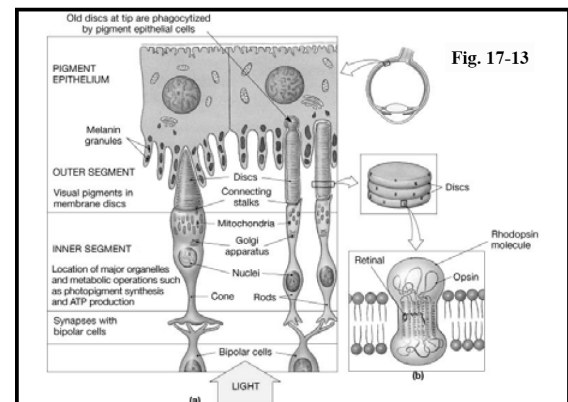


Fig. 17-13

MAT 200C

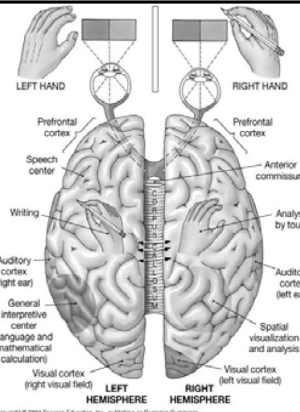
MAT 200C

264

Visual Processing in the Brain

3. From the primary visual and association areas, sensory signals enter the parietal lobe. The left hemisphere processes language in response to the vision; the right hemisphere recognizes pattern. The 2 hemispheres share responses via the corpus callosum.

Fig. 14-18



Properties of Scenes

- Global properties
- Objects and their properties
 - Position, geometry, surfaces
- Lighting
- Medium

Visual Semantics

- Object and scene ID
- Semantics of surfaces and visual frequency
- Higher-level semantics
- Context in vision

Visual Data

- $X*Y*D$ pixel map (dimension and pixel depth)
- Surfaces and objects (models of objects in a scene)
- Analog frame sequence (film)
- Pixmap sequence (video)

Time and Dimension in Media Data

- Transducers have no effect on time-domain
- Most transducers generate a 1-D signal, which may however represent a 2- or 3-D phenomenon
- Sampling quantizes time and (generally) generates an isochronous signal

Time and Transformations

- Feature extraction, analysis and signal-to-symbol mapping
 - Translate the signal out of the isochronous domain
 - Create discrete clocked events or higher-level dynamic properties (control signals) that may be isochronous at a lower rate (e.g., tempo curves, lighting fades)

Data Quantization and Conversion

- Quantization is different from sampling
- A continuous range of values is mapped onto a fixed set of possible values, e.g., 16-bit sound samples or 256 color values
- Like sampling, this is a lossy (irreversible) process (more later)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

271

Representation of Signals and Symbols

- Signal vs. Symbol
- Data vs. Knowledge
- Implied semantics, interpretation, relationships
- Hierarchy and generalization
- Feature extraction and modeling

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

272

Data/Signals/Streams

- Implicit time
- Implicit semantics
- Little structure or hierarchy
 - Isochronous single-valued stream
- Few if any data types

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

273

Knowledge/Symbols/Events

- Explicit time
- Explicit semantics?
- Structure and hierarchy
 - Containment, membership
 - Generalization
 - Control relationships, parametric models
- Potentially diverse data types

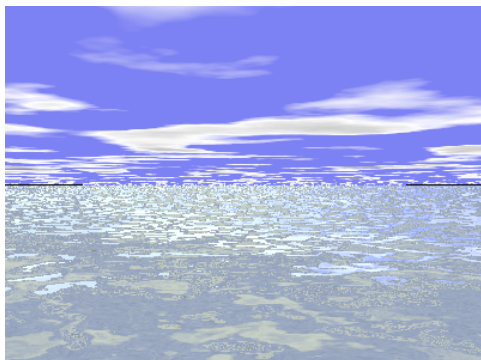
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

274

MEDIA ARTS & TECHNOLOGY PROGRAM

End of
Part 2.1



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

275

Topic 2: Media data, signals, and symbols

- Formats, languages, and protocols
 - Part 1: Definitions, theory
 - **Part 2: Practical issues**
 - Definitions
 - Time and media data
 - Sound and music models
 - Image and scene models

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

276

Media Information

- Representations, storage formats, and interchange formats
- Language issues
- Music representations
- Image and scene representations

© BEER, HOLT & THORNDYKE PUBLISHING

MAT 200C

277

What's the Goal?

- Representation
- Storage format
- Interchange format
- Not: Generational algorithm
- Not: Query/retrieval method

© BEER, HOLT & THORNDYKE PUBLISHING

MAT 200C

278

Representation

- “Abstract” description of media data, knowledge
- Of signals, symbols, data hierarchy
- Of object semantics?
- Relation to an underlying model?
- Typically human-readable
- Examples: score? name/value pairs?

© BEER, HOLT & THORNDYKE PUBLISHING

MAT 200C

279

Storage format

- In-memory or on-disk/disc, seen as a data structure, data coding
- Closely bound to a language binding and API
- Explicit time
- Rarely human-readable
- Examples: `short *sampleArray;`

© BEER, HOLT & THORNDYKE PUBLISHING

MAT 200C

280

Interchange format

- Streaming or file-based “external” I/O description, interchange protocol
- May be isochronous, implicit time
- Assumed independent of a language binding
- May go out of its way to be human-readable
- Examples: MIDI, HTML, VRML, S/P-DIF, OSC, DV, Flash, ...

© BEER, HOLT & THORNDYKE PUBLISHING

MAT 200C

281

Formal vs natural language

- Redundancy
- Ambiguity
- Grammar
- Others?

© BEER, HOLT & THORNDYKE PUBLISHING

MAT 200C

282

Elements of a “Programming” Language

- Reference and naming
- Assignment and evaluation
- Statements and expressions
- Scope and delineation
- Extension mechanism

“BETA” HDS & TECHNOLOGY PRESENTS
MAT 200C
283

A Few Language Design Issues

- Declarative vs. procedural
 - X is positive [state a fact = what]
 - $X := \max(X, \text{abs}(X))$ [state a procedure = how]
- Orthogonality and symmetry
 - Color vs. taste [different things look different]
 - Color(orange) vs. Color(berry) [like things look the same]

“BETA” HDS & TECHNOLOGY PRESENTS
MAT 200C
284

Knowledge Representation (KR)

- Part of AI (sort of)
- Goal is to express the relevant definitions and relationships from a domain in a natural and concise way in a formal language for ease of manipulation and “reasoning”
- Declarative vs. procedural KR

“BETA” HDS & TECHNOLOGY PRESENTS
MAT 200C
285

Computational Semantics

- Procedural
- Message-passing
- Functional - Application
- Logic - Unification
- Data-flow
- Others?

“BETA” HDS & TECHNOLOGY PRESENTS
MAT 200C
286

Grouping of State and Behavior

- Data vs state
- Functionality vs behavior
 - Procedure/function
 - Module/file
 - Object/method
 - Class

“BETA” HDS & TECHNOLOGY PRESENTS
MAT 200C
287

Dynamic Models Models of Dynamicity

- Stream, block, process, thread
- Synchronous (sample-clocked)
- Asynchronous (message/clocked)
 - Interrupt/event-driven
- Parallel (SIMD, MIMD) models
- Data-flow (parallel, unscheduled)

“BETA” HDS & TECHNOLOGY PRESENTS
MAT 200C
288

Time, Duration, and Events

- Calendar time
 - External reference
 - Schedule-less systems (immediate-time)
- Stream time (implicit)
 - May have memory, look-ahead
- Musical time
 - Symbolic units, non-linear map
 - Specified relatively

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

289

Time, Duration, and Events

- Logical time
 - Imprecise, relative, uncertain
 - Variable grain size, level of detail
 - “Persistence”
- “State as time”
 - Messages and state transitions
- Interval-based time
 - Only relative time
 - Only intervals are of interest

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

290

Examples

- Isochronous sample stream (with/without buffer)
- MIDI event stream, MIDI control (immediate)
- Music V SWSS score, function (logical, continuous functions)
- Data-flow “block-diagram compilers” (states)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

291

Dimensions of Media Representation

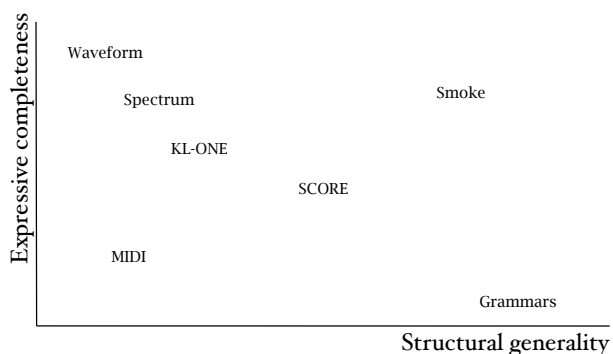
- Expressive completeness
 - Range of data semantics that can be (explicitly) represented
- Structural generality
 - Levels of structure that can be (explicitly) represented

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

292

Examples (after Wiggins et al.)



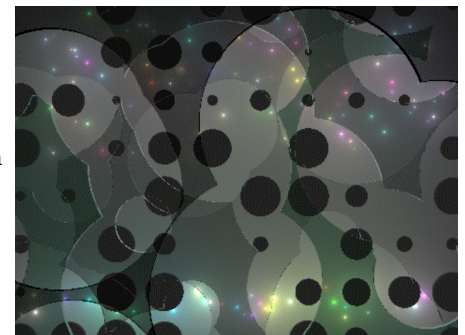
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

293

MEDIA ARTS & TECHNOLOGY PROGRAM

Music Representation



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

294

Music Representation, Notation, and Interchange

- History
 - Score, sketch, tablature
 - 1960s: Music input languages (MILs), compositional algorithms
 - 1970s: AI/KR and music representation
 - 1980s: Music theory, psychology, and MR-grammars, nets, rules, etc.
 - Abstractions used in MILs

© BEER, OWEN & HORNBY PUBLISHING
MAT 200C
295

Classes of Systems

- In the literature
 - Music input languages
 - Music programming languages
 - Music representation models
 - Music interchange protocols

© BEER, OWEN & HORNBY PUBLISHING
MAT 200C
296

Music Representation, Notation, and Interchange

- Music vs. sound vs. score
- Structure and content
 - Compositional algorithm as representation
- Performance scripts
 - Traditional score, screenplay
- Derived analysis
 - Spectrograms and reader's scores

© BEER, OWEN & HORNBY PUBLISHING
MAT 200C
297

Musical Objects

- Note event vs. stream (resource, instance)
- Function application (vibrato problem)
- What's an event or a note?
 - Note with known parameters and/or functions
 - Note as arbitrary name/value pairs
 - Note as event = simultaneous symbols at any level of granularity

© BEER, OWEN & HORNBY PUBLISHING
MAT 200C
298

Basic Models (Music Magnitudes)

- Time, tempo, beat (see above)
- Pitch, frequency
- Pitch gamut, key/scale
- Loudness, dynamic, amplitude
- Timbre, instrument, voice
- Space, position, distance, spatiality
- Others?

© BEER, OWEN & HORNBY PUBLISHING
MAT 200C
299

Time in Music

- Absolute time
- Beat and time quanta
- Tempo and tempo maps
- Functions of time, functions of tempo
- Semantics of drum-roll, glissando, and vibrato

© BEER, OWEN & HORNBY PUBLISHING
MAT 200C
300

Pitch Models

- Pitch and naming
- Pitch vs. frequency
 - Name within cycle
 - Ratio of pitch
 - Alterations and accidentals
- Pitch cycles and gamuts

MAT 200C

MAT 200C

301

Pitch Gamut

- Relationship to pitch cycle
- Named vs. computed gamut
- Selection criteria
- Repeating or not; definition of “octave”

MAT 200C

MAT 200C

302

Loudness Models

- Log scales
 - dBA, dBM
- Named ranges
 - pppp - ffff
- Sample value
- Amplitude scale

MAT 200C

MAT 200C

303

Timbre, Instrument, Voice

- Instrument name
- Performance technique
- Synthesis technique
- Physical model
- Resource/instance

MAT 200C

MAT 200C

304

Example: MIDI

- Serial interchange format, 31.25 kHz
- Note on/off = 3 bytes: cmd/chan, key, vel
- Time: immediate
- Pitch: key number 0 - 127
- Loudness: key velocity 0 - 127
- Timbre: channel 1 - 16
- Sampled/quantized control parameter updates

MAT 200C

MAT 200C

305

Example: Common Music Notation

- (Common-practise Western Music Notation)
- Time: logical, relative
- Pitch: dodecaphonic named pitch with accidental (potentially microtonal)
- Loudness: named dynamic and continuous inexact functions
- Timbre: instrument name and performance technique
- Continuous inexact control functions

MAT 200C

MAT 200C

306

Example: Smoke

- OO Music/media data representation
- (490 Hz, (1/7 beat), 56 dB, (#voice -> #flute), (#embrochure -> #tight))
- Time: relative, logical
- Pitch: many models
- Timbre: many models
- Loudness: many models
- Hierarchy: many levels, relationships

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

307

Examples: Other

- Musical grammars
- Control streams
- ZIPI, OSC: control + trigger
- FORMES: processes
- Petri Nets: control flow

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

308

MEDIA ARTS & TECHNOLOGY PROGRAM

Models and Representation of Visual Signals

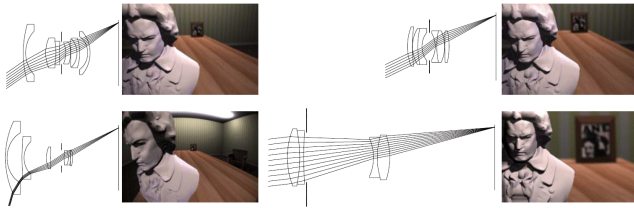


Figure 8: Four views of the same scene taken with a 16mm fisheye lens (bottom left), 35mm wide-angle lens (top left), 50mm double-Gauss lens (top right), and a 200mm telephoto lens (bottom right). A profile view of the lens system used to take each image is shown on the left. As with physical lenses, perspective is compressed with long focal lengths and expanded with short focal lengths. The fisheye image shows the lens' signature barrel distortion.

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

309

Geometry for Computers

- Mostly Cartesian x@y
- Origin in upper-left (as scanned)
- Origin in lower-left (1st quadrant)
- Where are the points?
 - Mid-pixel
 - Some corner
- Line-drawing algorithms and stroke thickness

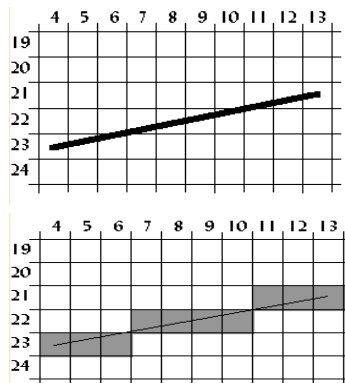
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

310

Line-Drawing

- How to denote the origin, destination points?
- Where are the pixels relative to the points?
- What pixels does the line cross through?
- How do I catch jaggies?



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

311

Computer Graphics 101

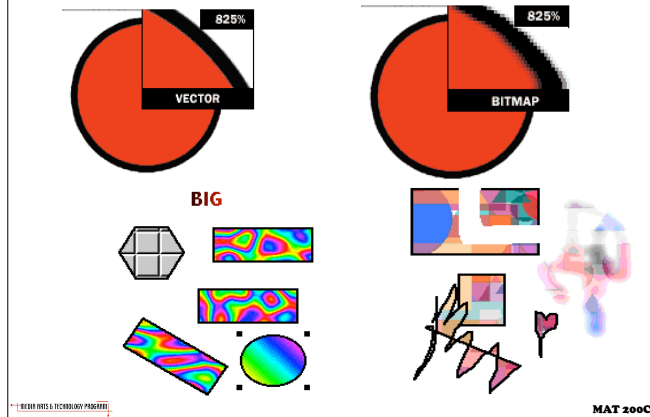
- 2D and 3D spatial models
- Cartesian and Polar
- Vectors, Planes, Solids
- Colors
- Textures and mapping
- Color and spatial dithering

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

312

Vectors (Draw) vs. Rasters (Paint)



313

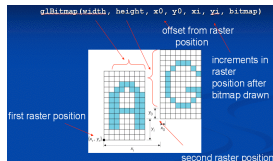
Representation

- Structured graphics display lists
- Bitmaps, pixel-maps
- Semantics of operations
 - Drawing operations
 - Scale, order, copy, stroke
 - Painting operations
 - Mask, fill, erase

314

BitBLT Operations

- Memory-to-memory copy with shift/mask
- Resulting destination pixels are some logical operation of the source, mask, alpha channel, and current destination pixel.
- Very simple to do; very hard to do fast



315

BitBLT Usage

```

CBitmap bitmap;
CDC dcStor;

// Load in our bitmap from our resources
bitmap.LoadBitmap(IDB_MYBITMAP);

// Make our "Memory CDC" have the same
// characteristics as our screen CDC
dcStor.CreateCompatibleDC(pDC);
dcStor.SelectObject(&bitmap);

// BitBlt message: src rect, dest map, origin, mode
pDC->BitBlt(10,10, 300,300, &dcStor, 0,0, SRCCOPY);
  
```

316

Structured Graphics

- Primitives
 - Line, arc, polygon, text, image
- Transformations
 - Move, scale, rotate
- Grouping and containment
- Window composition as display list
 - Window, header, layout, application pane
- Examples: QuickDraw, Xlib

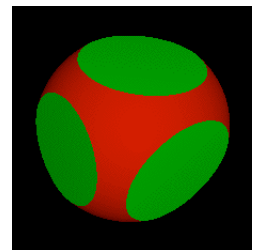
317

Constructive Solid Geometry

- Basic solids
- Union, intersection, difference











```

intersection{
  box{<0,0,0>, <4,4,4>
    pigment{color Green}}
  sphere{<2,2,2>, 2.5
    pigment{color Red}}
  scale 1.2
}
  
```



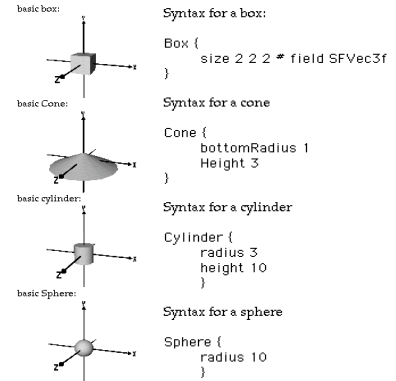
318

2-D Rendering Primitives (OpenGL)

Point	individual points	
Line	pair of vertices interpreted as individual line segments	
Polygon	boundary of a single, convex polygon	
Triangles	triple of vertices interpreted as triangles	
Quads	quadruple of vertices interpreted as four-sided polygons	
Line Strip	series of connected line segments	
Line Loop	series as above, with a segment added between last and first vertex	
Triangle Strip	linked strip of triangles	
Triangle Fan	linked fan of triangles	
Quad Strip	linked strip of quadrilaterals	

319

3-D (Platonic) VRML Primitives

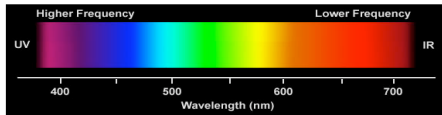


MAT 200C

320

Color

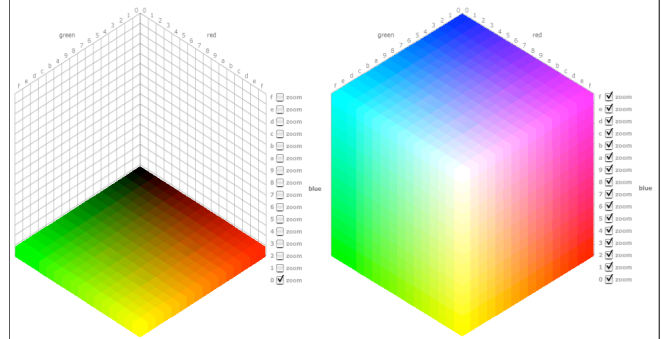
- Metamerism -- relation to Fourier
- Color spectra
- Color Correction
- Gamma Correction
 - "All CRT displays, almost all photographic film, and many electronic cameras have nonlinear signal-to-light-intensity or intensity-to-signal characteristics"
- Use power functions to correct



MAT 200C

321

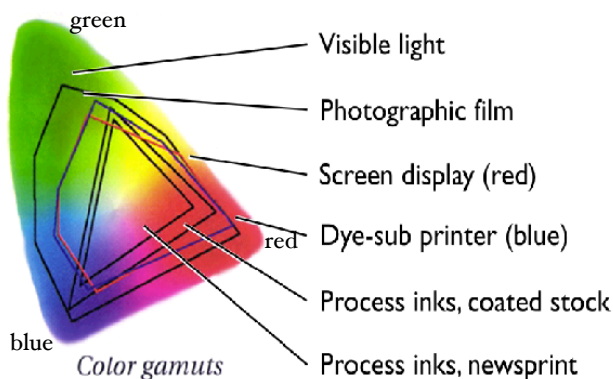
The Color Cube


<http://www.morecrayons.com/palettes/webSmart/colorcube.php>

MAT 200C

322

Visible and Displayable Colors

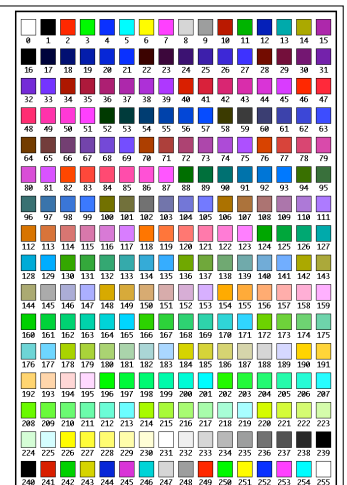


MAT 200C

323

256-Place Color Map

Most humans can differentiate approx. 250 colors.)



324

Gamma Gray Table

	3.0		1.8
	2.8		1.6
	2.6		1.4
	2.4		1.2
	2.2		1.0
	2.0		0.8
	1.8		0.6

<ftp://ftp.uu.net/graphics/png/images/suite/gamma.png>

MAT 200C

325

Raster/Image Graphics

- (It's the input, stupid)
- Formats: pixel-maps
- Operations
 - Draw, fill,
 - BitBlt (Bit-wise block logical transfer)
 - Source, destination, operation
- Structure extraction
 - Edge-detection
 - Region-building

MAT 200C

MAT 200C

326

Images, Masks, Opacity and the Alpha Channel

- Image combination
- "Opaque forms" and masking
- The stencil-paint model
- Mixed Formats
- Apple PICT
- Adobe PostScript

MAT 200C

MAT 200C

327

Alpha Channel

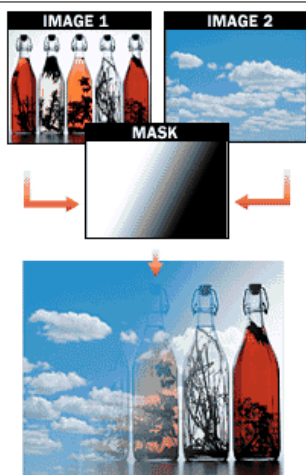


MAT 200C

MAT 200C

328

Masking



MAT 200C

329

Coordinate Systems

- Object Space
 - The coordinates in which an object is defined.
- World Space
 - The global system independent of the camera. The interface moves objects from the object space to the world space.
- Camera Space
 - A coordinate system with the viewpoint at the coordinate origin and the direction of the view along the positive Z axis.
- Screen Space
 - Normalized after the scene is projected onto a viewing plane.
- Raster Space
 - A two-dimensional space in which the top left corner of the top left pixel of the output image lies at (0,0) and pixel corners lie at non-negative locations.

MAT 200C

MAT 200C

330

CG ToCome

- Graphics and scene models
- Dynamic world models and object interaction
- Data formats for raster and vector data

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

331

Review

- Representations, storage formats, and interchange formats
- Language issues in representation
- Sound/music representation systems and notations
- Graphical models for static and dynamic images and scenes

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

332

Exercises, Demos

- Signal/symbol theory
 - AI and feature extraction
 - Signal transmission
- Models and representations
 - Sound/music representation languages
 - DASP languages
 - Models of images and scenes
 - Structured graphics tools
 - Image/scene tools

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

333

What's Next? (Topic 3 Readings)

- Data models, representations, and formats; data storage and transfer
 - Representation of sound, image
 - Sound and image data formats
 - Audio/visual rendering: models of objects, space and light
 - Storage and retrieval of media data: media databases
 - Real-time and isochronous transfer of media data over networks

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

334

MEDIA ARTS & TECHNOLOGY PROGRAM

End of
Topic 2



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

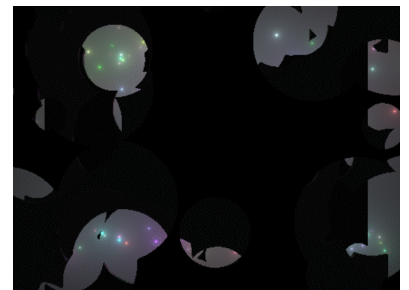
335

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C: Survey of Media Technology

Topic 3: Media Data Representations and Formats

Stephen T. Pope
MAT/UC Santa Barbara
stp@mat.ucsb.edu
Winter Quarter, 2008



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

336

Topic 3: Representations and Formats for Media Data

- Part 1: Sound and Music Storage and Interchange (snd, AIFF, MP3)
- Part 2: Still and Dynamic Image Storage and Interchange (GIF, JPEG/MPEG)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

337

Topic 3 Readings

- Wikipedia entries for *Audio file format*, *Image file formats*, *Format war*, and *Comparison of high-definition optical disc formats*
- D. Meyer. “Graphics File Formats” (www)
- B. Lipchak. “Overview of OpenGL” (www)
- M. Altmann. “About NURBS.” (www)
- J. Goubeaux. “Color Theory and Digital Color Management.”

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

338

See Also

- S. T. Pope and G. Van Rossum. “A Child's Garden of Sound File Systems” CMJ 19:1
- Graphics file format overviews
- Container formats
- Other MAT 201B Readings

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

339

Compression & Formats

- Background on MPEG TV Compression (www)
- Newsgroup comp.compression FAQ (www: news.comp.compression)
- MPEG-1 Data Structures (www)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

340

Extra References

- Music I/O and applications
- Sound file formats
- CORBA
- SMIL
- Image/scene formats
- Image processing
- Structured graphics formats and tools

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

341

Digital Audio Formats

- History, Background
- OS, HW, and API considerations
- Sampling rates
- Sample formats and resolutions
- Headers and header-less formats

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

342

Part I: Sound and Music

- Music Descriptions
- Sound File Formats
- Models and Compression
- Sound Streaming

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

343

Music Descriptions

- Music input languages (MILs) and software sound synthesis (SWSS)
 - Orchestra/Score model
- Score interchange formats
 - Adagio -- terse note-oriented
 - NIFF -- CWM notation interchange
 - MIDI -- musical instrument digital interface
- Music databases
 - CCARH books
 - Paleo project at CREATE

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

344

Sampling and Quantization Again

The **digitization** of a signal $x(t)$ comprises two steps: $x(t) \rightarrow x_s(n) \rightarrow x_q(n)$

– **Sampling**: $x_s(n) = x(nT)$ where

- T is the **sampling period**
- $F = 1/T$ is the **sampling frequency**
- The inverse transformation ($x_s(n)$ to $x(t)$) is called **interpolation**

– **Quantization**: $x_q(n) = Q(x_s(n))$

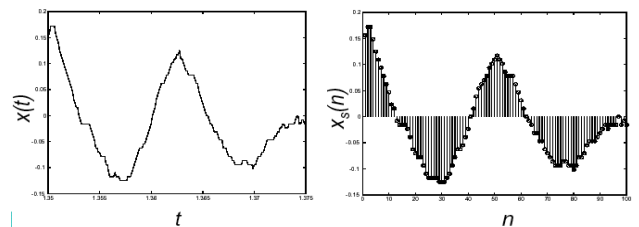
- $Q(\cdot)$ is a **rounding function** which maps the values of $x_s(n)$ into N levels (Δ is the **quantization step**)
- Typically, $N = 2^{N_b}$, therefore we need N_b bits to represent one quantized sample.

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

345

Sampling Example

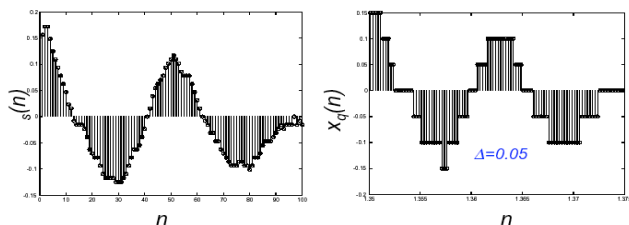


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

346

Quantization Example

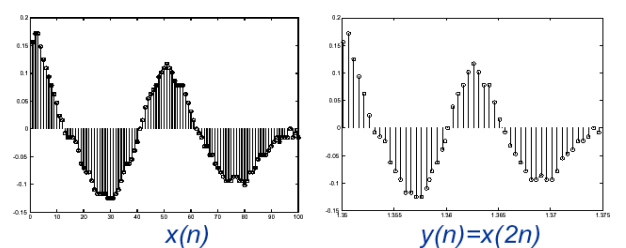


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

347

Sub-sampling



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

348

The Wonderful History of Sound File Formats

- Tape-based interchange (1960s)
 - SWSS and D/A in multiple stages
- Disks and sound file systems (1970s)
 - Special-purpose CCSS
 - General-purpose SFS
- Large-scale sound storage (1980s)
- Streaming audio (1990s)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

349

(Sound) File Systems

- Issues
 - Throughput, file size, mark-up, compression, multiple formats
- UNIX special-device model
 - open(), close(), read(), write(), ioctl()
 - Special directory operations
- OO Model
 - Directory, Filename, IOStream objects

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

350

The Play/Record Program

- Steps, functions
 - Command parsing, set-up
 - File I/O
 - Buffering
 - Driver and scheduling
 - Interrupt servicing
- The Sound I/O API
 - OS-, HW- and format-dependent

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

351

Sampling Rates

- Early: 10 kHz, 12-bit samples (determined by 1960s DAC HW)
- See table in Pope & van Rossum
- Modern rates: 5500 - 44100 - 192000 Hz
 - Common divisors of high frequencies
 - Common multiples of powers of two or other sampling frequencies

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

352

Sample Formats, Resolutions

- 8-bit compressed - 24-bit linear - 32- or 64-bit floating-point
- Format: linear, μ -law, floating-point
- Resolution: 4-64-bits, relationship to computer word size
- Perception and limen testing (70s, 90s)
- Linearity, monotonicity, residual noise of contemporary playback systems

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

353

Sampled Sound Format Examples

Speech/Audio Type	Frequency Range	Sampling Rate	Bits/Sample	Uncompressed Bit Rate
Narrowband Speech	200-3200 Hz	8 kHz	16	128 kb/s
Wideband Speech	50-7000 Hz	16 kHz	16	256 kb/s
CD Audio	20-20000 Hz	44.1 kHz	16 x 2 channels	1.41 Mb/s

- Others?

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

354

The Sound File

- Header vs. header-less
 - Type implicit in name
 - xxx.au, xxx.snd
 - Format described in file header or resource fork
 - Fixed parameters
- Fixed-format vs. “chunked” formats
 - Description and data chunks

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

355

The Sound File Header

- “Magic number” (keyword or file ID, 32-bit, identifies file)
- Basic parameters
 - Sampling rate
 - Format/resolution
 - Number of channels
- Sample data block (typically 1)
- Other data
 - Comment, text
 - Pitch
 - MIDI data
 - Loop points

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

356

The NeXT/Sun File Header (1986)

```
typedef struct {
    int magic;          /* magic number SND_MAGIC */
    int dataLocation;   /* offset/pointer to the data */
                        /* (header can be >28 bytes) */
    int dataSize;       /* number of bytes of data */
    int dataFormat;     /* the data format code */
    int samplingRate;   /* the sampling rate */
    int channelCount;   /* the number of channels */
    char info[4];       /* optional text information */
} SNDSoundStruct;
```

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

357

Extended Fixed Headers

- MOD, AVR, SndDesigner
 - Sample Loop points (begin/end)
 - Pitch, MIDI key
 - MIDI keyboard split, multiple samples
- IRCAM/BICSF, EBICSF
 - MaxAmp, ampl. envelopes
 - Pitch envelopes, LPC
 - Processing scripts
 - Cues, virtual files

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

358

Tagged Formats: AIFF/WAV

- Chunked file formats: simple, ubiquitous
 - Chunk = ID, Size, Data
- ```
struct ChunkHeader {
 int32 ckID; // char[4]
 int32 ckSize; // size in bytes
};
```
- File: FORM, COMM, DATA, etc.

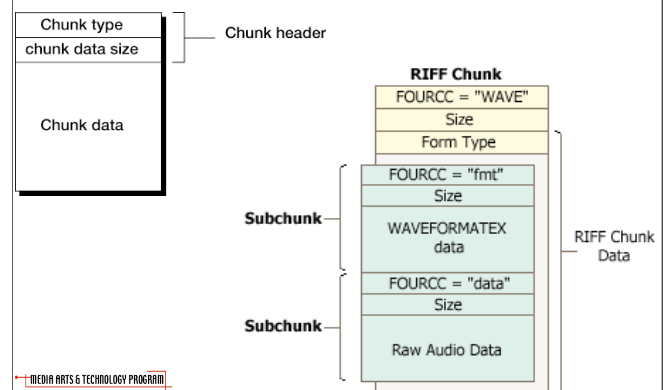
```
struct SoundDataChunk {
 int32 ckID;
 int32 ckSize;
 unsigned int32 offset;
 unsigned int32 blockSize;
};
```

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

359

## AIFF/WAV Chunk & File Structure



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

360

## LibSNDFile Coding

```
// built-in types
SNDFILE *file ;
SF_INFO sfinfo ;

// header struct members
sfinfo.format = filetype | SF_FORMAT_PCM_16 ;
sfinfo.samplerate = 48000 ;
sfinfo.channels = 2 ;

// UNIX file model
file = sf_open (filename, SFM_READ, &sfinfo_rd);

// typed read/write fcns.
k = sf_write_short (file, data, this_write);
```

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

361

## Sound Data Processing

- CARL - UNIX shell pipes
  - sndin/out
  - Filter, gain, reverb,
  - par/chan
- SoundHack
  - Mac format conversion
  - Analysis/synthesis
- SOX
  - Portable UNIX conversion, transformation

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

362

## Sound Formats and Bandwidths

- Surround HiFi (8 @ 24/96) 18 M b/s
- “CD-quality” (2 @ 16/44) ~ 1.4 M b/s
  - Losses (rel. to 24/96) easily heard over good systems
- MP3 (DCT, masking) ~128 k b/s (VBR)
  - Debatable losses relative to “CD-quality” over “mid-fi” systems for  $\geq 128$  k b/s bandwidth
  - Obvious losses for lower bit-rates
- RealAudio (CELP) ~ 14 k b/s
  - Monophonic, “telephone-quality”
- Best speech compression ~ 1.5 k b/s

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

363

## Sound Streaming Issues

- Database server (disk I/O)
- Network server (100BaseT, ATM)
- Client-side player (lightweight, plug-in)
  - Network interface
  - Buffer
  - I/O driver
- Network considerations
  - Quality-of-service (QoS)
  - Rate negotiation
  - Distributed buffers (multicast)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

364

## Streaming Examples

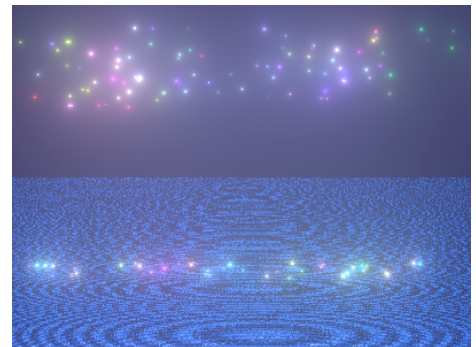
- RealAudio
  - Variable QoS, bandwidth (5-96 kbps, lo-fi)
  - Simple (proprietary) client
  - Mono-directional 1:n, no distributed buffers
- Network Audio System (NAS)
  - Fixed QoS (CD/au)
  - Symmetrical client/server
  - Bi-directional n:m, distributed buffers
- Others: QT, Liquid, MS, SMIL, ...

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

365

## Part 2: Still and Dynamic Images



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

366

## Topic 3.2: Representations and Models

- Part 1: Sound and Music Storage and Interchange (snd, AIFF, MP3, SMIL)
- **Part 2:** Still and Dynamic Image Storage and Interchange (GIF, JPEG, MPEG)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

367

## Part 2: Still and Dynamic Images

- Visual Transducers
- Graphics Background
- Geometry Software
- Structured Graphics
- Images
- Fonts and text
- Storage/Interchange Formats

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

368

### Visual Transducers



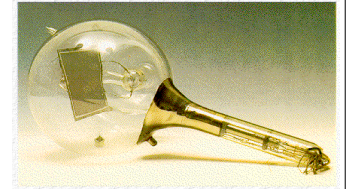
- Chemical Transducers (Daguerre, 1830s)
  - Copper plate coated with silver iodide was exposed to focused light, then fumed with mercury vapor and fixed with salt
- Photoelectric effect (May, 1873)
  - Selenium bars, exposed to sunlight, show a variation in resistance.

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

369

### TV



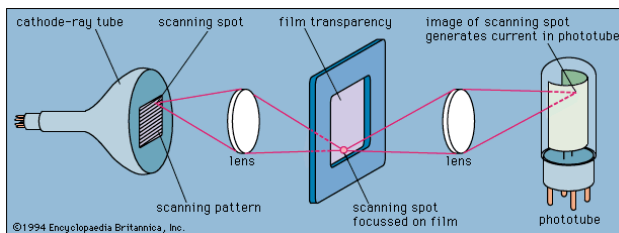
- Cathode Ray Tube (Zworkyin, 1923)
- Interlaced scanning (Schroeter, 1930)
- Iconoscope camera (1935)
- Solid-state Still/Video Cameras
- LCD and other new displays

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

370

### Electrical Scanner

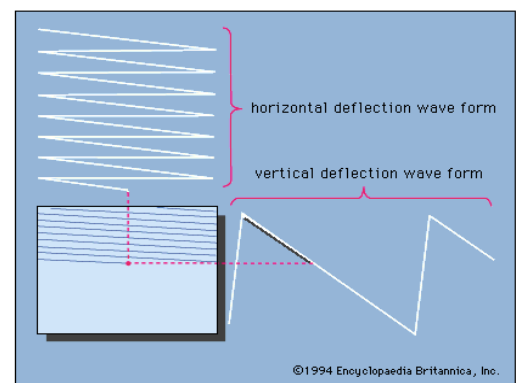


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

371

### CRT Scanning

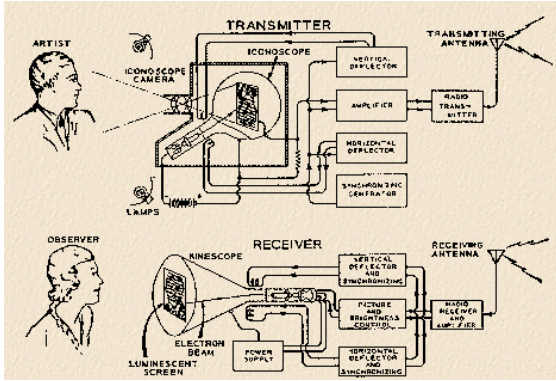


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

372

## TV I/O: Iconoscope and Kinescope



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

373

## CRTs

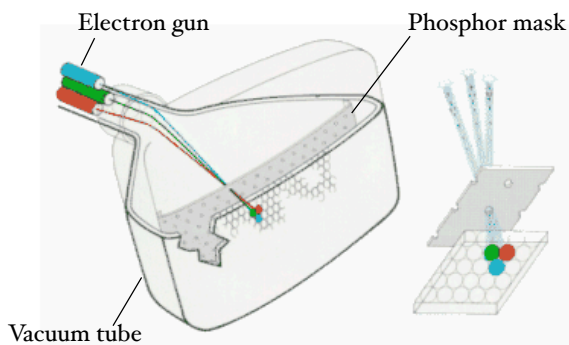
- Vector tubes (non-uniform scanning)
- Constant-raster tubes
- Masks and colors
- CRT Properties
  - Resolution
  - Scan frequency
  - Color temperature
  - Geometry
  - Aspect ratio
  - Pixel shape

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

374

## Color CRT



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

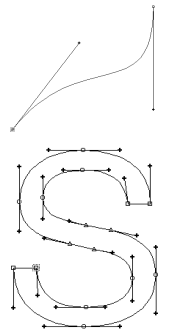
MAT 200C

375

## Representing Fonts



- Raster and stroked fonts
- Kerning (inter-character spacing)
  - Kerning tables
- Character processing
  - Rotation, scaling, shadow, outline
- Font storage
- Character formats



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

376

## Animation

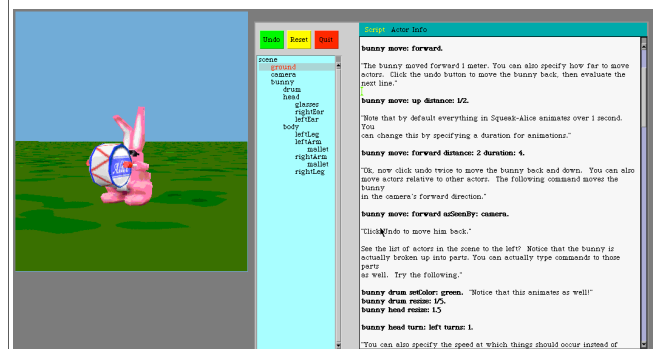
- Vector vs. raster-based
  - Same issues only more
- Agents, Actors, and Sprites
  - Distribution of behavior and parallelism
  - Real-world model
  - Time, control, interaction
  - Rendering

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

377

## Alice Scripting Example



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

378

## Color Management

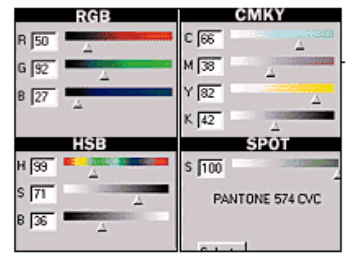
- ICC: Int'l Color Consortium  
characterization of I/O media
- CIE: International Commission on  
Illumination (CIE XYZ)
- Color look-up tables

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

379

## Color Models



- RGB Color
  - Red, green, blue -- additive (CRT)
  - Values may be ints or floats
- CMY Color
  - Cyan, magenta, and yellow -- subtractive (gels)
- CMY and RGB colors are complementary

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

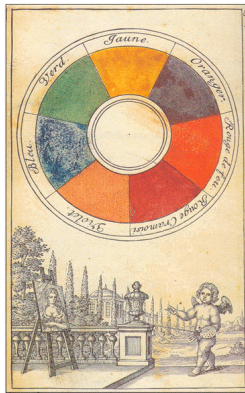
MAT 200C

380

## Color Models 2

- HSV Color
  - Hue, saturation, and value

```
struct HSVColor {
 float hue; // Fraction of circle,
 // red at 0
 float sat; // 0-1, 0 for gray,
 // 1 for pure color
 float value; // 0-1, 0 for black,
 // 1 for most intensity
};
```
- HSL Color
  - Hue, saturation, and lightness
  - Value = 0-1, 0 for black, 1 for white



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

381

## Color Maps

- Map between an image's fixed (low) number of colors and a medium's (larger) number.
- Typically size is a power of two (pixel depth of image)
- There may be one per window or per screen
- There are several standards (see above)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

382

## Image Format Standards (I)

| Format name | Lines/frame | Pixels/line | Bits/Pixel |
|-------------|-------------|-------------|------------|
| FAX         | 2200        | 1700        | 1          |
| VGA         | 480         | 640         | 8          |
| XVGA        | 768         | 1024        | 24         |

| Format name | Lines/frame | Pixels/line | Frames/s | Interlaced | SS scheme | Aspect ratio |
|-------------|-------------|-------------|----------|------------|-----------|--------------|
| CIF         | 288         | 352         |          | N          | 4:2:0     | 4:3          |
| QCIF        | 144         | 176         |          | N          | 4:2:0     | 4:3          |
| SQCIF       | 128         | 96          |          | N          | 4:2:0     | 4:3          |
| SIF-325     | 240         | 352         | 30       | N          | 4:2:0     | 4:3          |
| SIF-625     | 288         | 352         | 25       | N          | 4:2:0     | 4:3          |
| NTSC        | 486         | 720         | 29.97    | Y          | 4:2:2     | 4:3          |
| PAL         | 576         | 720         | 25       | Y          | 4:2:2     | 4:3          |
| HDTV        | 720         | 1280        | 59.94    | N          | 4:2:0     | 16:9         |
| HDTV        | 1080        | 1920        | 29.97    | Y          | 4:2:0     | 16:9         |

- **Example:** what is the bit-rate for interlaced HDTV format?  
 $N_l = 1080$  lines per frame,  $N_p = 1920$  pixels per line,  $R_f = 29.97$  frames/s  
frames per second, 12 bits per pixels (luminance + subsampled chrominances):  $N_l N_p R_f \cdot 12 = 745,749,504$  bits/s.

MEDIA

200C

383

## Video Formats (Rowe 1994)

### Many Video Formats

Analog - NTSC, PAL, SECAM

Digital - CCIR 601, D1-D5, HDTV

### Relative Comparison

|               | Image Size | Bytes/Pixel | MBytes/sec |
|---------------|------------|-------------|------------|
| NTSC          | 640x480    | 3           | 27.6       |
| PAL           | 768x576    | 3           | 33.2       |
| CCIR 601 (D2) | 720x485    | 2           | 21.0       |
| HDTV          | 1920x1080  | 3           | 78.0       |
|               | 1280x720   | 3           | 35.0       |

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

384

## Operations on Images

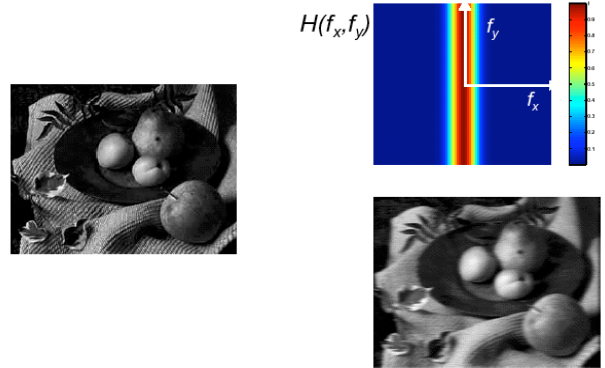
- 1D and 2D operations
- Visual frequency and filtering
- Sampling and quantization
- Color map operations
- Image combination, keying, masking

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

385

## Image Filtering (horiz-lopass)

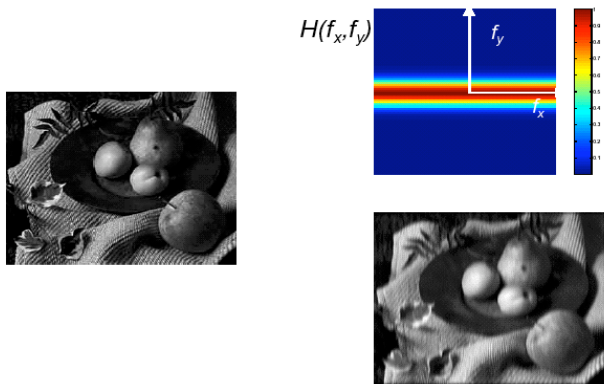


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

386

## IF: v-lp

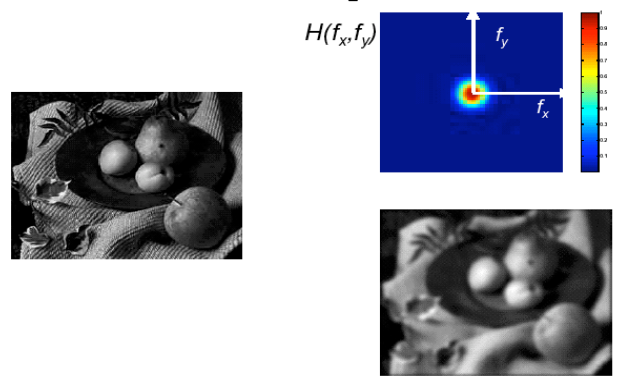


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

387

## IF: c-lp

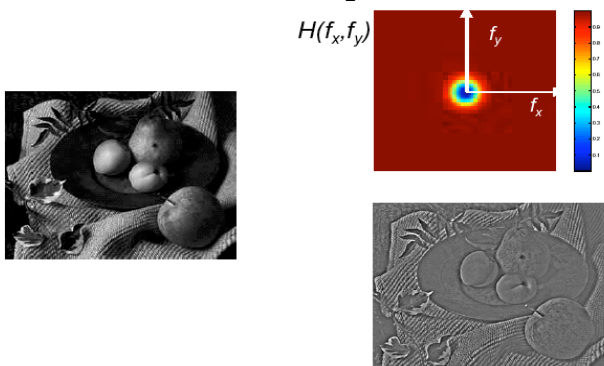


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

388

## IF: c-hp



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

389

## Image Sub-sampling

*Subsampled by 2*



*Without prefilter*

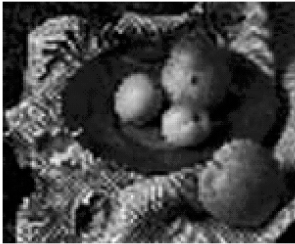
*With prefilter*

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

390

## IS 2

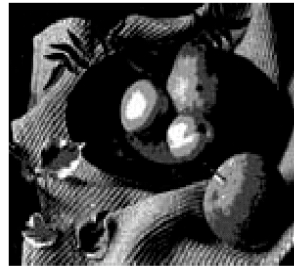
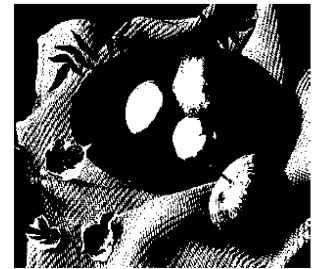
*Subsampled by 4**Without prefilter**With prefilter*

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

391

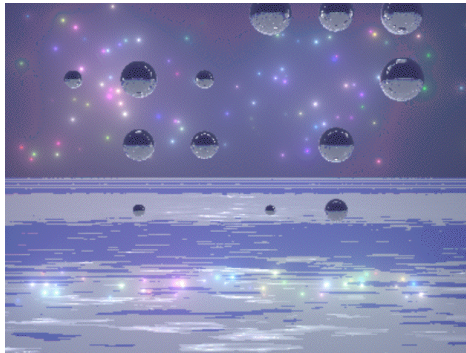
## Bit-depth/quantization

 $N_b=2$  $N_b=1$ 

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

392

Graphics  
Formats

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

393

## Graphics Formats

- Scene Description (modeling)
- Computer display (presentation)
- Page description (printing)
- Animation (key frame rendering)
- Issues
  - Units
  - Colors
  - Structure/pixels

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

394

## Storage and Interchange Formats

## • Simple Pixmap Structures

- `int x, y, d, *pixel_data`

## • Sun RAS format

```
struct rasterfile {
 int ras_magic;
 int ras_width;
 int ras_height;
 int ras_depth;
 int ras_length;
 int ras_type;
 int ras_maptype;
 int ras_maplength;
};
```

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

395

## BMP FILE FORMAT

- Images with 1, 4, 8, or 24 bits per pixel
- Stored by scan line, bottom to top (!!)
- Two incompatible versions of this format, one introduced with OS/2 & Windows 3.0
- Resolution-independent (!!)
- Hierarchical color map (!!)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

396

## BMP File Header (MSC Format)

| Data                   | Description                        |
|------------------------|------------------------------------|
| WORD Type;             | File type. Set to "BM."            |
| DWORD Size;            | Size in DWORDs of the file         |
| DWORD Reserved;        | Reserved. Set to zero.             |
| DWORD Offset;          | Offset to the data.                |
| DWORD headerSize;      | Size of rest of header. Set to 40. |
| DWORD Width;           | Width of bitmap in pixels.         |
| DWORD Height;          | Height of bitmap in pixels.        |
| WORD Planes;           | Number of Planes. Set to 1.        |
| WORD BitsPerPixel;     | Number of bits per pixel.          |
| DWORD Compression;     | Compression. Usually set to 0.     |
| DWORD SizeImage;       | Size in bytes of the bitmap.       |
| DWORD XPixelsPerMeter; | Horizontal pixels per meter.       |
| DWORD YPixelsPerMeter; | Vertical pixels per meter.         |
| DWORD ColorsUsed;      | Number of colors used.             |
| DWORD ColorsImportant; | Number of "important" colors.      |

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

397

## TIFF - "Tagged" format

- Basic chunked format
- Four TIFF Classes have been defined:
  - Class B for bilevel (1-bit) images
  - Class G for grayscale images
  - Class P for palette color images
  - Class R for RGB full color images

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

398

## GIF

- Chunked format
- LZW (statistical) compression
- Unisys/Compuserve ownership (\$5000 per program that generates GIFs)
- Move to PNG!

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

399

## GIF89a File Structure (chunks)

- GIF89a HEADER
- LOGICAL SCREEN DESCRIPTOR BLOCK
- GLOBAL COLOR TABLE
- optional NETSCAPE APPLICATION EXTENSION BLOCK
- a stream of graphics, each graphic being composed of:
  - optional GRAPHIC CONTROL BLOCK (one for each IMAGE)
  - a single IMAGE DESCRIPTOR or PLAIN TEXT BLOCK which can include an optional LOCAL COLOR TABLE for an image and the actual IMAGE or TEXT data table
- GIF TRAILER ends the series of images

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

400

## Animated GIF

- Series of GIFs with delays
- Compressor does inter-frame relative differentials
- Recognizes
  - Fades
  - Panning
  - Scaling
  - Other transformations

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

401

## PNG: Portable Network Graphics

- Lossless compression
- Supports up to 48bit color or 16 bit gray scale
- Full alpha channel and transparency support
- Palette-mapped images to 256 colors
- Streamability and progressive display
- "Legally unencumbered" compression
- Hardware and platform independence
- W<sub>3</sub>C Standard

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

402

## PNG Required Chunks

- The header chunk (IHDR) - contains basic information about the image data and must appear as the first chunk, and there must only be one header chunk in a stream.
- The palette chunk (PLTE) - stores the colormap data associated with the image data. Required if the image data uses a color palette.
- The image data chunk (IDAT) - stores the image data; multiple image data chunks may occur in a data stream and must be stored in contiguous order.
- The image trailer chunk (IEND) - must be the final chunk and marks the end of the PNG file or data stream.

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

403

## PNG File Header

- A PNG file starts with an 8-byte signature: 89 50 4E 0D 0A 1A 0A. Each of these is there for a good reason.
- 89 - Has the high bit set to detect systems that do not support 8 bit data (i.e., 7 bits + parity).
- 504E47 - In ASCII, the letters "PNG", allowing a person to identify the format if it is viewed in a text editor.
- 0D0A - A DOS style line ending (CRLF) to detect DOS-UNIX line ending conversion of the data.
- 1A - A byte that stops display of the file under DOS when the command TYPE has been used
- 0A - A unix line ending (LF) to detect UNIX-DOS line ending conversion.

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

404

## PNG File Header Chunk

- The IHDR chunk must appear FIRST. It contains:
- Width: 4 bytes (2 GPixel!)
- Height: 4 bytes
- Bit depth: 1 byte
- Color type: 1 byte
- Compression method: 1 byte
- Filter method: 1 byte
- Interlace method: 1 byte

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

405

## PNG File Color types

- Color Allowed Interpretation  
Type Bit Depths

|     |            |                                                                   |
|-----|------------|-------------------------------------------------------------------|
| • 0 | 1,2,4,8,16 | Each pixel is a grayscale sample.                                 |
| • 2 | 8,16       | Each pixel is an R,G,B triple.                                    |
| • 3 | 1,2,4,8    | Each pixel is a palette index;<br>a PLTE chunk must appear.       |
| • 4 | 8,16       | Each pixel is a grayscale sample,<br>followed by an alpha sample. |
| • 6 | 8,16       | Each pixel is an R,G,B triple,<br>followed by an alpha sample.    |

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

406

## Using PNG

- LibPNG.org open source libraries

```
png_structp png_ptr = png_create_read_struct(
 PNG_LIBPNG_VER_STRING, 0, 0, 0);
png_infop info_ptr = png_create_info_struct(png_ptr);
png_init_io(png_ptr, some_file);
png_read_png(png_ptr, info_ptr, 0, 0);
height = info_ptr->height;
width = info_ptr->width;
channels = info_ptr->channels;
png_bytep * row_pointers = new png_bytep[height];
row_pointers = png_get_rows(png_ptr, info_ptr);
```

- TweakPNG et al. utilities

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

407

## JPEG

- JPEG: Joint Photographic Experts Group
- JBIG: Joint Bi-level Image experts Group (fax)
- Lossy image compression scheme for images to be viewed by humans
- Better for natural scenes than text
- Trade-offs between compression, loss, and speed (pick any 2 again)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

408

## JPEG/MPEG Compression

- Spatial effects
  - Color gradient vs. luminosity gradient
- Luminosity effects
  - Independent of color
- Color effects
  - Independent of luminosity
- (Remember Masking?)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

409

## JPEG Formats

- JFIF (JPEG File Interchange Format)
  - Low-end, pixels only
  - Standard on the Web
- TIFF/JPEG, aka TIFF 6.0
  - High-end, includes annotations
  - Used in graphics programs

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

410

## Structured Graphics



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

411

## Structured Graphics Formats

- Declarative/Procedural
- QuickDraw and PICT format
- PostScript format as a language
- Hierarchy, naming, ordering
- Renderers

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

412

## PostScript Programming / Page-description Language

- Derived from Xerox INTERPRESS
- Based on FORTH
- Stencil-paint imaging model
- All positions are arbitrary-precision floats
- All paths are splines
- Context and transformations
- Paths and clipping
- Outline Fonts

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

413

## Example: Draw a box

```
%!
%% Draws a one square inch box and inch in
%% from the bottom left
/inch {72 mul} def
%% Convert inches -> points (1/72 inch)
Newpath % start a path
1 inch 1 inch moveto
2 inch 1 inch lineto
2 inch 2 inch lineto
1 inch 2 inch lineto
Closepath % Finish and close the path
Stroke % Draw the outline (no fill)
Showpage % Show the page
```

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

414

## Use a box as a clipping region

```

%!
% operator box: xcoord ycoord box -
% Creates one inch box at xcoord, ycoord
/box {
 newpath
 moveto % gets args off stack
 72 0 rlineto
 0 72 rlineto
 -72 0 rlineto
 closepath
} def
/Times-Roman findfont 30 scalefont setfont
gsave % Save the old clip path
72 72 box % Set up our box
gsave % save box
stroke
grestore % Restore the box path
clip % Clip to the box
60 60 moveto
(This is Times-Roman clipped to a box) show
grestore % Get the clip path back
showpage

```

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

415

## Rotation and Transformation

```

gsave
100 450 translate % Set the origin
45 rotate % Rotate by 45 degrees
0.5 1 scale % Scale coordinates
0 0 box stroke % Draw box
75 0 moveto
(Everything) show
grestore

```

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

416

## EPS FILE FORMAT

- File obeys Adobe PostScript Document Structuring Conventions (DSC)
- May include 1 or more pages
- May contain a low-resolution thumbnail

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

417

## VRML

- History
  - SGI, VRML 1.0, Web3D and VRML 97
- Worlds and coordinate spaces
- Transforms
- Light sources
- Hyperlinks and WWW
- Object behaviors and scripting

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

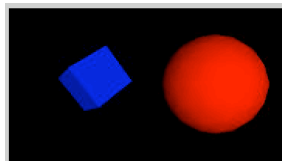
418

## VRML Example

```

#VRML V2.0 utf8
Transform {
 children [
 NavigationInfo { headlight FALSE } # We'll add a light
 DirectionalLight { # First child
 direction 0 0 -1 # Light illuminating the scene
 }
 Transform { # Second child - a red sphere
 translation 3 0 1
 children [
 Shape {
 geometry Sphere { radius 2.3 }
 appearance Appearance {
 material Material { diffuseColor 1 0 0 } # Red
 }
 }
]
 }
]
}

```



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

419

## VRML Example, cont'd

```

Transform { # Third child - a blue box
 translation -2.4 .2 1
 rotation 0 1 1
 children [
 Shape {
 geometry Box {}
 appearance Appearance {
 material Material { diffuseColor 0 0 1 } # Blue
 }
 }
] # end of children for world
}

```

See <http://www.vrml.org/technicalinfo/specifications/vrml97/part1/exampleD.2.wrl>

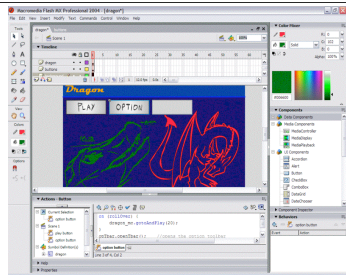
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

420

## Flash

- Manages animation
- User input can be managed via actionscript and widgets (see below)
- Stored with SWF ("swiff") file format (compact, streamable)
- Macromedia (now Adobe) streaming and storage format
- Supports both raster and vector representations
- Supports video and audio



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

421

## Flash Tools

- Macromedia Flash suite
- Open-source readers/renderers: FLIRT, GPLFlash, OpenLaszlo and JSFL JavaScript-to-Flash API (see OSFlash.org)
- Plug-in renderers for most web browsers
- Stand-alone execution engine

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

422

## Scalable Vector Graphics: SVG

- Text- (XML-) based
- Blessed by the W3C and Adobe (open spec.)
- Not as flexible or compact as Flash
- SVG Example (draw a box)

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://
www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="500" height="500">
 <rect x="111" y="78" width="187" height="160"
 style="fill:rgb(192,192,255);
 stroke:rgb(0,0,128);
 stroke-width:1"/>
</svg>
```

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

423

## AVI Movie File Format

- Specialization of RIFF

```
'RIFF' (4 byte file length) 'AVI '
// file header (a RIFF form)
'LIST' (4 byte list length) 'hdlr'
// list of headers for AVI file
```

The 'hdlr' list contains:

```
'avih' (4 byte chunk size) (data)
// the AVI header (a chunk)
'strl' lists of stream headers for each stream
(audio, video, etc.) in the AVI file.
```

For an AVI file with one video and one audio stream:

```
'LIST' (4 byte list length) 'strl'
// video stream list (a list)
```

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

424

## Video Stream

The video 'strl' list contains:

```
'strh' (4 byte chunk size) (data)
// video stream header (a chunk)
'strf' (4 byte chunk size) (data)
// video stream format (a chunk)
```

```
'LIST' (4 byte list length) 'strl'
// audio stream list (a list)
```

The audio 'strl' list contains:

```
'strh' (4 byte chunk size) (data)
// audio stream header (a chunk)
'strf' (4 byte chunk size) (data)
// audio stream format (a chunk)
'JUNK' (4 byte chunk size) (data - usually all zeros)
// an OPTIONAL junk chunk to align on 2K byte
boundary
```

```
'LIST' (4 byte list length) 'movi'
// list of movie data (a list)
```

The 'movi' list contains the audio and video data.

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

425

## MOVI List

This 'movi' list contains one or more ...

```
'LIST' (4 byte list length) 'rec '
// list of movie records (a list)
```

```
'##wb' (4 byte chunk size) (data)
// sound data (a chunk)
```

```
'##dc' (4 byte chunk size) (data)
// video data (a chunk)
```

```
'##db' (4 byte chunk size) (data)
// video data (a chunk)
```

A 'rec' list (a record) contains the audio and video data for a single frame.

```
'##wb' (4 byte chunk size) (data)
// sound data (a chunk)
```

```
'##dc' (4 byte chunk size) (data)
// video data (a chunk)
```

```
'##db' (4 byte chunk size) (data)
// video data (a chunk)
```

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

426

## Other Formats

- ...so many to choose from
  - 1100-page book (<http://www.oreilly.com/catalog/gffcd/>)
  - Web references
- QuickTime
- New image formats
  - Geographical data
- New OO DisplayList formats
  - VR

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

427

## File Format Convertors

- All can import and save many formats
- GraphicConvertor -- also has many transformations: scaling, dithering, etc.
- xv
- Netpbm
- TweakPNG
- many others

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

428

## Exercises and Demos

- Concrete sound I/O libraries
- Streaming
- File I/O
- Structured graphics processing
- Image processing
- CORBA next steps
- SMIL authoring and rendering

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

429

## Review

- Part 1: Sound and Music Storage and Interchange (snd, AIFF, MP3)
- Part 2: Still and Dynamic Image Storage and Interchange (GIF, JPEG/MPEG)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

430

## What's Next? (Topic 4 Readings)

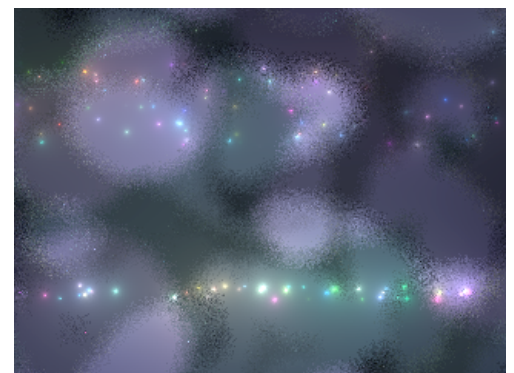
- Digital manipulation of signals and symbols
  - Data processing in the time, spectral, spatial, and other domains
- Basic models and synthesis techniques
  - Sound analysis/synthesis techniques
  - Control and interaction
  - Image/scene description and rendering
  - Video generation and animation

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

431

End of  
Part 3



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

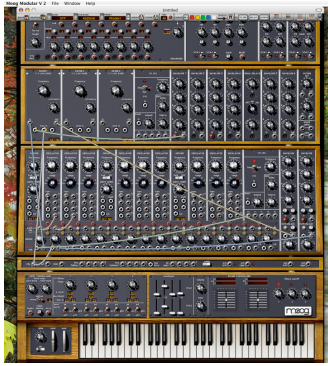
432

**MEDIA ARTS & TECHNOLOGY PROGRAM**

## MAT 200C: Survey of Media Technology

### Topic 4: Media Data Manipulation

Stephen T. Pope  
MAT/UC Santa Barbara  
stp@mat.ucsb.edu  
Winter Quarter, 2008



**MAT 200C**

433

## Topic 4: Digital Manipulation of Signals and Symbols, Selected Topics

- Part 1: Sound analysis/synthesis, processing, rendering and performance
- Part 2: image and video analysis/synthesis and processing

## Readings: Sound/Music

- J. O. Smith. "Physical Modeling Synthesis Update" CMJ 20:2
- M-H Serra. "Introducing the Phase Vocoder" Musical Signal Processing (pp. 31-55)
- G. Kendall. "A 3-D Sound Primer" CMJ 19:4
- V. Galloway and S. Wilkinson. "Surrounded by Sound" Electronic Musician 12/99
- Optional:
  - J-M Jot. "Efficient Models for Reverberation and Distance Rendering" Proc 1997 ICMC
  - R. Vaananen and J. Huopaniemi. "Virtual Acoustics Rendering in MPEG-4" Proc 1999 ICMC

## Readings: Image/Graphics

- POV-Ray Beginning Tutorial (www)
- S. Martin. "Animation Tricks." (www)
- S. Dunn. "Digital Color." (www)
- M. Altmann. "Hardware Acceleration for Window Systems." (www)

## Extra References

- Sound analysis/synthesis
  - Basic techniques
  - Physical models
- Audio rendering and performance
  - Pluriphonic processing and projection
- Image and video
  - Models, formats, rendering, ray-tracing

## Sound Analysis/Synthesis and Processing

- Survey of modern media analysis/synthesis techniques
- Focus on physical modeling
- Processing and effects
- Spatial hearing and spatial sound projection

## Review: Sound Synthesis

- (See MAT 240D)
- Additive (Fourier analysis, sum-of-sines or IFFT synthesis)
- Subtractive (filter-based)
- Non-linear (FM, look-up tables)
- Sample-based (or grain-based)
- Physical model-based
- Synthesis control and I/O

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

439

## Audio Synthesis Programming

- Write a function that fills a buffer with samples based on some static state (e.g., inst. phase or prev. outputs)
- Register this function as a call-back for your sound driver or plug-in API
- Extend to support multiple sound sources (callback uses list of playing sources & mixer)
- Most APIs use similar callback signatures
- See MAT 240A, 240D and CSL

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

440

## Sound Synthesis Callback

- Example: PortAudio in C

```
static int paCallbackSignature(
 const void *inputBuffer,
 void *outputBuffer,
 unsigned long framesPerBuffer,
 const PaStreamCallbackTimeInfo* tInfo,
 PaStreamCallbackFlags statusFlags,
 void *userData);
```

- Implementer holds synth. state in userData

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

441

## Background: SW Technologies

- Late 1970s
  - OOP
  - “Simulation as Application”
  - GUIs
  - HW: DSPs, PCs
- 1980s
  - “Alternate Reality Kit,” VR/VE
  - New UI Devices (gloves, HMD)
  - 3D Rendering
  - HW: “3M” machines, Window systems

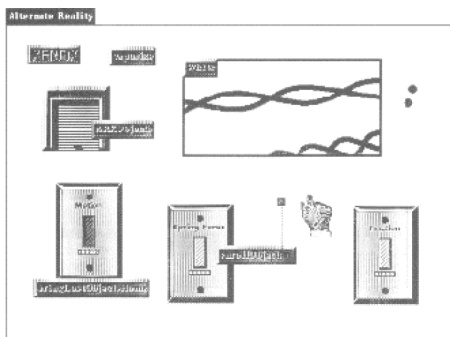
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

442

## Early Physical Models

- Simulation and “Alternate Reality”

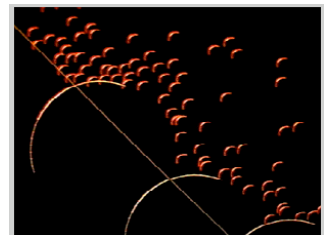


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

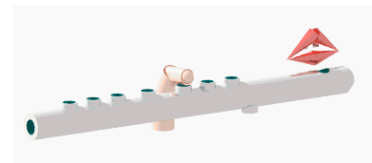
MAT 200C

443

## Combined PM of Image and Sound



- <http://www-acroe.imag.fr/mediatheque/videotheque/videotheque.html>

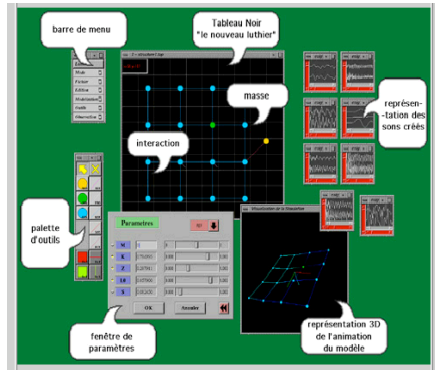


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

444

## CORDIS/ANIMA Today



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

445

## Types of Physical models

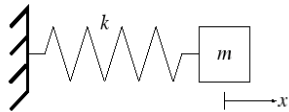
- Finite elements (mass-spring)
- Direct (Differential equations)
- Waveguides (delay-lines)
- Modal (resonance)
- Others

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

446

## Mass-and-Spring Oscillators

By Newton's Second Law:  $F = ma = m \frac{dv}{dt} = m \frac{d^2x}{dt^2}$ System equation:  $m \frac{d^2x}{dt^2} + R \frac{dx}{dt} + kx = 0$ 

$$x = e^{-\alpha t} A \cos(\omega_d t + \phi).$$

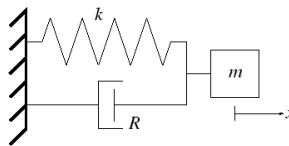


Figure 2: An ideal mass-spring-damper system.

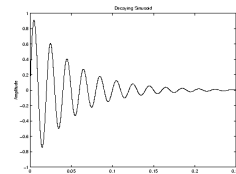


Figure 3: A decaying sinusoid.

- <http://cde.rpi.edu/MassSpring.html>

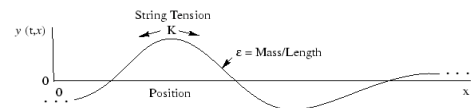
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

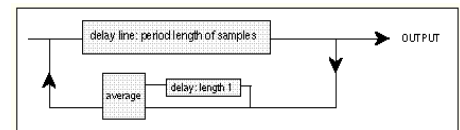
447

## Physics of the Ideal String

- String Physics



- Karplus-Strong Plucked String



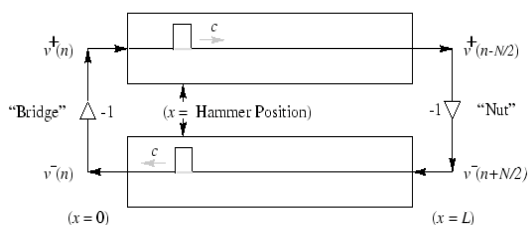
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

448

## Initial String State after Striking

- Traveling waves

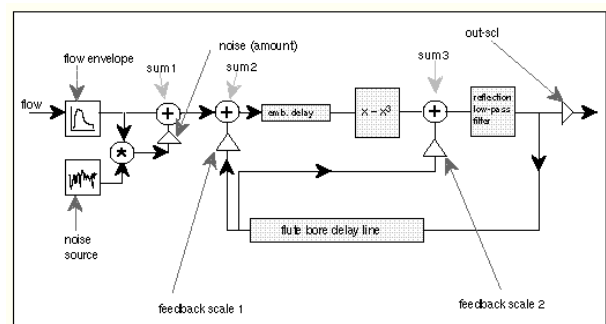


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

449

## PM Flute

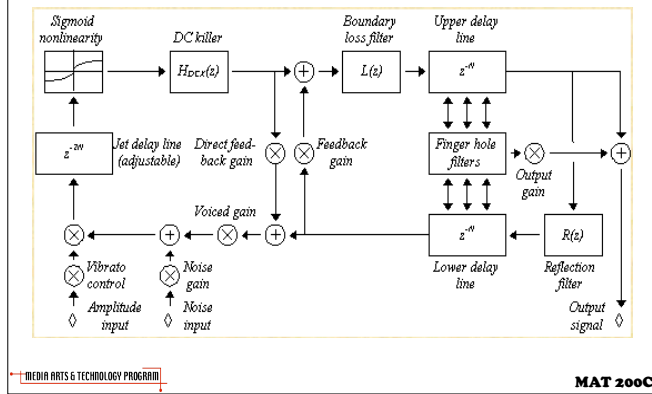


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

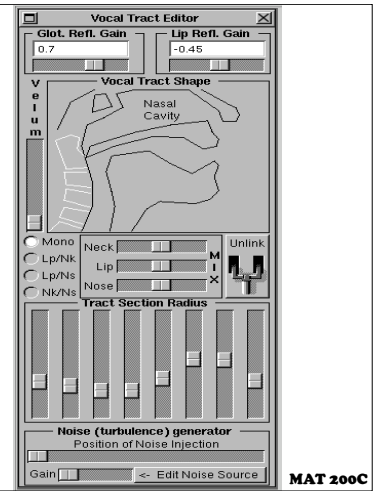
450

## Refined Waveguide Flute

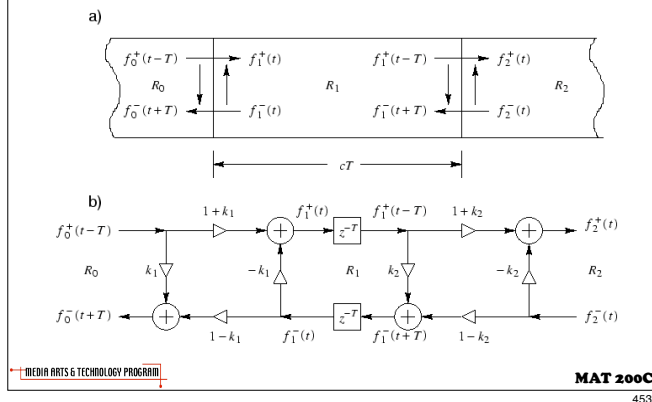


## PM Voice

- Vocal tract seen as a varying-cross-section tube driven by glottal pulses

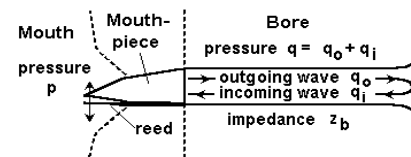


## Modeling Scattering



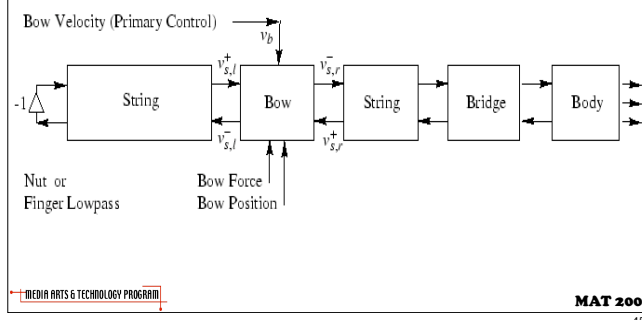
## Simple Clarinet

### Clarinet Model -- Pressures and Flows

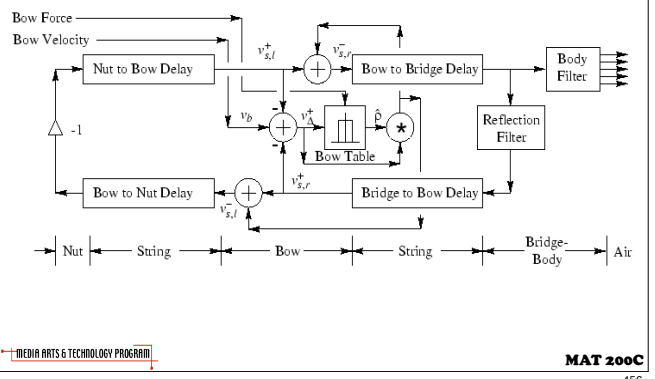


## Bowed String + Body

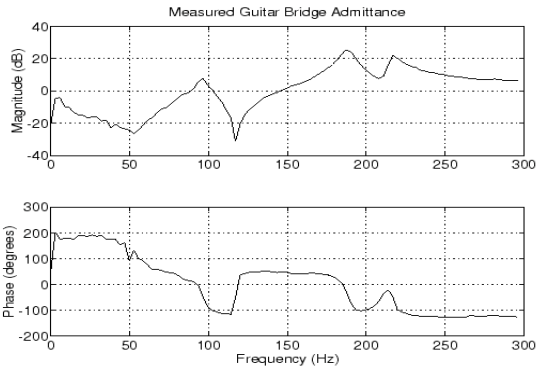
### • Source-filter model



## Bowed String Instrument Details



## Measurements of Physical Objects

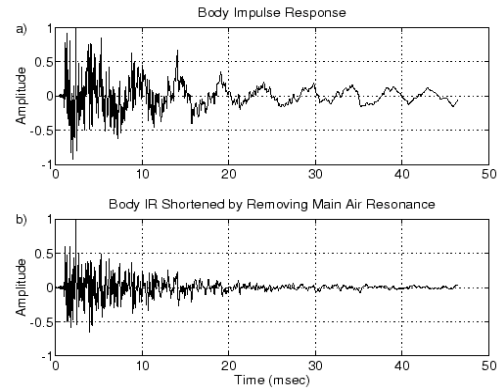


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

457

## Violin Body Impulse Response



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

458

## Other Physical Models

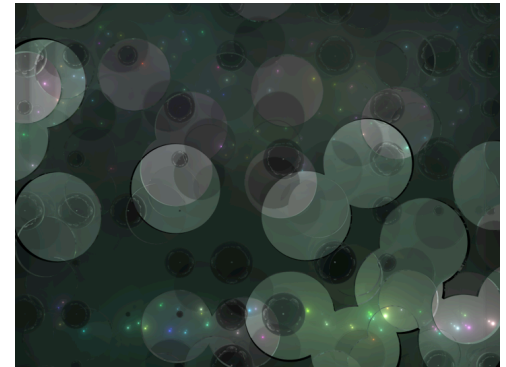
- Struck and bowed strings
- Coupled strings
- Other wind instruments
- Struck objects
- More on the way...
- PM in graphics (see below)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

459

## DASP



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

460

## DASP and Sound Effects

- Time-domain processing
- Frequency-domain processing
- Special techniques
- Nostalgic techniques

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

461

## Time-Domain Processing

- Delay
  - Tape effects
- Echo
  - Spring, plate reverb
- Reverb
- Spatialization
- Flanging

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

462

## Frequency-Domain Processing

- Filters
  - Wah-wah (swept band-pass filter)
  - Comb filters (phasing)
  - Resonators
- Vocoder
  - FFT, processing, IFFT

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

463

## Hybrid Techniques

- Chorus, doublers
- Spatial techniques
- DJ sound (scratching)
- Voice processing
  - De-essing
  - Voice removal
  - Karaoke

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

464

## Nostalgic Techniques

- Analog synth processing
  - Analog-sound filters
    - BWAP
  - Ring modulators
    - Multiplication, convolution
- Vintage instruments
  - Analog synthesizers
- Vintage processors
  - Analog vocoder, filter banks

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

465

## Spatial Sound

- Perception of sound source direction
- Perception of rooms and spaciousness
- Synthesis of spatial sound cues
- Synthesis of room acoustics

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

466

## Spatial Sound Processing

- Applications
  - Complex supervisory control systems such as telecommunications and air traffic control systems
  - Civil and military aircraft warning systems
  - Teleconferencing and telepresence applications
  - Virtual environments
  - Computer-user interfaces and auditory displays, especially those intended for use by the visually impaired
  - Arts and entertainment, especially video games and music

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

467

## How is Hearing Different from Vision?

- Multi-stream (i.e., chords or counterpoint)
- Multi-directional ( $2p * 2p * 2p$ )
- Sources have position, orientation, radiation pattern, etc.
- Timbre is as rich a space as visual shape/texture
- We're used to sounds changing in time

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

468

## Uses of Sound in “Immersive” Systems

- For more “real” VE/VR worlds
- For more channels of data
- For non-visual worlds (dark, blind)
- For VR-based artistic performance

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

469

## Auditory Display or Sonification

- Since the late 1970s, exploration into “audio rendering” of data
  - Chemical spectra as chords (Bly)
  - Population data as “events” (De Campo)
  - SonicFinder (Gaver)
  - See ICAD literature

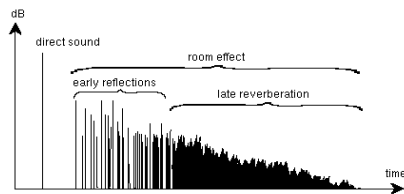
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

470

## Reverberation

- Reverberation gives us important cues as to the space we’re in, even if we don’t perceive it as “reverberant”



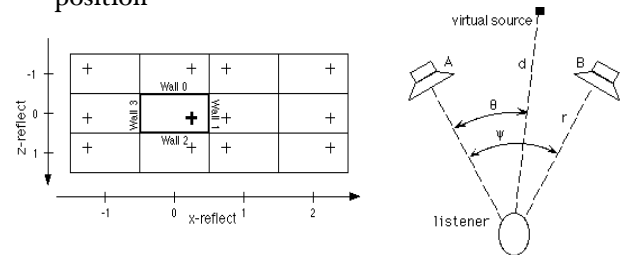
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

471

## Virtual Sources and Panning

- Each wall reflection implies a virtual source
- Panning tries to simulate a virtual source’s position

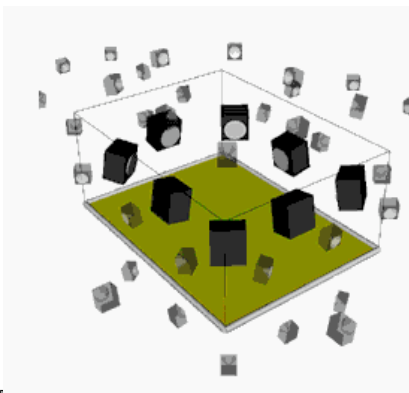


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

472

## Virtual Sources: Example



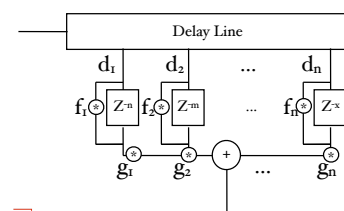
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

473

## Schroeder Reverberator (1963)

- Multi-tap delay line followed by a comb filter bank
  - $d_n$  = tap delay (early reflections)
  - $Z^n$  = comb delays,  $g_n$  = gain,  $f_n$  = feedback



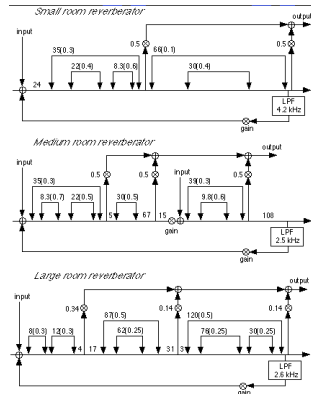
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

474

## Reverberator Architecture

- Early reflections: based on exact room geometry (acoustical ray tracing)
- Late reverb according to general characteristics



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

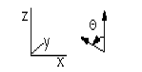
MAT 200C

475

## Mapping Geometry to Spatial Hearing Cues

### Geometry

(seen from the top)



(seen from the back)

Source  
Filter (medium)

Listener

### Spatial Cues

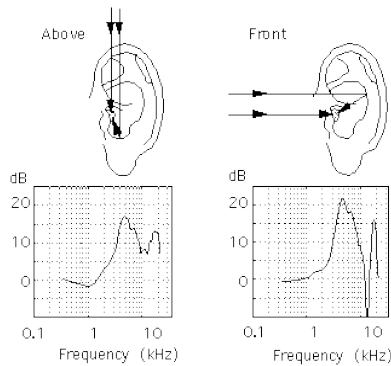
- Distance  $(d) = (b + a) / 2$
- Loudness  $\propto d^2$
- L/R ratio  $\propto (b - a) / e$
- Low-pass filter  $\propto d^2$
- Direct/reverb ratio  $\propto d^2$
- Spatial low-pass filter ratio  $\propto \theta$
- Spatial band-reject filter  $\propto \psi$
- Inter-aural time delay  $\propto (a - b) / e$
- Initial/decay reverb ratio  $\propto \psi$
- (many more possible)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

476

## Individual Differences and Pinna Geometry



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

477

## HRTF/HRIR Filters

- The Head-related Transfer Function (HRTF) is the 4-D acoustical filter of the shoulders, head, and pinna

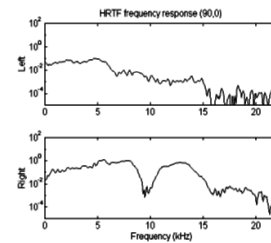


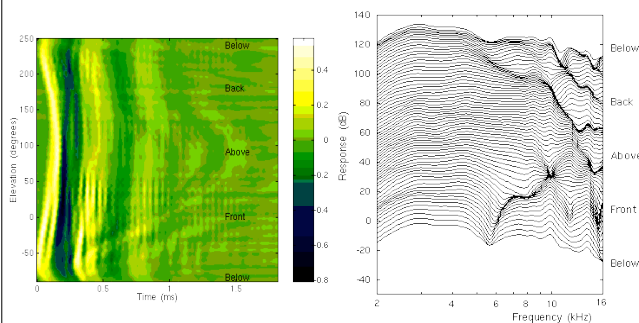
Figure 4.2: Example HRTF magnitude for 90° azimuth, 0° elevation from SDO set.

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

478

## HRTF in the Median Plane



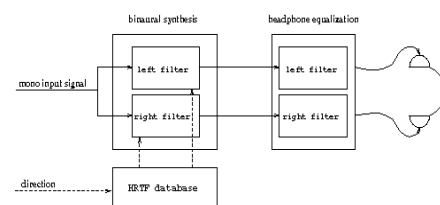
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

479

## HRTF Processing

- Given a measured HRTF, it's straightforward to apply it to a given sound source and relative position



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

480

## The Problem with HRTFs

- Mine don't sound good to you
- The average of mine and yours don't sound good to either one of us
- We have many years training with each of ours
- It's a 4-D function!
- But, if every user could have it measured...

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

481

## Spatial Sound Renderers

- Need model of the space
  - Walls and their characteristics
- Need position(s) of source(s)
  - Possibly moving
- Need position of listener's ears
  - Possibly moving
- Decide which cues to synthesize and how...

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

482

## VRML Sound Node Description

```

Sound {
 exposedField SFVec3f direction 0 0 1
 exposedField SFFloat intensity 1
 exposedField SFVec3f location 0 0 0
 exposedField SFFloat maxBack 1
 exposedField SFFloat maxFront 1
 exposedField SFFloat minBack 1
 exposedField SFFloat minFront 1
 exposedField SFFloat priority 0
 exposedField SFNode source NULL
 field SFBool spatialize TRUE
}

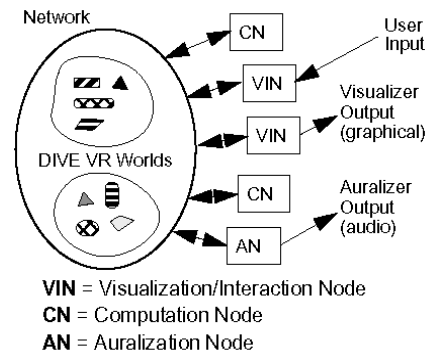
```

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

483

## DRIVE Architecture

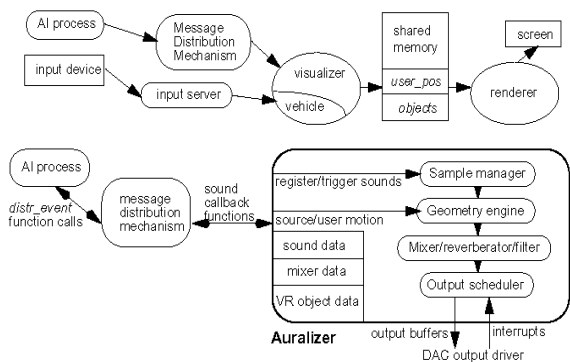


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

484

## DRIVE Flow and Aural Renderer

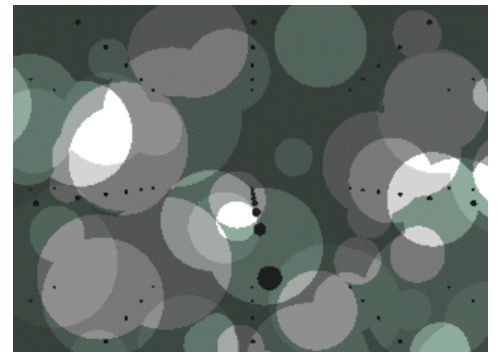


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

485

End of  
Part 4.1



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

486

## Topic 4: Digital Manipulation of Signals and Symbols

- Part 1: Sound Analysis/Synthesis and Processing
- **Part 2: Image and Video Analysis/Synthesis and Processing**

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

487

## Topic 4 Part 2

- Topics in Graphics
- Rendering
- Ray-Tracing

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

488

## Topics in Graphics Synthesis

- Rendering
  - Coordinates and Geometry
  - Frame buffers
  - Rendering Processes
- Ray-Tracing
  - Basic formulation
  - Reflection and refraction
  - Implementation

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

489

## Rendering

- Take a 2D or 3D display list (read as data structures or commands) and generate a frame buffer for a particular vantage point
  - Treat surfaces as polygons (triangles)
  - Transform coordinates to viewer's plane of view
  - Sort by z-plane relative to user perspective

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

490

## Transformation Math

Translation in space:

$$\text{Trans}(x,y,z) = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about the x-axis

$$\text{Rot}(x,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about the y-axis

$$\text{Rot}(y,\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about the z-axis

$$\text{Rot}(z,\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

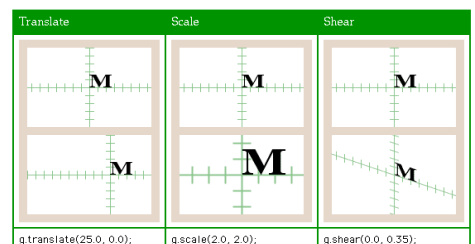
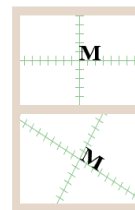
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

200C

491

## Transformations

```
Graphics2D g = (Graphics2D)graphics;
...
g.rotate(30.0 * Math.PI / 180.0);
g.setFont(new Font("Serif", Font.BOLD, 24));
g.drawString("M", 0, 0);
```



g.translate(25.0, 0.0);

g.scale(2.0, 2.0);

g.shear(0.0, 0.35);

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

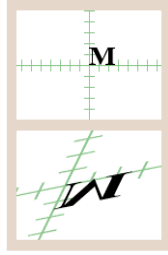
492

## Transformations

```
Graphics2D g = (Graphics2D)graphics;
```

```
g.scale(2.5, -1.5);
g.translate(-10.0, 0.0);
g.shear(0.5, 0.15);
g.rotate(10.0 * Math.PI / 180.0);
```

```
g.setFont(new Font("Serif", Font.BOLD, 24));
g.drawString("M", 0, 0);
```

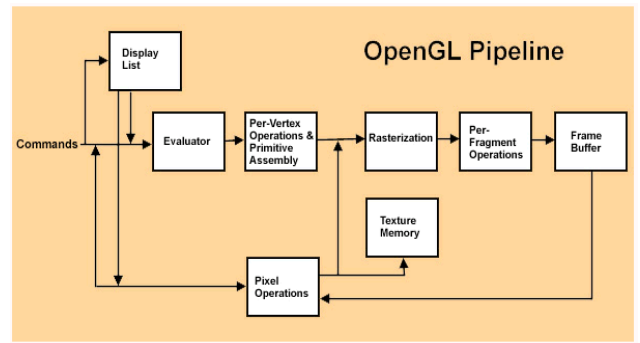


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

493

## The Rendering Pipeline



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

494

## Rendering Processes (1)

- Solid Modeling
  - Define solid models as collections of surfaces and combine them using the set operations intersection, union and difference.
- Trim Curves
  - Specify a subset of a parametric surface by giving a region in parameter space.
- Level of Detail
  - Specify several definitions of the same model and have one selected based on the estimated screen size of the model.
- Motion Blur
  - Process moving primitives and anti-alias them in time.

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

495

## Rendering Processes (2)

- Programmable Shading
  - Perform shading calculations using user-supplied Shading Language programs.
- Special Camera Projections
  - Perform nonstandard camera projections such as spherical or Omnimax projections.
- Deformations
  - Handle nonlinear transformations such as bends and twists.
- Spectral Colors
  - Calculate colors with an arbitrary number of spectral color

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

496

## Rendering Processes (3)

- Texture Mapping
  - Index a texture map with the surface's texture coordinates.
- Environment Mapping
  - Model the environmental illumination by indexing a texture map with a direction vector.
- Bump Mapping
  - Perturb just surface normals by giving a displacement map.
- Depth of Field
  - Simulate focusing at different depths.

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

497

## Rendering Processes (4)

- Volume Shading
  - Attach and evaluate volumetric shading procedures.
- Ray Tracing
  - Evaluate global illumination models using ray tracing.
- Radiosity
  - Evaluate global illumination models using radiosity.
- Area Light Sources
  - Illuminate surfaces with area light sources.
- Shadow Depth Mapping
  - Index a shadow map with a position.

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

498

## Frame Buffers

- Fragments and bitplanes
- Color buffer
- Depth buffer
- Stencil buffer
- Accumulation buffer
- Stereo and double-buffering
  - Front/back, left/right

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

499

## Texture-Mapping

- Provide color detail for intricate surfaces., e.g. woodgrain, by modifying the surface color
  - linear interpolation between four texel values
  - weighted sum of 4x4 array of texels
  - final fragment color – texture environment function
  - modulating the original fragment color with the texel color

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

500

## Textured Rendering Steps

- Generate white polygons
- Light them
- Use this lit value to modulate the texture image
- Texel selection/mixing
- Perform specular lighting after texturing

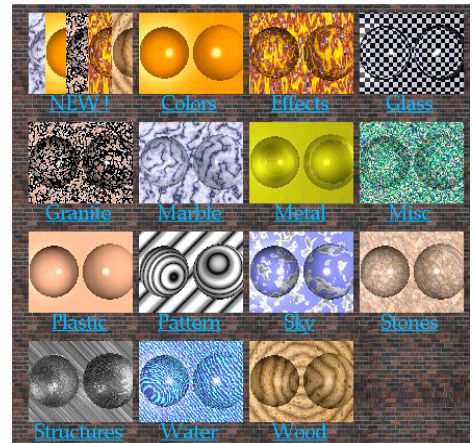
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

501

## Texture Examples

- Povray textures



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

502

## RenderMan Demo Shot

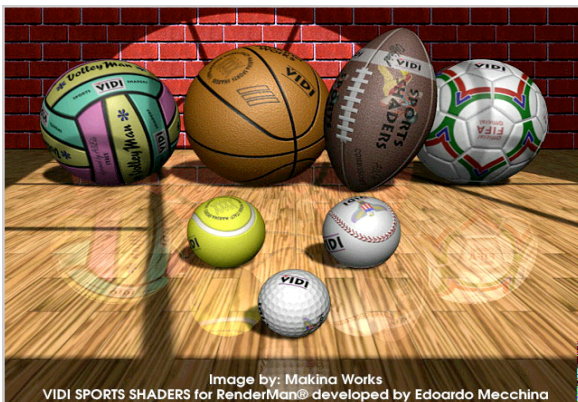


Image by: Makina Works  
VIDI SPORTS SHADERS for RenderMan® developed by Edoardo Mecchina

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

503

## DIVE Rendering



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

504

## Rendering Issues

- Frame storage
  - Fragmentation and copy buffers
- Texture storage
  - Accesses and block operations
- Update throughput
  - Pipeline-feeding

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

505

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

## Ray-Tracing



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

506

## Ray-Tracing

- Generating a screen by following light from sources to the viewer
  - Trace light rays from the user's eye through the display surface to objects in the scene, reflecting to light sources as appropriate
  - Pixel is a function of surface "texels"
  - Integrate illumination function over reflective function for each pixel
- Looks great, really inefficient

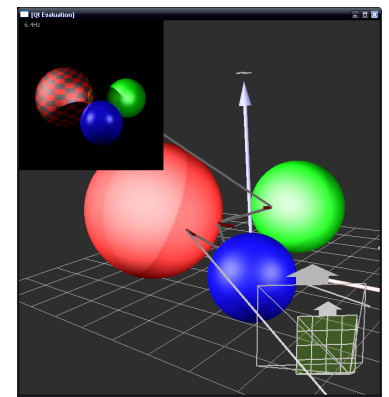
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

507

## Ray-tracing Example

- Follow light rays from source, bounces off of surfaces, then through screen plane to eye



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

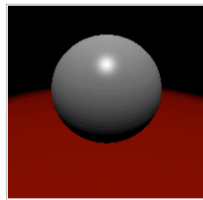
508

## RenderMan Example

```

LightSource "spotlight" 1 "from" 0 20 -10 "to" 0 0 0
 "intensity" 500 "coneangle" 0.250
AttributeBegin
 Surface "plastic"
 Color [1 1 1]
 Sphere 1 -1 1 360
AttributeEnd
AttributeBegin
 Translate 0 -1 0
 Surface "plastic"
 Color [1 0 0]
 Polygon "P" [-10 0 -10 -10 0 10 10 0 10 10 0 -10]
 "s" [0 0 1 1]
 "t" [0 1 1 0]
AttributeEnd

```

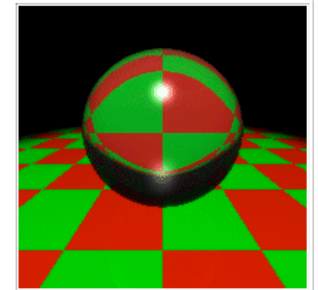
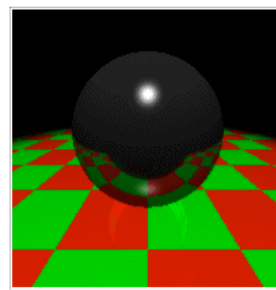


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

509

## Reflection and Refraction



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

510

## Reflection and Refraction

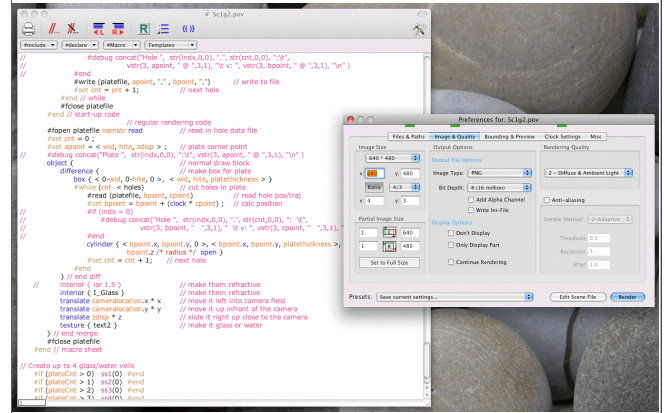


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

511

## POV-Ray Example



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

512

## Ray Tracing Issues

- Glossiness and fuzzy reflections
  - Spread reflective function
  - Gel reflection function
  - Reflection coefficient of textures
- Fuzzy translucency
  - Smoke and clouds
- Penumbras (soft shadows)
  - Math issues

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

513

## Ray Tracing Issues

- Depth of field and focus
  - Z-buffer and ray length
  - Motion blur
- Numerical issues (edge aliasing and oversampling)
  - High-resolution math
- Refraction and computer optics
  - Colored prisms

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

514

## Efficient Ray-Tracing

- Parallel transformations
  - Frame caches
- Physical models
- Distributed systems
  - Render-farms
- Instruction set support
  - MMX-style
  - VLIW
- OpenGL shading language

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

515

## Animation and Control

- Physical models of solid geometry
  - Laws of motion:  $x/y/z = kt$
  - Gravity and inter-object forces
  - Object extension, collision detection
- Control and motion capture
- Video processing and feature extraction, computer vision

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

516

## Exercises & Demos

- Audio synthesis systems, languages, tools
- Runnable physical models
- Spatial sound playback
- Modeling languages and rendering tools
- Ray-tracing systems (POV-ray)
- OpenGL

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

517

## Summary

- Synthesis of sound and image
  - Audio synthesis techniques
  - Physical Models
  - Spatial Sound
  - Rendering processes
  - Ray-tracing for image synthesis

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

518

## What's Next? (Topic 5)

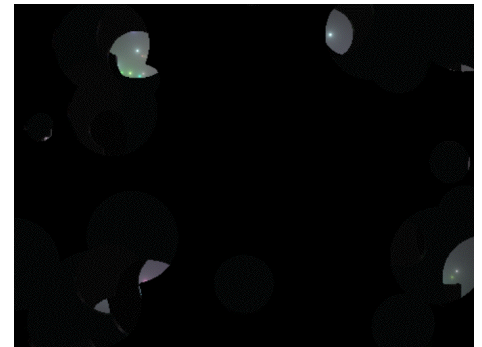
- Readings on Haptics and IO Devices
- Media I/O devices
  - Perception, psychology, and media
  - Issues of communication theory
  - Media data capture and performance
  - Transducers for signals, control variables, and events
  - Keyboard ergonomics
  - I/O Interface survey

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

519

End of  
Part 4



MEDIA ARTS & TECHNOLOGY PROGRAM

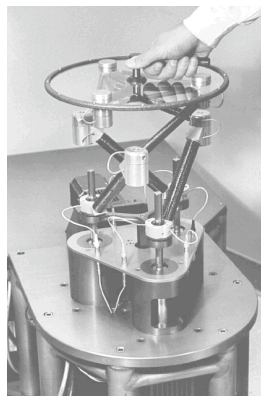
MAT 200C

520

## MAT 200C: Survey of Media Technology

### Topic 5: Haptics, Media I/O Devices and HCI

Stephen T. Pope  
MAT/UC Santa Barbara  
stp@mat.ucsb.edu  
Winter Quarter, 2008



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

521

## Topic 5: Media I/O Devices

- Controls, Triggers, and Signals
- Ergonomics and Haptic Input
- Visual I/O
- Sound I/O

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

522

## Topic 5 Readings

- Wikipedia entries for *Human interface device*, *Computer keyboard*, *Computer mouse*, *Game controller*, *Speech recognition*, and *Motion capture*
- J. Pressing. "Cybernetic Issues in Interactive Performance Systems" *CMJ 14:1*
- M. Kölsch, M. Turk. "Keyboards without Keyboards: A Survey of Virtual Keyboards" *Proc 2002 SIMS Workshop, UCSB*
- See also MAT 240E reader, presentation

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

523

## Domain Readings

- Buxton on haptics
- SIMS Proceedings
- SIGCHI Literature
- NIME Proceedings
- Code and tools

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

524

## Extra References

- Haptics and HCI references
- Sensors and controllers
- Capture and gesture feature extraction
- Text capture
- Multi-modal input
- Video I/O

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

525

## Signals and Controls

- Signal (continuous physical quantity)
- Control (over a model)
- Trigger (of an event or condition)
- Dimensionality of control functions
- Latency, Jitter (delay, buffering)

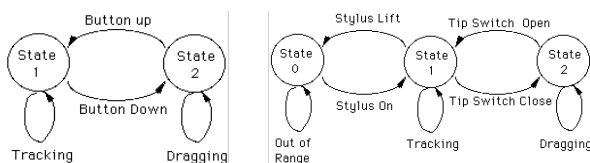
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

526

## State Machines and Control

- State conditions and transition actions (observed, actuated)



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

527

## Input Devices

- Human input to machines and tools
- Event and signal-like inputs
- Degrees of freedom (DOF)
  - 1 DOF: knob, slider, keyboard
  - 2 DOF: mouse, touch-sensitive keyboard
- Haptic input
  - Physical transducer, mapping range, feedback

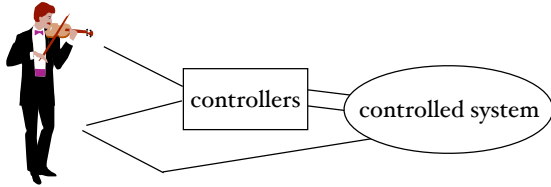
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

528

## The Feedback System

- Pressing Example
- Feedback loop



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

529

## Input Domains

- Triggers, selectors
  - Triggers and timing
  - Selectors and modes
  - State machines for mixed-mode inputs
- Continuous functions of 1 or more variables
  - Sampling/quantization
  - Range mapping and analysis

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

530

## Triggers, Selectors and Input

- Buttons, keyboards
  - Timing, sequence and mode
  - State machines for triggers
    - Control keys, chords
    - Multi-key sequences
- Audio/Video capture
  - Detection and feature extraction
  - Voice input
  - Video triggering
- UltraSound, video, etc.

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

531

## Continuous Function Inputs

- Position (3 DOF) and orientation (3 DOF) at any point
- Effort and force feedback (6 DOF)
- Range and value mapping (to model)
- More than 1 device – multiplicity of control
- Point, vector, and surface interfaces
  - Surface and vector field input

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

532

## Logical Tasks and Devices

- Gestures and device-independent logical input
- Generic Transactions
  - Select an object
  - Position an object in 1-3 dimensions
  - Orient an object in 1-3D
  - Ink: draw a line
  - Text input
  - Value specification

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

533

## Transducers and Sensors

- Electrical switches and potentiometers
- Magnetic sensors (Polhemus, Ascension)
- Electrical Fields (Paradiso)
- Sonar-based inputs (Hands, head-tracking)
- Optical inputs (camera-driven)
- Hall-effect transducers

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

534

## Input Issues

- Feedback and effort in haptic systems
  - Zero effort/feedback
  - Nominal/programmed
  - Physical model-derived
  - Coordination with other media
- Response and Learning
  - Latency and latency jitter

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

535

## The Physical Interface

- Hand/Arm – dexterity, speed, small range, Muscle group interfaces
- Foot/Leg – pedals, walking
- Neck/Head – head-tracking
- Eye tracking - FOV adaptation

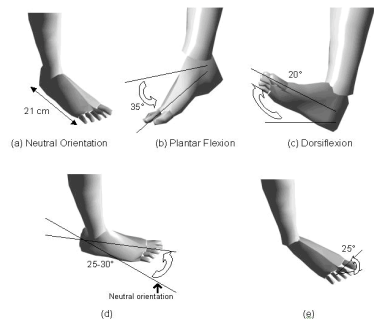
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

536

## Ergonomics of Humans

- Example: Ankle: DOF and limits of motion



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

537

## Survey

- A bit of history
- Text input
- Early tweezers and planar
- N DOF motor systems
- Robot arms
- 3 and 6 DOF pens
- Handles and joysticks
- New mice

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

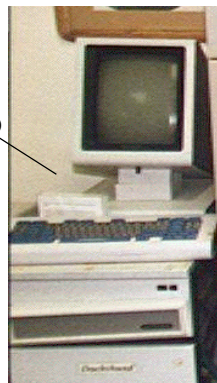
538

## Some History (1970s)



Xerox Alto (1973)

MIT LM-2 (1976)

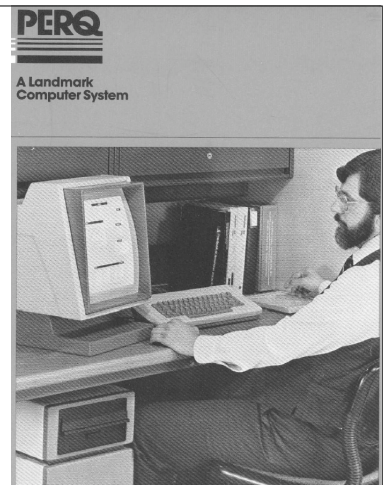


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

539

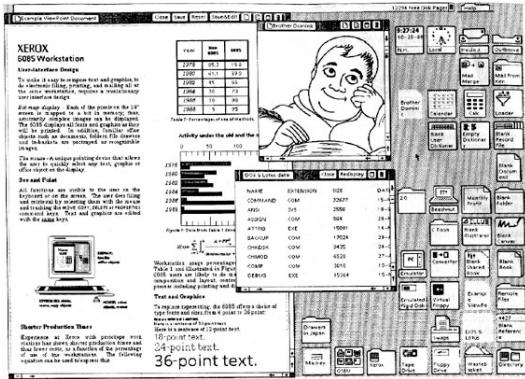
## Three Rivers Computer PERQ (1979)



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

540

## Xerox STAR Screen (1979)



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

541

## Early Mice



118

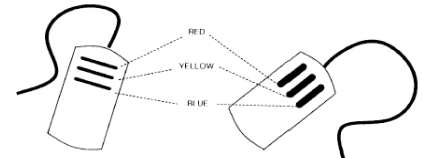
DRAW MANUAL

### Appendix A

#### Mouse and keyboard terminology

#### The mouse

The three mouse buttons on a mouse are labeled RED, YELLOW and BLUE. The physical arrangement of the buttons depends on the model:

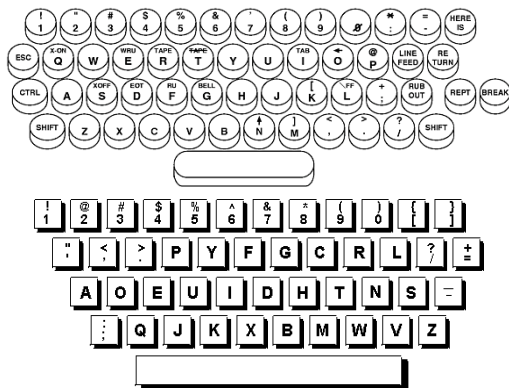


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

542

## Keyboard Layouts



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

543

## Symbolics Lisp Machine Keyboards



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

544

## TypeMatrix Keyboards



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

545

## TypeMatrix Demo



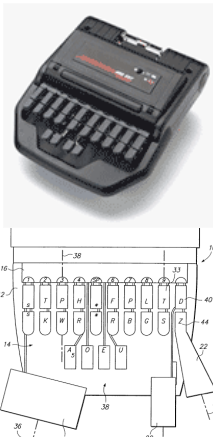
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

546

## Chording Keyboards

- Braille, stenographer's



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

547

## Chording Keyboard



- Velotype

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

548

## Alternative/Ergonomic Keyboards

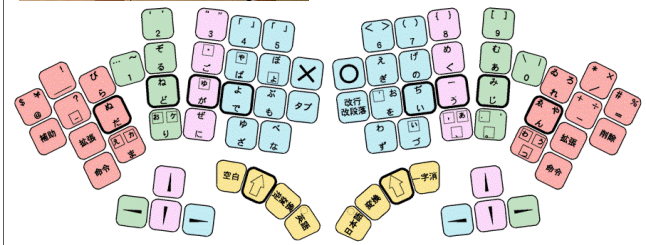


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

549

## TRON Project Keyboards



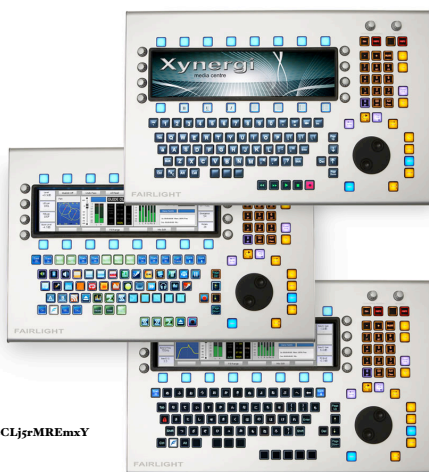
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

550

## Fairlight Xinergy Self-labelling Keyboard

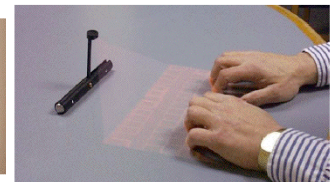
<http://www.youtube.com/watch?v=CLj5rMREmxy>



MEDIA ARTS & TECHNOLOGY PROGRAM

551

## Text Input Without Keyboards



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

552

## 2D Systems

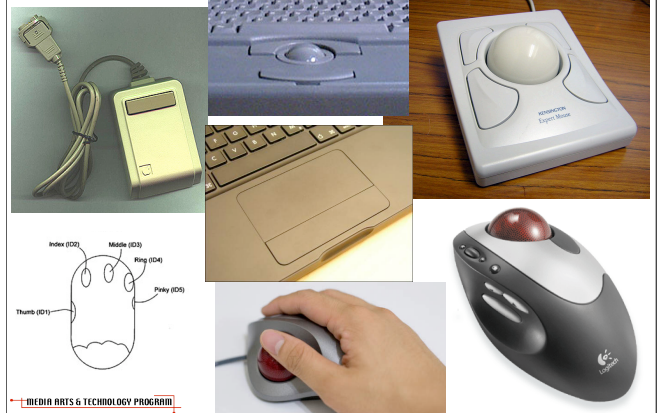
- Touch tablets and -screens
- Tablets, light-pens
- Joysticks + button
- Operations
  - Point, track
  - Drag, lasso, rubber band
  - Menus and modes
  - Character recognition

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

553

## Mice, Trackballs, Trackpads



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

554

## SpacePilot “3D Mouse” Video



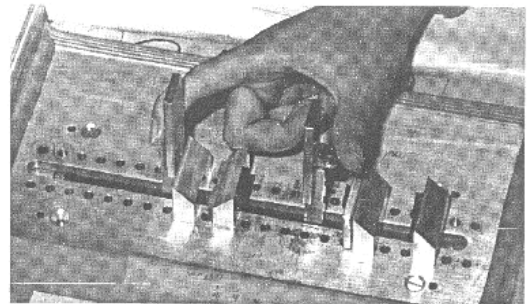
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

555

## Early 2D “tweezers”

- MIT Linear Grasper

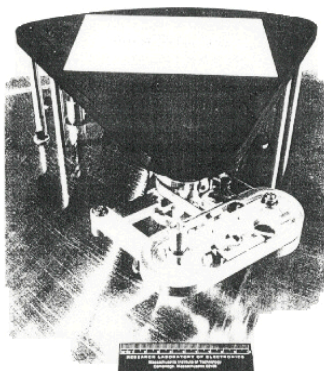


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

556

## 3D Planar Sensors/Graspers



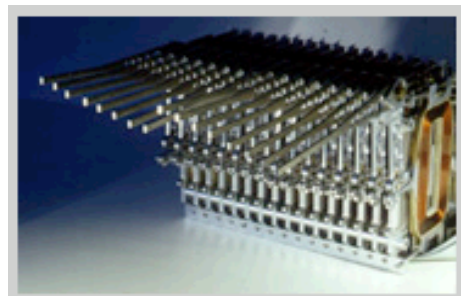
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

557

## ACROE Linear Motors

- Each arm its own position/force function
- Gears or pulleys to make 3 or 6 DOF devices

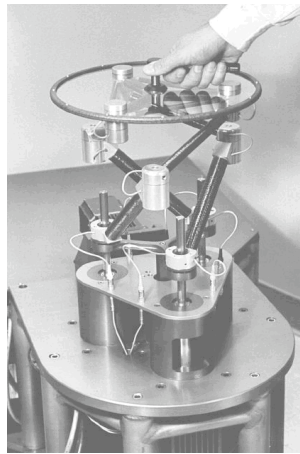


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

558

### NWU Planar 3DOF Handle



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

559

### Early 3/6DOF Arms



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

560

### SMU MasterArm

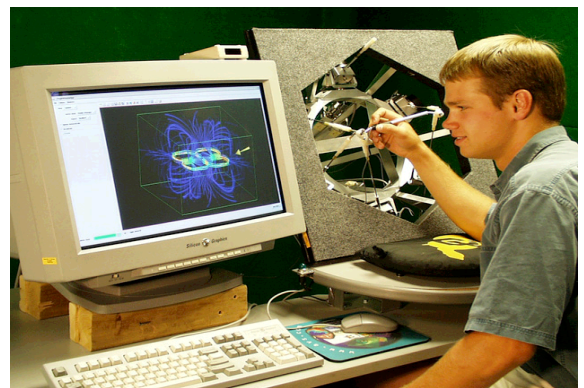


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

561

### Colorado 6DOF "Pen"

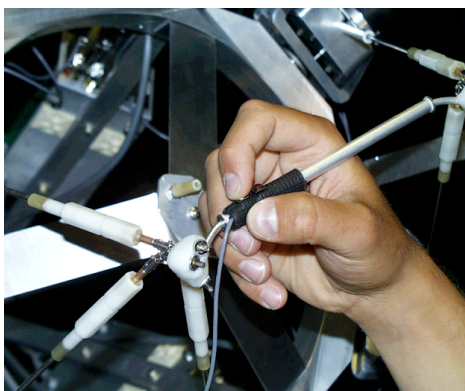


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

562

### 6DOF Pen Detail



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

563

### Sensible Phantom 6DOF Pens



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

564

## CMU 6DOF Handle



Magnetic Levitation  
Device Cabinet



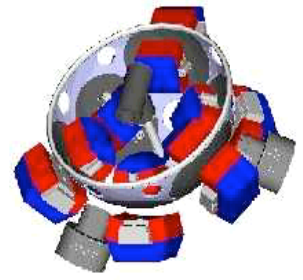
Hand Operation of Magnetic  
Levitation Haptic Device

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

565

## CMU Magnetics Details

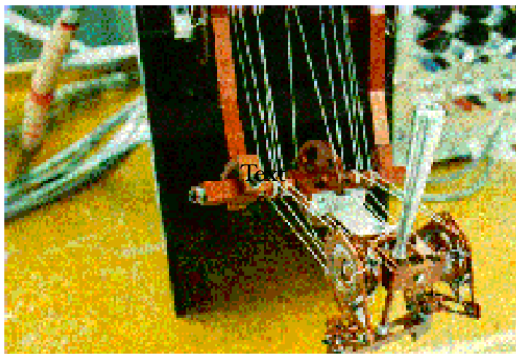


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

566

## Torque: Hayward's 7DOF Input



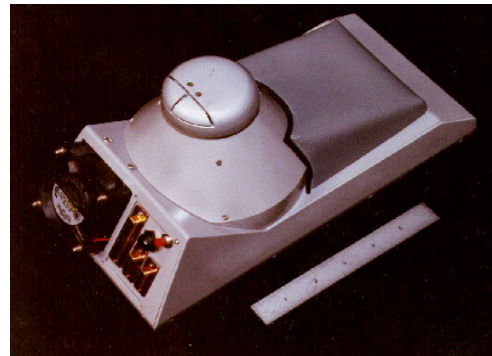
<http://www.cim.mcgill.ca/~haptic/devices.html>

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

567

## UBC MagLev PowerMouse



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

568

## UCB Mouse



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

569

## Rutgers 6DOF Pedal



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

570

## 3 Finger Tweezer Video



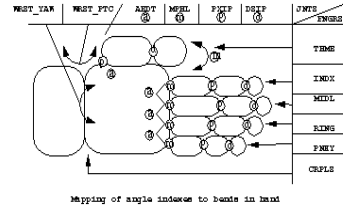
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

571

## Gloves

- Position/orientation of “hand” and orientation of fingers



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

572

## P5 Glove



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

573

## Hand Gesture-mapping

- Hand positions
- Arm motions
- Analogous gestures
- P5-osc code examples



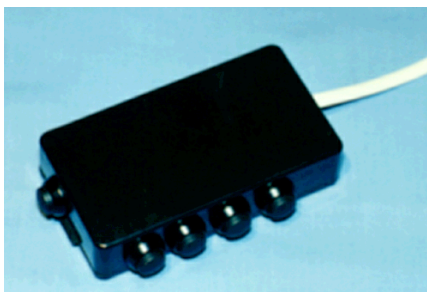
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

574

## 5DT Space Controller

- 6 DOF + triggers



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

575

## Game Controllers



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

576

## Examples (other)

- CMJ Slides
- ICMC Examples
- Pisa Interface Slides
- SIMS
- NIME
- SIGCHI

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

577

## Coding for Haptic Input

- Serial inputs (older gloves)
  - RS232 interface
  - Serial data packet protocol
- USB HID devices
  - Example: P5 glove-to-OSC convertor
  - Example: CREATE CUI
- OSC for input

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

578

## Visual I/O



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

579

## Visual I/O

- Input: cameras and video parsing
- Output: real-time rendering
- Presentation
  - Screen
    - Projection and r-r
    - Multi-screen (cockpit, CAVE)
  - HMD
- Immersive environments
  - User involvement, interactivity

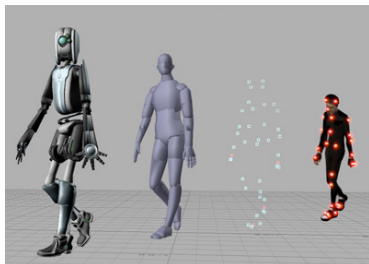
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

580

## Video Input

- Motion capture and tracking for animation

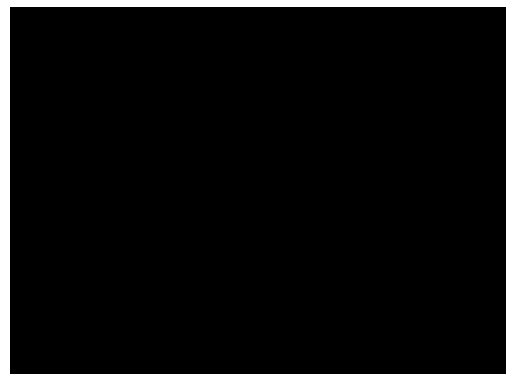


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

581

## Vicon Motion Capture

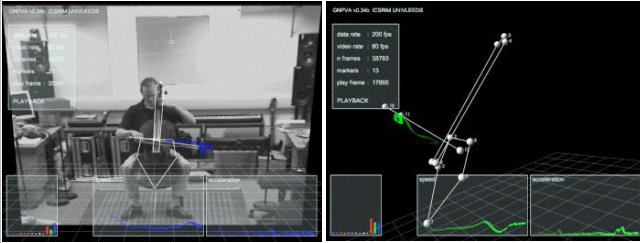


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

582

## iMaestro Motion Capture



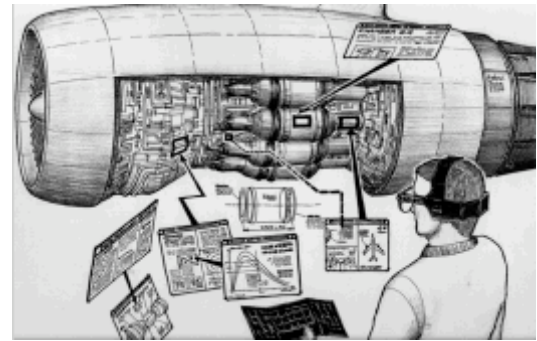
[http://www.i-maestro.org/contenuti/contenuto.php?contenuto\\_id=52&tool=gp](http://www.i-maestro.org/contenuti/contenuto.php?contenuto_id=52&tool=gp)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

583

## The VR/HMD Scenario

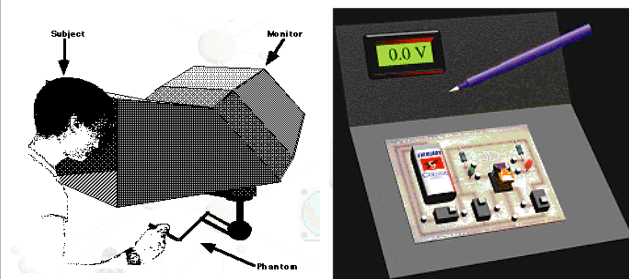


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

584

## MIT Virtual Workbench



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

585

## Early and Contemporary HMD



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

586

## Sony HMD Features



	LDI-100B	LDI-50B
LCD	0.7" 1.55M (832x3x624)	0.7" 180K (800 x 225)
H Angle of view	28 degree	30 degree
Virtual Image size	30 inch	52 inch
Focal length	1.2 m (4 feet)	2 m (7 feet)
See through mode	LCD shutter	Mechanical shutter
Diffuser	Yes	Yes
Image Adjustment	Brightness, Hue, Color	Brightness, Hue, Color
Headphone	Yes	Yes (Surround)
Input	SVGA, Y/C and VBS	VBS
Weight (Display)	120 g	160 g
Dimension (mm)	149x48x86	165x56x110
Power Consumption	12W	3.8W
Power Requirement	AC100 to 240V, 50/60Hz (DC 7.5 to 10V)	AC100 to 240V, 50/60Hz (DC 7.5 to 9V)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

587

## General Reality HMD, FakeSpace Boom

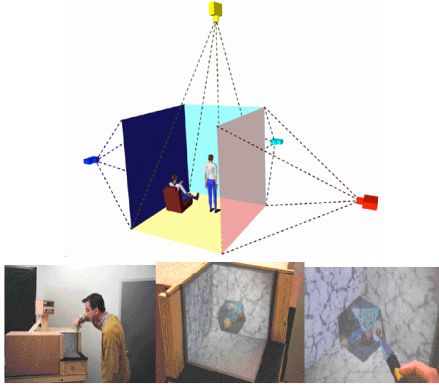


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

588

## The CAVE and the Cubby



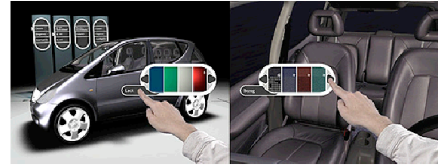
MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

589

## Art+Com Display

- Boom-mounted LCD monitor



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

590

## Head-Trackers

Ultra-sound and optical

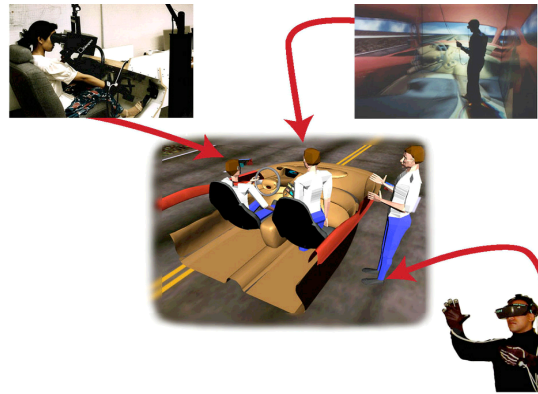


MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

591

## Multi-Modal Interfaces



MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

592

## HMD Issues

- Initial/final adjustment time
- Screen parameters (FOV, binocular)
- Panning and head motion
- Latency and latency jitter
- Transparency of live-video mix-in
- HMD-induced vertigo and illness

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

593

## Sound I/O

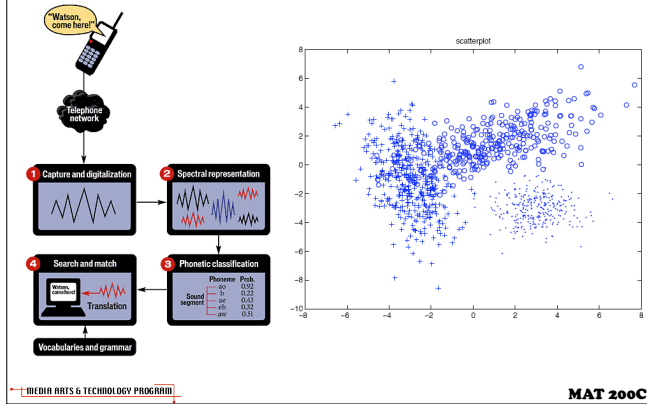
- See above...
- Immersive sound environments and media
- Play-through and real-world sound
- 3D Sound via headphones vs loudspeakers
- Data/environment sonification

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

594

## Voice Input: Speech Recognition



## Sound Input

- Simple mono or stereo sound field
- Spatial microphone arrays and source positioning
- Linking sound and visual input and output
- Sound as trigger, sound as control
- DIVE architecture: vehicles and renderers

## Exercises

- MAT sensor course, lab, group; Dan Overholt
- OSC programming examples
- P5 glove
- Kaiser HMD
- CNSI Allosphere

## Summary

- Controls, Triggers, and Signals
- Ergonomics and Haptic Input
- Visual I/O
- Sound I/O

## What's Next? (Topic 6 References)

- Applications of media technology in arts, entertainment, and education
  - Composition, arrangement, production, data management
  - Interactive hardware/software tools, instruments, and installations
  - Media content and distribution/dissemination
  - New (digital/distributed/interactive) media
  - Virtual environments, telepresence, and remote collaboration
  - Media data in education

End of  
Part 5



## MEDIA ARTS & TECHNOLOGY PROGRAM

### MAT 200C: Survey of Media Technology

#### Topic 6: Applications

Stephen T. Pope  
MAT/UC Santa Barbara  
stp@mat.ucsb.edu  
Winter Quarter, 2008



MAT 200C

601

### Topic 6: Applications

- Applications of media technology in arts, entertainment, and education
  - Composition, arrangement, production, data mgmnt
  - Interactive HW/SW tools, instruments, installations
  - Media content and distribution/dissemination
  - Virtual worlds, telepresence, and remote collaboration
  - Media data in education and edutainment
  - New (digital/distributed/interactive) media
  - New applications (networks, simulations, etc.)

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

602

### Topic 6 Readings

- Wikipedia entries for *Scientific visualization*, *Hypertext*, *HTML editor*, *Graphic art software*, *Digital audio workstation*, *Collaborative software*, *Edutainment*, and *MMORPG*
- TOC from baddesigns.com
- TOC from B. Schneidman's 1993 *Encyclopedia of Virtual Environments* (HITL @ UW)
- G. Wang, A. Misra and P. R. Cook. "Building Collaborative Graphical Interfaces in the Audicle." *Proc NIME 2006*

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

603

### Content Production

- Composition, arrangement, editing, post-production, data management
- Content generation and editing
- Content data management and interchange
- Models in production suite software
- Music notation
- Graphics models and animation suites

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

604

### Tasks & Process

- Process model (e.g.,)
  - Capture
  - Logging
  - Sync
  - Mix/merge
  - Postproduction, mastering
  - Encoding/compression
  - Archiving
  - Publication, distribution

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

605

### MM Data Models

- Content data models
- Synthesis models
- System state, response, capture mappings
- User model, goals, character
- Application state, narrative, dramaturgy
- Structured content: narrative, musical form
- Metadata, all the metadata

MEDIA ARTS & TECHNOLOGY PROGRAM

MAT 200C

606

## Application Survey

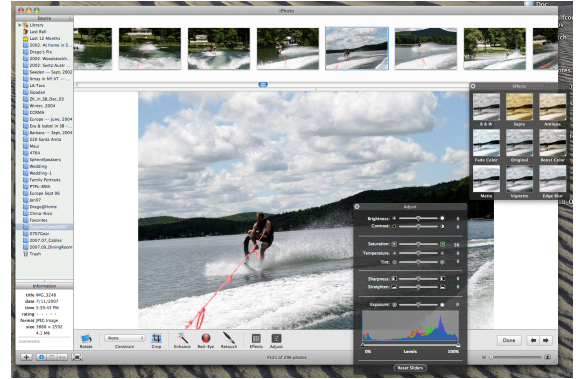
- Medium
- Task
- Process stage
- I/O & storage formats
- Application model
- User model

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

607

## Application: iPhoto Image Mgmt



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

608

## MODE (1992): Signals & Symbols

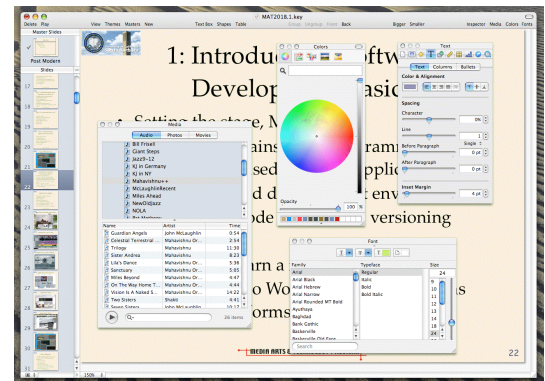


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

609

## Application: Keynote Authoring

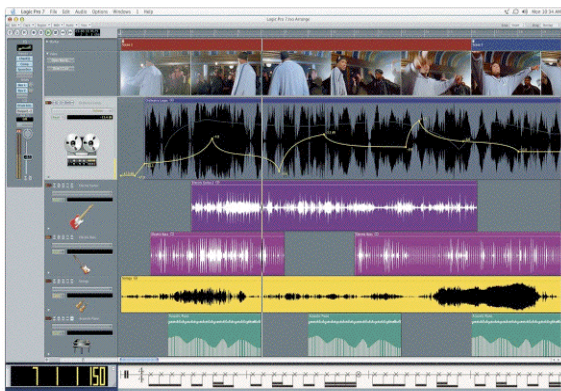


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

610

## Logic Audio: MM Composition



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

611

## Logic Sculpture: Synth. Model



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

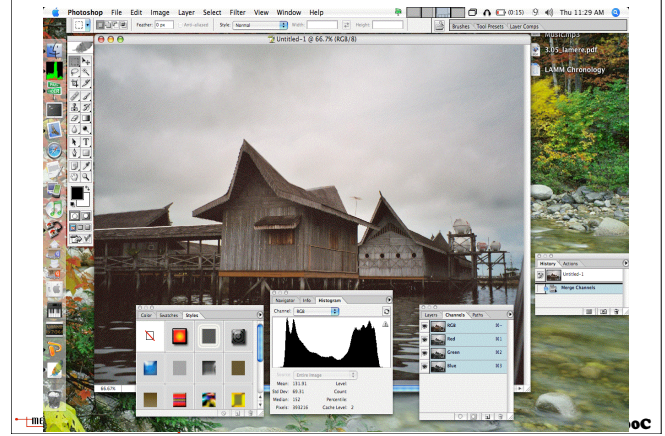
612

## Native Instr. Kontakt: Content Sample DB



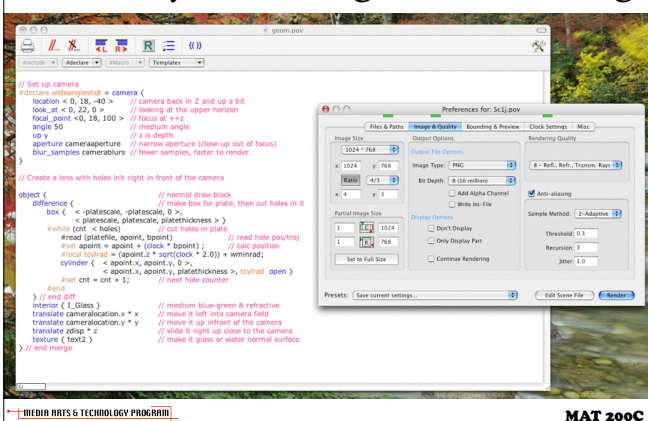
613

## Adobe Photoshop: Image Manipulation



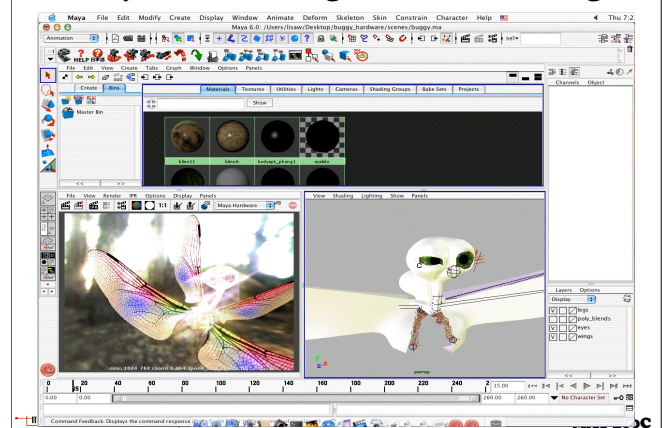
614

## POV-Ray: Modeling and Rendering



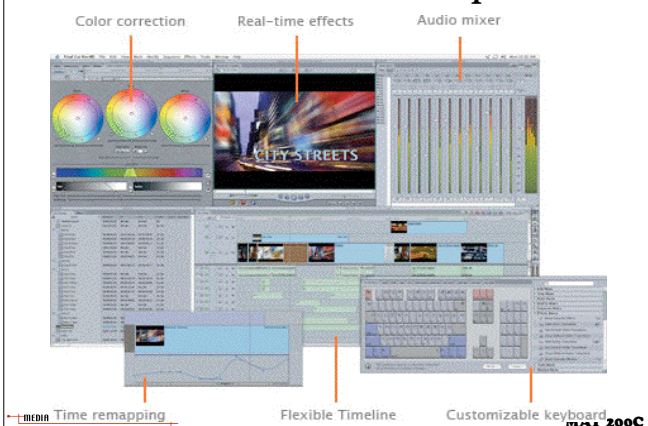
615

## Maya: Modeling and Rendering



616

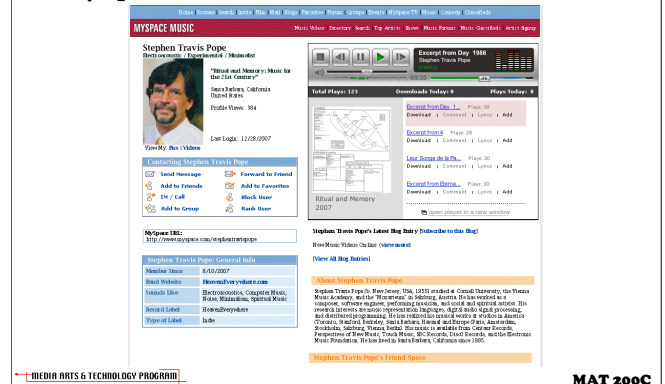
## Final Cut Pro: MM Composition



617

## MM in Social Networks

- MySpace, Facebook



618

## Content on Demand Services

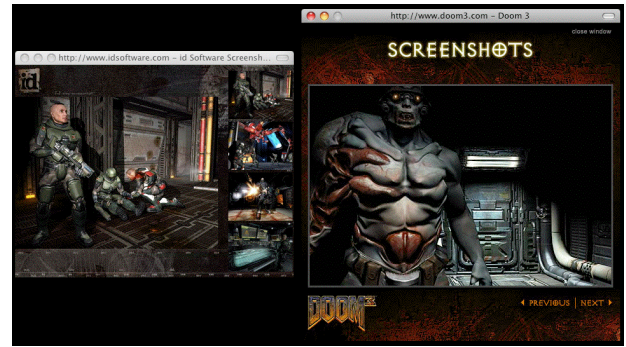


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

619

## Games: Quake & Doom

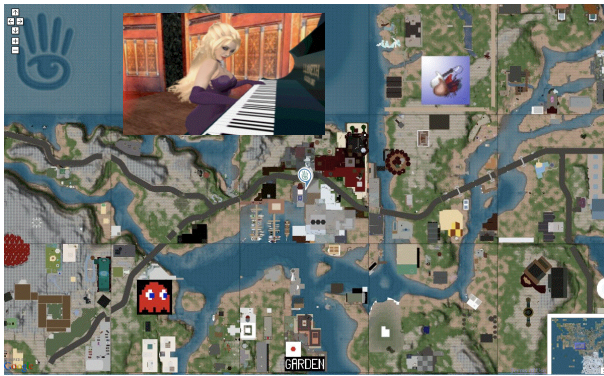


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

620

## Simulations: SecondLife



- <http://youtube.com/watch?v=i6D9I36zsec&feature=related>

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

621

## Character Animation: The Evolution of L. C.



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

622

## Multi-player Games

- MMORPGs



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

623

## EverQuest 2 Character

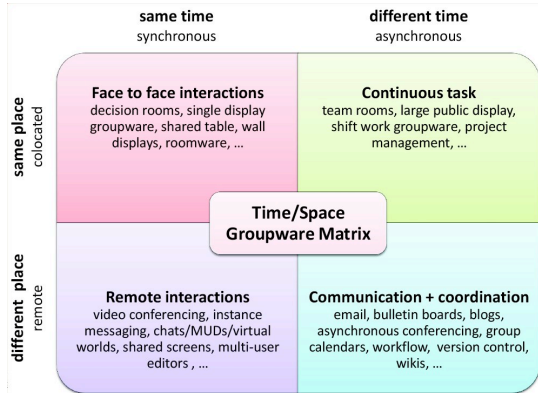


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

624

## Remote Collaboration: CSCW



## Other Applications

- Content creation
- Model-building
- “Composition”
- Sequencing, time flow
- Postproduction
- Data management
- Encoding for distribution
- Other process/task models

## Content Distribution

- Content and distribution/dissemination
- Delivering silver disks
- Web-based down-load options
- Streaming players (& their protocols)
- New web services for packaged content
- Search, databases and playlist management
- Delivery of metadata (cover art, DRM)
- Commercial models (napster, u2be, web radio)

## Content Delivery Platforms

- Silver disc player
  - Home theater, car
- Personal media player, phone, PDA
- Game device
- Web site
  - Streaming content
  - Playlist generation
  - On-line community
- Web application delivery
- LAN-gaming center

## Tools and Instruments

- Interactive hardware/software tools for media content
- Role of precomposition
- Role of interactivity
- Tools vs. instruments
- Performance vs. installation
- New categories of tool, teacher, instrument

## Next Media

- Digital/Distributed/Interactive
- Services: Pandora.com
- Communities: digg.com
- Delivery vehicles: iPod/phone/PDA/camera
- Commercial models: craigslist, virtual money
- Multi “player” software
- Remote communication: iChat, skype, netMeeting, GoToMyPC
- Merged media: video chat, edutainment, infomercial games

## Immersive Interfaces

- Immersive UIs and applications
- Virtual environments and simulation
- Augmented reality and half-transparent UIs
- Telepresence and remote UIs
- Remote collaboration applications
- HCI for immersion
- Ubiquitous computing

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

631

## Media in Education

- (Static/dynamic) content delivery
- Testing and test data management
- Integrated HW/SW systems (piano, drawing)
- Remote access to content, expertise and test data
- Merger with games, entertainment content, communication, etc.

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

632

## The Next Step: Trends

- Media categories and merged media
- Application domains
- Delivery/interaction devices
- Online services and communities
- WAN capacity swamping local compute power
- I/O devices: haptics and display
- Programming/content creation paradigms

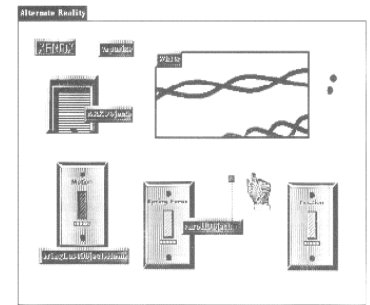
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

633

## Novel Application Models: ARK

- Alternate Reality Kit (ARK) (Smith & Ungar, 1986)
- Direct manipulation of software simulation
- Visual programming language
- Multi-user virtual world



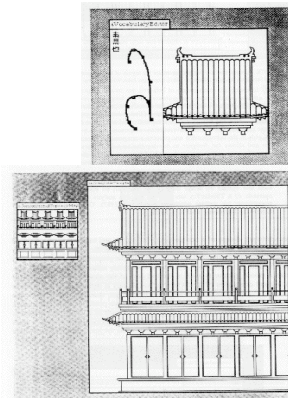
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

634

## Novel Application: CDTK

- Chinese Temple Design Toolkit (Makkuni, 1987)
- Map gestures to features of vocabulary elements
- Design grammars for combination of elements
- Manage libraries of refinement processes



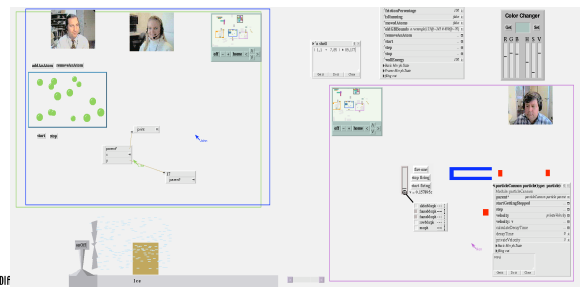
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

635

## Novel Application: Self/Kansas

- OO development environment with no tools or windows, "only objects on the screen"
- Multi-user shared workspace



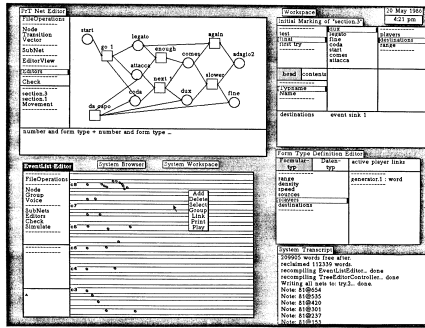
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

636

## Novel Application: DoubleTalk (1986)

- Use logic-marked Petri nets for music composition
- Net and token editors
- Checkers and simulators

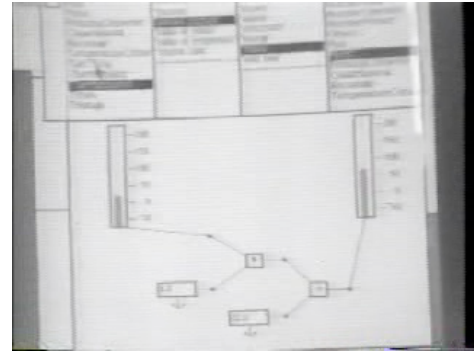


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

637

## Novel Application: ThingLab (1978)

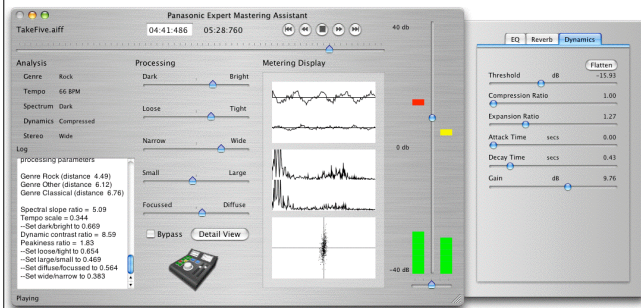


MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

638

## Novel Application: EMA (2003)



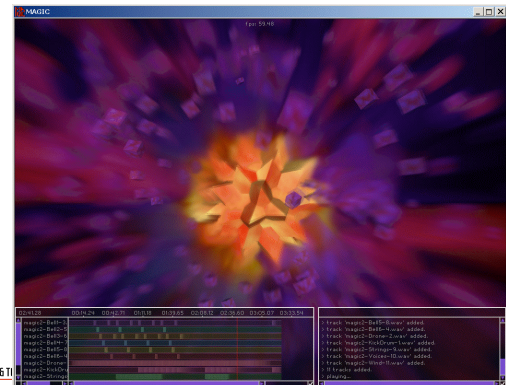
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

639

## Novel Application: MAGIC (2006)

- Eric Newman's "Synaesthetic Editor"



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

640

## Brain-storming New Models

- Applications
  - Sound: how to get away from tape recorder, score, cue sheet, patch bay models?
  - Image: the same?
  - Animation: is 2ndLife it?
  - Programming: where's the scalable visual programming language?
- Widgets
  - MVC for new models of state/interaction
  - Generic apps as high-level widgets

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

641

## The Computer Revolution...

*hasn't happened yet!*

- Early prototypes
  - Memex (Bush, 1948: HyperText)
  - Augment/NLS (Engelbart, 1969: CSCW + HyperText)
  - Dynabook (Kay et al, 1970s: "personal dynamic media" in Smalltalk)
  - Where are they now?
    - <http://sloan.stanford.edu/mousesite/1968Demo.html>
    - <http://www.educause.edu/ir/library/ra/COM9802.ram>
    - <http://video.google.com/videoplay?docid=2950949730059754521>
    - <http://video.google.com/videoplay?docid=5776880551404953752>

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

MAT 200C

642



# Course Announcement

MEDIA ARTS & TECHNOLOGY PROGRAM

## MAT 201A Media Signal Processing (Fall, 2009)

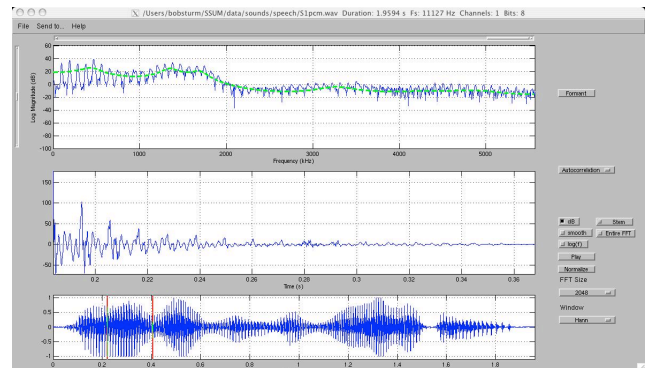
### Overview

The MAT 201A course is dedicated to introducing students to the multimedia digital signal processing methods. We will explore a range of topics from theoretical principals to practical considerations in multimedia (speech, audio, still images, and video) signal representation, synthesis, analysis and processing. The course assignments will consist of reading and homework tasks where students explore the concepts introduced in the lectures through concrete software applications using the MATLAB programming language and custom-developed SSUM framework.

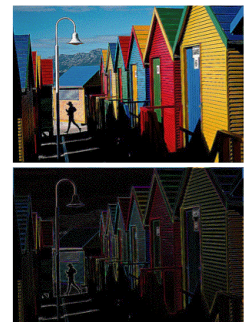
The premise of MAT 201A is that the multimedia (MM) expert of the future will need to have a firm grasp of the mathematical foundations of MM data representation and to understand (and even be able to extend) the algorithms used on common MM signal synthesis/processing/analysis applications.

### Outline

- Multimedia applications tour
- Data representations
- Operations on signals
- Information theory of signals
- Transforms and mappings
- Techniques in applications
- Projects



$$B_{\text{Edge Enhance}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & -1 & 9 & -1 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$D_{\text{Edge Enhance}} = 1$$
$$B_{\text{Find Edges}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & -1 & 8 & -1 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$D_{\text{Find Edges}} = 1$$



### Prerequisites

Students are expected to have graduate standing (or the instructor's approval), experience with trigonometry, complex numbers, and algebra, and to be functionally proficient in some programming or scripting language. The majority of the course grade will be based upon a final project report, presentation, and demonstration, in which the students explore one of the lecture topics in the form of a multimedia software application in MATLAB.

### Course Materials

The class text will consist of selected readings from the signal processing and multimedia literature; the course web site includes many links as well as down-loadable example software in MATLAB. Students are expected to purchase a personal license to MATLAB.

### Instructor

Stephen T. Pope (stephen@mat.ucsb.edu)

### Meeting Time and Place

Tuesday/Thursday, 3:00 - 5:00 PM

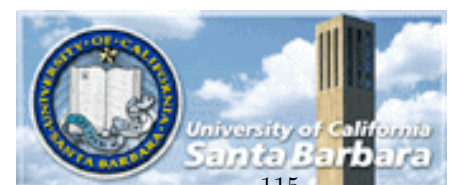
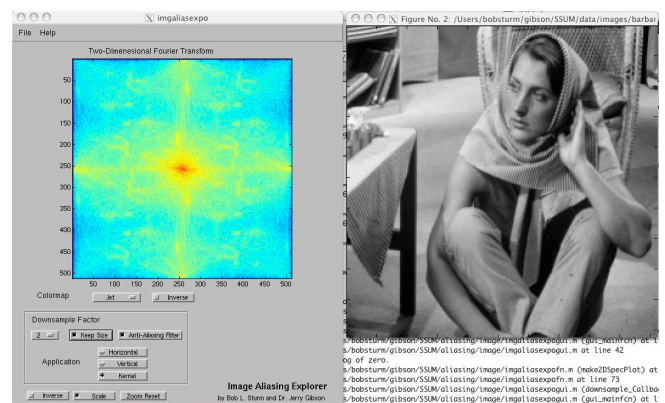
CREATE class room, Music 2215

First meeting: Thursday September 24, 2009

### Course Web Site

The course site includes required readings and mailing list info.

<http://www.mat.ucsb.edu/201A>





# MAT 201A Course Outline

1. Multimedia applications tour
2. Data representations
3. Operations on signals
4. Information theory of signals
5. Transforms and mappings
6. Techniques in applications
7. Projects

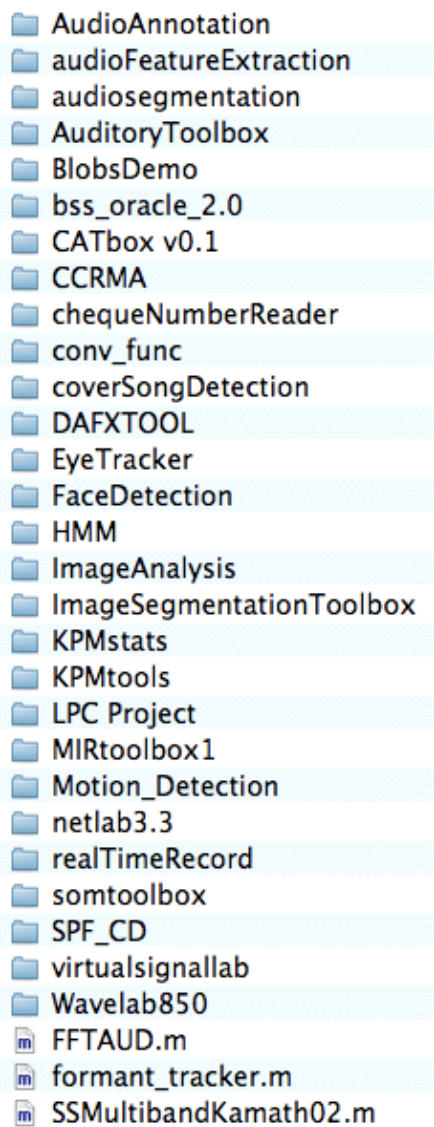
## Detailed Outline

1. Multimedia applications tour
  - Image, video, sound
  - Capture/logging, archiving, synthesis, transcoding
  - How are apps written?
2. Math & MATLAB introduction
3. Data representations
  - Image, sound data structures
  - Signal sampling and quantization
  - Signal magnitude and phase
  - Other properties
4. Operations on signals
  - Signal creation and manipulation
  - Signal arithmetic and statistics
  - Signal-processing functions and flow-charts
  - Memory, difference equations and digital filters
  - Autoregression and moving-average processes
5. Information theory of signals
  - Applications
  - Error correction
  - Compression
  - Data embedding and hiding
6. Transforms and mappings
  - Time- and frequency-domain signals
  - Spectral representations, Fourier transform
  - Linear prediction, signal modeling by polynomials
7. Techniques in applications
  - Image edge/object/patch detection
  - Motion estimation in video
  - Feature extraction from speech and music
  - Gesture capture and analysis
8. Projects
  - SSUM examples throughout
  - Image processing, feature extraction
  - Video processing, computer vision
  - Audio feature extraction, onset detection
  - Gestural input and gesture detection
  - Capture, archival, distribution, transcoding

## Course Schedule

Week 1 (9/29)	Introduction; Multimedia applications tour; Math & MATLAB introduction
Week 2 (10/6)	Data representations
Week 3 (10/13)	Operations on signals
Week 4 (10/20)	Operations on signals
Week 5 (10/27)	Information theory of signals
Week 6 (11/3)	Transforms and mappings
Week 7 (11/10)	Transforms and mappings
Week 8 (11/17)	Techniques in applications
Week 9 (11/24)	Techniques in applications
Week 10 (12/1)	Projects

## MAT 201A Code Archive



# MAT 201A Readings

*Why should an artist study signal processing?* Curtis Roads

*Why DSP?* Bob Sturm

*A Geometric Review of Linear Algebra.* Eero Simoncelli

*Lab 01: Introduction to MATLAB.* J. McClellan, GaTech

*Little Bits of MATLAB.* J. McClellan, GaTech

*Index of SSUM Examples* (Appendix to dissertation), Bob Sturm

MAT 201A homework assignments of years past

MAT 201A 2006 Lab Descriptions

DSP Tutorial TOC

How do I Learn DSP?

Bores DSP Tutorial

*Image Formation* chapter from *Computer Vision* by Richard Szeliski

*Lode's Computer Graphics Tutorial: Image Filtering*

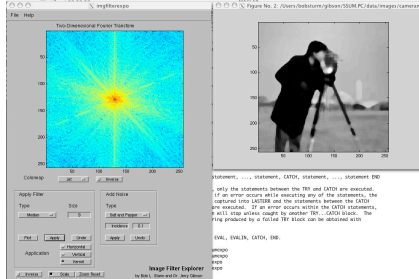
*Fundamentals of DSP for Music Analysis*

*MIR in MATLAB.* O. Lartillot & P Toivainen

# Media Signal Processing

## UCSB MAT 201A: Media Signal Processing (Fall, 2009)

Stephen Travis Pope,  
Graduate Program in Media  
Arts and Technology,  
University of California,  
Santa Barbara  
stp@mat.ucsb.edu



1

2

## MAT 201A Topics

- ◆ Multimedia applications tour
- ◆ Data representations
- ◆ Operations on signals
- ◆ Information theory of signals
- ◆ Transforms and mappings
- ◆ Techniques in applications
- ◆ Projects

3

## Logistics

- ◆ Course Meeting
  - ◆ Lectures: Tu/Th 3:00 - 5:00 PM, Music 2215
  - ◆ Lab: TBD
  - ◆ Instructor: Stephen Pope (stp@mat.ucsb.edu)
  - ◆ TA: Javier Villegas (jvillegas@umail.ucsb.edu)
- ◆ Grading
  - ◆ Attendance, participation
  - ◆ Homework?
  - ◆ In-class presentations?
  - ◆ Final project

4

## Course Materials

- ◆ MATLAB software
- ◆ SSUM software (on MAT 201A web site)
- ◆ Other MATLAB example code
- ◆ Book: *Signal Processing First* (McClellan, Schafer & Yoder) (or other introductory DSP text)
- ◆ Presentation slides
- ◆ Web site & readings: <http://www.mat.ucsb.edu/201A>
- ◆ Course mailing list

5

## Why MAT 201A?

- ◆ **Course objective**
  - ◆ The multimedia (MM) expert of the future will need to have a firm grasp of the mathematical foundations of MM data representation and to understand (and even be able to extend) the algorithms used in common MM signal synthesis/processing/analysis applications.

6

# Topical Overview

- ◆ Multimedia applications tour
- ◆ Data representations
- ◆ Operations on signals
- ◆ Information theory of signals
- ◆ Transforms and mappings
- ◆ Techniques in applications
- ◆ Projects

7

# Outline 1

- ◆ 1. Multimedia applications tour
  - ◆ Image, video, sound
  - ◆ Capture/logging, archiving, synthesis, transcoding
  - ◆ How are apps written?
- ◆ 2. Math & MATLAB introduction
- ◆ 3. Data representations
  - ◆ Image, sound data structures
  - ◆ Signal sampling and quantization
  - ◆ Signal magnitude and phase
  - ◆ Other properties

8

# Outline 2

- ◆ 4. Operations on signals
  - ◆ Signal creation and manipulation
  - ◆ Signal arithmetic and statistics
  - ◆ Signal-processing functions and flow-charts
  - ◆ Memory, difference equations and digital filters
  - ◆ Autoregression and moving-average processes
- ◆ 5. Information theory of signals
  - ◆ Applications
    - ◆ Error correction
    - ◆ Compression
    - ◆ Data embedding and hiding

9

# Outline 3

- ◆ 6. Transforms and mappings
  - ◆ Time- and frequency-domain signals
  - ◆ Spectral representations, Fourier transform
  - ◆ Linear prediction, signal modeling by polynomials
- ◆ 7. Techniques in applications
  - ◆ Image edge/object/patch detection
  - ◆ Motion estimation in video
  - ◆ Feature extraction from speech and music
  - ◆ Gesture capture and analysis

10

# Outline 4

- ◆ 8. Projects
  - ◆ SSUM examples throughout
  - ◆ Image processing, feature extraction
  - ◆ Video processing, computer vision
  - ◆ Audio feature extraction, onset detection
  - ◆ Gestural input and gesture detection
  - ◆ Capture, archival, distribution, transcoding

11

# The MAT 201A Approach

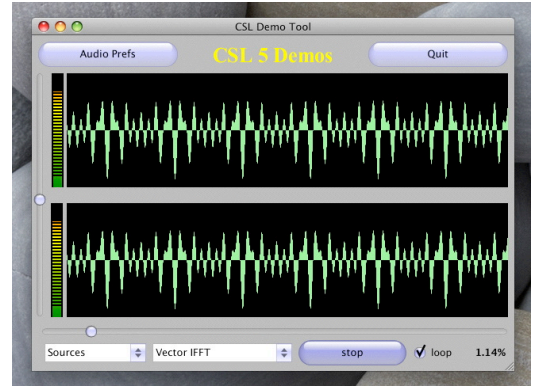
- ◆ Learn concepts and definitions
- ◆ Learn to read the math
- ◆ Learn to turn math into code
- ◆ Learn to customize/alter code
- ◆ Learn to use a code development tool

12

# DSP Textbooks

- ♦ *Signal Processing First*. (AKA *DSP First*) McClellan, Schafer, and Yoder <http://users.ece.gatech.edu:80/~dspfirst>
- ♦ *Digital Signal Processing: A Computer Based Approach*. S. K. Mitra
- ♦ *A Digital Signal Processing Primer with Applications to Digital Audio and Computer Music*. K. Steiglitz
- ♦ *Signals and Systems*. A. Oppenheim and A. Willsky
- ♦ Bores DSP Tutorial (WWW, in reader)
- ♦ Many on-line DSP Tutorials (see MAT 201A web site)

## Topic 1



13

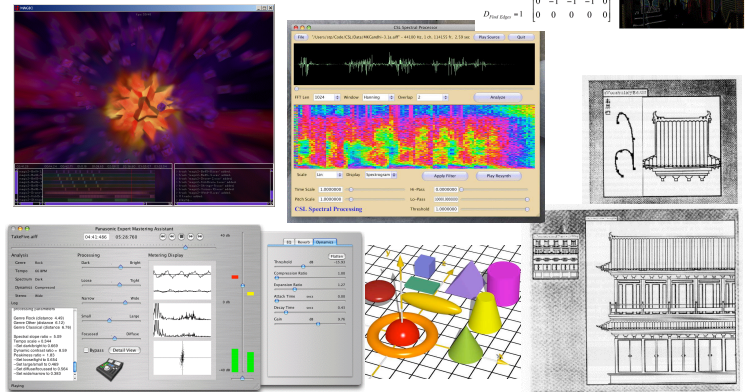
14

# Application Examples

- ♦ What media?
  - ♦ Image, video, sound, gesture, biometric, environmental, others?
- ♦ What task?
  - ♦ Capture/logging, archiving, synthesis, transcoding, feature extraction, classification, matching, ...

## Application Survey

MAT 200C slides

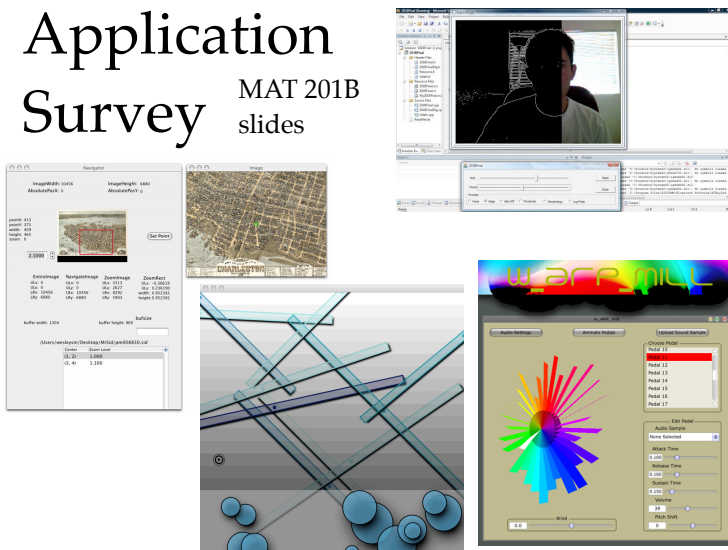


15

16

# Application Survey

MAT 201B slides



17

## MODE (1992)



18

# Application Examples

- ◆ How are apps written?
  - ◆ Media data representation
    - ◆ Storage, IO formats
  - ◆ Operations on media data
    - ◆ Mathematical transformations
    - ◆ Perceptual considerations

19

# Creating Applications

- ◆ How are apps written?
  - ◆ Language (C, C++, Java, scripting, MATLAB)
  - ◆ Platform (PC, mobile, embedded, ASIC)
  - ◆ Libraries used (APIs)
  - ◆ Algorithms used (synthesis, analysis, processing, compression, ...)
  - ◆ Evaluate app examples

20

## Exercise

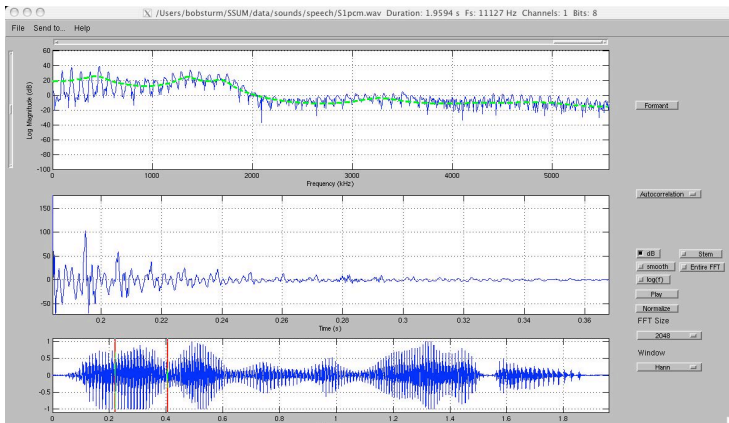
- ◆ Select a multimedia application you use
  - ◆ Image, video, sound, music, etc.
- ◆ Analyze its data type(s) and I/O
- ◆ What non-trivial operations does it support?
- ◆ How might these be implemented?

21

## What will we be doing?

- ◆ MAT 201A
  - ◆ The multimedia (MM) expert of the future will need to have a firm grasp of the mathematical foundations of MM data representation and to understand (and even be able to extend) the algorithms used in common MM signal synthesis/processing/analysis applications.

22



## Topic 2: MATLAB & SSUM

## Why MATLAB?

- ◆ High-level, abstract, typeless, scripting
- ◆ Cross-platform, well-integrated with OS
- ◆ Extensive libraries for DSP of all media
- ◆ Interactive development environment
- ◆ GUI and GUIDE
- ◆ Commercial and free versions

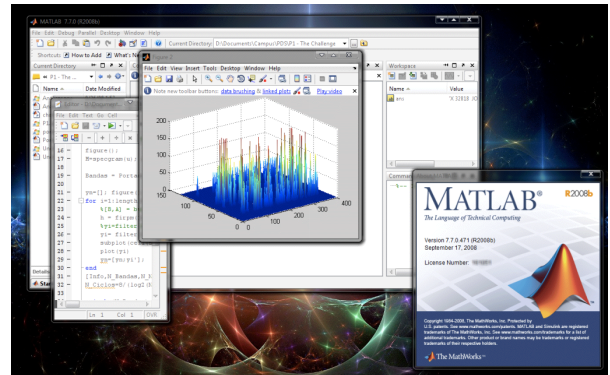
24

23

# Installing/ running MATLAB

- ♦ MATLAB purchase
- ♦ Open computer labs in Phelps Hall
- ♦ GNU Octave (free)
- ♦ SSUM Sources (see MAT 201A web site)
- ♦ SPFirst code (on CD-ROM)
- ♦ Other MATLAB demo code

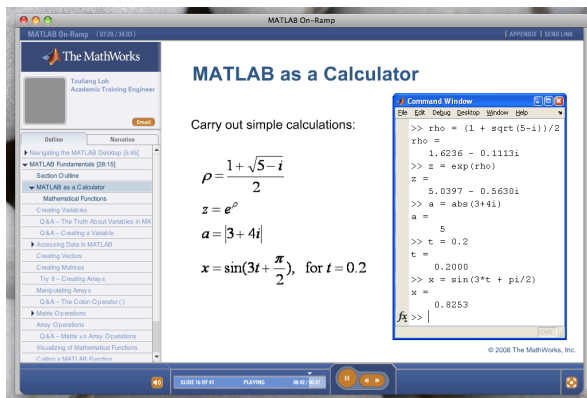
# MATLAB GUI



25

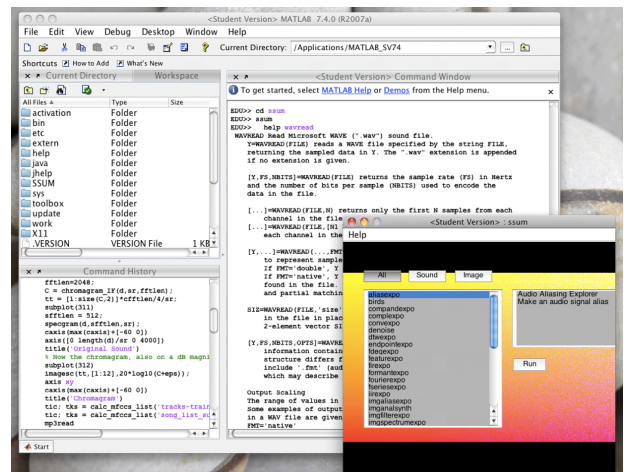
26

# MATLAB Tutorial Movies



27

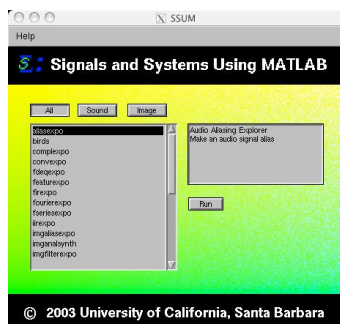
# MATLAB & SSUM



28

# SSUM Demos

- ♦ Bob Sturm's MS thesis
- ♦ ICMC Publication
- ♦ List of demos
- ♦ Source code available!



29

# SSUM Demos

9. Appendix A: List of SSUM Applications.....	49	IIR Filter Explorer.....	64
SSUM Main GUI.....	49	Image Aliasing Explorer.....	65
Additive Synthesis Forest.....	50	Image Analysis/Resynthesis Explorer .....	66
Additive Synthesis Waveform Explorer.....	51	Image Filter Explorer .....	67
Audio Aliasing Explorer .....	52	Image Spectrum Explorer.....	68
Catastrophic Additive Synthesis Composition Machine.....	53	LPC Explorer .....	69
Communication Models Explorer.....	54	Modulation Explorer.....	70
Complex Number Explorer.....	55	Pole-Zero Explorer.....	71
Concatenative Synthesis Explorer.....	56	Pole-Zero Filter Explorer .....	72
Convolution Explorer .....	57	Sampling Explorer .....	73
Cross-Synthesis Explorer.....	58	Signal Feature Explorer .....	74
Finite Difference Equation Explorer.....	59	Sinewave Speech Synthesis Explorer.....	75
FIR Filter Explorer.....	60	Sinusoidal Explorer.....	76
Formant Explorer.....	61	Sonogram Explorer.....	77
Fourier Explorer.....	62	Sound Analysis/Resynthesis Explorer.....	78
Fourier Series Explorer.....	63	Spectrum Explorer.....	79

30

# SPFirst MATLAB Examples

Table C-1: Laboratory material on SP-First CD-ROM.

Lab	Subject	Cross-Reference
Lab #1	Introduction to Matlab	Ch. ??
Lab #2a	Introduction to Complex Exponentials—Multipath	Ch. ??
Lab #2b	Introduction to Complex Exponentials—Direction Finding	Ch. ??
Lab #3	AM and FM Sinusoidal Signals	Ch. ??
Lab #4	Synthesis of Sinusoidal Signals	Ch. ??
Lab #5	FM Synthesis for Musical Instruments	Ch. ??
Lab #6	Digital Images: A/D and D/A	Ch. ??
Lab #7	Sampling, Convolution, and FIR Filtering	Ch. ??
Lab #8	Frequency Response: Bandpass & Nulling Filters	Ch. ??
Lab #9	Encoding and Decoding Touch-Tone Signals	Ch. ?? and ??
Lab #10	Octave Band Filtering	Ch. ??
Lab #11	PcZ—The $z$ , $n$ , and $\omega$ Domains	Ch. ??
Lab #12	Two Convolution GUIs	Ch. 9
Lab #13	Numerical Evaluation of Fourier Series	Ch. 10
Lab #14a	Design with Fourier Series—Power Supply	Ch. 12
Lab #14b	Design with Fourier Series—Distortion	Ch. 12
Lab #15	Fourier Series	Ch. 12
Lab #16	AM Communication System	Ch. 12
Lab #17	Digital Communication: FSK Modem (Encoding)	Ch. 13
Lab #18	Digital Communication: FSK Modem (Decoding)	Ch. 13
Lab #19	The Fast Fourier Transform	Ch. 13
Lab #20	Extracting Frequencies of Musical Tones	Ch. ??

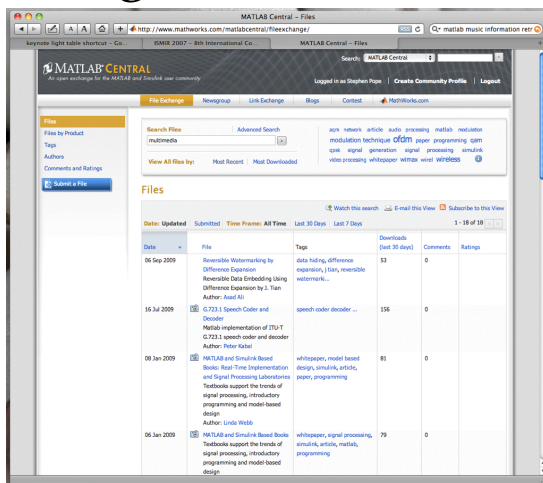
# MATLAB Labs

- ◆ First steps: using the interpreter
- ◆ Using built-in help
- ◆ Variables and dimension; the colon
- ◆ Simple expressions
- ◆ Plotting functions
- ◆ File IO (data and code)
- ◆ Writing new functions
- ◆ Code files

31

32

# Finding MATLAB Code



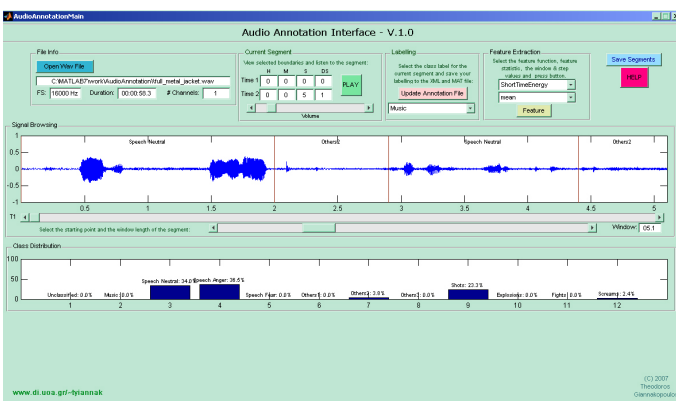
33

# MAT 201A Code Archive

- AudioAnnotation
- audioFeatureExtraction
- audiosegmentation
- AuditoryToolbox
- BlobsDemo
- bss\_oracle\_2.0
- CATbox v0.1
- CCRM
- chequeNumberReader
- conv\_func
- coverSongDetection
- DAFXTOOL
- EyeTracker
- FaceDetection
- HMM
- ImageAnalysis
- ImageSegmentationToolbox
- KPMstats
- KPMtools
- LPC Project
- MIRtoolbox1
- Motion\_Detection
- netlab3.3
- realTimeRecord
- somtoolbox
- SPF\_CD
- virtualsignallab
- Wavelab850
- FFTAUD.m
- formant\_tracker.m
- SSMultibandKamath02.m

34

# Example App: Audio Annotation



35

# A Bit of Math

- ◆ Scalars and vectors
- ◆ Complex numbers
- ◆ Arithmetic expressions
- ◆ Algebraic expressions
- ◆ Basics of trigonometry

36

# Scalar, Vector, Matrix

- ◆ Scalar: single-values
  - ◆ **1, 3.14**
- ◆ Vector: 1-D list, array, sequence, buffer
  - ◆ **[ 1, 2, 3 ]**
- ◆ Matrix: 2-D table, chart, spreadsheet
  - ◆ **[ 1, 2, 3 ] [ 4, 5, 6 ]**

37

# Math Topics

- ◆ Arithmetic: simple operations on scalars, vectors, matrices
- ◆ Algebra: expressions using variables and higher-level functions
- ◆ Trigonometry: angular functions, unit circle
- ◆ Exponentials: relationships of potentiation  $x^y$

38

# Operations

- ◆ Scalar operations: simple, straight-forward
  - ◆ Basic arithmetic
  - ◆ Statistics
- ◆ Vector operations
  - ◆ Vector-vector (element-by-element, same size) (+ \* cross-correlate)
  - ◆ Vector-scalar (sum, avg, mean, etc.)
  - ◆ Vector-vector (different size), vector transpose, sub-sample, interpolate

39

# Matrix Operations

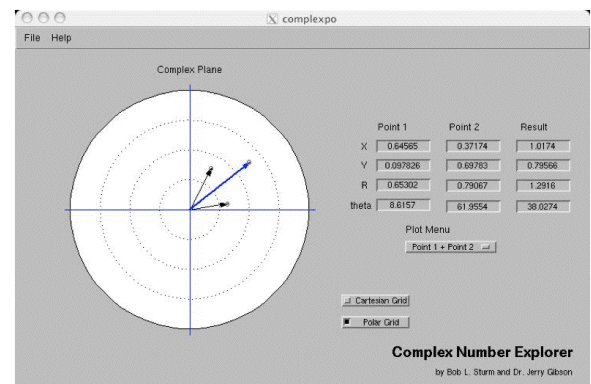
- ◆ *Linear algebra*
- ◆ Matrix-matrix (element-by-element, row / column format) arithmetic, special matrix operations
  - ◆ Affine transformations  $x \mapsto Ax + b$ .
- ◆ Matrix-vector (\*) (row / column-major format)
- ◆ Matrix-scalar (\*)
- ◆ Matrix statistics
  - ◆ Determinant, Eigen-values

40

# Complex Numbers

- ◆ Real numbers: regular scalars
- ◆ “Imaginary” numbers”  $\sqrt{-1}$
- ◆ Complex: pair-values (r / i) (x / y) with dimensions related by exponentials or trigonometry
  - ◆ Plotted on X/Y graph as R/I
  - ◆ Relationship to “unit circle”
  - ◆ Relationship to trigonometry and exponentials

41



# SSUM Complex # Explorer

42

# Trigonometry

- ◆ Geometry of angles
- ◆ Relation to the “unit circle”
- ◆ Relation to complex numbers (Euler)
- ◆ Algebra of trig. functions

$$\sin(A + B) = \sin A \cos B + \cos A \sin B$$

$$\begin{aligned} \cos 2A &= \cos^2 A - \sin^2 A \\ \sin 2A &= 2 \sin A \cos A \\ \cos 2A &= 1 - 2 \sin^2 A \\ \cos 2A &= 2 \cos^2 A - 1 \end{aligned} \quad \tan 2A = \frac{2 \tan A}{1 - \tan^2 A}$$

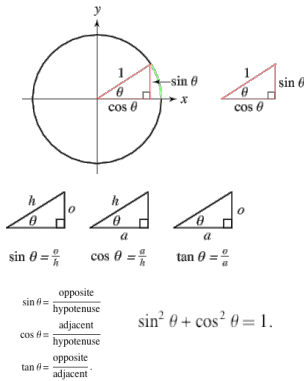


Table 2-1: Basic properties of the sine and cosine functions.

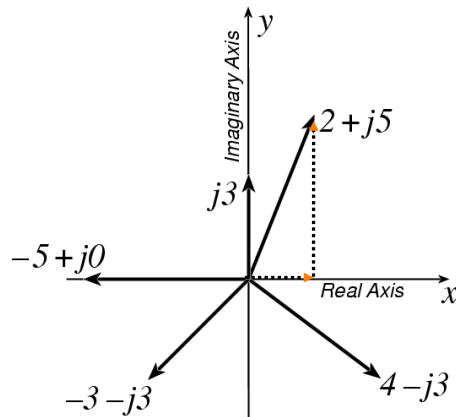
Property	Equation
Equivalence	$\sin \theta = \cos(\theta - \pi/2)$ or $\cos(\theta) = \sin(\theta + \pi/2)$
Periodicity	$\cos(\theta + 2\pi k) = \cos \theta$ , when $k$ is an integer
Evenness of cosine	$\cos(-\theta) = \cos \theta$
Oddness of sine	$\sin(-\theta) = -\sin \theta$
Zeros of sine	$\sin(\pi k) = 0$ , when $k$ is an integer
Ones of cosine	$\cos(2\pi k) = 1$ when $k$ is an integer.
Minus ones of cosine	$\cos[2\pi(k + \frac{1}{2})] = -1$ , when $k$ is an integer.

Table 2-2: Some basic trigonometric identities.

Number	Equation
1	$\sin^2 \theta + \cos^2 \theta = 1$
2	$\cos 2\theta = \cos^2 \theta - \sin^2 \theta$
3	$\sin 2\theta = 2 \sin \theta \cos \theta$
4	$\sin(\alpha \pm \beta) = \sin \alpha \cos \beta \pm \cos \alpha \sin \beta$
5	$\cos(\alpha \pm \beta) = \cos \alpha \cos \beta \mp \sin \alpha \sin \beta$

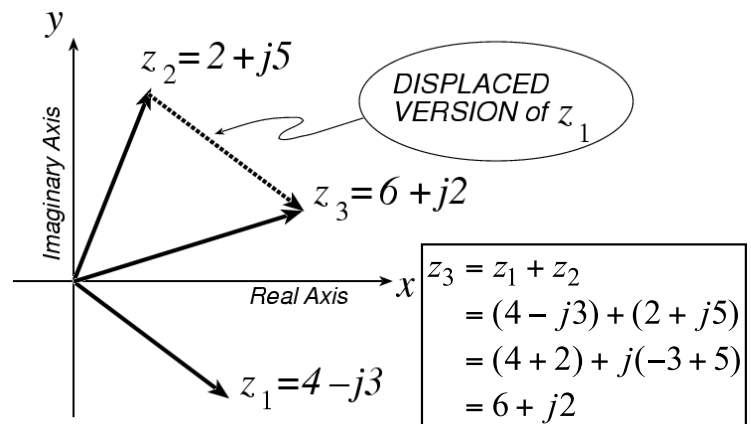
## Trig Function Properties

## PLOT COMPLEX NUMBERS



45

## COMPLEX ADDITION = VECTOR Addition



46

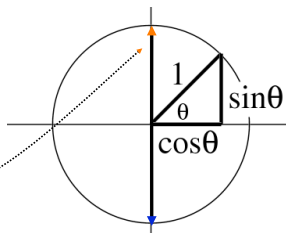
## POLAR FORM

### Vector Form

- Length = 1
- Angle =  $\theta$

### Common Values

- $j$  has angle of  $0.5\pi$
- $-1$  has angle of  $\pi$
- $-j$  has angle of  $1.5\pi$
- also, angle of  $-j$  could be  $-0.5\pi = 1.5\pi - 2\pi$
- because the PHASE is **AMBIGUOUS**



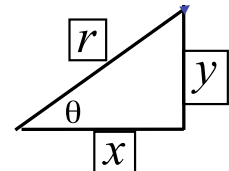
47

## POLAR <--> RECTANGULAR

- Relate (x,y) to (r,θ)

$$\begin{aligned} r^2 &= x^2 + y^2 \\ \theta &= \tan^{-1}\left(\frac{y}{x}\right) \end{aligned}$$

Most calculators do  
Polar-Rectangular



$$\begin{aligned} x &= r \cos \theta \\ y &= r \sin \theta \end{aligned}$$

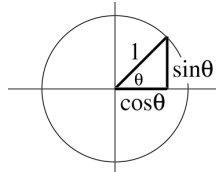
**Need a notation for POLAR FORM**

48

## Euler's FORMULA

### Complex Exponential

- Real part is cosine
- Imaginary part is sine
- Magnitude is one



$$e^{j\theta} = \cos(\theta) + j \sin(\theta)$$

$$re^{j\theta} = r \cos(\theta) + jr \sin(\theta)$$

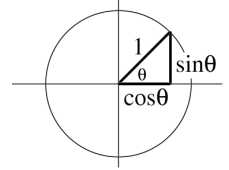
49

## COMPLEX EXPONENTIAL

$$e^{j\omega t} = \cos(\omega t) + j \sin(\omega t)$$

### Interpret this as a Rotating Vector

- $\theta = \omega t$
- Angle changes vs. time
- ex:  $\omega = 20\pi$  rad/s
- Rotates  $0.2\pi$  in 0.01 secs



$$e^{j\theta} = \cos(\theta) + j \sin(\theta)$$

50

## cos = REAL PART

Real Part of Euler's

$$\cos(\omega t) = \Re\{e^{j\omega t}\}$$

General Sinusoid

$$x(t) = A \cos(\omega t + \varphi)$$

So,

$$\begin{aligned} A \cos(\omega t + \varphi) &= \Re\{Ae^{j(\omega t + \varphi)}\} \\ &= \Re\{Ae^{j\varphi} e^{j\omega t}\} \end{aligned}$$

51

## COMPLEX AMPLITUDE

General Sinusoid

$$x(t) = A \cos(\omega t + \varphi) = \Re\{Ae^{j\varphi} e^{j\omega t}\}$$

### Complex AMPLITUDE = X

$$z(t) = Xe^{j\omega t} \quad X = Ae^{j\varphi}$$

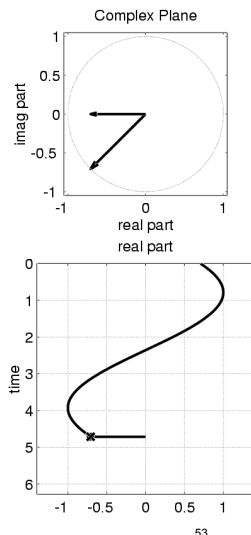
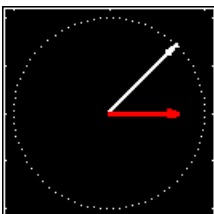
Then, any Sinusoid = REAL PART of  $Xe^{j\omega t}$

$$x(t) = \Re\{Xe^{j\omega t}\} = \Re\{Ae^{j\varphi} e^{j\omega t}\}$$

52

## Rotating Phasor

See Demo on CD-ROM  
Chapter 2



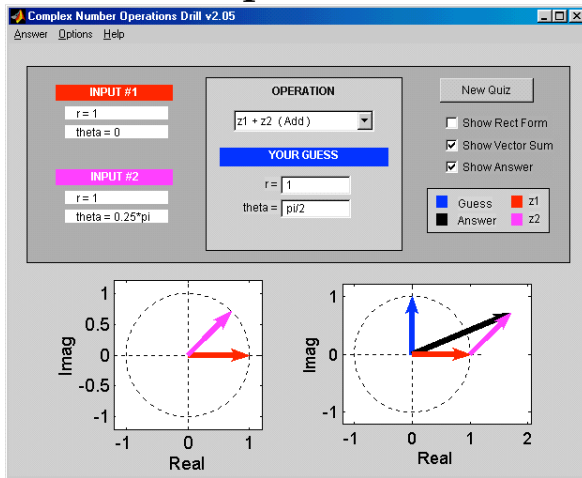
53

## More Linear Algebra

- See MATLAB tutorials
- See MATLAB online videos
- See readings

52

## Z DRILL (Complex Arith)



55

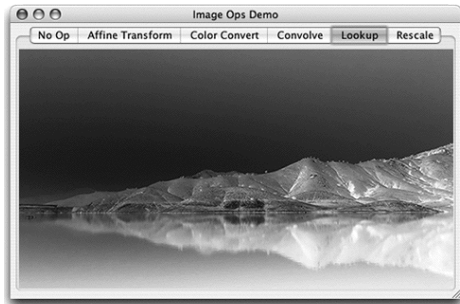
## Exercise

- ◆ Complex number demos and exercises in MATLAB
- ◆ Trigonometry demos and exercises in MATLAB
- ◆ Make MATLAB “programs” for simple operations on complex numbers and phasors
- ◆ Explore MATLAB linear algebra functions

56

## Topic 3: Media Data

### Topic 3



57

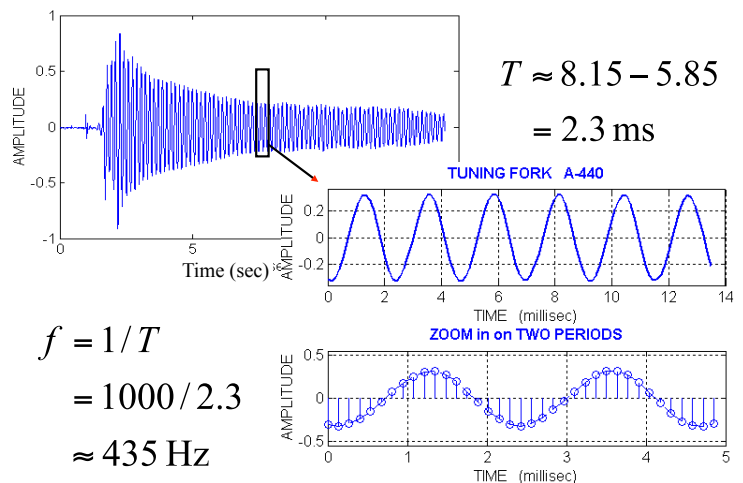
- ◆ Data representations
  - ◆ Image, sound data structures
  - ◆ Signal sampling and quantization
  - ◆ Signal magnitude and phase
  - ◆ Other properties

58

## Media and Signals

- ◆ Start with a few examples
- ◆ Audio perception: hearing
  - ◆ Tuning fork example
  - ◆ Auditory scene analysis
- ◆ Vision and visual perception
  - ◆ Scanned image example
  - ◆ Scene analysis

### TUNING FORK A-440 Waveform



59

60

# Scanned Image Example



♦ Original = 644w \* 558h \* 24d

61

# Media Data

- ♦ Data representation
  - ♦ Analog & digital
  - ♦ Continuous-time and discrete-time
  - ♦ Time-domain
  - ♦ Frequency-domain
  - ♦ Other domains

62

# Analog & Digital

- ♦ Analog
  - ♦ Real-world signals and perception
  - ♦ Theoretically unlimited precision
  - ♦ Limited by noise
  - ♦ Simple transducers (speaker, microphone, camera...)
- ♦ Digital
  - ♦ Limited resolution
  - ♦ No noise
  - ♦ Complex transducers

63

# Continuous- and Discrete-time

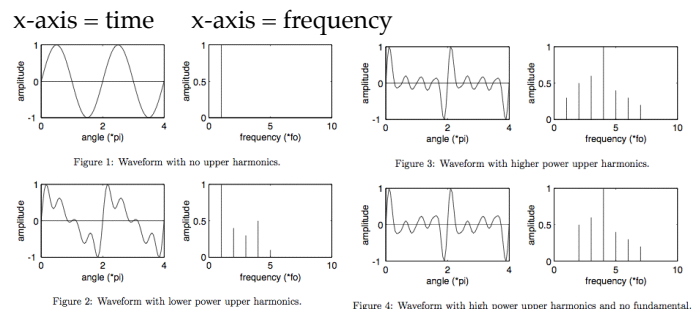
- ♦ Continuous-time
  - ♦ Real-world signals and perception
  - ♦ Theoretically unlimited frequency response
  - ♦ Requires infinite bandwidth to transmit
- ♦ Discrete-time
  - ♦ Limited frequency response (Nyquist)
  - ♦ Limited bandwidth required

64

# Signal Domains

- ♦ Signal as a function of time
  - ♦ Sound
  - ♦ Motion
- ♦ Signal as a function of frequency
  - ♦ Spectrum
  - ♦ Transform back and forth (analysis/synthesis)
- ♦ Other domains of representation

# Functions of Time (wave) and Functions of Frequency (spectrum)



65

66

# Signal Domains

- ◆ Time-domain
  - ◆ Sampled discrete audio signal
  - ◆ Control signal
    - ◆ 1-D, n-D, geometry
  - ◆ Impulse response
  - ◆ Other T-D semantics
- ◆ Frequency-domain
  - ◆ Spectrum (inst. frame or continuous 3-D)
  - ◆ F0 track
  - ◆ Wavelet-modulus-domain

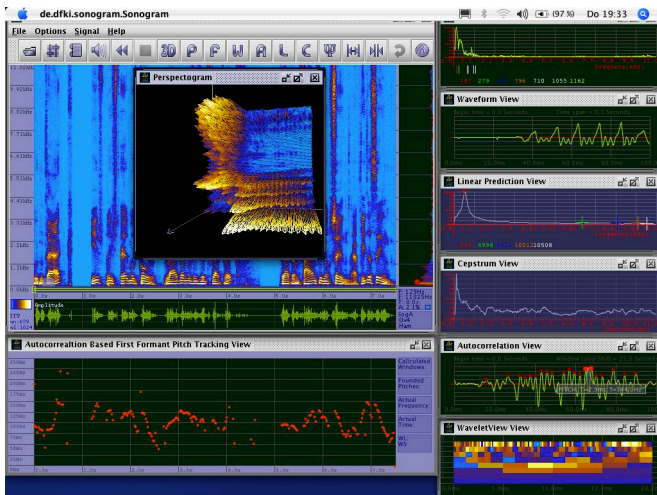
67

# Frequency-domain Transforms

- ◆ How to derive a spectrum from a time-domain signal?
  - ◆ Swept band-pass filter
  - ◆ Swept oscillator (heterodyne)
  - ◆ Filter bank
  - ◆ Fourier transform
  - ◆ Polynomial approximation (linear prediction)
  - ◆ Others?

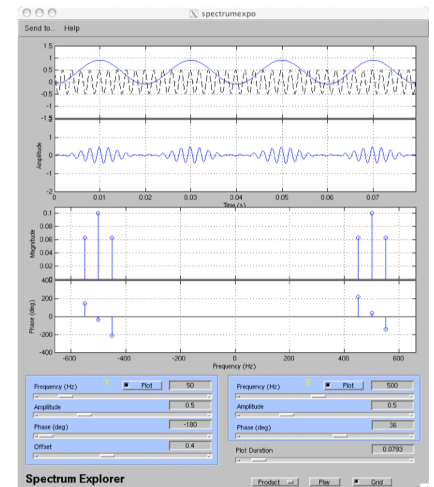
68

## Example: Spectrogram



69

## SSUM Spectrum Explorer



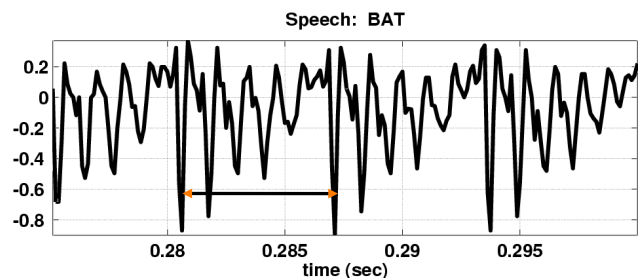
70

# Real-world Signals

- ◆ Are complex (as in, not simple) and noisy
- ◆ Are often composed of simple components that are harmonically related (sums of sines)
- ◆ Can be disassembled into components that differ in *frequency, amplitude, and phase*

## Speech Signal: BAT

- Nearly **Periodic** in Vowel Region
  - Period is (Approximately)  $T = 0.0065$  sec



72

72

## SINUSOIDAL SIGNAL

$$A \cos(\omega t + \varphi)$$

### ▪ FREQUENCY $\omega$

- Radians/sec
- Hertz (cycles/sec)

$$\omega = (2\pi)f$$

### ▪ PERIOD (in sec)

$$T = \frac{1}{f} = \frac{2\pi}{\omega}$$

### ▪ AMPLITUDE

- Magnitude

 $A$ 

### ▪ PHASE

 $\varphi$ 

## EXAMPLE of SINUSOID

$$5 \cos(0.3\pi t + 1.2\pi)$$

### ▪ Formula defines

- amplitude  $A$ ,
- freq  $\omega$ , and
- phase  $\phi$

$$\begin{aligned} A &= 5 \\ \omega &= 0.3\pi \\ \varphi &= 1.2\pi \end{aligned}$$

74

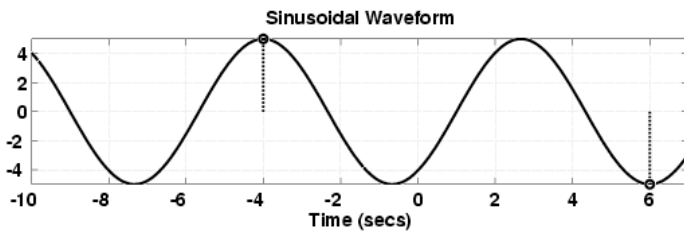
74

## Plot the Signal

### ▪ Given the Formula

$$5 \cos(0.3\pi t + 1.2\pi)$$

### ▪ Make a plot



73

## PHASE <--> TIME-SHIFT

### ▪ Equate the formulas:

$$A \cos(\omega(t - t_m)) = A \cos(\omega t + \varphi)$$

### ▪ and we obtain:

$$-\omega t_m = \varphi$$

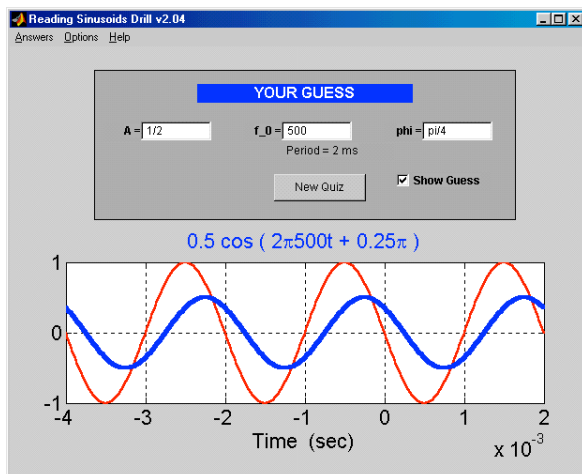
### ▪ or,

$$t_m = -\frac{\varphi}{\omega}$$

76

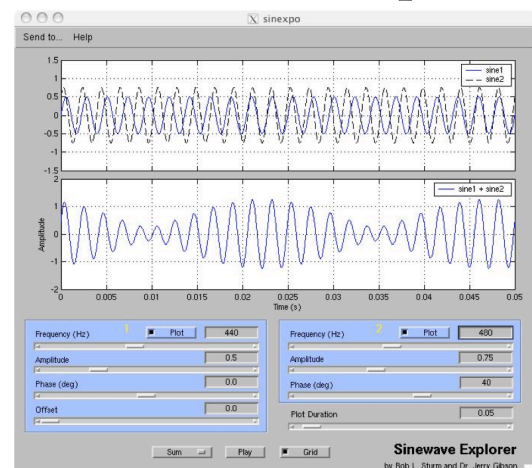
76

## SINE DRILL (MATLAB GUI)



77

## SSUM Sinewave Explorer



78

# Signals and Systems

- ◆ Building signal-processing systems
- ◆ A/D and D/A conversion
- ◆ Sampling and sample-rate
- ◆ Quantization and signal resolution
- ◆ Other considerations

## SYSTEMS Process Signals



- PROCESSING GOALS:
  - Change  $x(t)$  into  $y(t)$ 
    - For example, more BASS
  - Improve  $x(t)$ , e.g., image deblurring
  - Extract Information from  $x(t)$

80

80

## System IMPLEMENTATION

- ANALOG/ELECTRONIC:
  - Circuits: resistors, capacitors, op-amps



- DIGITAL/MICROPROCESSOR
  - Convert  $x(t)$  to numbers stored in memory



81

81

## DIGITIZE the WAVEFORM

- $x[n]$  is a SAMPLED SINUSOID
  - A list of numbers stored in memory
- Sample at 11,025 samples per second
  - Called the SAMPLING RATE of the A/D
  - Time between samples is
    - $1/11025 = 90.7$  microsec
- Output via D/A hardware (at  $F_{\text{samp}}$ )

82

## STORING DIGITAL SOUND

- $x[n]$  is a SAMPLED SINUSOID
  - A list of numbers stored in memory
- CD rate is 44,100 samples per second
- 16-bit samples (2 Bytes)
- Stereo uses 2 channels
- Number of bytes for 1 minute is
  - $2 \text{ ch} * 2 \text{ B/s} * 60\text{sec} * 44100 \text{ Hz} = 10.584 \text{ Mbytes}$

83

84

## Sampling & Quantization

- ◆ Sampling
  - ◆ Convert continuous-time to discrete-time
  - ◆ ...at a fixed sample-rate
- ◆ Quantization
  - ◆ Convert continuous-value (analog) to discrete-value (digital)
  - ◆ ...at a known signal resolution

# Sampling and Quantization

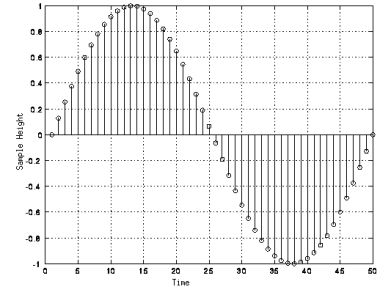
The **digitization** of a signal  $x(t)$  comprises two steps:  $x(t) \rightarrow x_s(n) \rightarrow x_q(n)$

- **Sampling**:  $x_s(n) = x(nT)$  where
  - $T$  is the **sampling period**
  - $F = 1/T$  is the **sampling frequency**
  - The inverse transformation ( $x_s(n)$  to  $x(t)$ ) is called **interpolation**
- **Quantization**:  $x_q(n) = Q(x_s(n))$ 
  - $Q(\cdot)$  is a **rounding function** which maps the values of  $x_s(n)$  into  $N$  levels ( $\Delta$  is the **quantization step**)
  - Typically,  $N = 2^{N_b}$ , therefore we need  $N_b$  bits to represent one quantized sample.

85

# Sampling Example

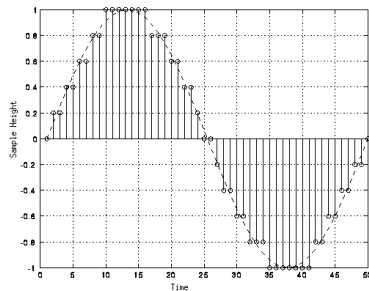
- ◆ Sampling is time-quantization
- ◆ The sample rate determines the fixed upper frequency limit for the signal



86

# Quantization Example

- ◆ Value-quantization
- ◆ Quantization step determines the smallest signal change that can be represented



87

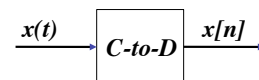
# SAMPLING $x(t)$

## ■ SAMPLING PROCESS

- Convert  $x(t)$  to numbers  $x[n]$
- “ $n$ ” is an integer;  $x[n]$  is a sequence of values
- Think of “ $n$ ” as the storage address in memory

## ■ UNIFORM SAMPLING at $t = nT_s$

- IDEAL:  $x[n] = x(nT_s)$



88

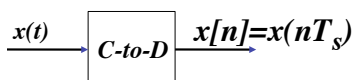
# SAMPLING RATE, $f_s$

## ■ SAMPLING RATE ( $f_s$ )

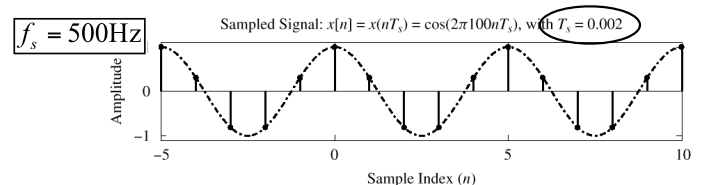
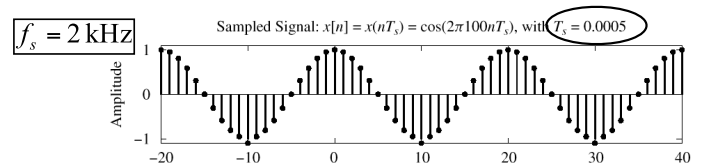
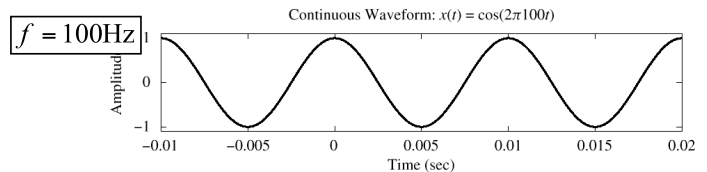
- $f_s = 1/T_s$ 
  - NUMBER of SAMPLES PER SECOND
- $T_s = 125 \text{ microsec} \rightarrow f_s = 8000 \text{ samples/sec}$ 
  - UNITS ARE HERTZ: 8000 Hz

## ■ UNIFORM SAMPLING at $t = nT_s = n/f_s$

- IDEAL:  $x[n] = x(nT_s) = x(n/f_s)$



89



90

90

## SAMPLING THEOREM

- HOW OFTEN ?
  - DEPENDS on FREQUENCY of SINUSOID
  - ANSWERED by SHANNON/NYQUIST Theorem
  - ALSO DEPENDS on **"RECONSTRUCTION"**

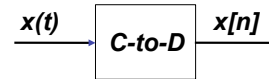
### Shannon Sampling Theorem

A continuous-time signal  $x(t)$  with frequencies no higher than  $f_{\max}$  can be reconstructed exactly from its samples  $x[n] = x(nT_s)$ , if the samples are taken at a rate  $f_s = 1/T_s$  that is greater than  $2f_{\max}$ .

91

## SAMPLING $x(t)$

- UNIFORM SAMPLING at  $t = nT_s$ 
  - IDEAL:  $x[n] = x(nT_s)$

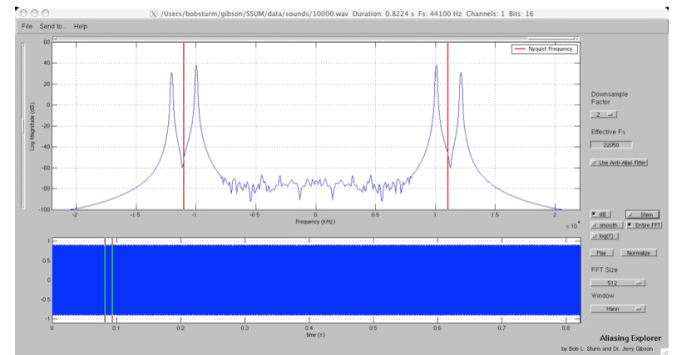


92

## NYQUIST RATE

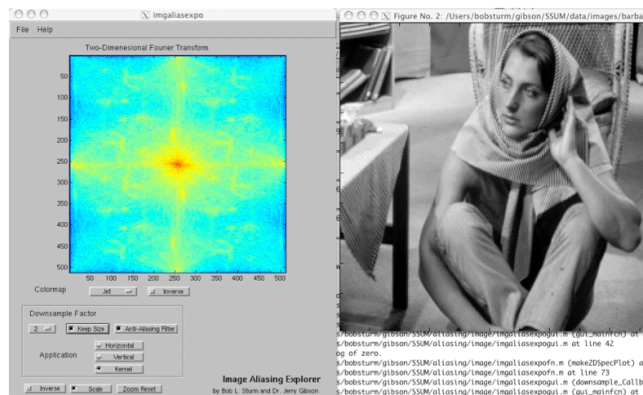
- "Nyquist Rate" Sampling
  - $f_s > \text{TWICE}$  the HIGHEST Frequency in  $x(t)$
  - "Sampling above the Nyquist rate"
- BANDLIMITED SIGNALS**
  - DEF:  $x(t)$  has a HIGHEST FREQUENCY COMPONENT in its SPECTRUM
- NON-BANDLIMITED EXAMPLE
  - TRIANGLE WAVE is NOT BANDLIMITED

93



## SSUM Audio Aliasing Explorer

94

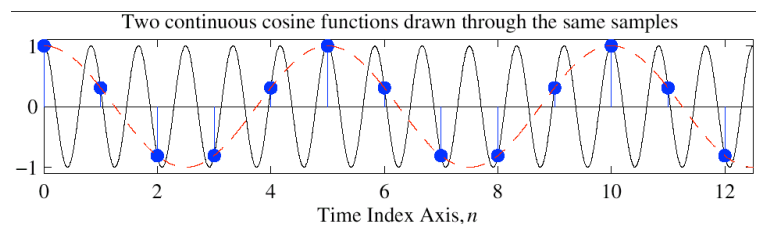


## SSUM Image Aliasing Explorer

95

## Reconstruction? Which One?

Given the samples, draw a sinusoid through the values



$$x[n] = \cos(0.4\pi n)$$

When  $n$  is an integer  
 $\cos(0.4\pi n) = \cos(2.4\pi n)$

96

96

## DISCRETE-TIME SINUSOID

- Change  $x(t)$  into  $x[n]$  **DERIVATION**

$$x(t) = A \cos(\omega t + \varphi)$$

$$x[n] = x(nT_s) = A \cos(\omega nT_s + \varphi)$$

$$x[n] = A \cos((\omega T_s)n + \varphi)$$

$$x[n] = A \cos(\hat{\omega} n + \varphi)$$

$$\hat{\omega} = \omega T_s = \frac{\omega}{f_s} \quad \text{DEFINE DIGITAL FREQUENCY}$$

97

## DIGITAL FREQUENCY

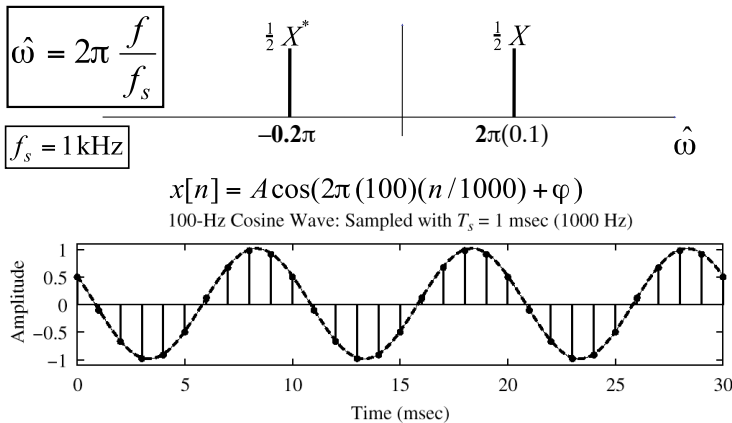
 $\hat{\omega}$ 

- $\hat{\omega}$  VARIES from 0 to  $2\pi$ , as  $f$  varies from 0 to the sampling frequency
- UNITS are radians, **not** rad/sec
  - DIGITAL FREQUENCY is NORMALIZED

$$\hat{\omega} = \omega T_s = \frac{2\pi f}{f_s}$$

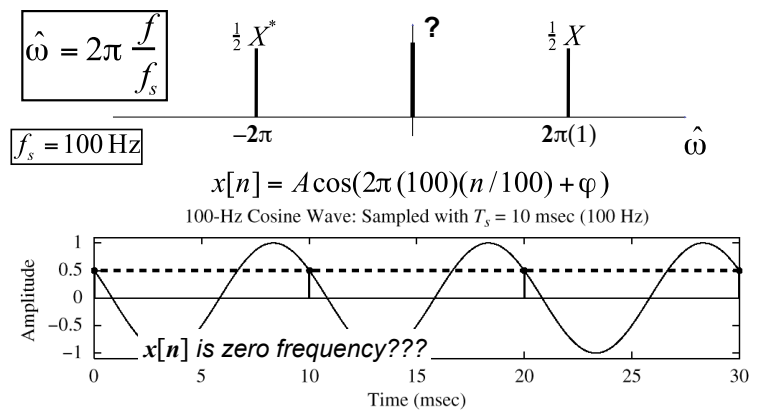
98

## SPECTRUM (DIGITAL)



99

## SPECTRUM (DIGITAL) ???



100

## The REST of the STORY

- Spectrum of  $x[n]$  has more than one line for each complex exponential
  - Called **ALIASING**
  - MANY SPECTRAL LINES**
- SPECTRUM is PERIODIC with period =  $2\pi$ 
  - Because

$$A \cos(\hat{\omega} n + \varphi) = A \cos((\hat{\omega} + 2\pi)n + \varphi)$$

101

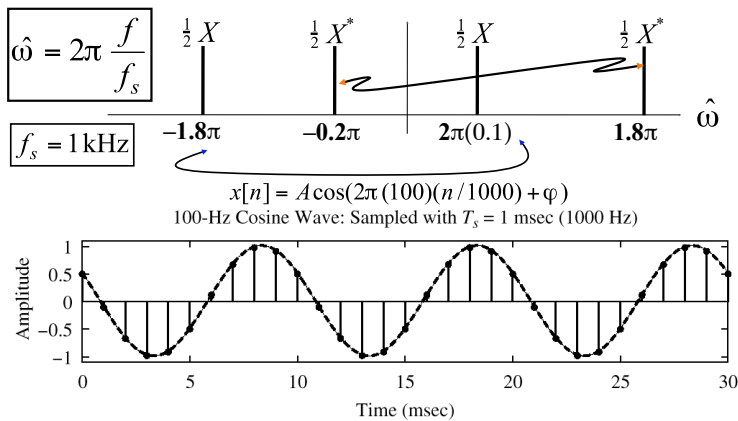
## SPECTRUM for $x[n]$

- PLOT versus NORMALIZED FREQUENCY
- INCLUDE **ALL** SPECTRUM LINES
  - ALIASES
    - ADD MULTIPLES of  $2\pi$
    - SUBTRACT MULTIPLES of  $2\pi$
  - FOLDED ALIASES
    - (to be discussed later)
    - ALIASES of NEGATIVE FREQS

102

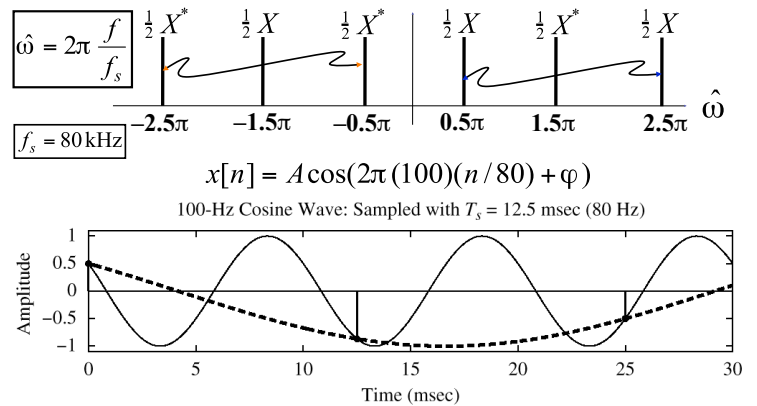
102

## SPECTRUM (MORE LINES)



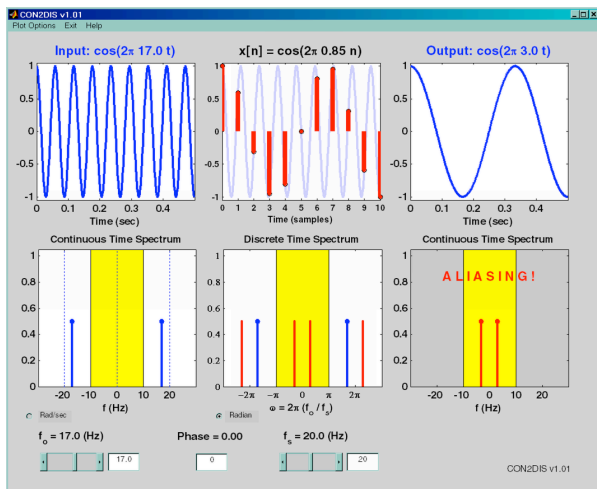
103

## SPECTRUM (ALIASING CASE)

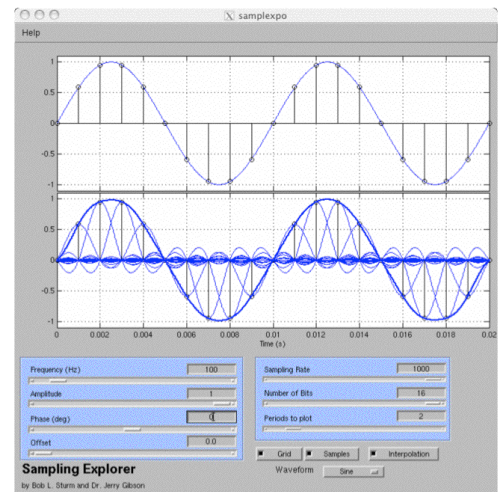


104

## SAMPLING GUI (con2dis)

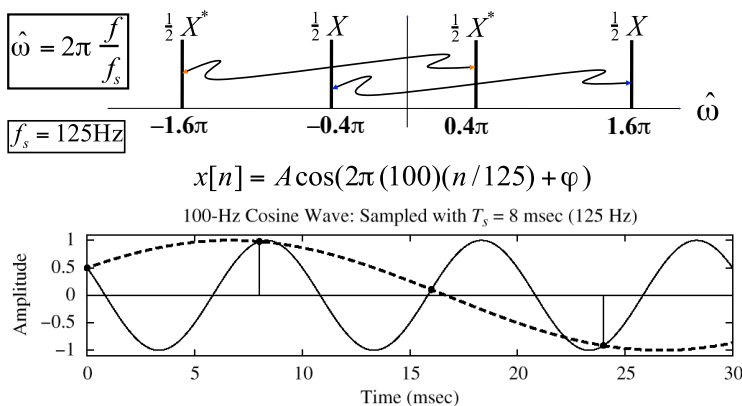


105

SSUM  
Sampling  
Explorer

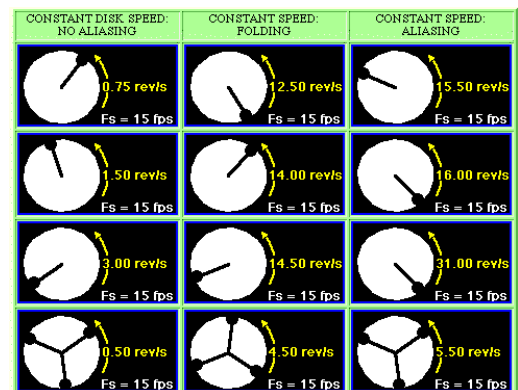
106

## SPECTRUM (FOLDING CASE)



107

## STROBE DEMO (Synthetic)



108

108

## SIGNAL TYPES



- **A-to-D**
  - Convert  $x(t)$  to numbers stored in memory
- **D-to-A**
  - Convert  $y[n]$  back to a “continuous-time” signal,  $y(t)$
  - $y[n]$  is called a “discrete-time” signal

109

## SPECTRUM for $x[n]$

- **INCLUDE ALL SPECTRUM LINES**
  - **ALIASES**
    - ADD INTEGER MULTIPLES of  $2\pi$  and  $-2\pi$
  - **FOLDED ALIASES**
    - ALIASES of NEGATIVE FREQS
- **PLOT versus NORMALIZED FREQUENCY**
  - i.e., DIVIDE  $f_0$  by  $f_s$

$$\hat{\omega} = 2\pi \frac{f}{f_s} + 2\pi \ell$$

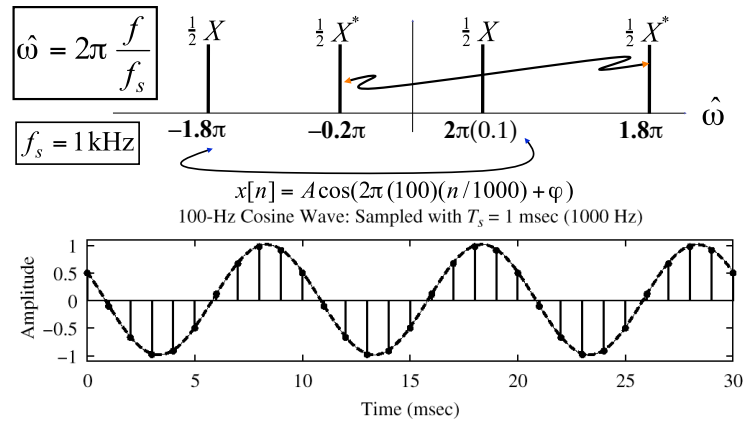
110

## EXAMPLE: SPECTRUM

- $x[n] = A \cos(0.2\pi n + \phi)$
- **FREQS @  $0.2\pi$  and  $-0.2\pi$**
- **ALIASES:**
  - $\{2.2\pi, 4.2\pi, 6.2\pi, \dots\}$  &  $\{-1.8\pi, -3.8\pi, \dots\}$
  - EX:  $x[n] = A \cos(4.2\pi n + \phi)$
- **ALIASES of NEGATIVE FREQ:**
  - $\{1.8\pi, 3.8\pi, 5.8\pi, \dots\}$  &  $\{-2.2\pi, -4.2\pi, \dots\}$

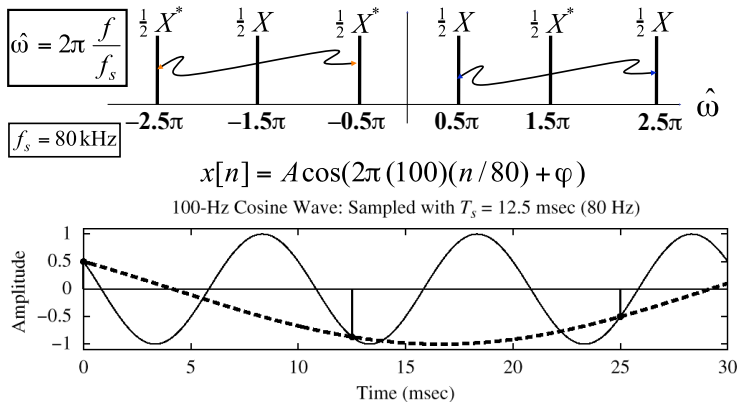
111

## SPECTRUM (MORE LINES)



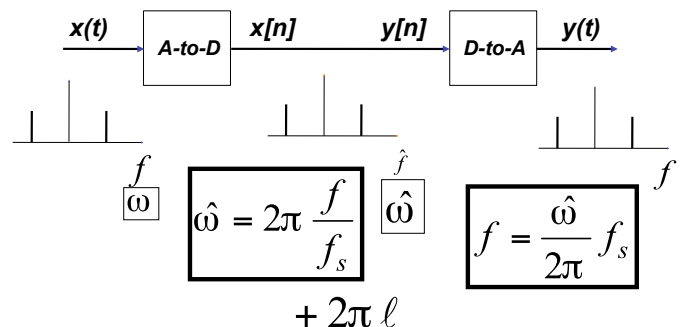
112

## SPECTRUM (ALIASING CASE)



113

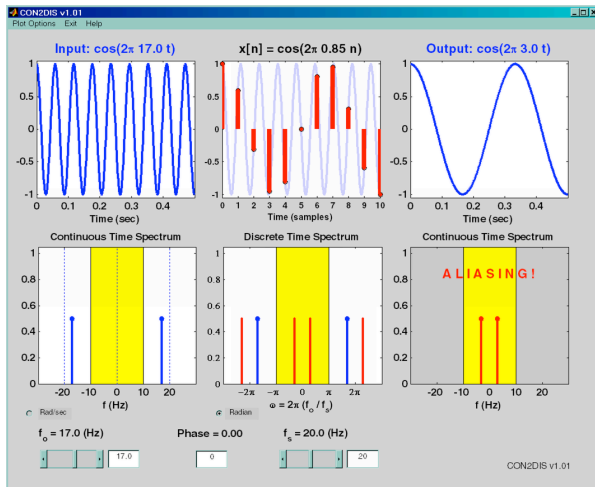
## FREQUENCY DOMAINS



114

114

## SAMPLING GUI (con2dis)



115

## D-to-A Reconstruction



- Create continuous  $y(t)$  from  $y[n]$ 
  - **IDEAL**
    - If you have formula for  $y[n]$
  - Replace  $n$  in  $y[n]$  with  $f_s t$
  - $y[n] = A \cos(0.2\pi n + \phi)$  with  $f_s = 8000$  Hz
  - $y(t) = A \cos(2\pi(800)t + \phi)$

116

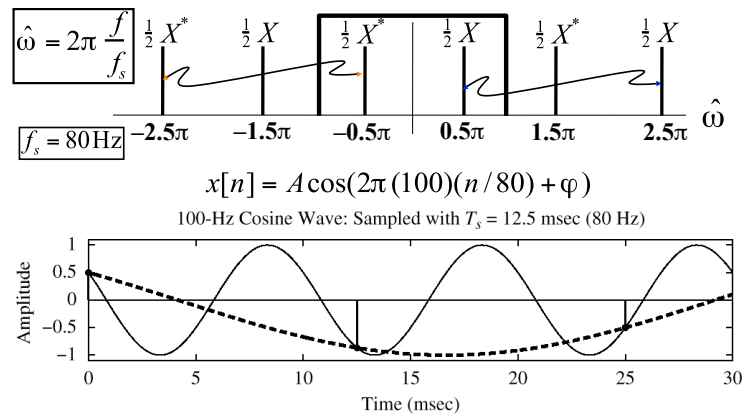
116

## D-to-A is AMBIGUOUS !

- **ALIASING**
  - Given  $y[n]$ , which  $y(t)$  do we pick ???
- INFINITE NUMBER of  $y(t)$ 
  - PASSING THRU THE SAMPLES,  $y[n]$
- D-to-A RECONSTRUCTION MUST CHOOSE ONE OUTPUT
- RECONSTRUCT THE SMOOTHEST ONE
  - THE LOWEST FREQ, if  $y[n]$  = sinusoid

117

## SPECTRUM (ALIASING CASE)

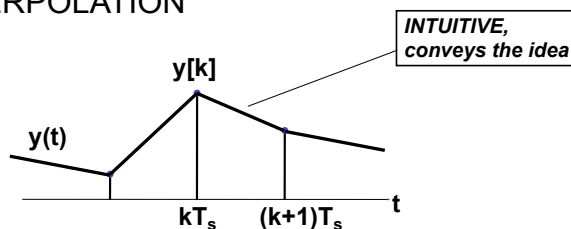


117

118

## Reconstruction (D-to-A)

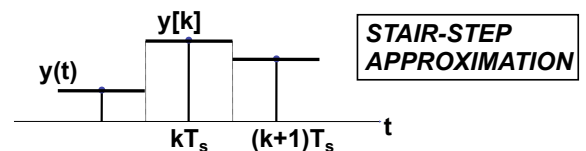
- CONVERT STREAM of NUMBERS to  $x(t)$
- "CONNECT THE DOTS"
- INTERPOLATION



119

## SAMPLE & HOLD DEVICE

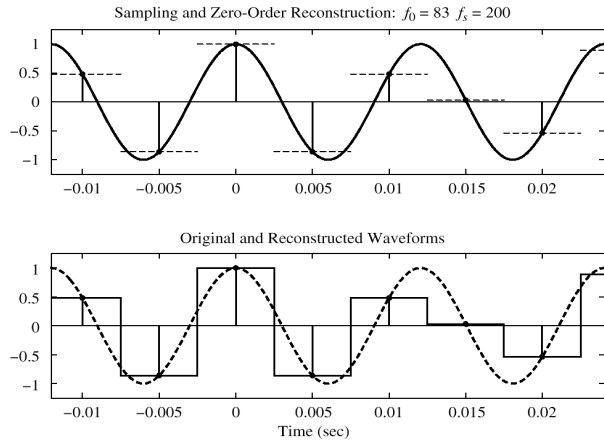
- CONVERT  $y[n]$  to  $y(t)$ 
  - $y[k]$  should be the value of  $y(t)$  at  $t = kT_s$
  - Make  $y(t)$  equal to  $y[k]$  for
    - $kT_s - 0.5T_s < t < kT_s + 0.5T_s$



119

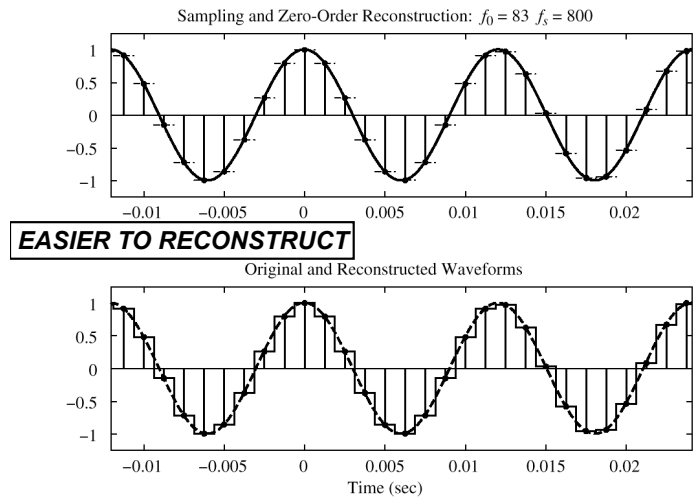
120

## SQUARE PULSE CASE



121

## OVER-SAMPLING CASE



122

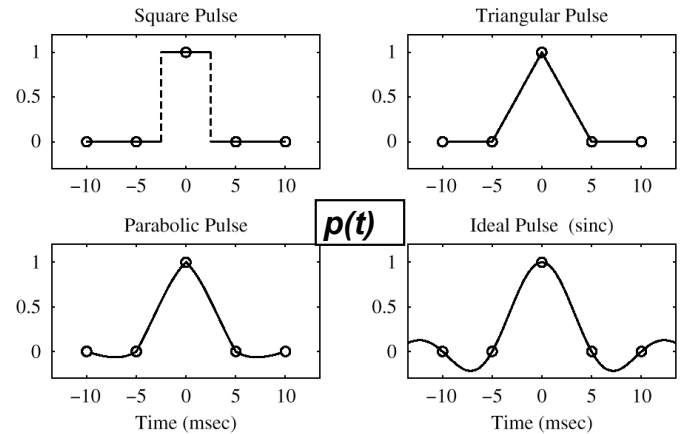
## MATH MODEL for D-to-A

$$y(t) = \sum_{n=-\infty}^{\infty} y[n]p(t - nT_s)$$

SQUARE PULSE:

$$p(t) = \begin{cases} 1 & -\frac{1}{2}T_s < t \leq \frac{1}{2}T_s \\ 0 & \text{otherwise} \end{cases}$$

123



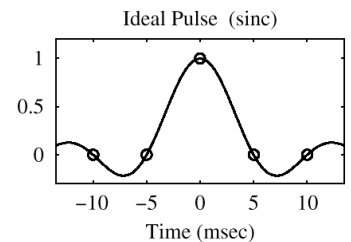
**Figure 4.17** Four different pulses for D-to-C conversion. The sampling period is  $T_s = 0.005$ , i.e.,  $f_s = 200$  Hz. Note that the duration of each pulse is approximately one or two times  $T_s$ .

123

124

## OPTIMAL PULSE ?

**CALLED  
"BANDLIMITED  
INTERPOLATION"**

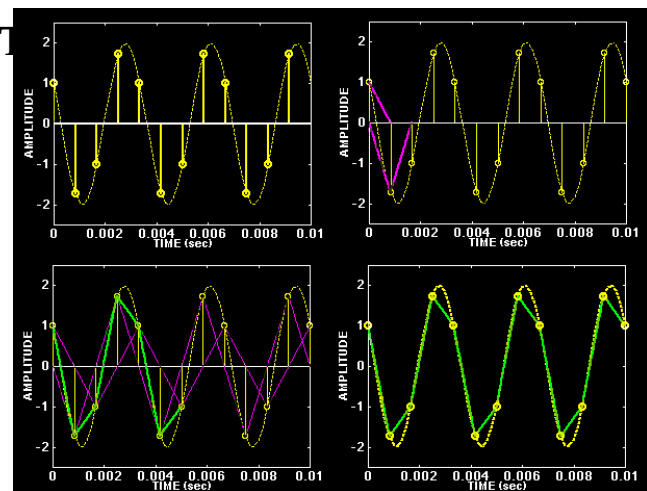


$$p(t) = \frac{\sin \frac{\pi t}{T_s}}{\frac{\pi t}{T_s}} \quad \text{for } -\infty < t < \infty$$

$$p(t) = 0 \quad \text{for } t = \pm T_s, \pm 2T_s, \dots$$

126

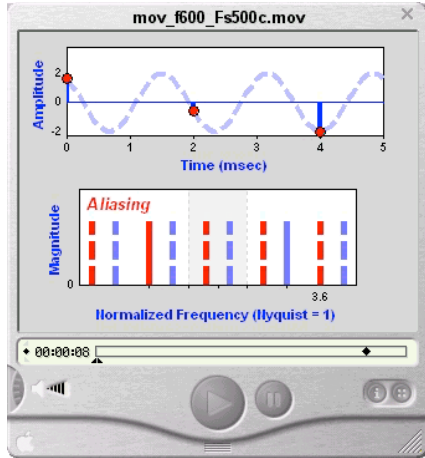
126



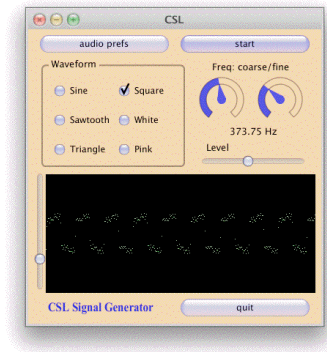
125

125

## SAMPLING DEMO (Ch. 4)



127



128

## Media Data Formats

## Media Data Formats

- ◆ Kind of signal
  - ◆ Audio, image, gesture
- ◆ Encoding
  - ◆ Assume discrete time digital representation
  - ◆ Basic rate and resolution
- ◆ Concrete format
  - ◆ Standard formats

129

## Images

- ◆ Assume uniform X/Y sampling (sample-spacing) and constant picture-element (pixel) format
- ◆ 3 variables
  - ◆ X dimension, Y dimension, Pixel depth
  - ◆ Pixel depth leads to a power of 2 # of possible colors
  - ◆ E.g., 1024 wide \* 768 high \* 8-bit deep (= 3/4 Mpixel @ 256 colors each)

130

## Image Representation

- Simple Pixmap Structures
  - int x, y, d, \*pixel\_data
- Sun RAS file format header
 

```
struct rasterfile {
 int ras_magic;
 int ras_width;
 int ras_height;
 int ras_depth;
 int ras_length;
 int ras_type;
 int ras_mapttype;
 int ras_maplength;
};
```

131

## Image Sequences and Video

- ◆ Sequence of images as a fixed frame rate
- ◆ Leads to the possibility of visual aliasing
- ◆ Many compression opportunities
- ◆ Many standard formats
  - ◆ Different resolutions
  - ◆ Different handling of color
  - ◆ Different frame rates

132

# Image / Video Standards

Format name	Lines/frame	Pixels/line	Bits/Pixel
FAX	2200	1700	1
VGA	480	640	8
XVGA	768	1024	24

Format name	Lines/frame	Pixels/line	Frames/s	Interlaced	SS scheme	Aspect ratio
CIF	288	352		N	4:2:0	4:3
QCIF	144	176		N	4:2:0	4:3
SQCIF	128	96		N	4:2:0	4:3
SIF-525	240	352	30	N	4:2:0	4:3
SIF-625	288	352	25	N	4:2:0	4:3
NTSC	486	720	29.97	Y	4:2:2	4:3
PAL	576	720	25	Y	4:2:2	4:3
HDTV	720	1280	59.94	N	4:2:0	16:9
HDTV	1080	1920	29.97	Y	4:2:0	16:9

- **Example:** what is the bit-rate for interlaced HDTV format?  
 $N_l = 1080$  lines per frame,  $N_p = 1920$  pixels per line,  $R_f = 29.97$  frames/s  
frames per second, 12 bits per pixels (luminance + subsampled chrominances):  $N_l N_p R_f \cdot 12 = 745,749,504$  bits/s.

133

# Video Formats (Rowe 1994)

## Many Video Formats

Analog - NTSC, PAL, SECAM

Digital - CCIR 601, D1-D5, HDTV

## Relative Comparison

	Image Size	Bytes/Pixel	MBytes/sec
NTSC	640x480	3	27.6
PAL	768x576	3	33.2
CCIR 601 (D2)	720x485	2	21.0
HDTV	1920x1080	3	78.0
	1280x720	3	35.0

134

# Sound

- ◆ Basic features
  - ◆ Sample rate
  - ◆ Sample size (in bits, assumed linear)
  - ◆ # of channels
- ◆ Optional: statistical or perceptual compression
- ◆ Many standard formats

135

# Sound File Header

```
typedef struct {
 int magic; /* magic number SND_MAGIC */
 int dataLocation; /* offset/pointer to the data */
 /* (header can be >28 bytes) */
 int dataSize; /* number of bytes of data */
 int dataFormat; /* the data format code */
 int samplingRate; /* the sampling rate */
 int channelCount; /* the number of channels */
 char info[4]; /* optional text information */
} SNDSoundStruct;
```

136

# Sound Formats / Bandwidths

- ◆ Surround HiFi (8 @ 24/96) 18 M b/s
- ◆ "CD-quality" (2 @ 16/44) ~ 1.4 M b/s
  - ◆ Losses (rel. to 24/96) easily heard over good systems
- ◆ MP3 (DCT, masking) ~128 k b/s (VBR)
  - ◆ Debatable losses relative to "CD-quality" over "mid-fi" systems for  $\geq 128$  k b/s bandwidth
  - ◆ Obvious losses for lower bit-rates
- ◆ RealAudio 14 k b/s, cell phone (CELP) ~ 8 k b/s
  - ◆ Monophonic, "telephone-quality"
- ◆ Best speech compression ~ 1.5 k b/s

137

# Other Media Data

- ◆ Gesture
- ◆ Biometric signals
- ◆ Environmental data
- ◆ Same issues

138

# Review

- ◆ Data representations
  - ◆ Image, sound data structures
  - ◆ Signal sampling and quantization
  - ◆ Signal magnitude and phase
  - ◆ Other properties

## Topic 4



139

140

## Topic 4

- ◆ Operations on signals
  - ◆ Signal creation and manipulation
  - ◆ Signal arithmetic and statistics
  - ◆ Signal-processing functions and flow-charts
  - ◆ Memory, difference equations, digital filters
  - ◆ Autoregression and moving-average processes

141

## Working with Signals

- ◆ Theory
  - ◆ Math of vectors
  - ◆ Trigonometry
  - ◆ Transformations
- ◆ Practice
  - ◆ Math programming languages
  - ◆ Vector/matrix data types
  - ◆ Libraries of DSP functions

142

## Operations on Signals

- ◆ Signal arithmetic & algebra
  - ◆ Create a sinusoidal signal
    - ◆  $x = \sin(t)$  for  $0 < t < 2\pi$
  - ◆ Create a step function
    - ◆  $x = \begin{cases} 0 & \text{for } t < 0 \\ 1 & \text{for } t \geq 0 \end{cases}$
- ◆ Add signals
  - ◆  $x = x1 + x2$

143

## Programming with Signals

- ◆ “Mainstream” programming languages (C/C++...)
  - ◆ Signal buffer stored as `int[]`, `float[]`, `int[][]`, etc.
  - ◆ Signal operations written as `for()` loops
    - for ( $i = 0$ ;  $i < \text{signal\_length}$ ;  $i++$ )**
      - `sigY[i] = sigX1[i] + sigX2[i];`**
  - ◆ Many kinds of function/class libraries and APIs for signal processing (TNT, CSL, ...)

144

# Programming with Signals

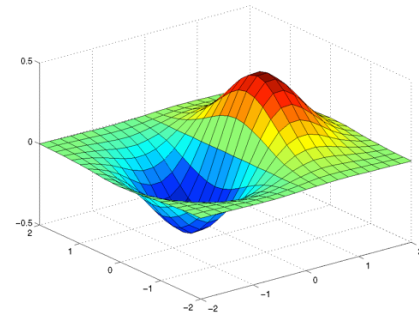
- ◆ “Special-purpose” programming languages  
(MATLAB, Mathematica, Maple, ...)
- ◆ Everything (including signal buffers) is stored as high-level n-dim. vectors **t = 0 : 3 : 10; y = sin(t)**
- ◆ Signal operations written as simple arithmetic operations **y = x1 + x2**
- ◆ Many kinds of function/class libraries for DSP

145

# MATLAB Examples

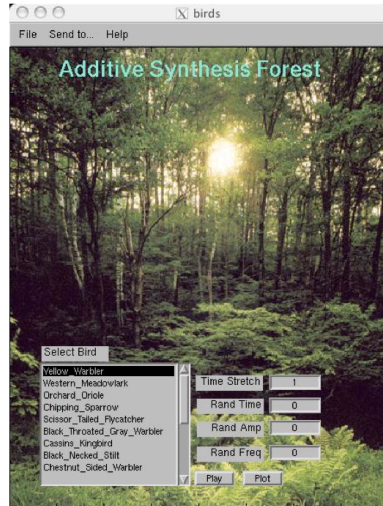
Here is how we graph the function  $z(x,y) = x \exp(-x^2 - y^2)$ :

```
>> [x,y] = meshgrid(-2:.2:2, -2:.2:2);
>> z = x .* exp(-x.^2 - y.^2);
>> surf(x,y,z)
```



146

## SSUM Additive Synthesis



147

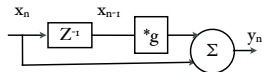
# Basic DSP Techniques

- ◆ Filters
  - ◆ Change the ratios of different (visual/ audio) frequencies
- ◆ Transforms
  - ◆ Convert signals between representation domains
- ◆ Getting started
  - ◆ Look at simple delay-based filters

148

# Filters: DSP with Delays

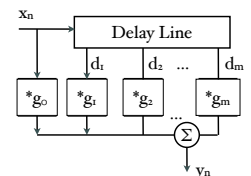
- ◆ Simplest Example
  - ◆ Forward Sample Averager
    - ◆  $y_n = (x_n + x_{n-1}) / 2$  (or)
    - ◆  $y_n = (x_n + (g * x_{n-1}))$
  - ◆ “Function smoother”
  - ◆ = Low-pass filter
- ◆ Sample-averaging player app



149

# Using Multiple Delays

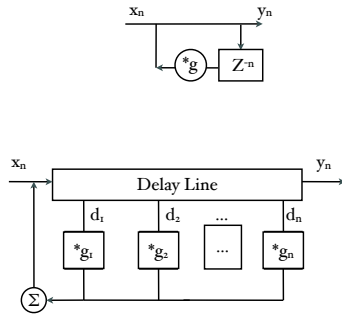
- ◆ General case: Multi-tap delay line
 
$$y_n = (g_0 * x_n) + (g_1 * x_{n-d_1}) + \dots + (g_m * x_{n-d_m})$$
- ◆ Short delays: filter
- ◆ Long delays: reverberator



150

## Past Output Sample Averager

- ◆ Feedback loop
  - ◆  $y_n = (x_n + g * y_{n-1})$
  - ◆ = comb filter
- ◆ General form: feedback delay line



151

## Filter Classes

- ◆ Feed-forward delay-lines
  - ◆ Play an impulse through it, output ends at most  $d_m$  samples later
  - ◆ = Finite impulse response (**FIR**) filter
- ◆ Feed-back delay-lines
  - ◆ Play an impulse through it, output recirculates “forever”
  - ◆ = Infinite impulse response (**IIR**) filter

152

## Filter Classes

- ◆ IIR Filters (Feedback systems)
  - ◆ Speed/space efficient and flexible
- ◆ FIR Filters (Delay lines)
  - ◆ Impulse response is simply the coefficients
  - ◆ Phase is linear if coefficients are symmetrical (i.e.,  $b_0 = b_n, b_1 = b_{n-1}$ , etc)

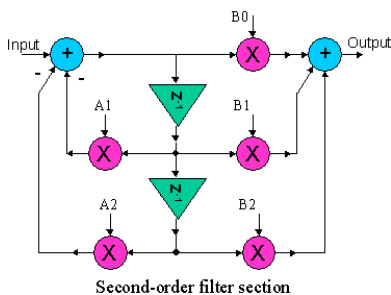
153

## General Form

- ◆ Mix of feed-forward and feed-back terms (poles and zeros)
- ◆ Feed-forward weighted averager = 1 zero
 
$$y(n) = A_0 * x(n) + A_1 * x(n-1)$$
- ◆ Feed-back = comb filter = 1 pole
 
$$y(n) = A_0 * x(n) - B_1 * y(n-1)$$
- ◆ General 2-pole/2-zero
 
$$y[n] = g_0 * x[n] + g_1 * x[n-1] + g_2 * x[n-2] - h_1 * y[n-1] - h_2 * y[n-2]$$

154

## 2nd-Order General Form



$$y(z) = x \prod_1^n \frac{1 - a_1 z^{-1} - a_2 z^{-2}}{b_0 + b_1 z^{-1} + b_2 z^{-2}}$$

155

## General Form to Z-form

- ◆  $Y[n]$  polynomial
  - ◆ Substitute  $z$  for  $n-1$ ,  $z^2$  for  $n-2$ , etc.
  - ◆ Solve polynomial for roots in  $z$
- ◆ Example IIR system
 
$$y[n] = x[n] + k * y[n-1]$$
- ◆ Expressed in Z:
 
$$Y(z) = X(z) + K * (Y(z) * z^{-1})$$

$$H(z) = Y(z) / X(z) = 1 / (1 - kz^{-1}) = z / (z - k)$$
- ◆ This is 0 for  $z = 0$  and infinite for  $z = k$
- ◆ These are the 1 zero and 1 pole of the function

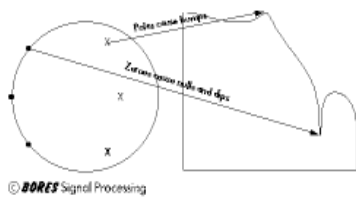
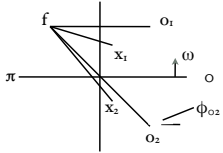
156

# Pole-Zero Diagrams

- ◆ Z-plane and the unit circle (rubber sheet analogy)
- ◆ Poles and zeros as roots of xfer function

$$|H(\omega)| = \frac{\text{Product of vectors to zeros}}{\text{Product of vectors to poles}}$$

$$\Theta(H(\omega)) = \sum \text{zero angles} - \sum \text{pole angles}$$

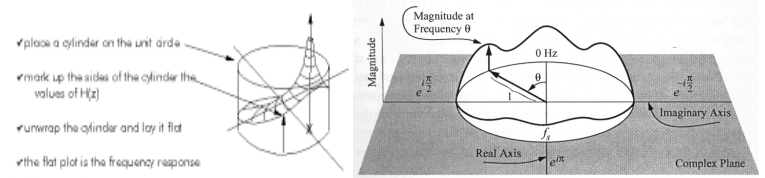


© BORES Signal Processing

157

# Pole-Zero Diagram Examples

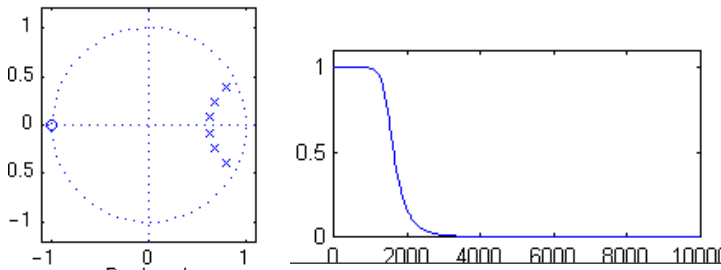
- ◆ See DSP Skriptum p. 80 (or other reference)
- ◆ Other topics
  - ◆ Frequency transformation (see below)

Figure 5.17  
Complex frequency response.

158

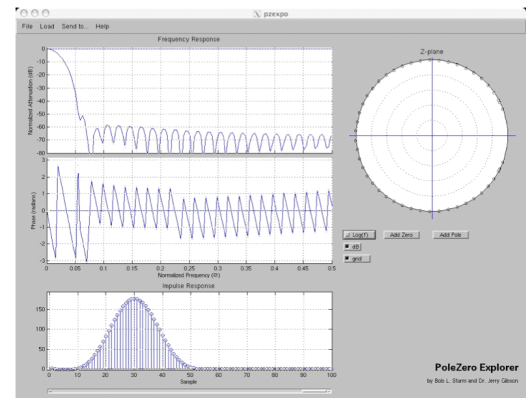
Example: 6-pole low-pass Butterworth Filter (wmin.ac.uk)

- ◆  $F_s = 20 \text{ kHz}$ ,  $F_0 = 1500 \text{ Hz}$
- ◆ Designed using MATLAB



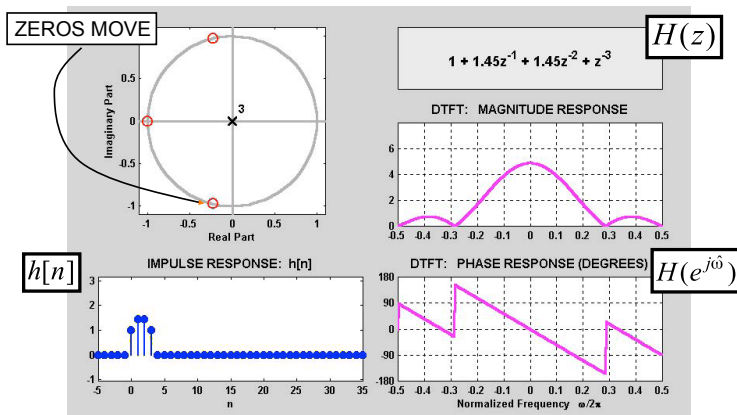
159

# SSUM P/Z Explorer



160

# 3 DOMAINS MOVIE: FIR



161

# NULLING PROPERTY of $H(z)$

- When  $H(z)=0$  on the unit circle.
  - Find inputs  $x[n]$  that give zero output

$$H(z) = 1 - 2z^{-1} + 2z^{-2} - z^{-3}$$

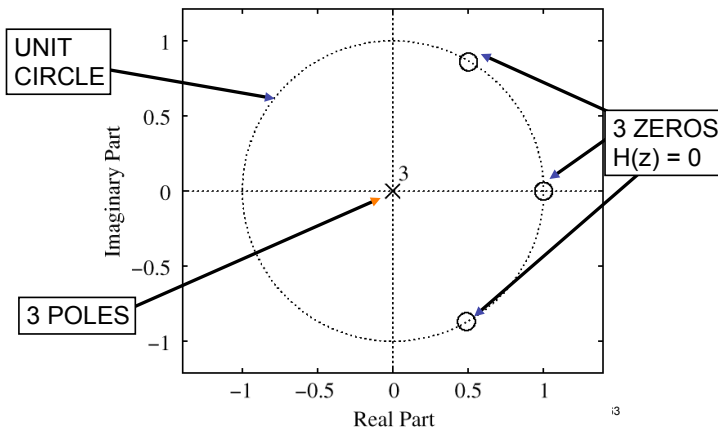
$$H(e^{j\omega}) = 1 - 2e^{-j\omega} + 2e^{-j2\omega} - e^{-j3\omega}$$

$$x[n] = e^{j(\pi/3)n} \rightarrow H(z) \rightarrow y[n] = H(e^{j\pi/3}) \cdot e^{j(\pi/3)n}$$

162

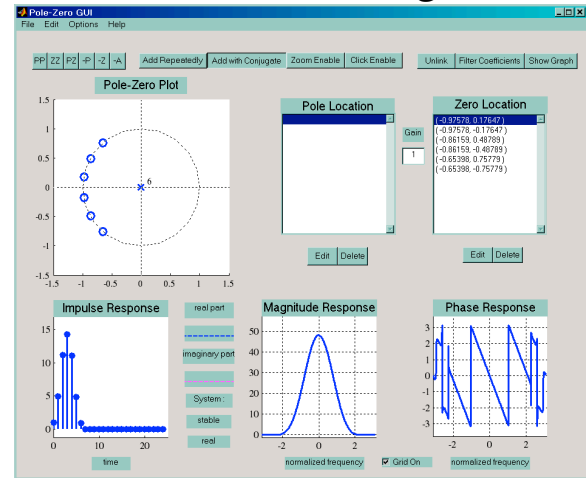
162

## PLOT ZEROS in z-DOMAIN



163

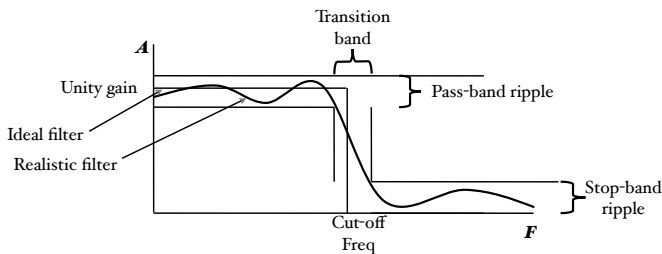
## PeZ Demo: Zero Placing



164

## Real-world Filters

- ◆ General filter design methods and parameters



165

## Complex Filters

- ◆ Two Options
  - ◆ Higher-order filters
  - ◆ Series/Parallel combinations of simple filter
- ◆ Some design programs decompose, some combine
- ◆ Parametric Filters
  - ◆ Some topologies allow Q to be separated from  $F_0$

166

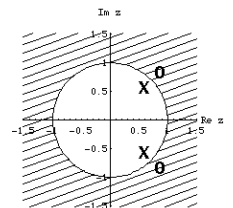
## Problems with Digital Filters

- ◆ Numerical stability (poles on the unit circle are very very bad, lead to ringing)
- ◆ Round-off errors are everywhere (floats are limited precision)
- ◆ Efficiency (higher-order filters may be slower than FFT/IFFT combination)

167

## All-Pass Filters (?)

- ◆ If the p/z pairs are each on the same radius (symmetrical on the unit circle), the frequency response remains flat, but one can design an arbitrary phase/delay response.
- ◆ This is very useful.



168

## FIR Spec Example

Specifications:  $\omega_p = 0.33\pi$ ,  
 $\omega_s = 0.67\pi$ ,  $\alpha_p = 0.01$ ,  
 $\alpha_s = 0.01$

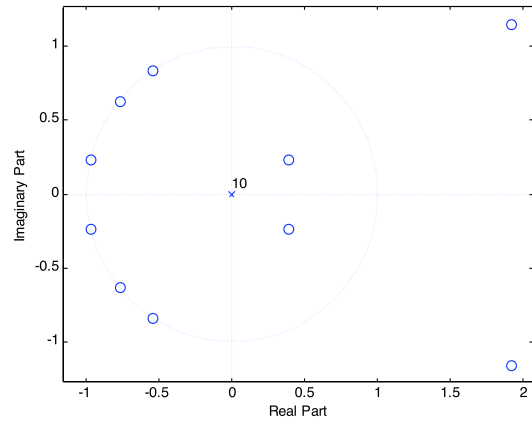
*Design of Constant-Coefficient Linear-Phase FIR Filter  
 Using the Remez Exchange Technique in MatLab  
 (OK State ECEN 377)*

```
[n,f0,a0,w]=remezord([.33 .67],[1 0],[.01 .01])
b=remez(n,f0,a0,w);
[h,w]=freqz(b,1,256);
hd=20*log10(abs(h));
zplane(b)
figure(2)
plot(w/pi,hd)
```

**Output**  
 $n = 10$ ,  $f0 = [0.33 \ .67 \ 1]$ ,  $a0 = [1 \ 1 \ 0 \ 0]$ ,  $w = [1 \ 1]$   
 $b = [0.0248 \ -0.0000 \ -0.0767 \ 0.0000 \ 0.3074$   
 $0.5000 \ 0.3074 \ 0.0006 \ -7.6719 \ -0.0046 \ 2.4776]$   
**NOTE:** Coeff. that appear as 0 are small non-0.

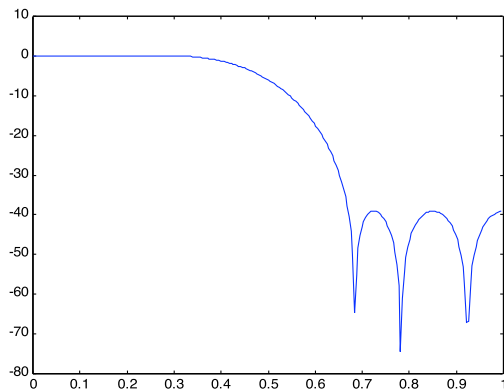
169

## Pole-Zero Diagram for Remez Filter Example



170

## Response Spectrum for Remez Filter Example



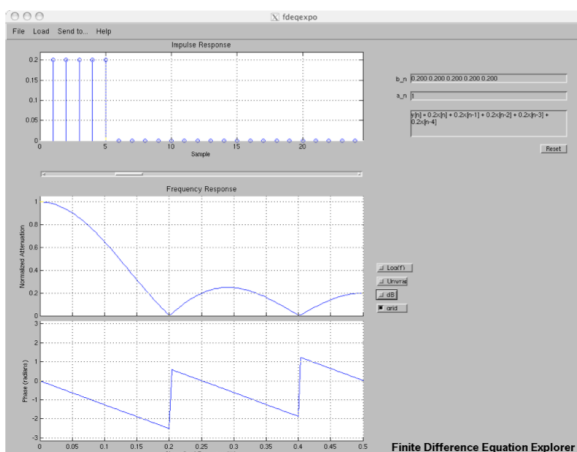
171

## Form a cascade of second-order sections

```
z=roots(b);
b1=poly([z(1) z(2)]); [1.0000 -3.8231 4.9915]
k1=b1(1)+b1(2)+b1(3); k1 = 2.1684
b2=poly([z(3) z(4)]); [1.0000 1.9429 1.0000]
k2=b2(1)+b2(2)+b2(3); k2 = 3.9429
b3=poly([z(5) z(6)]); [1.0000 1.5487 1.0000]
k3=b3(1)+b3(2)+b3(3); k3 = 3.5487
b4=poly([z(7) z(8)]); [1.0000 1.0955 1.0000]
k4=b4(1)+b4(2)+b4(3); k4 = 3.0955
b5=poly([z(9) z(10)]); [1.0000 -0.7659 0.2003]
k5=b5(1)+b5(2)+b5(3); k5 = 0.4344
```

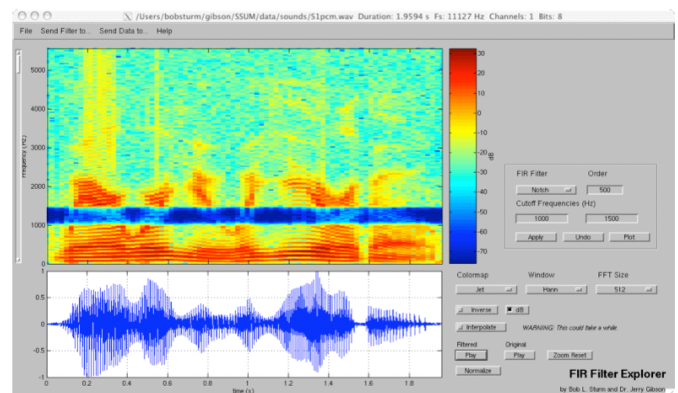
172

## SSUM FinDiffEq Explorer



173

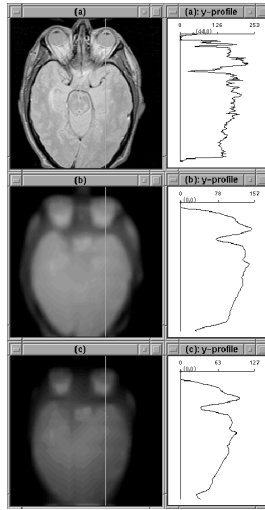
## SSUM FIR Filters



174

# N-Dimensional Filters

- ◆ Image filtering
  - ◆ Filter in the X and Y dimensions separately
  - ◆ Low-pass = smoothing
  - ◆ High-pass = edge-sharpen



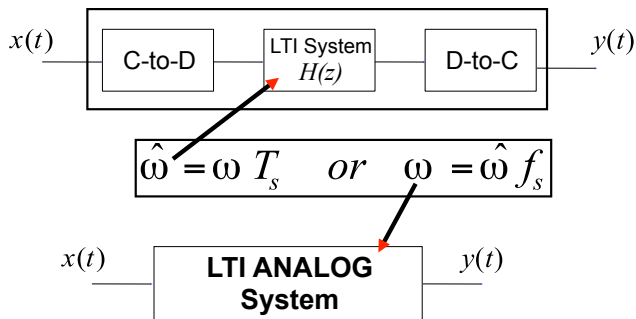
175

# Complex Filters

- ◆ Long FIR delay lines can be very computationally intensive
- ◆ Solve with convolution in the frequency domain (see below)
- ◆ Image-processing: “kernel” matrices

176

## D-T Filtering of C-T Signals



177

177

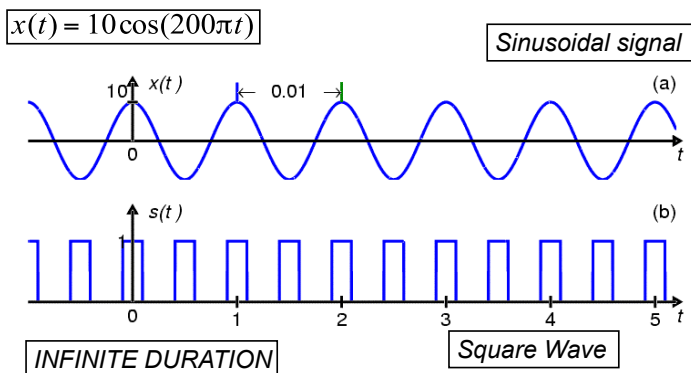
## ANALOG SIGNALS $x(t)$

- INFINITE LENGTH
  - SINUSOIDS: (t = time in secs)
  - PERIODIC SIGNALS
  - ONE-SIDED, e.g., for  $t > 0$
  - UNIT STEP:  $u(t)$
- FINITE LENGTH
  - SQUARE PULSE
- IMPULSE SIGNAL:  $\delta(t)$
- DISCRETE-TIME:  $x[n]$  is list of numbers

178

178

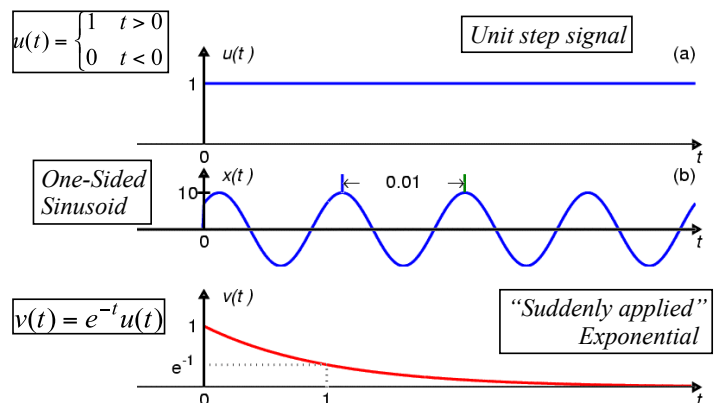
## CT Signals: PERIODIC



179

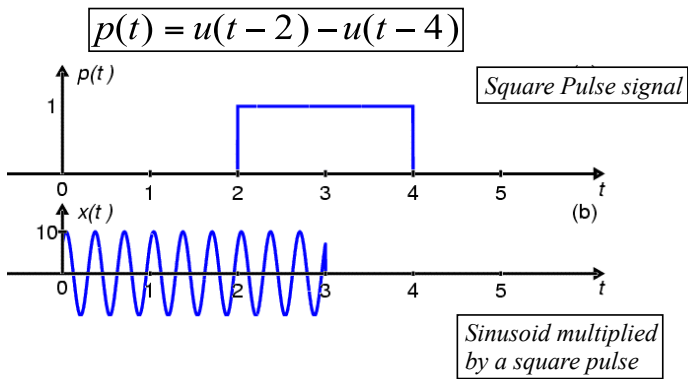
179

## CT Signals: ONE-SIDED



180

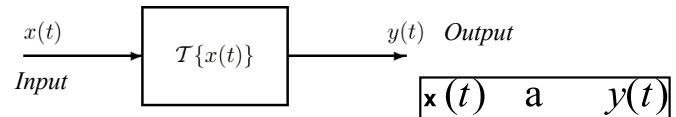
## CT Signals: FINITE LENGTH



181

181

## Continuous-Time Systems



### Examples:

#### Delay

Modulator  $y(t) = x(t - t_d)$

#### Integrator

$$y(t) = [A + x(t)] \cos \omega_c t$$

$$y(t) = \int_{-\infty}^t x(\tau) d\tau$$

182

182

## CT BUILDING BLOCKS

- INTEGRATOR (CIRCUITS)
- DIFFERENTIATOR
- DELAY by  $t_0$
- MODULATOR (e.g., AM Radio)
- MULTIPLIER & ADDER

183

183

## Ideal Delay:

### Mathematical Definition:

$$y(t) = x(t - t_d)$$

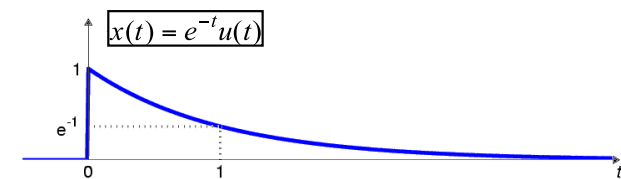
- To find the IMPULSE RESPONSE,  $h(t)$ , let  $x(t)$  be an impulse, so

$$h(t) = \delta(t - t_d)$$

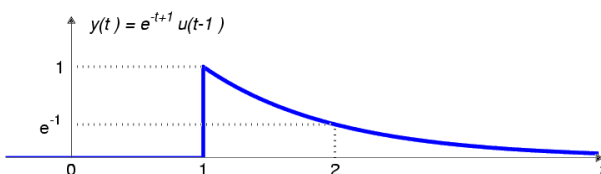
184

184

## Output of Ideal Delay of 1 sec



$$y(t) = x(t-1) = e^{-(t-1)}u(t-1)$$



185

## Integrator:

### Mathematical Definition:

$$y(t) = \int_{-\infty}^t x(\tau) d\tau$$

Running Integral

- To find the IMPULSE RESPONSE,  $h(t)$ , let  $x(t)$  be an impulse, so

$$h(t) = \int_{-\infty}^t \delta(\tau) d\tau = u(t)$$

186

186

**Integrator:**  $y(t) = \int_{-\infty}^t x(\tau) d\tau$

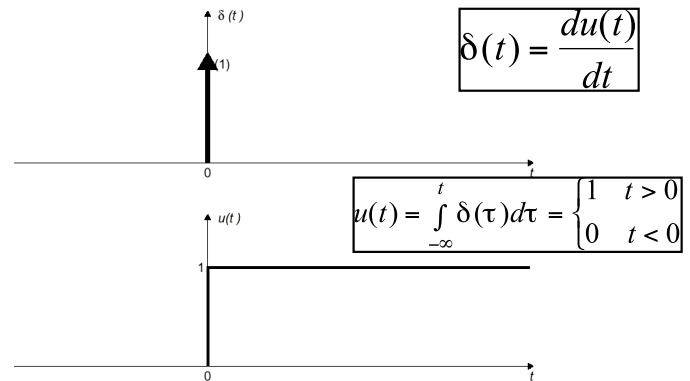
- Integrate the impulse

$$\int_{-\infty}^t \delta(\tau) d\tau = u(t)$$

- IF  $t < 0$ , we get zero
- IF  $t > 0$ , we get one
  - Thus we have  $h(t) = u(t)$  for the integrator

187

## Graphical Representation

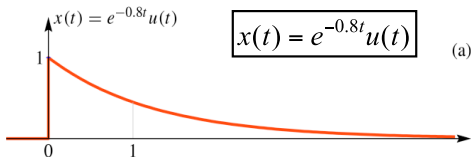


188

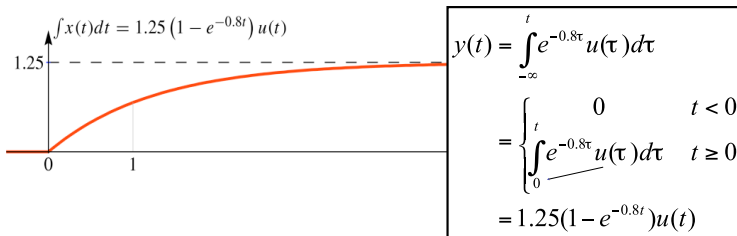
188

## Output of Integrator

$$y(t) = \int_{-\infty}^t x(\tau) d\tau = x(t) * u(t)$$



(a)



189

## Differentiator:

- Mathematical Definition:

$$y(t) = \frac{dx(t)}{dt}$$

- To find  $h(t)$ , let  $x(t)$  be an impulse, so

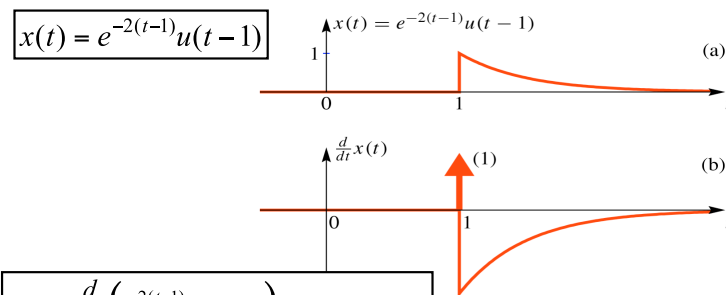
$$h(t) = \frac{d\delta(t)}{dt} = \delta^{(1)}(t) \quad \text{Doublet}$$

190

190

## Differentiator Output:

$$y(t) = \frac{dx(t)}{dt}$$



(a)

(b)

$$\begin{aligned} y(t) &= \frac{d}{dt} (e^{-2(t-1)} u(t-1)) \\ &= -2e^{-2(t-1)} u(t-1) + e^{-2(t-1)} \delta(t-1) \\ &= -2e^{-2(t-1)} u(t-1) + 1\delta(t-1) \end{aligned}$$

191

## Linear and Time-Invariant (LTI) Systems

- If a continuous-time system is both linear and time-invariant, then the output  $y(t)$  is related to the input  $x(t)$  by a **convolution integral**

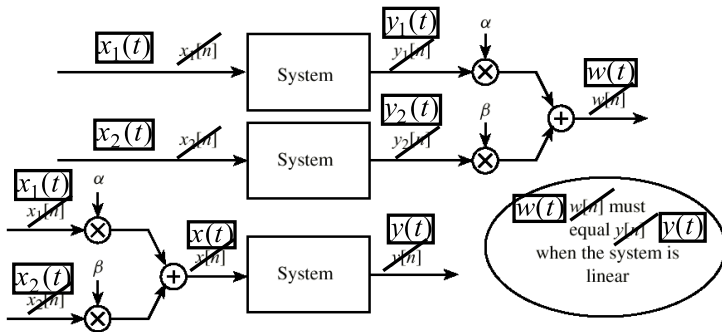
$$y(t) = \int_{-\infty}^{\infty} x(\tau) h(t - \tau) d\tau = x(t) * h(t)$$

where  $h(t)$  is the **impulse response** of the system.

192

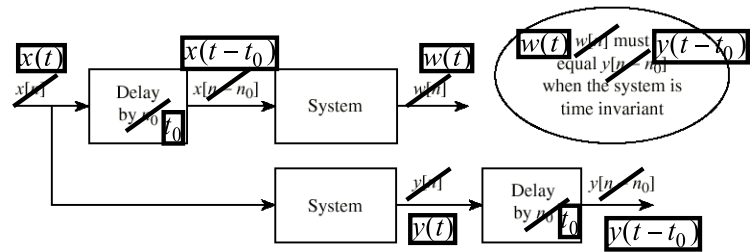
192

## Testing for Linearity



193

## Testing Time-Invariance



194

194

**Integrator:** 
$$y(t) = \int_{-\infty}^t x(\tau) d\tau$$

- Linear

$$\int_{-\infty}^t [ax_1(\tau) + bx_2(\tau)] d\tau = ay_1(t) + by_2(t)$$

- And Time-Invariant

$$w(t) = \int_{-\infty}^t x(\tau - t_0) d\tau \quad \text{let } \sigma = \tau - t_0$$

$$\Rightarrow w(t) = \int_{-\infty}^{t-t_0} x(\sigma) d\sigma = y(t-t_0)$$

195

195

**Modulator:** 
$$y(t) = [A + x(t)] \cos \omega_c t$$

- **Not** linear--obvious because

$$[A + ax_1(t) + bx_2(t)] \neq$$

$$[A + ax_1(t)] + [A + bx_2(t)]$$

- **Not** time-invariant

$$w(t) = [A + x(t-t_0)] \cos \omega_c t \neq y(t-t_0)$$

196

196

## DIGITAL FILTERING

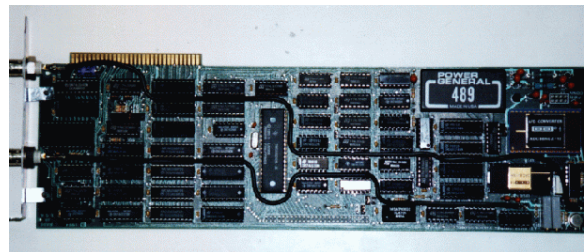


- CONCENTRATE on the COMPUTER
  - PROCESSING ALGORITHMS
  - SOFTWARE (MATLAB)
  - HARDWARE: DSP chips, VLSI
- DSP: DIGITAL SIGNAL PROCESSING

197

197

## The TMS32010, 1983



First PC plug-in board from Atlanta Signal Processors Inc.

198

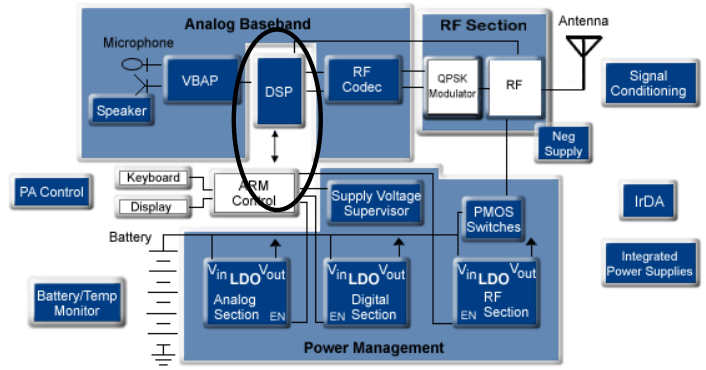
198

## Rockland Digital Filter, 1971



For the price of a small house, you could have one of these.

## Digital Cell Phone (ca. 2000)



Modern designs include MM DSPs and video

## DISCRETE-TIME SYSTEM



- OPERATE on  $x[n]$  to get  $y[n]$
- WANT a **GENERAL** CLASS of SYSTEMS
  - **ANALYZE** the SYSTEM
    - TOOLS: TIME-DOMAIN & FREQUENCY-DOMAIN
  - **SYNTHESIZE** the SYSTEM

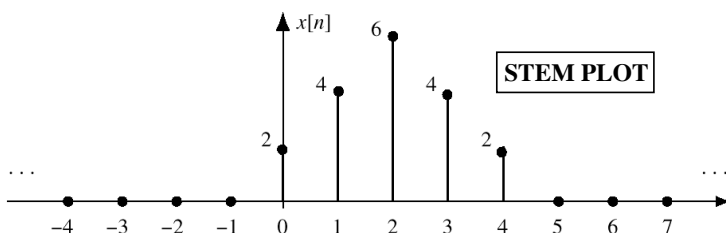
## D-T SYSTEM EXAMPLES



- EXAMPLES:
  - POINTWISE OPERATORS
    - SQUARING:  $y[n] = (x[n])^2$
  - RUNNING AVERAGE
    - RULE: "the output at time  $n$  is the average of three consecutive input values"

## DISCRETE-TIME SIGNAL

- $x[n]$  is a LIST of NUMBERS
  - INDEXED by " $n$ "



## 3-PT AVERAGE SYSTEM

- ADD 3 CONSECUTIVE NUMBERS
  - Do this for each " $n$ "

the following input-output equation

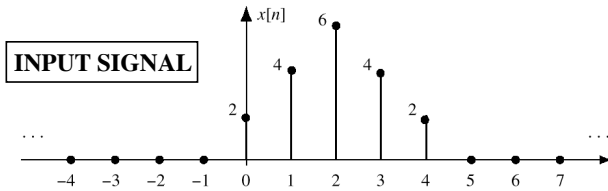
**Make a TABLE**

$$y[n] = \frac{1}{3}(x[n] + x[n+1] + x[n+2])$$

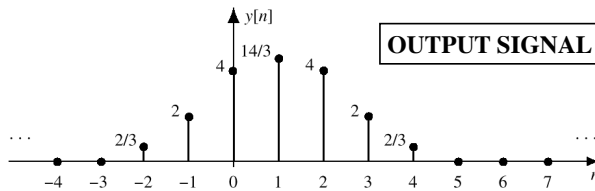
$n$	$n < -2$	-2	-1	0	1	2	3	4	5	$n > 5$
$x[n]$	0	0	0	2	4	6	4	2	0	0
$y[n]$	0	$\frac{2}{3}$	2	4	$\frac{14}{3}$	4	2	$\frac{2}{3}$	0	0

$$n=0 \quad y[0] = \frac{1}{3}(x[0] + x[1] + x[2])$$

$$n=1 \quad y[1] = \frac{1}{3}(x[1] + x[2] + x[3])$$

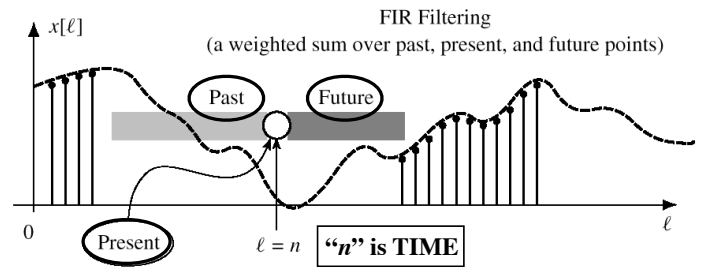
Figure 5.2 Finite-length input signal,  $x[n]$ .

$$y[n] = \frac{1}{3}(x[n] + x[n+1] + x[n+2])$$

Figure 5.3 Output of running average,  $y[n]$ .

## PAST, PRESENT, FUTURE

Sec. 5.2 The Running Average Filter 123

Figure 5.4 The running-average filter calculation at time index  $n$  uses values within a sliding window (shaded). Dark shading indicates the future ( $\ell > n$ ); light shading, the past ( $\ell < n$ ).

205

206

## ANOTHER 3-pt AVERAGER

- Uses "PAST" VALUES of  $x[n]$ 
  - IMPORTANT IF "n" represents REAL TIME
    - WHEN  $x[n]$  &  $y[n]$  ARE STREAMS

$$y[n] = \frac{1}{3}(x[n] + x[n-1] + x[n-2])$$

$n$	$n < -2$	-2	-1	0	1	2	3	4	5	6	7	$n > 7$
$x[n]$	0	0	0	2	4	6	4	2	0	0	0	0
$y[n]$	0	0	0	$\frac{2}{3}$	2	4	$\frac{14}{3}$	4	2	$\frac{2}{3}$	0	0

207

207

## GENERAL FIR FILTER

- FILTER COEFFICIENTS  $\{b_k\}$

- DEFINE THE FILTER

$$y[n] = \sum_{k=0}^M b_k x[n-k]$$

- For example,

$$b_k = \{3, -1, 2, 1\}$$

$$y[n] = \sum_{k=0}^3 b_k x[n-k] = 3x[n] - x[n-1] + 2x[n-2] + x[n-3]$$

208

208

## GENERAL FIR FILTER

- FILTER COEFFICIENTS  $\{b_k\}$

$$y[n] = \sum_{k=0}^M b_k x[n-k]$$

- FILTER **ORDER** is  $M$
- FILTER **LENGTH** is  $L = M+1$ 
  - NUMBER of FILTER COEFFS is  $L$

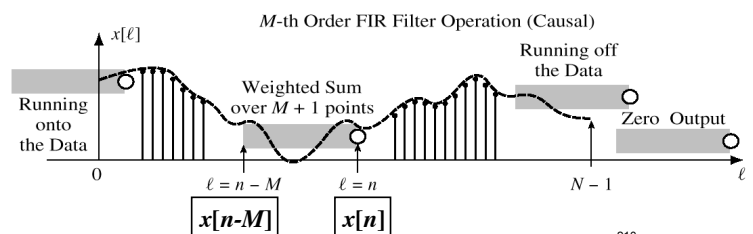
209

209

## GENERAL FIR FILTER

- SLIDE a WINDOW across  $x[n]$

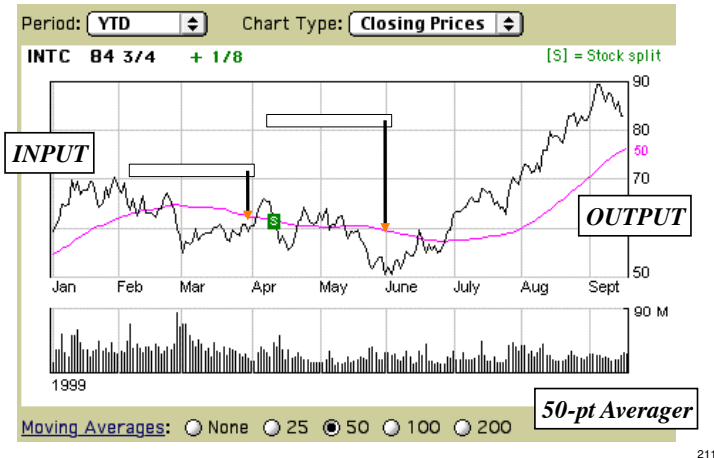
$$y[n] = \sum_{k=0}^M b_k x[n-k]$$



210

210

## FILTERED STOCK SIGNAL



211

## 4-pt AVERAGER

- CAUSAL SYSTEM: USE PAST VALUES

$$y[n] = \frac{1}{4}(x[n] + x[n-1] + x[n-2] + x[n-3])$$

- INPUT = UNIT IMPULSE SIGNAL =  $\delta[n]$

$$x[n] = \delta[n]$$

$$y[n] = \frac{1}{4}\delta[n] + \frac{1}{4}\delta[n-1] + \frac{1}{4}\delta[n-2] + \frac{1}{4}\delta[n-3]$$

- OUTPUT is called "IMPULSE RESPONSE"

$$h[n] = \{\dots, 0, 0, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0, 0, \dots\}$$

212

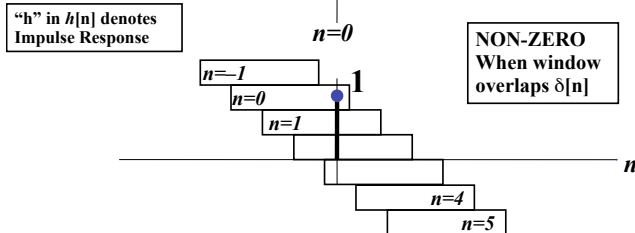
212

## 4-pt Avg Impulse Response

$$y[n] = \frac{1}{4}(x[n] + x[n-1] + x[n-2] + x[n-3])$$

$\delta[n]$  "READS OUT" the FILTER COEFFICIENTS

$$h[n] = \{\dots, 0, 0, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0, 0, \dots\}$$



213

213

## FIR IMPULSE RESPONSE

- Convolution = Filter Definition
  - Filter Coeffs = Impulse Response

$n$	$n < 0$	0	1	2	3	...	$M$	$M+1$	$n > M+1$
$x[n] = \delta[n]$	0	1	0	0	0	0	0	0	0
$y[n] = h[n]$	0	$b_0$	$b_1$	$b_2$	$b_3$	...	$b_M$	0	0

$$y[n] = \sum_{k=0}^M b_k x[n-k] \quad y[n] = \sum_{k=0}^M h[k] x[n-k]$$

**CONVOLUTION**

214

214

## FILTERING EXAMPLE

- 7-point AVERAGER

- Removes cosine

- By making its amplitude (A) smaller

$$y_7[n] = \sum_{k=0}^6 \left(\frac{1}{7}\right) x[n-k]$$

- 3-point AVERAGER

- Changes A slightly

$$y_3[n] = \sum_{k=0}^2 \left(\frac{1}{3}\right) x[n-k]$$

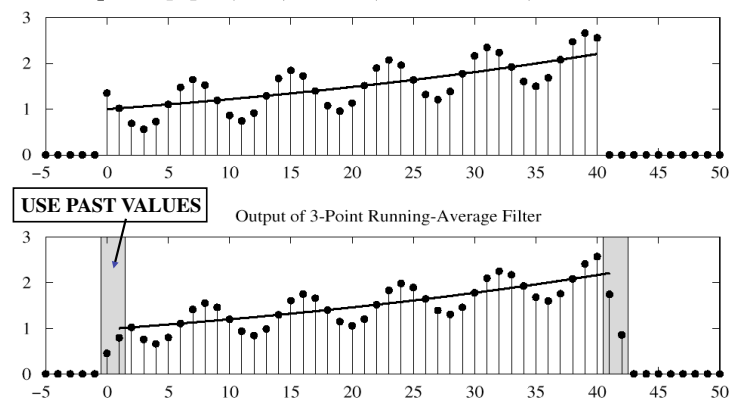
215

215

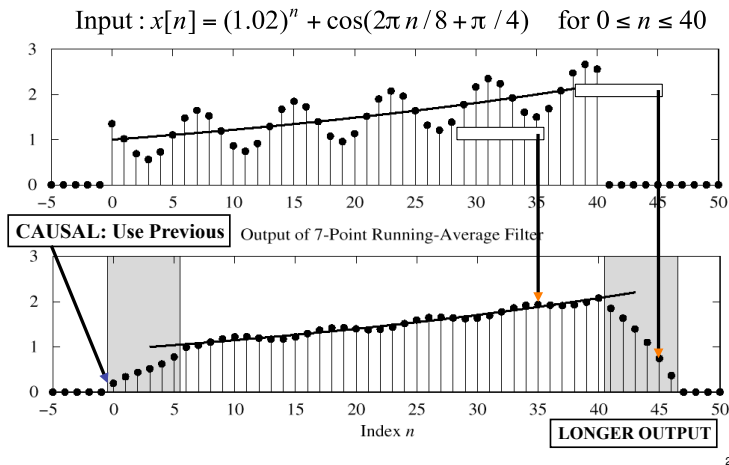
216

## 3-pt AVG EXAMPLE

Input :  $x[n] = (1.02)^n + \cos(2\pi n/8 + \pi/4)$  for  $0 \leq n \leq 40$



## 7-pt FIR EXAMPLE (AVG)



## FREQUENCY RESPONSE

- At each frequency, we can DEFINE

$$H(e^{j\hat{\omega}}) = \sum_{k=0}^M b_k e^{-j\hat{\omega} k} \quad \leftarrow \text{FREQUENCY RESPONSE}$$

- Complex-valued formula
  - Has MAGNITUDE vs. frequency
  - And PHASE vs. frequency
- Notation:  $H(e^{j\hat{\omega}})$  in place of  $H(\hat{\omega})$

218

218

## MATLAB: FREQUENCY RESPONSE

- HH = freqz(bb,1,ww)**
  - VECTOR **bb** contains Filter Coefficients
  - SP-First: **HH = freqz(bb,1,ww)**
- FILTER COEFFICIENTS  $\{b_k\}$

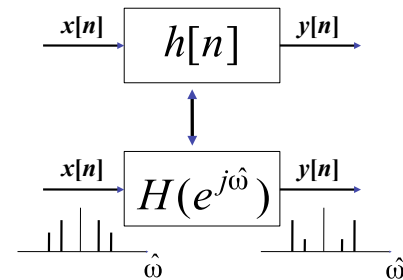
$$H(e^{j\hat{\omega}}) = \sum_{k=0}^M b_k e^{-j\hat{\omega} k}$$

219

219

## BLOCK DIAGRAMS

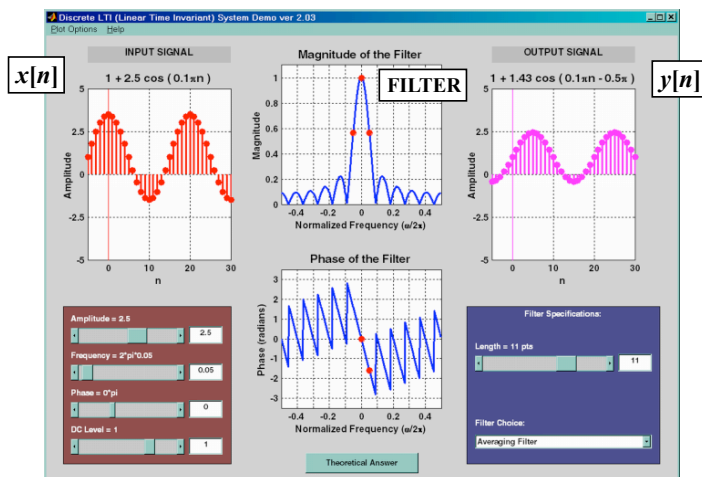
- Equivalent Representations



220

220

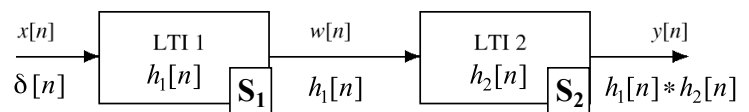
## DLTI Demo with Sinusoids



221

## CASCADE SYSTEMS

- Does the order of  $S_1$  &  $S_2$  matter?
  - NO, LTI SYSTEMS can be rearranged !!!
  - WHAT ARE THE FILTER COEFFS?  $\{b_k\}$
  - WHAT is the overall FREQUENCY RESPONSE ?

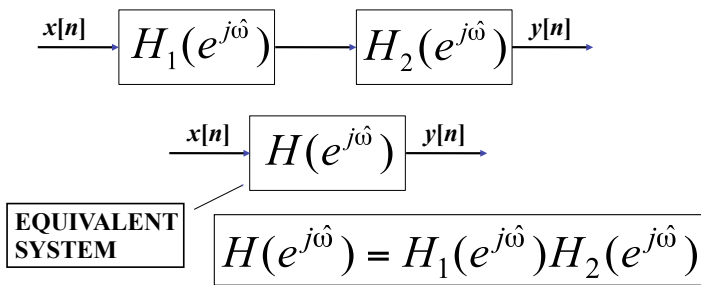


222

222

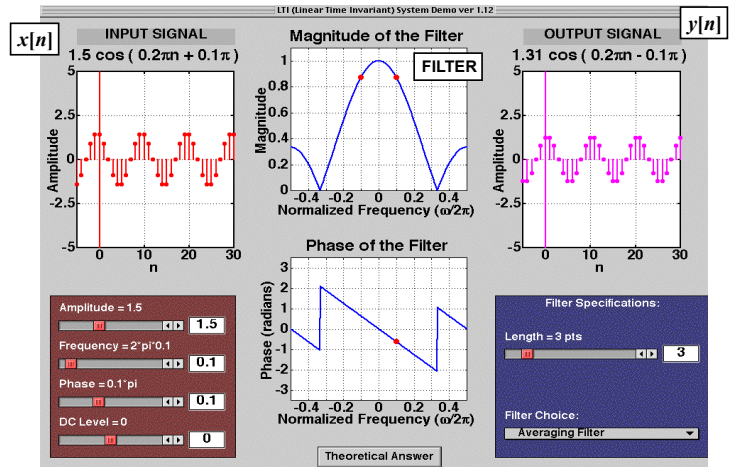
## CASCADE EQUIVALENT

- MULTIPLY the Frequency Responses



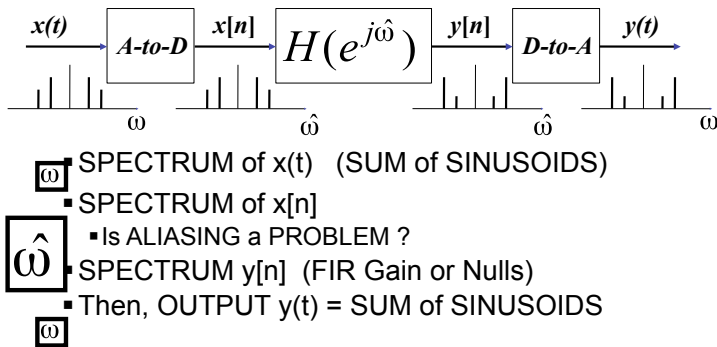
223

## LTI Demo with Sinusoids



224

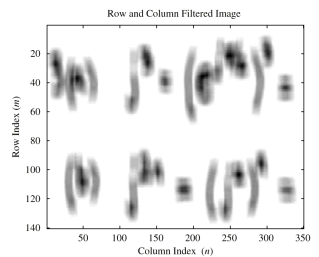
## DIGITAL "FILTERING"



225

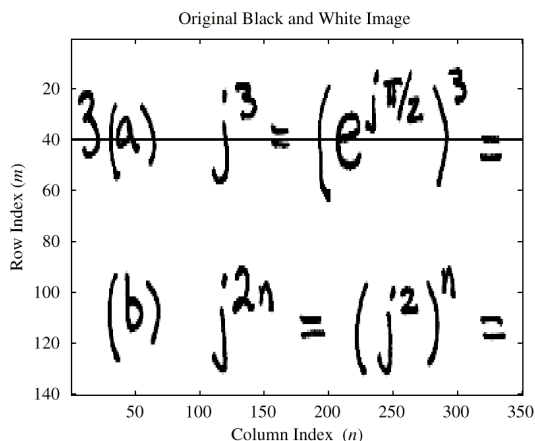
## FILTER TYPES

- LOW-PASS FILTER (**LPF**)
  - BLURRING
  - ATTENUATES HIGH FREQUENCIES
- HIGH-PASS FILTER (**HPF**)
  - SHARPENING for IMAGES
  - BOOSTS THE HIGHS
  - REMOVES DC
- BAND-PASS FILTER (**BPF**)



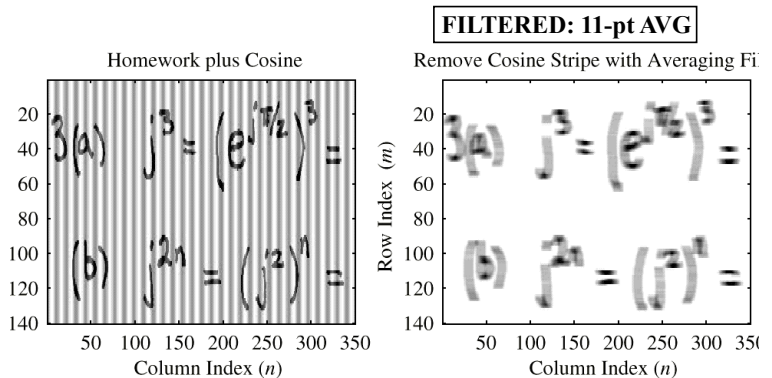
226

## B & W IMAGE



227

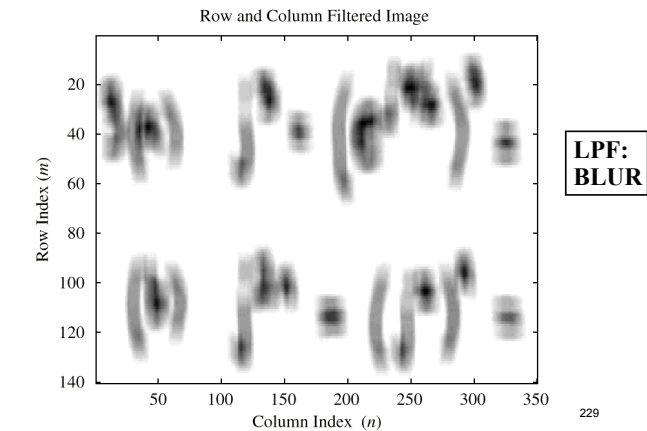
## B&W IMAGE with COSINE



228

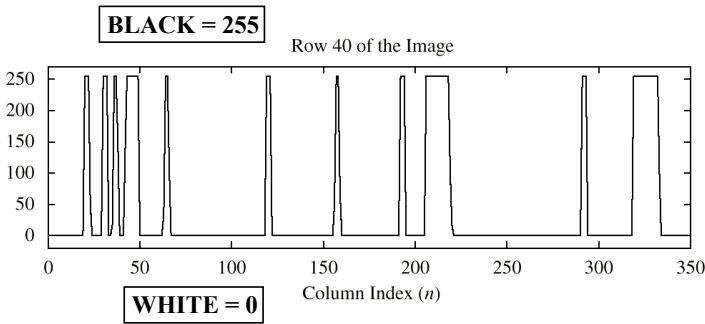
228

FILTERED B&W IMAGE



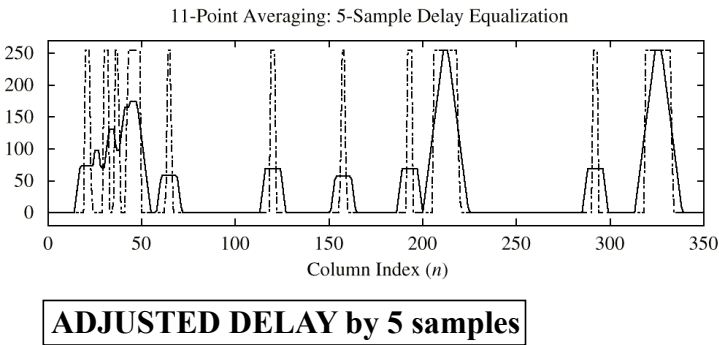
229

ROW of B&W IMAGE



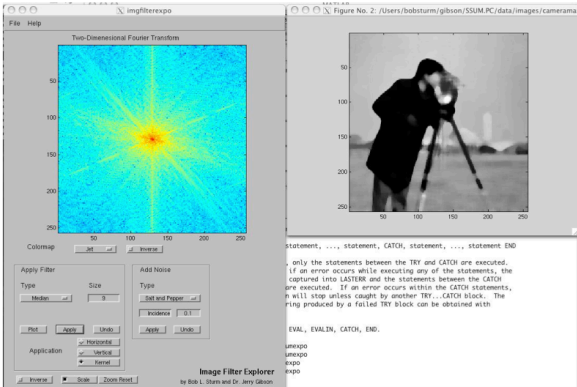
230

FILTERED ROW of IMAGE



231

SSUM Image Filtering

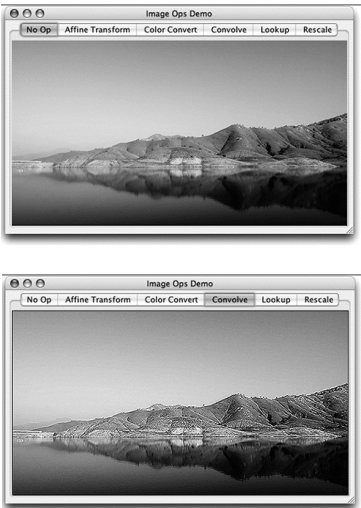


232

Image Sharpening

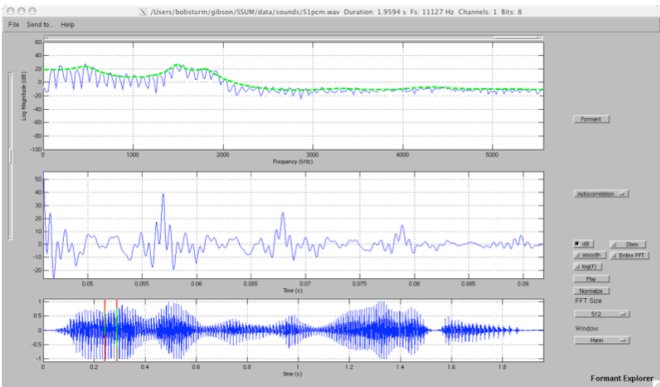
Using a 3\*3 pixel convolution kernel

$$\begin{bmatrix} 0.0f & -1.0f & 0.0f \\ -1.0f & 5.0f & -1.0f \\ 0.0f & -1.0f & 0.0f \end{bmatrix}$$



233

SSUM Formant Filters



234

## CONVOLUTION Example

n	0	1	2	3	4	5	6	7	8
x[n]	2	4	6	4	2				
h[n]	3	-1	2	1					
h[0]x[n-0]	6	12	18	12	6				
h[1]x[n-1]		-2	-4	-6	-4	-2			
h[2]x[n-2]			4	8	12	8	4		
h[3]x[n-3]				2	4	6	4	2	
y[n]	6	10	18	16	18	12	8	2	

## MATLAB for FIR FILTER

▪ **yy = conv(bb,xx)**

▪ VECTOR **bb** contains Filter Coefficients

▪ DSP-First: **yy = firfilt(bb,xx)**

▪ FILTER COEFFICIENTS {b<sub>k</sub>}

**conv2()**  
for images

$$y[n] = \sum_{k=0}^M b_k x[n-k]$$

236

236

## SPECIAL INPUT SIGNALS

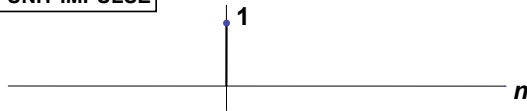
▪ x[n] = SINUSOID

**FREQUENCY RESPONSE**

▪ x[n] has only one NON-ZERO VALUE

$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$

**UNIT-IMPULSE**



237

## FIR IMPULSE RESPONSE

▪ Convolution = Filter Definition

▪ Filter Coeffs = Impulse Response

n	n < 0	0	1	2	3	...	M	M + 1	n > M + 1
x[n] = δ[n]	0	1	0	0	0	0	0	0	0
y[n] = h[n]	0	b <sub>0</sub>	b <sub>1</sub>	b <sub>2</sub>	b <sub>3</sub>	...	b <sub>M</sub>	0	0

$$h[n] = \sum_{k=0}^M b_k \delta[n-k]$$

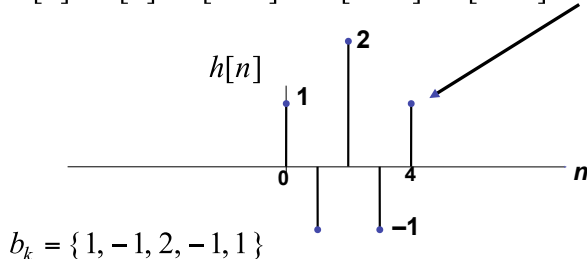
238

238

## MATH FORMULA for h[n]

▪ Use SHIFTED IMPULSES to write h[n]

$$h[n] = \delta[n] - \delta[n-1] + 2\delta[n-2] - \delta[n-3] + \delta[n-4]$$



239

## LTI: Convolution Sum

▪ Output = Convolution of x[n] & h[n]

▪ NOTATION:  $y[n] = h[n] * x[n]$

▪ Here is the FIR case:

$$y[n] = \sum_{k=0}^M h[k] x[n-k]$$

**FINITE LIMITS**

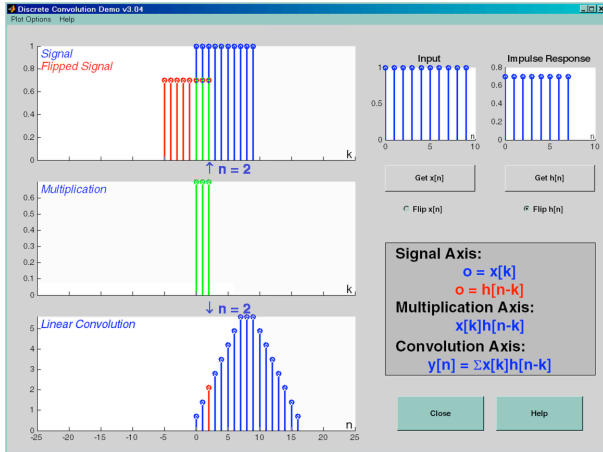
Same as **b<sub>k</sub>**

**FINITE LIMITS**

240

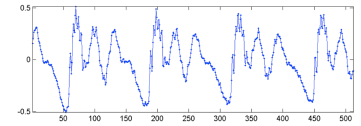
240

## DCONVDEMO: MATLAB GUI

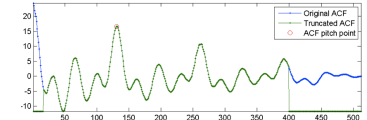


241

## Auto-correlation Example



- ◆ Problem: what's the pitch of this sound?
  - ◆ Waveform's period is obvious from visual inspection
  - ◆ Less-so to the computer...
- ◆ Solution: auto-correlation
 
$$R_x(\nu) = \sum_{n=-\infty}^{\infty} x[n]x[n+\nu]$$
  - ◆ Slide the wave "across itself" taking the sum of the vector product for each delay value
  - ◆ Gives a low value for most delays, but a high one for delays close to the period
  - ◆ This is one of a family of "regression" techniques



242

## Applied DSP

### ◆ Where's Николай Ежов



243

## Examples

- ◆ Audio processing
  - ◆ Conversion: audio DAC, ADC, format convertors, compression
  - ◆ Time-domain: gain-control, dynamic-range processing, echo, chorus, flanging
  - ◆ Filtering: equalization, "loudness," hum removal, feedback control, cross-synthesis

244

## Examples

- ◆ Image processing
  - ◆ Conversion: digital camera, projectors
  - ◆ Spatial-domain: clipping/cropping, scaling, image rotation, color correction, compositing, brightness/contrast, registration, color selection (chroma keying)
  - ◆ Filtering: edge detection, sharpening, de-noising, smoothing, effects

245

## Review

- ◆ Operations on signals
  - ◆ Signal creation and manipulation
  - ◆ Signal arithmetic and statistics
  - ◆ Signal-processing functions and flow-charts
  - ◆ Memory, difference equations and digital filters
  - ◆ Autoregression and moving-average processes

246

## Topic 5: Information Theory

### Topic 5

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
actqgwrzdevfbhinsymujxplok

Using this key, the plaintext:

THE SECURITY OF THE RSA ENCODING SCHEME RELIES ON THE  
FACT THAT NOBODY HAS BEEN ABLE TO DISCOVER HOW TO TAKE  
CUBE ROOTS MOD N WITHOUT KNOWING NS FACTORS

becomes the ciphertext:

UZG MGTJYDUO IW UZG YMA GHTIQDHR MTZGBG YGFDGM IH UZG  
WATU UZAU HICIQO ZAM CGGH ACFG UI QDMTIXGY ZIP UI UAVG  
TJCG YIIUM BIQ H PDUZIJU VHDPDR HM WATUIYM

247

- ◆ Information theory of signals
  - ◆ Randomness, noise, random walk, pseudonoise
  - ◆ Entropy and mutual information
- ◆ Applications
  - ◆ Error correction
  - ◆ Data compression
  - ◆ Data embedding and hiding

248

## Information Theory

- ◆ History
- ◆ Topics: signals, symbols and information
- ◆ Aspects of signals (later)
- ◆ Units of information
- ◆ Coding of information
  - ◆ Error detection and correction
  - ◆ Encryption, compression, embedding

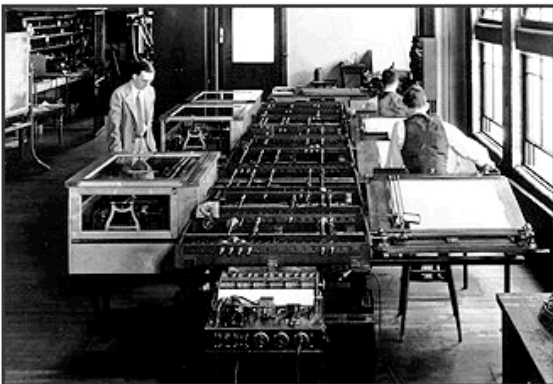
249

## Aspects of Information

- ◆ Technical: accuracy of symbol transmission
- ◆ Semantic: symbol/meaning map
- ◆ Effectiveness: of received meaning

250

## Differential Analyser



251

## History of Information

- ◆ Claude Shannon
  - ◆ 1938: "Application of Symbolic Logic to Relay Circuits" (Vannevar Bush and the "Differential Analyser" -- Boolean algebra)
  - ◆ "It just happened that no one else was familiar with both fields at the same time."
  - ◆ 1948 "A Mathematical Theory of Communication" Bell System Technical Journal

252

# Information Theory

- ◆ The crux of information theory is the realization that the information content of a message stream is directly connected to the probability of appearance of each possible message, and that we can take advantage, on average, of any non-uniformity of these probabilities.
- ◆ (From <http://ece.wpi.edu/infoeng/textbook/main.html>)

253

# Information System

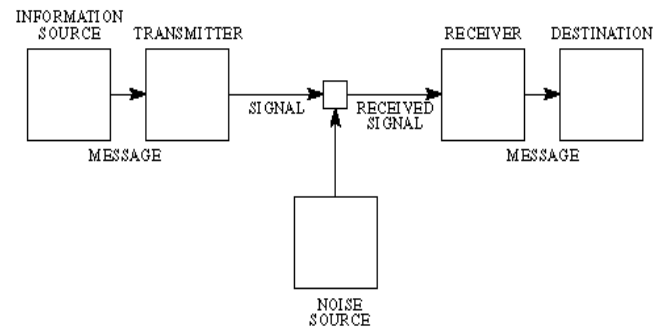


Fig. 1—Schematic diagram of a general communication system.

254

# Error Correction

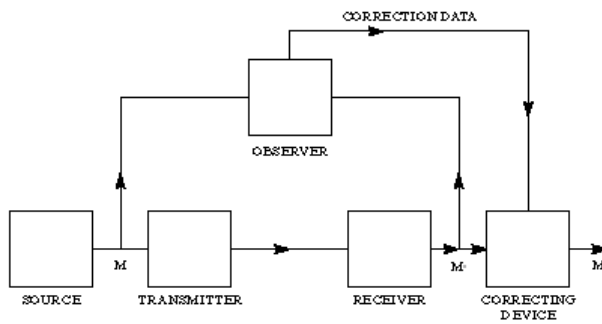


Fig. 8—Schematic diagram of a correction system.

255

# Signals (first pass)

- ◆ Fidelity of transmission
  - ◆ Real vs. reproduced signals
- ◆ The impact of transducers
  - ◆ Impact on perception and on information content
- ◆ Mapping signals onto information
- ◆ “Entropy” in signal sources
  - ◆ Information density

256

# Information

- ◆ Units: bits
- ◆ Data rate vs. information rate
  - ◆ Coding and redundancy
- ◆ Tangent: Ontology
  - ◆ (1) Die Welt ist alles, was der Fall ist.
  - ◆ (1.1) Die Welt ist die Gesamtheit der Tatsachen, nicht der Dinge.
  - ◆ Ludwig Wittgenstein, *Tractatus Logico-Philosophicus*

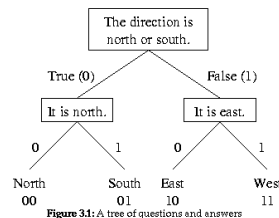


Figure 34: A tree of questions and answers

# Information Translation, Coding

- ◆ Codings
  - ◆ Thought as Speech
  - ◆ Speech as ASCII
- ◆ Representations: should be unique, standardized, technology-appropriate, lossless
- ◆ These are language-design issues (lots to come later)
- ◆ Design of codes and alphabets

258

257

# Coding Examples

- ◆ Example: Integer-as-bits
  - ◆ 1s-complement little-endian --  
 $749_{10} = 1011101101_2 = 2ED_{16} = 1355_8$
  - ◆ 2s-complement --  $749 = (1)1011101101$
  - ◆ BCD --  $749 = 0111\ 0100\ 1001$
  - ◆ 32-bit floating-point --  $749 =$   
 $10000001000011101101000000000000$
  - ◆ And that's just a small number!
- ◆ Finite- vs. infinite-precision numbers
- ◆ Scale and resolution (sci. notation)
- ◆ Transcendental data ( $\pi$ ,  $e$ )

259

# Standard Codings

- ◆ Text
  - ◆ ASCII, EBCDIC, Unicode, ...

- ◆ Numbers
  - ◆ IEEE floating-point
  - ◆ Extended numbers
- ◆ Sound
- ◆ Images

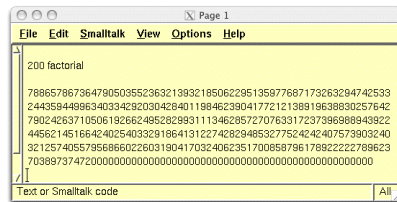
00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	\$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	
0D	CR	1D	GS	2D	=	3D	=	4D	M	5D	]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

NUL	Null	FF	Form feed	CAN	Cancel
SOH	Start of heading	CR	Carriage return	EM	End of medium
STX	Start of text	SO	Shift out	SUB	Substitute
ETX	End of text	SI	Shift in	ESC	Escape
EOT	End of transmission	DLE	Data link escape	FS	File separator
ENQ	Enquiry	DC1	Device control 1	GS	Group separator
ACK	Acknowledge	DC2	Device control 2	RS	Record separator
BEL	Bell	DC3	Device control 3	US	Unit separator
BS	Backspace	DC4	Device control 4	SP	Space
HT	Horizontal tab	NAK	Negative acknowledge	DEL	Delete
LF	Line feed	SYN	Synchronous idle		
VT	Vertical tab	ETB	End of transmission block		

ASCII Character Set

260

# Media and Channels



- ◆ Information characterization of transducers, storage, and transmission codes
- ◆ Transmitter, channel, and receiver model
- ◆ Level of redundancy, information efficiency, and error-proness

261

# Applied Information Theory

- ◆ Apply insights into the characteristics of different kinds of information, codings and channels
- ◆ Design of efficient representations and encodings
- ◆ Codings that support error detection, correction
- ◆ Compression
- ◆ Encryption
- ◆ Information embedding (error handling, data hiding, watermarking)

262

# Coding and Errors

- ◆ How do you encode a signal to ease error handling (in transmission/storage)?
- ◆ What do you need to take into account?
- ◆ 1: Characterize the system's error sources
- ◆ 2: Design an encoding that enables you to detect certain classes of errors
- ◆ 3: Add redundancy/repetition as necessary to recover from detected errors of the given kinds

263

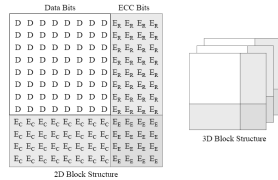
# Error Handling

- ◆ Error characterization: what kinds of errors are expected in a system? (describe via theory or measurement)
  - ◆ (e.g.) LAN: short bursty drop-outs (collisions)
  - ◆ DVD-ROM: large blocks across many tracks (scratches)
- ◆ Error detection: how can you be very sure whether you got a message right?
  - ◆ LAN: use hand-shake with seq # and check-sum
  - ◆ DVD-ROM: aggressive statistical checksums on blocks
- ◆ Error recovery: how do you get it then?
  - ◆ LAN: re-send packet, hand-shake again
  - ◆ DVD-ROM: look somewhere else on the disc
  - ◆ (distributed redundant data)

264

# Coding and Redundancy

- ◆ Formats and protocol design
  - ◆ Analysis of expected error allows the design of minimal effective redundancy
  - ◆ Trade-off between data rate and redundancy -- impact and error recovery
  - ◆ Coding vs. compression artifacts
- ◆ Pretty well-solved
  - ◆ (for existing media)
  - ◆ Parity-, polarity-based
  - ◆ Polynomial-based, etc.



265

# Statistical Techniques

Transmitted Character	Transmitted Information	Transmitted Parity	Received Information	Received Parity	Was there an Error
H	1001000	0	1000000	0	Yes
e	1100101	0	1100101	0	No
l	1101100	0	1101100	0	No
p	1110000	1	1110000	1	No

Table 3.1: Even Parity Example

- ◆ Parity-checking (parity = enforce even or odd # of 1's in binary words by added p-bits)
- ◆ Checksums (transmit sum of words in a block)
- ◆ Hierarchical or cyclical coding (data as under-specified polynomial of a certain family)
- ◆ Many others

266

# Encryption

- ◆ Encryption: make a signal look like noise to most observers, but easily recoverable to some
- ◆ Simple: substitution tables

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
actqgrwrzdevfbhinsymujxplok

Using this key, the plaintext:

THE SECURITY OF THE RSA ENCODING SCHEME RELIES ON THE  
FACT THAT NOBODY HAS BEEN ABLE TO DISCOVER HOW TO TAKE  
CUBE ROOTS MOD N WITHOUT KNOWING NS FACTORS

becomes the ciphertext:

UZG MGTJYDUO IW UZG YMA GHTIQDHR MTZGBG YGFDGM IH UZG  
WATU UZAU HICIQO ZAM CGGH ACPG UI QDMITXGY ZIP UI UAVG  
TJCG YIIUM BIQ H PDUZIJU VHDPDHR HM WATUIYM

267

# Advanced Encryption

- ◆ Multi-stage variable substitution encoding
- ◆ Use number theory: modulus congruences, remainders, primes and roots
- ◆ Often based on factorization of very large numbers
- ◆ Multi-part public/private keys
- ◆ No known “perfect” method, but many that are quite complex (hard to crack in the useful lifetime of the information)

268

# Steganography

- ◆ Data-embedding, data-hiding
- ◆ Can be used for error recovery
- ◆ Example application: add a “watermark” to a signal
- ◆ Watermark must be:
  - ◆ Indistinguishable (well-hidden)
  - ◆ Incontrovertable (un-removeable)
  - ◆ Robust across encoding or numerical operations

# Data-hiding Example



Before

Watermark

After

269

270

# Compression

- ♦ What assumptions can you make about the information and its coding?
  - ♦ Repetitions?
  - ♦ Noise?
- ♦ Does the reconstructed version have to be numerically or “perceptually” identical?
- ♦ Examples (lossless)
  - ♦ B/W scanned graphics: run-length-encoding
  - ♦ ASCII text: Huffman codes

271

# Run-Length Encoding

- ♦ Detect repeating pixels on each row; translate them to a pixel count and value pair
- ♦ Does very well for large areas of solid color
- ♦ Does not do well for stipple patterns or scanned images
- ♦ Used in BMP, TIFF, and PICT file formats
- ♦ Can be extended to longer patterns and 2-D runs (rectangle runs)
  - ♦ Original: AAAAAAABBBBCDE
  - ♦ Encoded: -8A-3BCDE

272

# Huffman Coding (ZIP, StuffIt, gzip, ...)

- ♦ Make a histogram of vocabulary usage
- ♦ Replace frequent input strings with n-bit codes (2n special terms in vocabulary)
- ♦ Can be very effective (e.g., almost any natural text, programs, machine code, DBMS)
- ♦ Fully reversible (lossless)
- ♦ Used in the GIF/ TIFF file formats (12-bit codes = 4096 coded patterns)

273

# Sound Compression

- ♦ Old rule:
  - ♦ Effective / real-time / artifact-free : pick any 2
- ♦ Linear/exponential:  $\mu$ -law ( $\sim$  A-law)
  - ♦ Non-linear quantization, look-up tables
  - ♦ 4:1 -- OK for speech only
- ♦ Statistical: Huffman
  - ♦ Generally ineffective for sound (try it!)

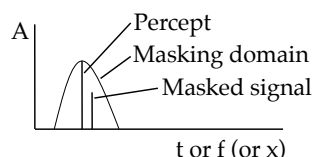
274

# Lossy Compression Models

- ♦ LPC model (subtractive synthesis)
  - ♦ CELP compression on cell phones
  - ♦ Very effective (100:1), complex, OK for speech

## ♦ Masking model

- ♦ MPEG, MP3
- ♦ Effective, complex, has artifacts



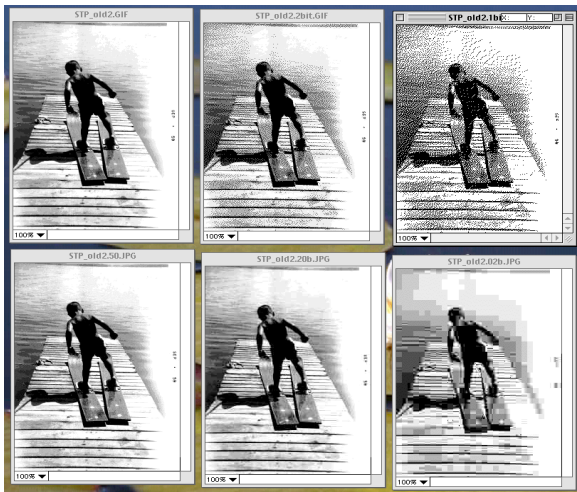
275

# Perceptual Coding & Compression

- ♦ MP3
  - ♦ Represent signal in spectral domain (DCT, like FFT)
  - ♦ Use freq/ampl-dependent masking threshold model to prune spectral data
  - ♦ Use linear algebra to compress resulting sparse matrices
- ♦ JPEG
  - ♦ Same except uses 2-D DCT

276

# Dithering vs. Compression



File sizes (kB):

GIF 1: 36  
GIF 1.z: 14  
GIF 2: 18  
GIF 2.z: 10  
GIF 3: 10

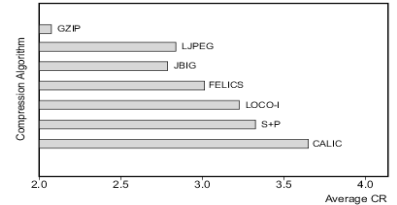
JPEG 1: 18  
JPEG 2: 6.6  
JPEG 3: 3.3

277

# Image Compression

- ◆ Lossless statistical methods (Huffman, LZW)
- ◆ Lossless image-specific (RLE)
- ◆ Lossy image-specific (JPEG / MPEG)
- ◆ Really fancy (wavelets, IFS)

Figure 2.3: Comparison among various lossless image compression techniques.



278

# Other Compression

- ◆ Fancy:
  - ◆ Wavelets
  - ◆ Chaos-based
- ◆ Color table compression and color dithering
- ◆ Color-to-grayscale conversion
- ◆ Custom techniques for special domains
  - ◆ Faxes
  - ◆ Video

279

# Review

- ◆ Information theory
  - ◆ Theory and history
  - ◆ Communication systems and their elements
  - ◆ Applications to data and media content
    - ◆ Coding
    - ◆ Compression
    - ◆ Encryption

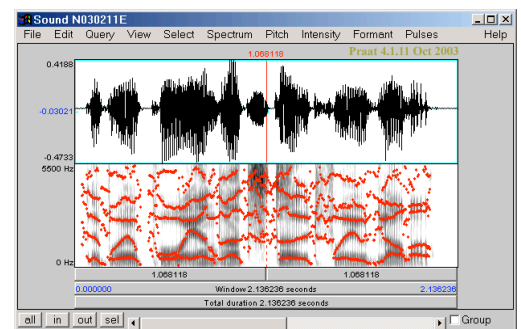
280

# Review

- ◆ Information theory of signals
  - ◆ Applications
    - ◆ Error correction
    - ◆ Data compression
    - ◆ Data embedding and hiding

281

# Topic 6



282

# Topic 6

- ◆ Transforms and mappings
  - ◆ Time- and frequency-domain signals
  - ◆ Spectral representations, Fourier transform
  - ◆ Linear prediction, signal modeling by polynomials

283

# Time, Frequency-domains

- ◆ Time-domain
  - ◆ Direct signals
  - ◆ Perception
- ◆ Frequency-domain
  - ◆ Derived signals
  - ◆ Cognition
  - ◆ Like Cartesian vs Polar coordinates, some operations are just simpler in the 1 or the other representation

284

## Fourier Analysis/

- ◆ Fourier theorem - periodic signals as sum of related sines
- ◆ Fourier transform - continuous/ discrete, infinite- /short-time

Time Duration		
Finite	Infinite	
Fourier Series (FS)	Fourier Transform (FT)	cont.
$X(k) = \frac{1}{T} \int_0^T x(t) e^{-j\omega_k t} dt$	$X(\omega) = \int_{-\infty}^{+\infty} x(t) e^{-j\omega t} dt$	time
$k = -\infty, \dots, +\infty$	$\omega \in (-\infty, +\infty)$	$t$
Discrete FT (DFT)	Discrete Time FT (DTFT)	discr.
$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\omega_k n}$	$X(\omega) = \sum_{n=-\infty}^{+\infty} x(n) e^{-j\omega n}$	time
$k = 0, 1, \dots, N-1$	$\omega \in (-\pi, +\pi)$	$n$
discrete freq. $k$	continuous freq. $\omega$	

285

## DSP Notations (from Smith)

Frequency and Time:

- $\omega$  denotes continuous radian frequency (rad/sec)
- $f$  denotes continuous frequency in Hertz (Hz)
- $\omega = 2\pi f$
- $\omega_k$  denotes discrete frequency,  $\omega_k = 2\pi(k/N)f_s$
- $\omega, \omega_k \in \mathbf{R}$  (frequencies are always real)
- $T$  = sampling interval (sec)
- $f_s$  = sampling rate,  $f_s = \frac{1}{T}$
- $t_n = nT$  (discrete time)
- $n, k \in \mathbf{Z}$  (integers)
- $t, t_n \in \mathbf{R}$  (times are always real)

Signal Notation:

- $x \in \mathbf{C}^N$  means  $x$  is a length  $N$  complex sequence
- $a = a(\cdot)$
- $X = \text{DFT}(x) \in \mathbf{C}^N$
- $X(k) = \text{DFT}_k(x)$
- $X(k) \in \mathbf{C}$
- $x(n) = \text{IDFT}_n(X)$
- $x(n) \in \mathbf{R}$  or  $\mathbf{C}$
- $n, k \in \mathbf{Z}$  or  $n, k \in \mathbf{Z}_N$  (integers modulo  $N$ )
- For  $x \in \mathbf{C}^\infty$ ,  $X = \text{DTFT}(x) \in \mathbf{C}_{2\pi}^\infty$
- $\bar{x}$  = conjugate of  $x$
- $\angle x$  = phase of  $x$

286

## DFT in Pseudocode

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\omega_k n}$$

$$k = 0, 1, \dots, N-1$$

- ◆ To get a spectrum  $X(k)$  ( $k$  is spectral bin #)
- ◆ For every output frequency  $k$  from 0 to  $F_s/2$ 
  - ◆ Compute the vector product of the input signal  $x(n)$  with a complex sine of freq.  $\omega_k$  (i.e.,  $e^{-j\omega_k n}$ , bin center freq.)
  - ◆ That (real/complex) val is the energy at freq.  $\omega_k$ , or  $X(k)$
- ◆ Process requires  $O(n^2)$  (or worse) operations
- ◆ Process is hard to accelerate (even with huge

287

## DFT in Pictures (from Loy)

Table 3.2  
Frequency Analysis Process of the DFT

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
$k$	Test Signal	Cosine Probe Phasor	Cosine Product Signal	Sine Probe Phasor	Sine Product Signal	Cosine Sum (Real)	Sine Sum (Imaginary)
0						0	0i
1						0	0i
2						0	-0.5i
3						0	0i
4						0	0i
5						0	0i
6						0	0.5i
7						0	0i

288

## Slow Fourier Transform in SuperCollider

```

sweepsize = 1024; // source: approx. pulse, 10
overtones
source = Signal.sineFill(sweepsize, [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]);
sweep = Signal.newClear(sweepsize); // create swept sine buffer
index = 0.0;
sweepfcn = { arg frq, i; // function to create a swept sine
 index = index + frq;
 sin(index)
};
sweep.waveFill(sweepfcn); // fill swept sine buffer
sinefcn = { arg x, i; sin(x) }; // function to create a static sine
sine = Signal.newClear(sweepsize); // create sine buffer
spect = Signal.newClear(sweepsize); // "convolve" sweep * source
spectfcn = { arg x, i; // fcn to "convolve" source & sine
 sine.waveFill(sinefcn, 0, (x*2pi)); // create correct freq. sine
 sum = 0.0; // mult sine * source and
 integrate
 (source * sine).integral / sweepsize
};

```

289

## Interpretations of the DFT

- ◆ Multiply the signal by a swept sine wave:  
 $x(t) e^{-j\omega t}$  for  $\omega$  in  $\pm\infty$
- ◆ Matrix multiplication  $X = Wx$  ( $W$  is square matrix of complex spectral coefficients)
- ◆ Projection coefficients of signal  $x$  onto  $N$  sinusoidal components (equal-spaced complex roots of 1)
- ◆ Coordinate transform from  $RN$  to  $X(k)$
- ◆ others...

290

## Properties of the DFT

- ◆ Linearity
- ◆ Time reversal
- ◆ Symmetry (phase, magnitude, conjugate)
- ◆ Shift
- ◆ Convolution
- ◆ Correlation
- ◆ Stretch/Repeat
- ◆ Downsampling/ Aliasing

291

## Why Fourier?

- ◆ Physical relevance (how the ear works)
- ◆ Link to filter banks and add. synth. (parameter est. for a model)
- ◆ Can represent any LTI as a FT system
- ◆ May be compact (and maybe not)
- ◆ Can be implemented efficiently (FFT)
- ◆ Some processes (complex filters, convolution) are more efficient in the frequency domain
- ◆ Many applications: analysis, coders, classifiers, processing, etc.

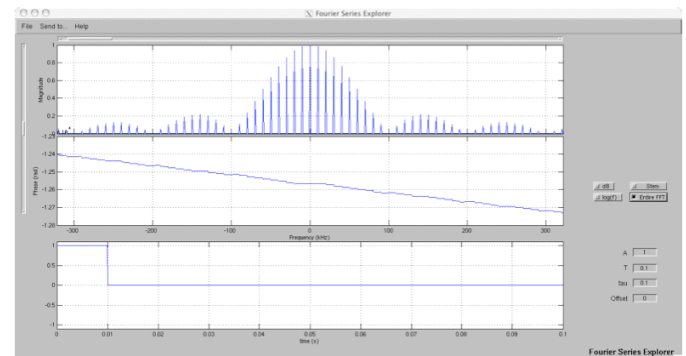
292

## Other Frequency Transforms

- ◆ Higher-level analysis based on FFT
  - ◆ Partial Trackers
  - ◆ Signature classifiers
  - ◆ Noise reduction
  - ◆ Pitch detection
- ◆ Linear Prediction and LPC
- ◆ Wavelet Transform

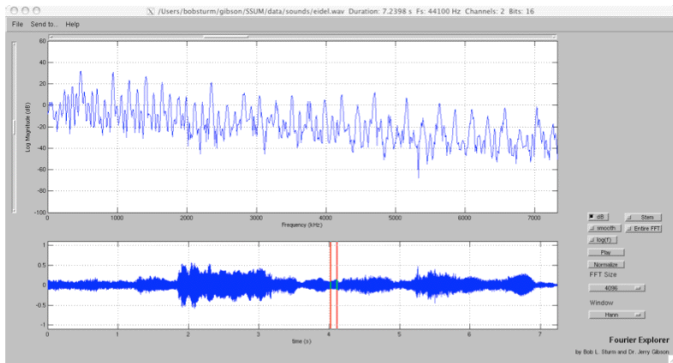
293

## SSUM Fourier Series



294

# SSUM Fourier Explorer



295

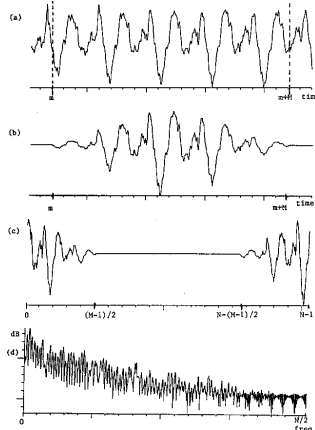
# Windowing

- ◆ Problem: the real world uses finite-length time-varying signals
- ◆ Solution: analyze signal in short “windows” of fixed size
- ◆ Impact: need to consider frequency response of the window function itself and the time/frequency accuracy trade-off

296

## Windowing Overview

- ◆ Input signal
- ◆ Windowed Signal
- ◆ Shifted (zero-phase) window
- ◆ FFT gives magnitude spectrum



297

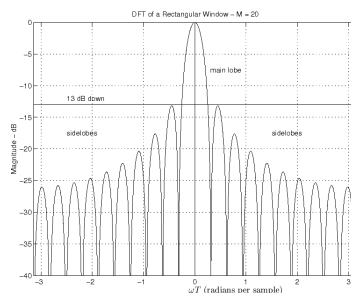
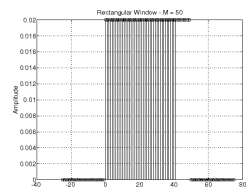
## Windows and Spectral Smear

- ◆ Any finite-length window will degrade the analysis.
- ◆ Window spectra have a central lobe and symmetrical side lobes.
- ◆ There are many kinds of window functions with varying properties.
- ◆ The important variables are
  - ◆ (a) the width of the central lobe,
  - ◆ (b) the 1st side-lobe level, and
  - ◆ (c) the side-lobe roll-off rate

298

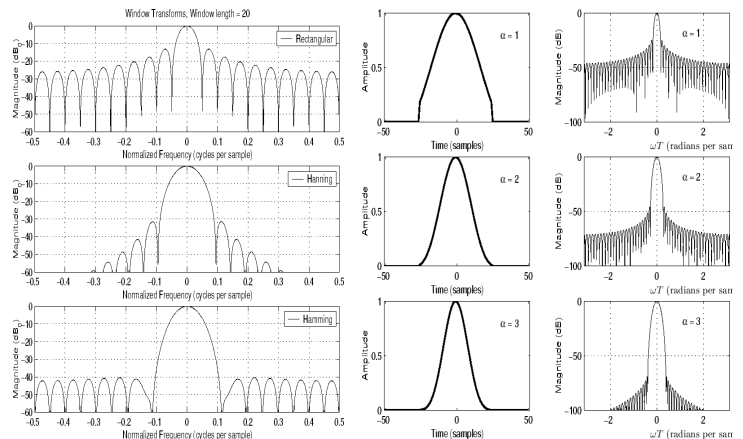
## Windows and their Spectra

- ◆ Rectangular (boxcar, null)



- ◆ Triangle (Bartlett) wider main lobe and lower 1st side-lobe (both by a factor of 2), as well as better roll-off

## Window Functions



299

300

## General Windowing Issues

- ◆ It's an underconstrained design problem!
- ◆ Both time-domain and frequency-domain (as well as magnitude and phase) characteristics are relevant.
- ◆ There are some new (2000) techniques that promise good time and frequency features.

## Main-lobe widths

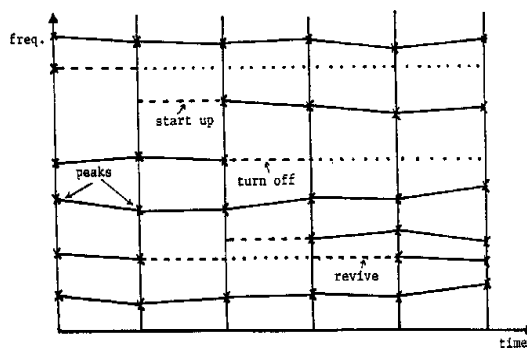
- ◆ For  $S = f_s /$  (side-lobe width in Hz)

Window Type	Main-Lobe Width $B_w$ (Hz)
Rectangular	$2S$
Hamming	$4S$
Hann	$4S$
Generalized Hamming	$4S$
Blackman	$6S$
$L$ -term Blackman-Harris	$2(L+1)S$
Kaiser	depends on $\beta$
Chebyshev	depends on ripple

301

302

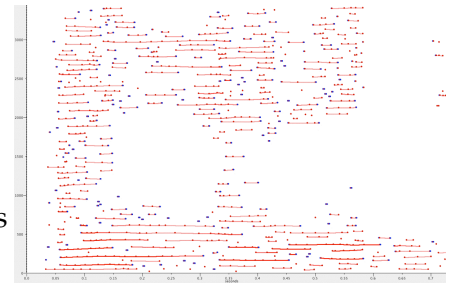
## Advanced FFTs



303

## What to do with a Spectrum?

- ◆ Peak-finding
- ◆ Formant-finding
- ◆ Peak-tracking
- ◆ F0 analysis
- ◆ MFCC coefficients
- ◆ Thresholding
- ◆ Freq.-shifting
- ◆ Additive cross-synthesis



304

## Spectral Metrics & Measures

- ◆ Spectrum reduction and (n-octave) band spectra
- ◆ Spectral weightings: centroid, slope, LF/HF, peakiness, ...
- ◆ Spectral change & dynamicity
- ◆ MPEG7 standard & MIR feature vectors

## Example: MIR Feature Vector

### Time-domain features

- Windowed RMS amplitude
- Max sample amplitude
- RMS (ratio?) of LP/HP-filtered signal
- Count of zero crossings
- RMS dynamic range of sub-windows
- RMS peak sub-window index
- Tempo estimates (several)
- Beat histograms & weights
- Tempo weight & off-by-2 confidence
- Time signature guess

### Frequency-domain features

- Windowed FFT data (stored?)
- 1-octave FFT data (10-12 points)
- 2.5-octave FFT data (4 spectral bands)
- List of spectral peak indices
- List of tracked peak frequencies
- Spectral peak track births/deaths
- Spectral measures: centroid, slope, variety
- Relative HF level & spectral variety
- Corr. between HF and audio-band
- MFCC coefficients (4-12)

### Spatial features

- L/R difference
- Front/Surround difference
- Center vs. L/R sum difference
- Spatial variety

### Pitch estimates

- Bass pitch guess in Hz
- Bass note (MIDI key number) guess
- Bass note dynamicity (size of histogram)
- Multi-pitch estimates?
- Chroma/key data

### LPC features

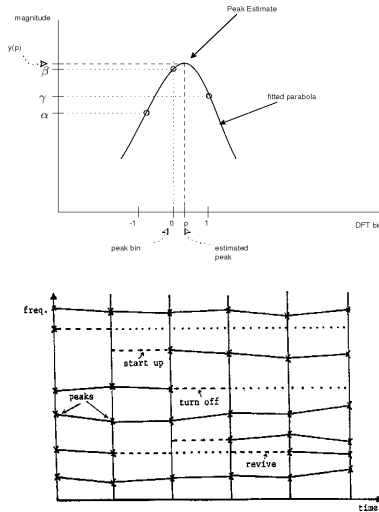
- List of LPC formant peaks
- List of tracked LPC formants
- LPC residual level (noisiness)
- LPC formant track births/deaths

### Fluctuation Pattern features

- FP flux
- FP gravity
- FP weight

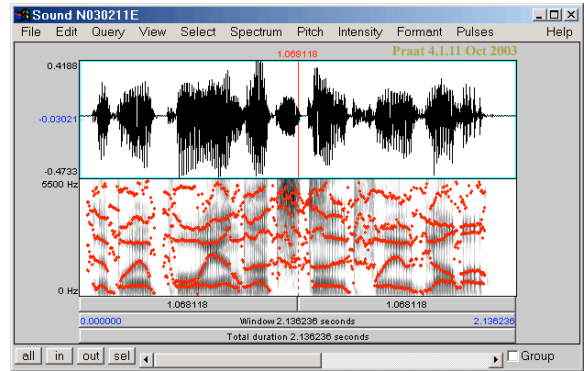
# Partial-Tracking

- ◆ Spectral peak detection (parabolic interpolation, possibly using phase unwrapping)
- ◆ Peak matching & tracking, continuation via guides
- ◆ Data on track births and deaths (important)



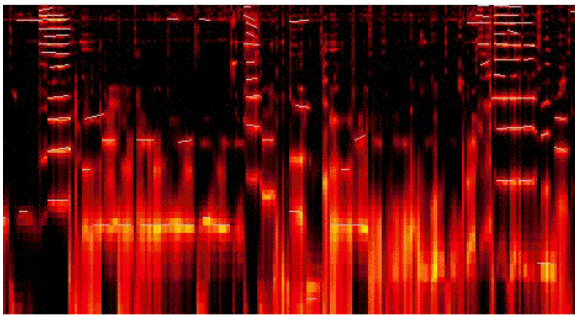
307

# Spectral Peaks



308

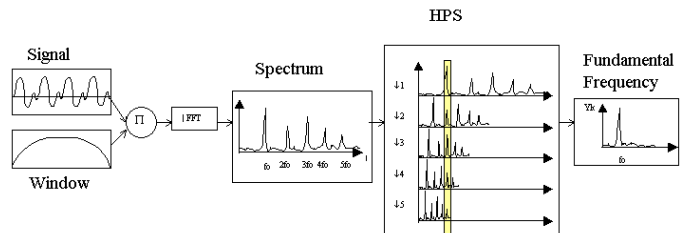
## Spectrum & Tracked Peaks



309

## Pitch-tracking with Harmonic Product Spectra

- ◆ Decimation of FFT spectra, summation, and spectral peak location
- ◆ Assumes overtones are significant, not that fundamental is (single-tone, multi-tone)



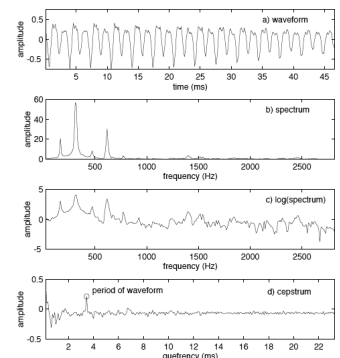
310

## Getting more from a Spectrum

- ◆ 1: Think of spectrum in signal correlation terms
- ◆ 2: Harmonic tones --> regularly spaced spectral peaks
- ◆ Take autocorrelation of spectrum and look for peaks
- ◆ The peaks relate to fundamental pitches in a warped frequency space (bin #)
- ◆ They are called Mel frequency spectral coefficients

## MFCC Analysis

- ◆ Analogy
  - ◆ Start with log spectrum of mixed complex tones: several sets of related partial peaks
  - ◆ Take, e.g., the autocorr. of the FFT PDS
  - ◆ Warped frequencies of peaks correspond to fundamental frequencies of overtone series



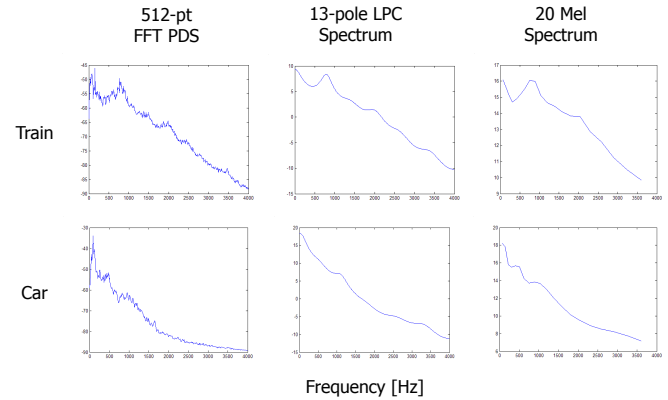
312

# Mel-Freq Cepstral Coefficients

- ◆ Steps:
    - ◆ Signal
    - ◆ FT
    - ◆ Log magnitude (PDS)
    - ◆ Phase unwrapping
    - ◆ FT (or DCT)
  - ◆ Name reversal
  - ◆ Interpretations
    - ◆ Quefrency
    - ◆ Mel-scale
    - ◆ Mel-scale filters
- Instead of AC, use FFT or DCT of PDS
- Leads to interesting statistics of higher-level spectral properties, see next section

# Comparison FFT / LPC / MFCC

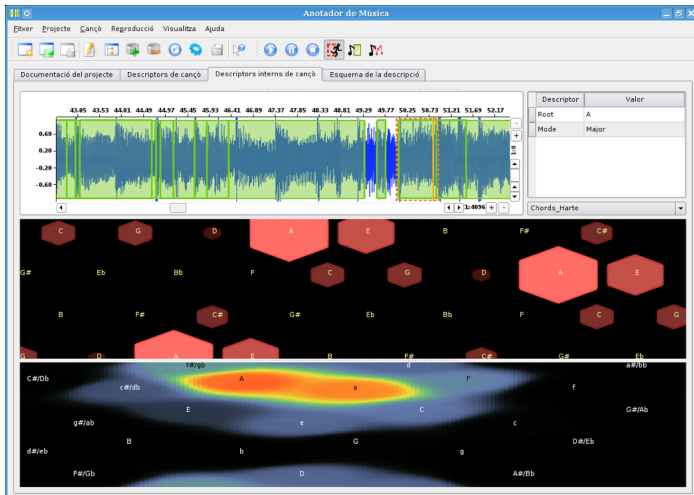
(by Andrianakis &amp; White)



313

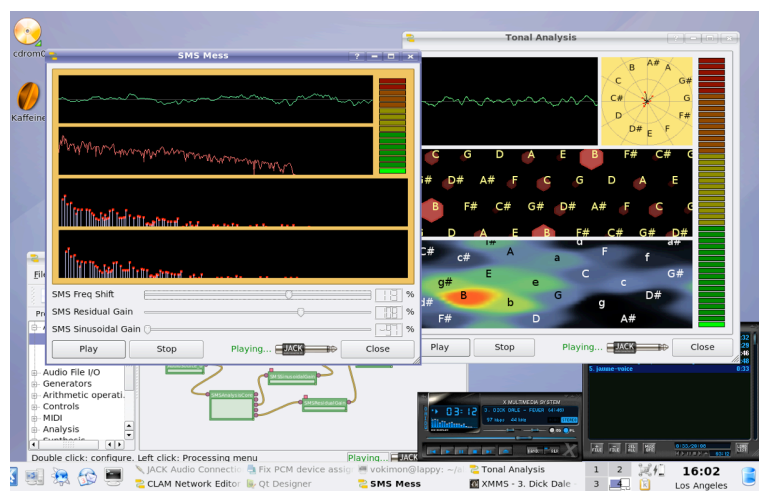
314

## MFCC ==> Chroma Features



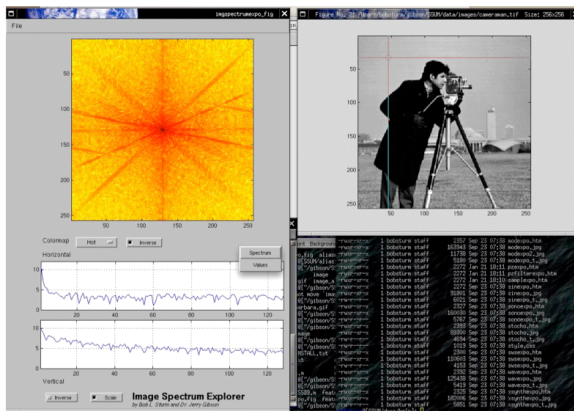
315

## Spectra, Peaks & Chroma in CLAM



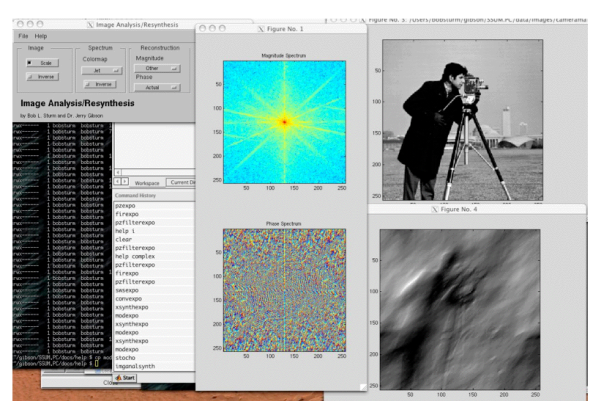
316

## SSUM Image Spectrum



317

## SSUM Image Analysis/Resynthesis



318

## FFT Applications & Extensions

- ◆ Vocoder and sinusoidal modeling
- ◆ Partial Tracking (MQAN)
- ◆ Transient Modeling
- ◆ Deterministic + Stochastic Modeling
- ◆ Bandwidth-enhanced partials
- ◆ DSL & ODFM
- ◆ Compression (MP3)

## Example: Synthetic Vowel

- Sum of 5 Frequency Components

$f_k$ (Hz)	$X_k$	Mag	Phase (rad)
200	$(771 + j12202)$	12,226	1.508
400	$(-8865 + j28048)$	29,416	1.876
500	$(48001 - j8995)$	48,836	-0.185
1600	$(1657 - j13520)$	13,621	-1.449
1700	$4723 + j0$	4723	0

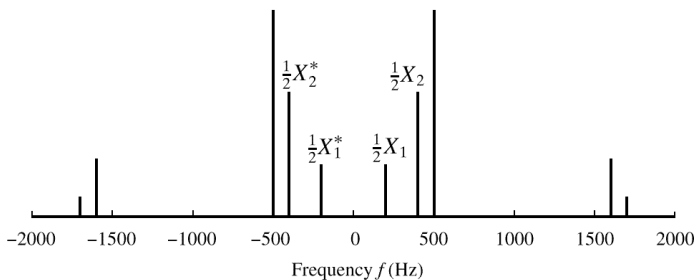
**Table 3.1:** Complex amplitudes for harmonic signal that approximates the vowel sound “ah”.

320

320

## SPECTRUM of VOWEL

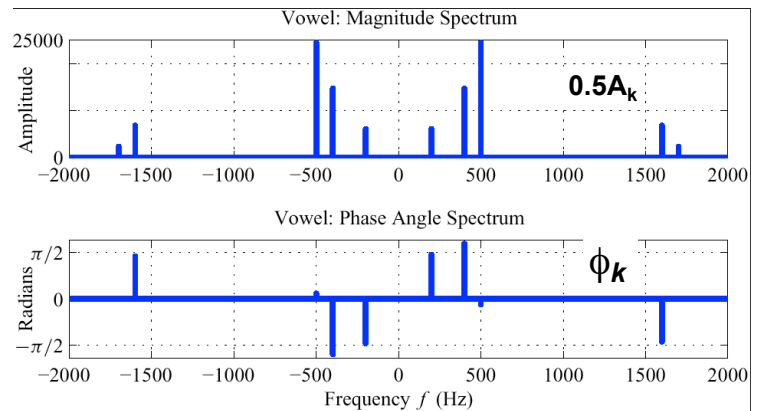
- Note: Spectrum has  $0.5X_k$  (except  $X_{DC}$ )
- Conjugates in negative frequency



321

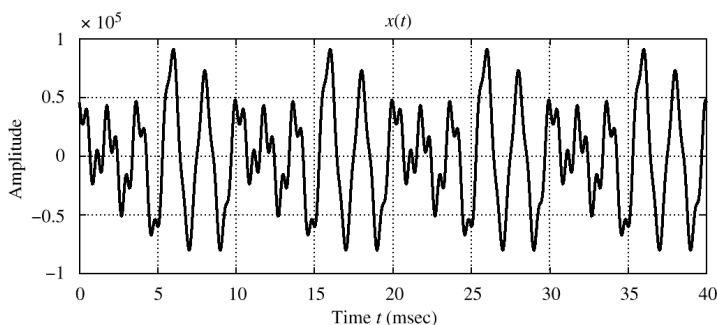
321

## SPECTRUM of VOWEL (Polar Format)



322

## Vowel Waveform (sum of all 5 components)



**Figure 3.11** Sum of all of the terms in (3.3.4). Note that the period is 10 msec, which equals  $1/f_0$ .

323

323

## Harmonic Signal Spectrum

Periodic signal can only have:  $f_k = k f_0$

$$x(t) = A_0 + \sum_{k=1}^N A_k \cos(2\pi k f_0 t + \varphi_k) \quad f_0 = \frac{1}{T}$$

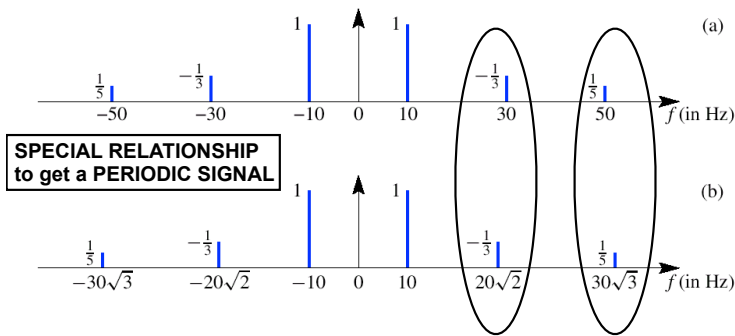
$$X_k = A_k e^{j\varphi_k}$$

$$x(t) = X_0 + \sum_{k=1}^N \left\{ \frac{1}{2} X_k e^{j2\pi k f_0 t} + \frac{1}{2} X_k^* e^{-j2\pi k f_0 t} \right\}$$

324

324

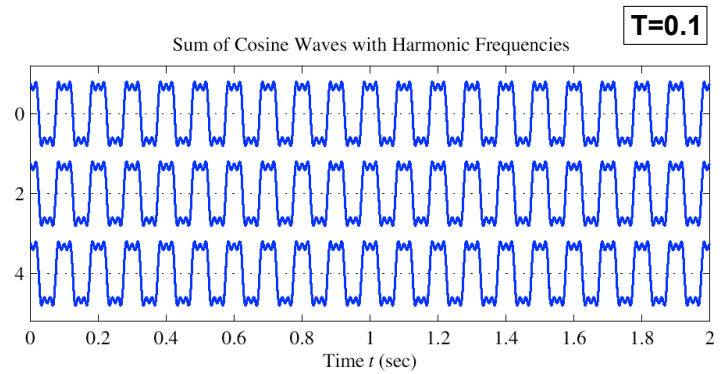
## IRRATIONAL SPECTRUM



325

325

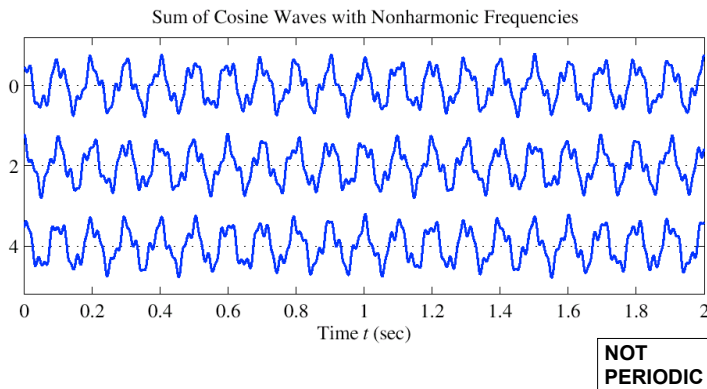
## Harmonic Signal (3 Freqs)



326

326

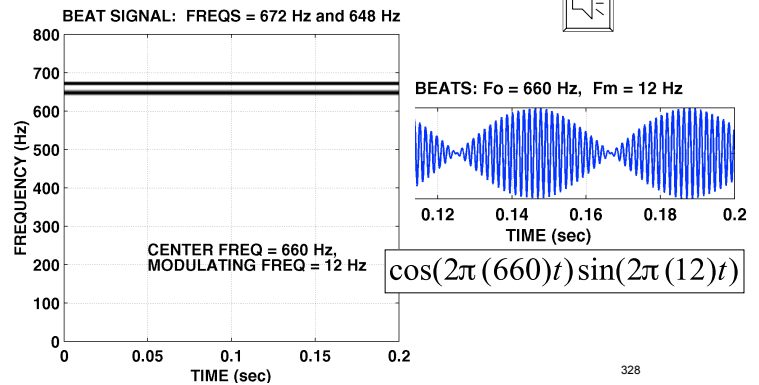
## NON-Harmonic Signal



327

## SPECTROGRAM EXAMPLE

### Two **Constant** Frequencies: Beats



328

328

## SYNTHESIS vs. ANALYSIS

- |                                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                   |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>▪ <b>SYNTHESIS</b> <ul style="list-style-type: none"> <li>▪ <b>Easy</b></li> <li>▪ Given <math>(\omega_k, A_k, \phi_k)</math> create <math>x(t)</math></li> </ul> </li> <li>▪ Synthesis can be <b>HARD</b> <ul style="list-style-type: none"> <li>▪ Synthesize Speech so that it sounds good</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>▪ <b>ANALYSIS</b> <ul style="list-style-type: none"> <li>▪ <b>Hard</b></li> <li>▪ Given <math>x(t)</math>, extract <math>(\omega_k, A_k, \phi_k)</math></li> <li>▪ How many?</li> <li>▪ Need algorithm for computer</li> </ul> </li> </ul> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

329

329

## STRATEGY: $x(t) \rightarrow a_k$

### ANALYSIS

- Get representation from the signal
- Works for **PERIODIC** Signals
- Fourier Series
  - Answer is: an INTEGRAL over one period

$$a_k = \frac{1}{T_0} \int_0^{T_0} x(t) e^{-j\omega_0 k t} dt$$

330

330

## Fourier Series Synthesis

- HOW do you **APPROXIMATE**  $x(t)$  ?

$$a_k = \frac{1}{T_0} \int_0^{T_0} x(t) e^{-j(2\pi/T_0)kt} dt$$

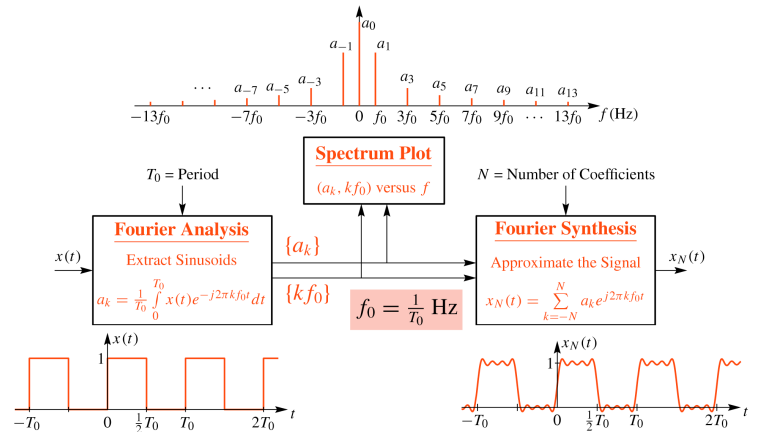
- Use **FINITE** number of coefficients

$$x(t) = \sum_{k=-N}^N a_k e^{j2\pi k f_0 t}$$

$a_{-k} = a_k^*$  when  $x(t)$  is real

331

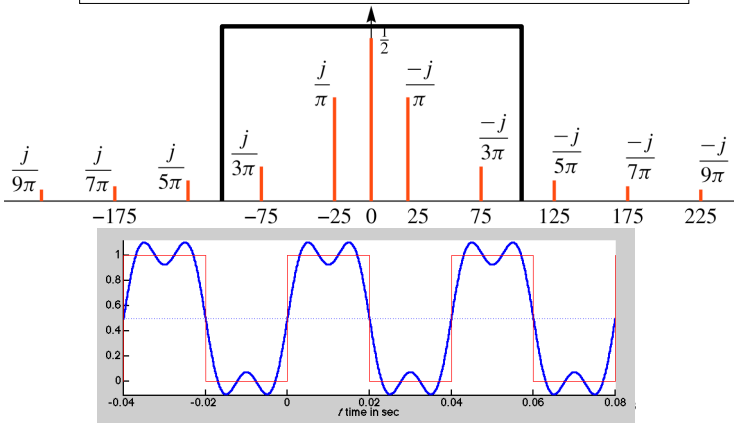
## Fourier Series Synthesis



332

## Synthesis: 1st & 3rd Harmonics

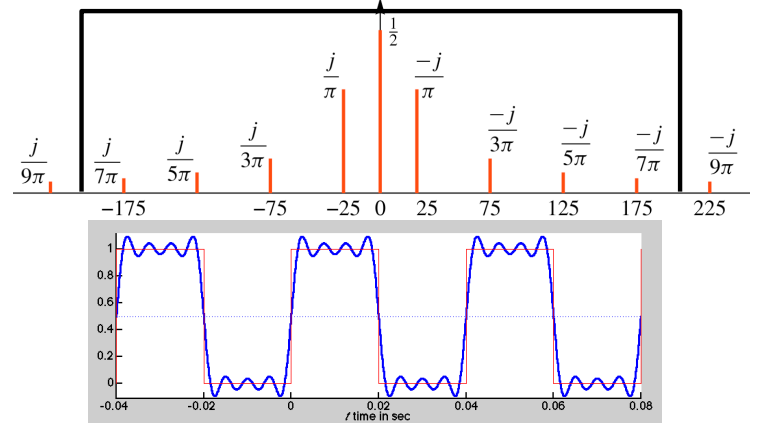
$$y(t) = \frac{1}{2} + \frac{2}{\pi} \cos(2\pi(25)t - \frac{\pi}{2}) + \frac{2}{3\pi} \cos(2\pi(75)t - \frac{\pi}{2})$$



333

## Synthesis: up to 7th Harmonic

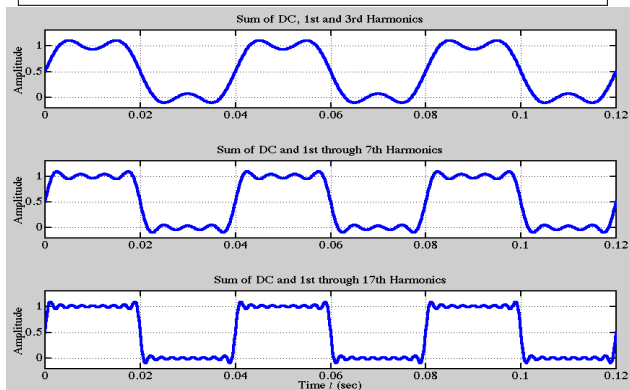
$$y(t) = \frac{1}{2} + \frac{2}{\pi} \cos(50\pi t - \frac{\pi}{2}) + \frac{2}{3\pi} \sin(150\pi t) + \frac{2}{5\pi} \sin(250\pi t) + \frac{2}{7\pi} \sin(350\pi t)$$



334

## Fourier Synthesis

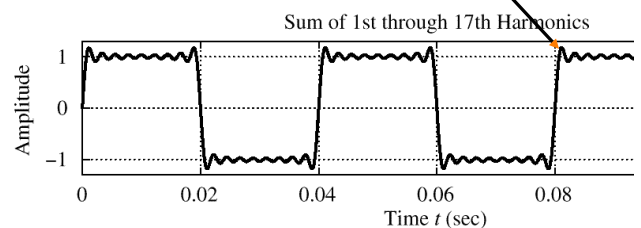
$$x_N(t) = \frac{1}{2} + \frac{2}{\pi} \sin(\omega_0 t) + \frac{2}{3\pi} \sin(3\omega_0 t) + \dots$$



335

## Gibbs' Phenomenon

- Convergence at **DISCONTINUITY** of  $x(t)$ 
  - There is always an overshoot
  - 9% for the Square Wave case



336

336

## Table of Easy FT Properties

### Linearity Property

$$ax_1(t) + bx_2(t) \Leftrightarrow aX_1(j\omega) + bX_2(j\omega)$$

### Delay Property

$$x(t - t_d) \Leftrightarrow e^{-j\omega t_d} X(j\omega)$$

### Frequency Shifting

$$x(t)e^{j\omega_0 t} \Leftrightarrow X(j(\omega - \omega_0))$$

### Scaling

$$x(at) \Leftrightarrow \frac{1}{|a|} X(j(\frac{\omega}{a}))$$

337

## Uncertainty Principle

- Try to make  $x(t)$  shorter
  - Then  $X(j\omega)$  will get wider
  - Narrow pulses have wide bandwidth
- Try to make  $X(j\omega)$  narrower
  - Then  $x(t)$  will have longer duration
- Cannot simultaneously reduce time duration and bandwidth**

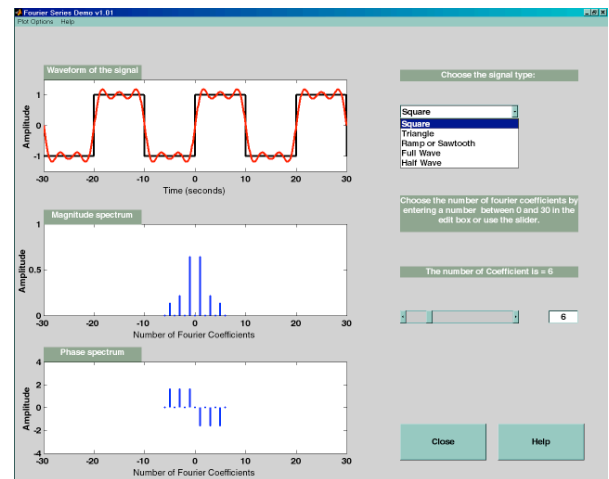
338

## Fourier Series Demos

- Fourier Series Java Applet
  - Greg Slabaugh
    - Interactive
  - [http://users.ece.gatech.edu/mcclella/2025/Fsdemo\\_Slabaugh/fourier.html](http://users.ece.gatech.edu/mcclella/2025/Fsdemo_Slabaugh/fourier.html)
- MATLAB GUI: fseriesdemo
  - <http://users.ece.gatech.edu/mcclella/matlabGUIs/index.html>

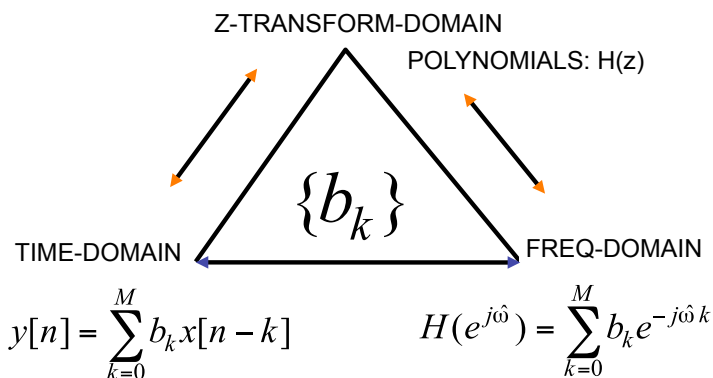
339

## fseriesdemo GUI



340

## TWO (no, THREE) DOMAINS



341

## Convolution



342

# Convolution and Filters

- ◆ Convolution  $(f * g)(t) = \int f(\tau)g(t - \tau) d\tau$ 
  - ◆ Definition, examples
- ◆ Time-domain solution (nested loop)
- ◆ FFT-based solution - use vocoder, then multiply spectral frames with FFT of (FIR) impulse response
- ◆ Same artifacts/problems as vocoder
- ◆ Constraints and work-arounds for long convolution
- ◆ Trade-offs

343

# Filter Performance

- ◆ Direct convolution is  $O(N^2)$
- ◆ FFT-based is  $O(N \log_2 N)$
- ◆ Number of Multiply / Adds

N	FFT	Direct Convolution
4	176	16
32	2560	1024
64	5888	4096
128	13,312	16,384
256	29,696	65,536
2048	311,296	4,194,304

344

## FIR Filter = CONVOLUTION

$x[n], X(z)$	0	+1	-1	+1	-1			
$h[n], H(z)$	1	2	3	4				

---

	0	+1	-1	+1	-1			
		0	+2	-2	+2	-2		
			0	+3	-3	+3	-3	
				0	+4	-4	+4	-4

---

$y[n], Y(z)$	0	+1	+1	+2	+2	-3	+1	-4
--------------	---	----	----	----	----	----	----	----

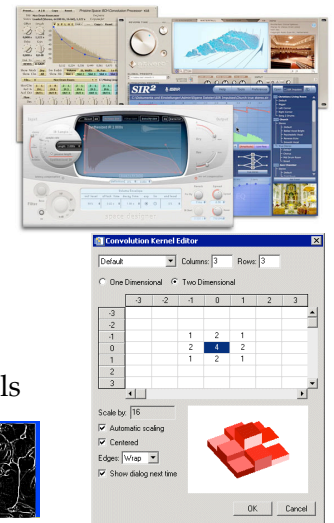
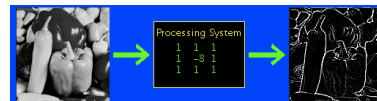
$$y[n] = \sum_{k=0}^M b_k x[n-k] = \sum_{k=0}^M h[k] x[n-k]$$

CONVOLUTION

345

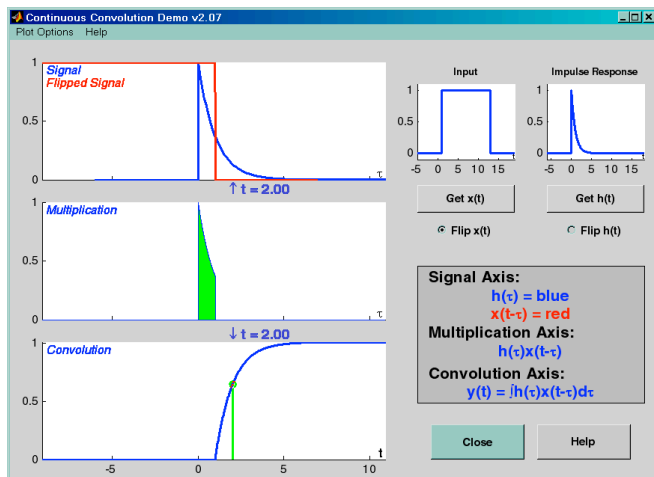
## Uses of Convolution

- ◆ Audio
  - ◆ Long FIR filters
- ◆ Reverb
- ◆ Images
  - ◆ Image-processing kernels



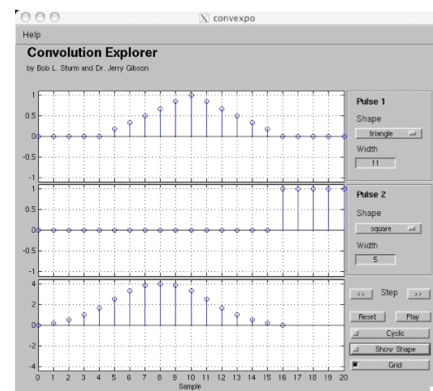
346

## Convolution GUI



347

## SSUM Convolution

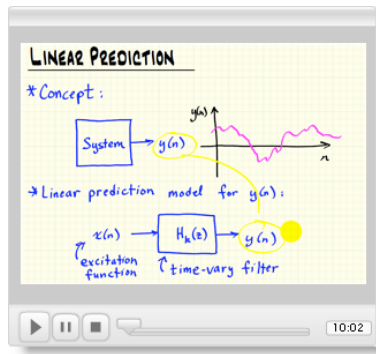


348

# Linear Predictive Coding

## The LPC Idea

### ◆ LPC



349

- ◆ Model a sequence as a weighted time series
  - ◆  $s(n) = a_1s(n-1) + a_2s(n-2) + \dots + a_ps(n-p)$
  - ◆  $= \sum_{k=1}^p a_k s(n-k) + \text{error}$
  - ◆ Think of this as a  $p^{\text{th}}$ -order polynomial (which it corresponds to in the z-domain)
- ◆ Expect some error and capture it.
- ◆ For periodic sounds, the error itself is periodic
- ◆ Polynomial can be turned into a filter!

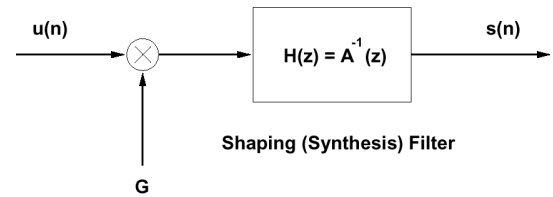
350

## Weights and Error Functions

- ◆ By a simple transformation, we can view the weights  $a_k$  as the coefficients of an all-pole filter
- ◆ For a large class of sounds, the error is either (a) noise (unvoiced speech) or (b) an impulse train (voiced speech)
- ◆ This leads to a very convenient physical model

351

## LPC Model



- ◆  $u(n)$  = excitation function derived from LPC error -- may include pitch function
- ◆  $G$  = gain
- ◆  $H(z)$  = time-varying filter derived from LPC polynomials

352

## So what?

- ◆ LPC analysis provides a nice source-filter model (glottal pulses + vocal tract)
- ◆ The LPC filter captures the spectral envelope of the signal well
- ◆ There are excitation functions that can be used to model a large class of signals (see CELP/VoIP)
- ◆ (or) An adaptive system can transmit the excitation functions that are characteristic of a given source
- ◆ LPC is especially applicable to speech
- ◆ Both analysis/resynthesis can be implemented efficiently

353

## LPC Analysis

- ◆ Two methods for deriving pole weights: autocorrelation and covariance
- ◆ Complexity generally scales with window size and number of poles
- ◆ Error can be classified into noise or periodic
- ◆ Pitch-tracking necessary to exactly determine pitch of excitation

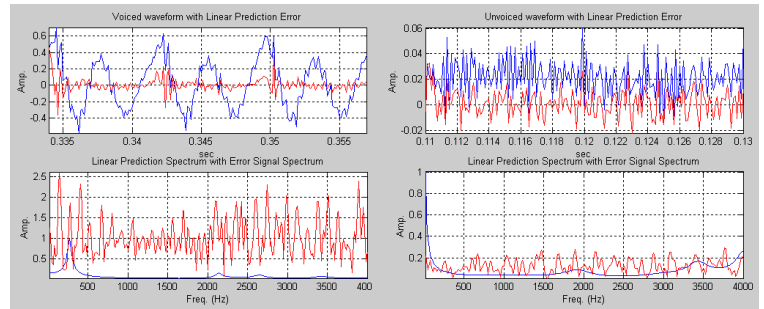
354

# LPC: The Math

- ◆ Polynomial  $s(n) \approx a_1s(n-1) + a_2s(n-2) + \dots + a_ps(n-p),$
- ◆ As summation  $s(n) = \sum_{i=1}^p a_i s(n-i) + Gu(n),$
- ◆ In Z-domain  $S(z) = \sum_{i=1}^p a_i z^{-i} S(z) + GU(z)$
- ◆ Error signal  $e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^p \alpha_k s(n-k)$
- ◆ Transfer function  $H(z) = \frac{S(z)}{GU(z)} = \frac{1}{1 - \sum_{i=1}^p a_i z^{-i}} = \frac{1}{A(z)}$

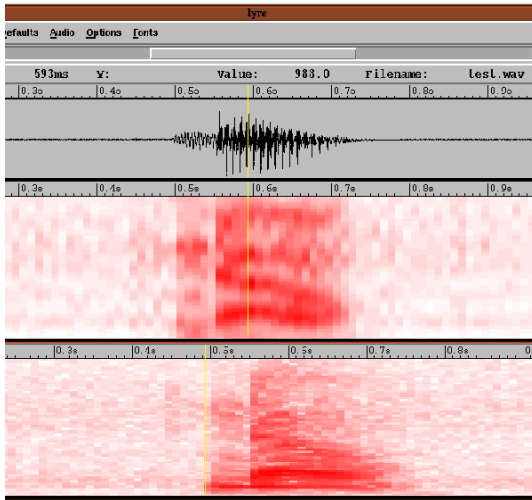
355

## Examples of Voiced and Unvoiced Speech



356

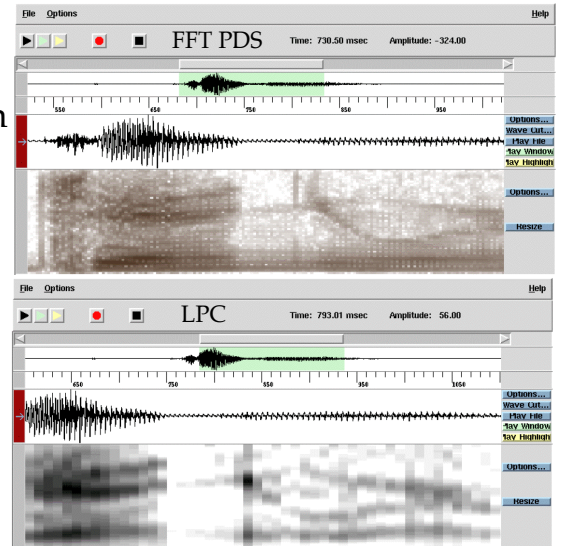
## LPC vs. Power Density Spectrum



357

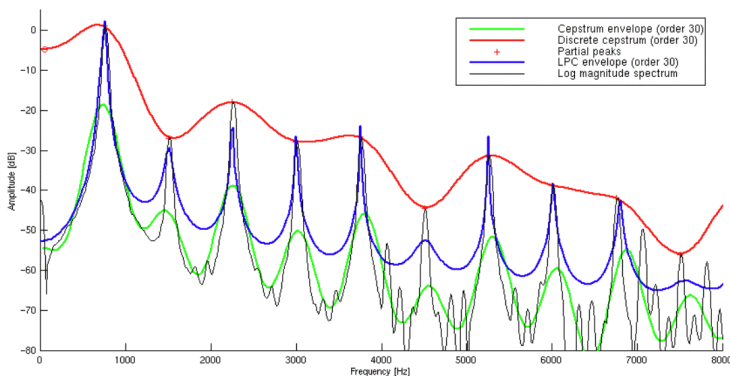
## Spectral Analysis Comparison

- ◆ Pitch/time trade-off
- ◆ Scalability of the model
- ◆ Feature extraction



358

## Smoothed Spectrum Types



359

## Review

- ◆ Transforms and mappings
- ◆ Time- and frequency-domain signals
- ◆ Spectral representations, Fourier transform
- ◆ Linear prediction, signal modeling by polynomials

360

# Topic 7: Applications

## Topic 7



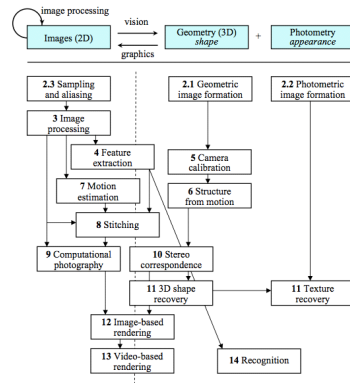
- ◆ Techniques in applications
- ◆ Image edge / object / patch detection
- ◆ Motion estimation in video
- ◆ Feature extraction from speech and music
- ◆ Gesture capture and analysis

361

362

## Image Processing

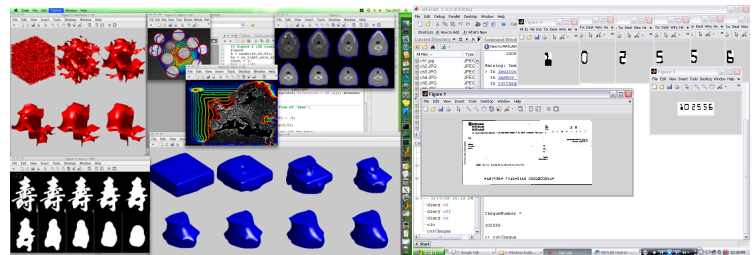
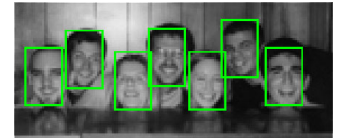
- ◆ Image creation, synthesis, rendering
- ◆ Image analysis, feature extraction
- ◆ <http://www.ph.tn.tudelft.nl/Courses/FIP/frames/fip.html>
- ◆ MATLAB image processing libraries
- ◆ Analysis, matching, OCR, etc.



363

## Examples

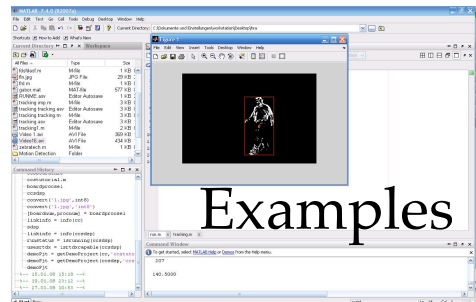
- ◆ Face detection
- ◆ Image segmentation toolbox
- ◆ Check-number reading



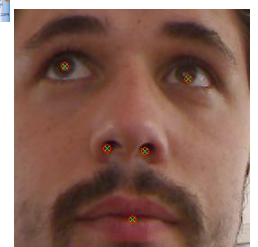
364

## Video Processing

- ◆ Image processing on video
- ◆ Motion processing
- ◆ Inter-frame processing
  - ◆ Compression
- ◆ Animation and video synthesis
- ◆ Video feature extraction and computer vision



- ◆ Video motion detection
- ◆ Eye-tracking from video



365

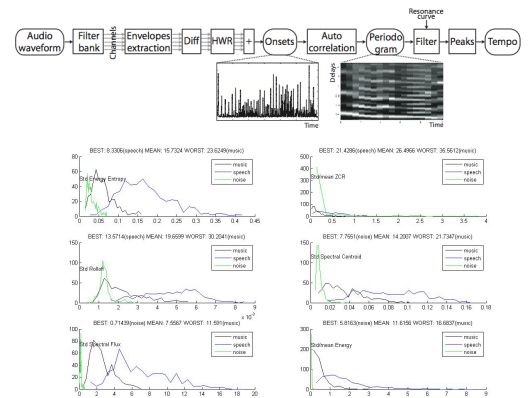
366

# Audio DSP (DASP)

- ◆ Synthesis techniques (MAT 240D)
- ◆ DASProcessing <http://www.musicdsp.org>
- ◆ Analysis, music information retrieval
- ◆ Computer Audition Toolbox
  - ◆ <http://music.ucsd.edu/~sdubnov/ComputerAudition.htm>
- ◆ MATLAB Auditory Toolbox
  - ◆ <http://cobweb.ecn.purdue.edu/~malcolm/interval/1998-010>

# DASP Examples

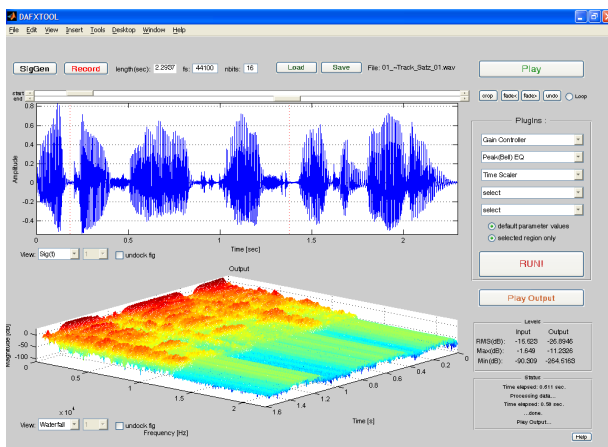
- ◆ MIR
- Toolbox
- ◆ Audio features



367

368

# DAFX Tool



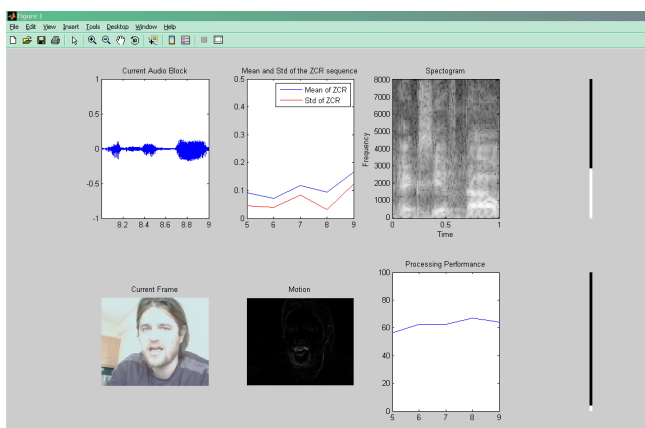
369

370

# Sensor Processing

- ◆ Gesture capture (sensor systems)
  - ◆ MAT 5940
- ◆ Gesture feature extraction
- ◆ Gesture classification
  - ◆ HMM, KNN

# Multi-media



371

372

# Topic 8: Projects

- ◆ SSUM examples
- ◆ Image processing, feature extraction
- ◆ Video processing, computer vision
- ◆ Audio feature extraction, onset detection
- ◆ Gestural input and gesture detection
- ◆ Capture, archival, distribution, transcoding

# MAT 201A Review

- ◆ Multimedia applications tour
- ◆ Data representations
- ◆ Operations on signals
- ◆ Information theory of signals
- ◆ Transforms and mappings
- ◆ Techniques in applications
- ◆ Projects

373

# Where do we go from here?

- ◆ MAT 201B: Computing with Media Data
- ◆ MAT 200C: Survey
- ◆ DASP courses (MAT 240A-F)
- ◆ DISP courses (Turk, Manjunath)
- ◆ Applications in MAT
- ◆ Creative systems

374

# DSP Courses @ UCSB

- ◆ ECE 158 (Digital Signal Processing)
- ◆ ECE 178 (Digital Image Processing)
- ◆ ECE 258A (Advanced DSP)
- ◆ ECE 596E (Speech Processing and Recognition)
- ◆ ECE 242 (Quantization and Signal Compression)
- ◆ CS 181B (Computer Vision)
- ◆ CS 182 (Multimedia Computing)

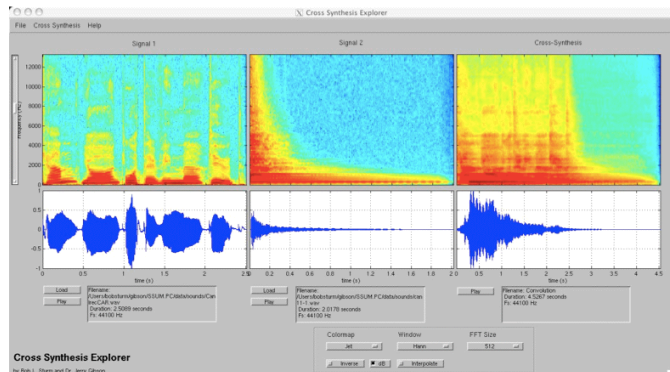
375

# What did we skip?

- ◆ Lots of more advanced topics
  - ◆ Representation details and trade-offs
    - ◆ Laplace/wavelet domains
  - ◆ Signal processing techniques galore
  - ◆ Higher-level math/stats
  - ◆ Signal classification
  - ◆ Applications

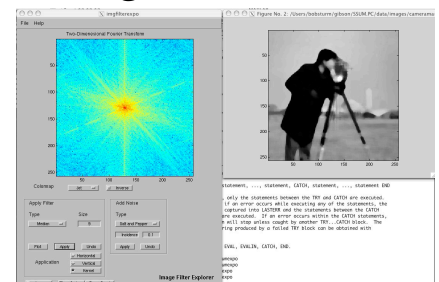
376

Thank you!



# UCSB MAT 201A: Media Signal Processing (Fall, 2009)

Stephen Travis Pope,  
Graduate Program in Media  
Arts and Technology,  
University of California,  
Santa Barbara



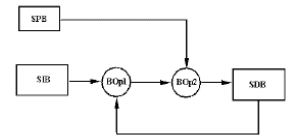
378

# Course Announcement MEDIA ARTS & TECHNOLOGY PROGRAM

## MAT 201B Programming with Media Data (Fall, 2007)

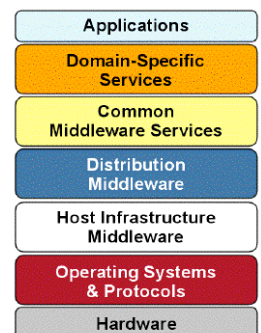
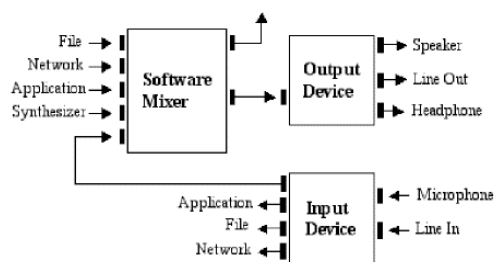
### Overview

MAT 201B is a hands-on graduate-level software development course related to media-rich applications and tools. The course reader consists of a mix of computer science and software engineering theory, and praxis-oriented how-to programming tutorials on media data processing and modern software development. The lecturers will concentrate on the concrete skills required for development of multimedia applications and tools in mainstream programming languages using the standard off-the-shelf software development libraries. In the practical part, students work on several platforms (Linux/UNIX, MS-Windows, and Macintosh OSX), and develop programs for multimedia tasks such as file I/O, data streaming and playback, content creation/editing, format conversion, and data analysis.



### Outline

- 1: Software development: languages and tools
- 2: Application organization and I/O models
- 3: Software engineering: objects, methodology
- 4: Computing with sound and music data
- 5: Computing with still and dynamic images
- 6: Programming GUIs and interactive software
- 7: Distributed systems and network programming
- 8: Control protocols and sensor input
- 9: Application domain frameworks



### Prerequisites

Students are expected to be functionally proficient in C, C++, or Java, to understand the principles of data structures and algorithms, and to be able to use an integrated development environment such as VisualStudio, Xcode, or Eclipse. Students may take MAT 233 (Introduction to Computer Software Development) at the same time as MAT 201B. Grading will be on the basis of in-class participation, quizzes, and programming projects.

### Course Materials

A reader (excellent reference) and the full presentation slides for this course (suitable for framing) will be available at the UCSB book store; the course web site (bookmark it) includes many links to further references as well as down-loadable example software.

### Instructor

Stephen T. Pope (stp@mat.ucsb.edu)

### Meeting Time and Place

Tuesday/Thursday 9:00 - 11:00 AM

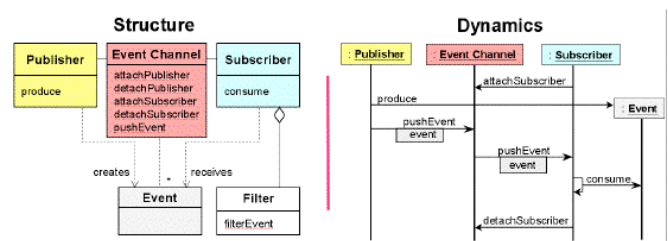
First meeting: September 25, 2007

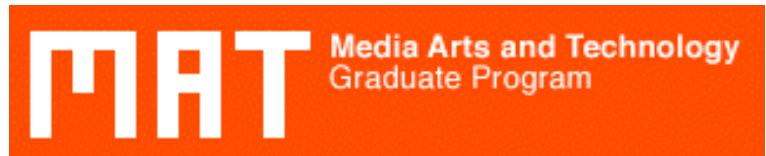
CREATE class room, Music 2215

### Course Web Site

Includes detailed outline and reader table of contents

See <http://www.create.ucsb.edu/201B>





# **MAT 201B: Computing with Media Data**

## **UCSB Fall Quarter 2007**

Instructor: Stephen Travis Pope, [stp@mat.ucsb.edu](mailto:stp@mat.ucsb.edu)

Teaching Asst: Alex Kouznetsov, [alex@create.ucsb.edu](mailto:alex@create.ucsb.edu)

URL: <http://create.ucsb.edu/201B>

Format: 2 lectures and 1 lab / discussion per week (all required)

Meeting Times: Lecture: Tues/Thurs 9:00 - 11:00, Music 2215

Lab: TBD

Materials: Readings book and presentation slides  
(both at the UCSB book store)

### **Topics: Computing with Media Data**

- 1: Software development: languages and tools
- 2: Application organization and I/O models
- 3: Software engineering: objects, methodology
- 4: Computing with sound and music data
- 5: Computing with still and dynamic images
- 6: Programming GUIs and interactive software
- 7: Distributed systems and network programming
- 8: Control protocols and sensor input
- 9: Application domain frameworks, review

### **Reader Contents**

#### *Topic 1. Software Development*

##### Programming Languages

OSData.com Programming Languages Overview ([www](http://www.osdata.com))

Programming Languages: A Collection of Home Pages (www)  
Survey of Object-Oriented Programming Languages. Chris Hostetter (www)  
Web Programming Languages (www)  
Scriptometer: Measuring the ease of Script-Oriented-Programming Languages  
(www)  
An Overview of Language Implementation. Terence Parr (www)

#### Software Development Tools

Requirements for an Experimental Programming Environment. L. P. Deutsch & E. A. Taft. Xerox PARC, 1980.  
GNU C Compiler (gcc) Manual Page  
What's New in Eclipse 3.1. Ed Burnette. *Java Developer's Journal*, 6/2005  
Review of JTest 7.0. Venkant. *Java Developer's Journal*, 6/2005  
Concurrent Version System (CVS) Overview (www)

#### APIs and Libraries

Introduction to the Standard Template Library (www)  
Boost Libraries and Documentation (www)  
OpenDirectory C++ Class Library List (www)

#### Programming Techniques for Multimedia

Clear, Efficient Audio Signal Processing in ANSI C, Adrian Freed (www)  
Music Programming with the new Features of Standard C++, Adrian Freed and Amar Chaudhary (www)  
Writing Portable Programs (Princeton CS 217)

### Topic 2: Application Organization

#### Software Architecture

The Classical OS Model in UNIX (Duke Systems and Architecture)  
UNIX Network Programming: 6.2: I/O Models. W. Richard Stevens, Bill Fenner, Andrew M. Rudoff. Addison/Wesley Publishers

#### Multi-threaded Programming

Getting Started with POSIX Threads. Tom Wagner and Don Towsley. UMass  
YoLinux Tutorial: POSIX Thread Libraries (www)

#### Databases

SQL: Chapter 2: Files, Databases and Database Management Systems (www)  
SQL: Chapter 3: Relational Databases  
SQL: Chapter 4: SQL Data Manipulation Language  
SQL: Chapter 5: SQL Data Definition Language

### *Topic 3: OO Software Engineering*

#### Errors and Exception Handling

C++.com Tutorial: 5.3: Exception Handling (www)

COP 2334 Lecture Notes: C++ Exception Handling (www)

#### Software Engineering and O-O Technology

Object-Oriented System Development: Chapter 2: Introduction to Analysis (www)

Object-Oriented System Development: Chapter 12: The Analysis Process

Object-Oriented System Development: Chapter 26: From Design to Implementation

Object-Oriented System Development: Appendix: Notation

MKS Integrity Suite 2005 Review. M. Sayko. *Java Developer's Journal*, 6/2005

#### O-O Design Patterns

Patterns and Software: Essential Concepts and Terminology, Brad Appleton

Software Patterns Lecture Slides (X. Amatriain)

The Design Patterns Java Companion. James W. Cooper. Addison-Wesley.

Java Tutorial: How do These Concepts Translate into Code? (www)

#### Testing

Eclipse Tutorial: Using JUnit in Eclipse. Christopher Batty. (www)

[Advanced UNIX Programming course (www)]

[C++ libraries: <http://www.thefreecountry.com/sourcecode/cpp.shtml>]

### *Topic 4: Digital Audio*

"A Child's Garden of Sound File Systems" Pope and Van Rossum. CMJ 19:1

PortAudio Tutorial, P. Burke. (www)

### *Topic 5: Computer Graphics*

"Graphics File Formats." D. Meyer.

"Overview of OpenGL." B. Lipchak.

Common Image File Formats (www)

Image Formats. Joe Burns. (www)

Compression, Encoding and Graphics Files: A Practical Tutorial (slides). Rich Holowczak

Parsing and Writing QuickTime Files in Java. Chris Adamson (www)

Star Trek Technology for Java 3D (SysCon)

[File formats -- <http://desktoppub.about.com/od/graphicformats>, <http://graphicssoft.about.com/od/aboutgraphics/a/bitmapvector.htm>]

### *Topic 6: Applications and GUIs*

QT 3.3 Whitepaper. Trolltech.

Eclipse Tutorial: Basic SWT Widgets. Shantha Ramachandran. (www)

*Topic 7: Distributed Applications and Networking*

Internet: "The Big Picture." Russ Haynal. (www)

Network Layers (www)

TCP/IP Networking Protocols (www)

Beej's Guide to Network Programming. Brian Hall.

Remote Method Invocation: Java RMI and Web Services. A. Konstantinou.

*Topic 8: Control and Sensor I/O*

PortMIDI Header file

OpenSoundControl Paper

Web References

*Topic 9: Domain-specific Application Frameworks*

AlloSphere ICMC 2007 Paper

Siren 7.5 ICMC 2007 Paper

Web References

# MAT 201B Code Archive

- Hello World examples
- APIs for media
- C, C++, Java
- Basic media types

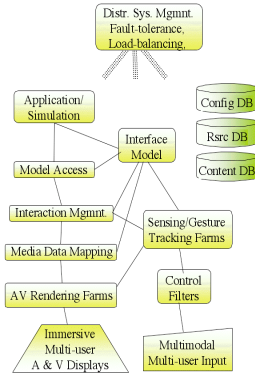
▶	2004_students	65.1 MB
▶	2005_Students	18 MB
▶	BasicCarbonApp	175.2 KB
▼	DrawScheduler	12.6 KB
	DrawScheduler.class	2.7 KB
	DrawScheduler.java	2.2 KB
	score copy.dat	824 bytes
	score.dat	816 bytes
▶	File_Formats	108.2 KB
▶	gamedesigndoc	180 KB
▶	Helix	42.9 MB
▼	HelloWorld	5.9 MB
	▶ HelloWorldInC	73.3 KB
	▶ HelloWorldInC++	56.9 KB
	▶ HelloWorldWithABug	75.1 KB
	▶ HelloWorldWithGUI	5.7 MB
	HelloWorld.c	455 bytes
	README	3.3 KB
▶	java2d	389.1 KB
▶	java3d11-examples	2.6 MB
▶	JavaAudioSamples	761.2 KB
▶	javashout	45.9 KB
▶	midi_sched	55.9 KB
▶	midi_test	277.2 KB
▼	PerfCode	7 MB
	a.out	12.5 KB
	Benchmarks.c	2.4 KB
	ras_file_info.c	7.1 KB
	slk.ras	125.3 KB
	slk.sun	25.2 KB
	Venice.ras	6.8 MB
▶	pm_mac_cocoa_play	258 KB
	Divider.java	1.7 KB
	lsf.class	2.8 KB
	lsf.java	3.4 KB
	opengl_demo.c	830 bytes
	SimpleAudioPlayer.java	9.8 KB
	snd_player	12.7 KB
	snd_player.c	3.5 KB
	snd_player2.c	4.7 KB



## MAT 201B: Computing with Media Data

UCSB, Fall 2007

Stephen T. Pope  
stp@mat.ucsb.edu



MEDIA ARTS & TECHNOLOGY PROGRAM

1

## Why MAT 201B?

- \* The big question about multimedia software development:
- \* Whereas: you've taken the 1-semester "intro-2-CS" course, followed by the 1-semester "algorithms and data structures" course.
- \* What's missing to be a high-paid state-of-the-art multimedia application, game, content, or tool developer?

MEDIA ARTS & TECHNOLOGY PROGRAM

2

## Ergo: MAT 201B Outline

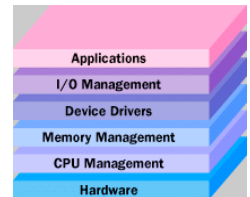
- \* 1: Software development: languages and tools
- \* 2: Application organization and I/O models
- \* 3: Software engineering: objects, methodology
- \* 4: Computing with sound and music data
- \* 5: Computing with still and dynamic images
- \* 6: Programming GUIs, interactive software
- \* 7: Distributed systems, networking
- \* 8: Control protocols, sensor input
- \* 9: Application domain frameworks

MEDIA ARTS & TECHNOLOGY PROGRAM

3

## MAT 201B: Lecture 1

- \* Logistics/Administrative
- \* Topical Overview
- \* Topic 1 Introduction
- \* What tools to use for MAT201B
- \* Exercises



MEDIA ARTS & TECHNOLOGY PROGRAM

4

## Logistics

- \* Meeting time: Tu/Th 9:00 - 10:50 AM
- \* Meeting place: CREATE class room (Music 2215, Studio Xenakis)
- \* Office hours: Tu/Th 1:00 - 3:00 PM
  - \* South Hall 4340F
- \* TA: Alex Kouznetsov
- \* TA office hours: TBD
- \* Lab meeting time/place: TBD
- \* Mailing list: <http://www.mat.ucsb.edu/mailman/201B>
- \* Machine and SW tool access

MEDIA ARTS & TECHNOLOGY PROGRAM

5

## Course Materials

- \* Course reader
  - \* Tutorials, background papers
- \* Course presentation slides
- \* Example code (mailed to list and on web site)
- \* Course web site and links
- \* Web references
  - \* Tutorials and API documentation
- \* Programming book
  - \* Pick a good C, C++, or Java tutorial (and/or O'Reilly pocket references)

MEDIA ARTS & TECHNOLOGY PROGRAM

6

## Prerequisites

- \* “Moderate” proficiency in a mainstream (low-level procedural/object-oriented) programming language (i.e., C, C++, or Java)
- \* Basics of language syntax, control flow, procedure/method decomposition, module/class source code structure
- \* Basics of numerics, IO, and other libraries
- \* Use of a modern integrated development environment (IDE)

MEDIA ARTS & TECHNOLOGY PROGRAM

7

## Grading

- \* Class attendance and participation
- \* Homework assignments
- \* In-class quizzes
- \* Programming projects
- \* Final presentation

MEDIA ARTS & TECHNOLOGY PROGRAM

8

## Introductions

- \* Stephen T. Pope
  - \* Education: EE/CS, recording engineering, music
  - \* Experience: academia, industrial R&D, start-ups, music, video composition
  - \* SW development: graphics (CORE/GKS, WSH, SPIM), audio (Siren, CSL, etc.), AI (UNCLE, EMA), GUI (ST80 MVC, etc.), distributed apps (CORBA, TINA/C DPE, CRAM)
- \* TA: Alex Kouznetsov

MEDIA ARTS & TECHNOLOGY PROGRAM

9

## What will we be learning?

- \* The big question about multimedia software development:
- \* Whereas: you’ve taken the 1-semester “intro-2-CS” course, followed by the 1-semester “algorithms and data structures” course,
- \* What’s missing to be a high-paid state-of-the-art multimedia application, game, content, tool developer?
- \* (Relation to MAT 200C)

MEDIA ARTS & TECHNOLOGY PROGRAM

10

## Topical Outline

- \* 1: Software development: languages and tools
- \* 2: Application organization and I/O models
- \* 3: Software engineering: objects, methodology
- \* 4: Computing with sound and music data
- \* 5: Computing with still and dynamic images
- \* 6: Programming GUIs and interaction
- \* 7: Distributed systems and networking
- \* 8: Control and sensor input
- \* 9: Application domain support
- \* (I know it’s a lot)

MEDIA ARTS & TECHNOLOGY PROGRAM

11

## Topic 1: Software Development

- \* Survey of current multimedia applications
- \* Programming languages: theory and practice
- \* Development tools and integrated development environments
- \* Application programmer’s interfaces (APIs)
- \* Source code control and versioning tools
- \* Exercises: “Hello World” in several versions on several platforms

MEDIA ARTS & TECHNOLOGY PROGRAM

12

## Topic 2: Application Organization

- \* Application models and architecture -- batch processing, command-line, WIMP model
- \* UNIX I/O model and APIs
- \* Real-time systems and schedulers
- \* Multi-threaded applications
- \* Databases for various kinds of data
- \* Programming techniques for media SW
- \* Exercises: "Hello World" GUIs and extended versions, schedulers

— MEDIA ARTS & TECHNOLOGY PROGRAM

13

## Topic 3: OO Software Engineering

- \* Object-oriented software technology: theory and practice
- \* Software analysis/design methodologies and notations
- \* OO design patterns
- \* Control flow, error handling, and exceptions
- \* Exercises: robust coding

— MEDIA ARTS & TECHNOLOGY PROGRAM

14

## Topic 4: Computing with Sound and Music Data

- \* Music/sound representation, notation, and interchange formats
- \* Representation: sampling and quantization
- \* Sound file formats for storage and interchange
- \* Sound I/O APIs

— MEDIA ARTS & TECHNOLOGY PROGRAM

15

## Topic 5: Computing with Still and Dynamic Images

- \* Geometry for computer graphics
- \* Raster and vector representations and transformations
- \* Image file formats
- \* Structured graphics APIs: OpenGL, VRML, Java3D, OpenSceneGraph, etc.
- \* Exercises: drawing things

— MEDIA ARTS & TECHNOLOGY PROGRAM

16

## Topic 6: Applications and GUIs

- \* Applications and window systems
- \* Geometry for GUIs
- \* GUI Frameworks and widget libraries
- \* GUI construction tools: platform-specific and cross-platform
- \* Code examples

— MEDIA ARTS & TECHNOLOGY PROGRAM

17

## Topic 7: Distributed Applications, Streaming, and Networks

- \* Distributed processing background: CORBA, RMI, SOAP, etc.
- \* IP networking basics
- \* Socket programming and network streaming
- \* Application-specific protocols
- \* Exercises: Socket-based streaming, content servers

— MEDIA ARTS & TECHNOLOGY PROGRAM

18

## Topic 8: Control and Sensor I/O

- \* Control vs. data
- \* MIDI data and MIDI APIs
- \* Open Sound Control
- \* Sensor input and drivers
- \* Control APIs
- \* HID devices

MEDIA ARTS & TECHNOLOGY PROGRAM

19

## Topic 9: Domain-specific Application Frameworks

- \* Performance systems
- \* Gaming platforms
- \* Virtual environments
- \* Network-based content delivery
- \* Exercises: applications
- \* Course review

MEDIA ARTS & TECHNOLOGY PROGRAM

20

## 201B Code Archive

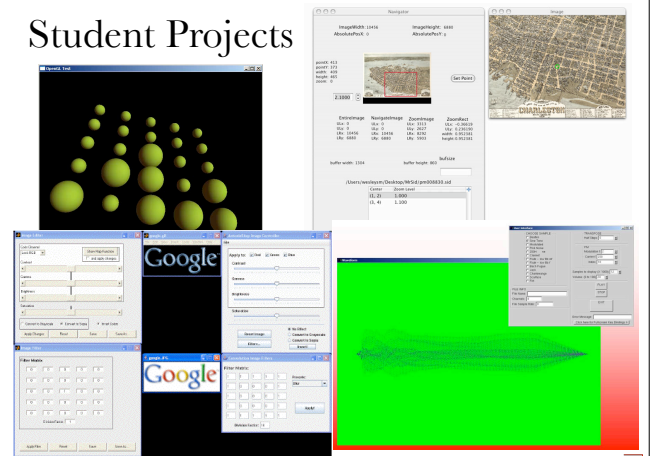
- \* Programming examples
- \* Audio code
- \* Graphics code
- \* GUI code
- \* Networking code
- \* APIs
- \* Upgrades
- \* BOOST, JUCE

BasicCarbonApp  
DrawScheduler  
File\_Formats  
gamedesigndoc  
Helix  
HelloWorld  
java2d  
java3d11-examples  
JavaAudioSamples  
javashout  
midi\_sched  
midi\_test  
PerfCode  
pm\_mac\_cocoa\_play  
CHUD\_4.2.2.dmg  
CoreAudioSDK1.4.2.dmg  
Divider.java  
Isf.java  
main.c  
SimpleAudioPlayer.java

MEDIA ARTS & TECHNOLOGY PROGRAM

21

## Student Projects



MEDIA ARTS & TECHNOLOGY PROGRAM

22

## Topic: MM Overview



MEDIA ARTS & TECHNOLOGY PROGRAM

23

## 1: Introduction: Software Development Basics

- \* Setting the stage, MM History
- \* Overview of mainstream programming languages (as used in media applications)
- \* Using integrated development environments
- \* Using source code control and versioning tools
- \* Tips: how to learn a new API
- \* Exercises: "Hello World" in several versions on several platforms

MEDIA ARTS & TECHNOLOGY PROGRAM

24

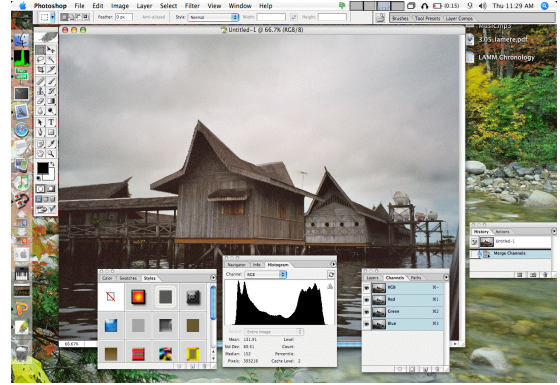
## Application Examples

- \* Sampling of modern media-rich applications (tools)
- \* What's the data
- \* What are the operations
- \* How is interaction managed?
- \* Can you describe it in terms of model, view, and controller classes?
- \* Can you add in your own favorite app?

MEDIA ARTS & TECHNOLOGY PROGRAM

25

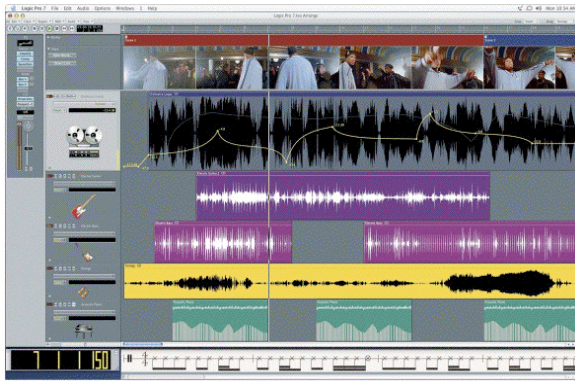
## Application: Photoshop



MEDIA ARTS & TECHNOLOGY PROGRAM

26

## Application: Logic Audio



MEDIA ARTS & TECHNOLOGY PROGRAM

27

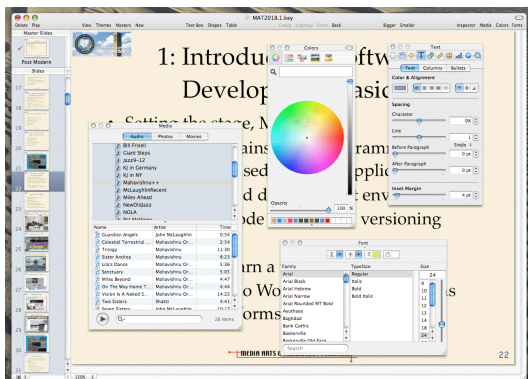
## Application: Ardour DAW



MEDIA ARTS & TECHNOLOGY PROGRAM

28

## Application: Keynote



MEDIA ARTS & TECHNOLOGY PROGRAM

29

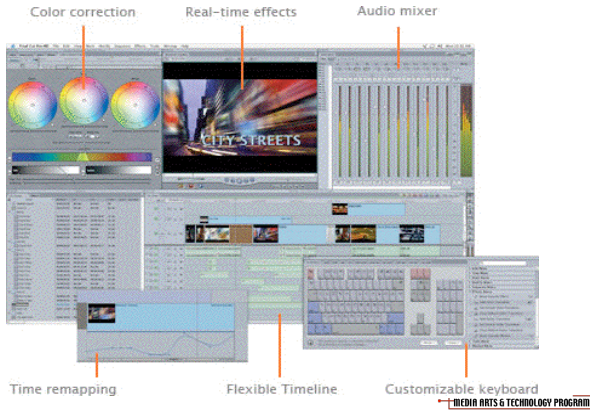
## Application: Logic Sculpture



MEDIA ARTS & TECHNOLOGY PROGRAM

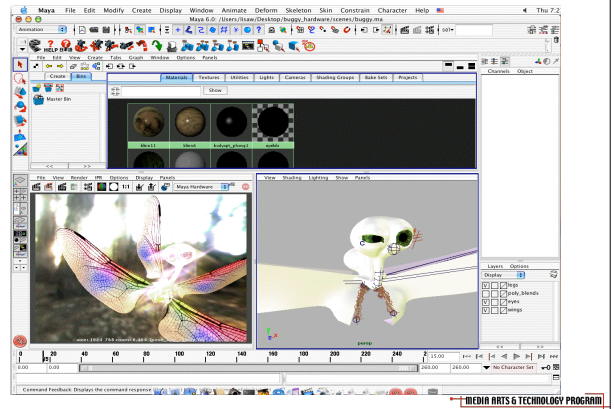
30

## Application: Final Cut Pro



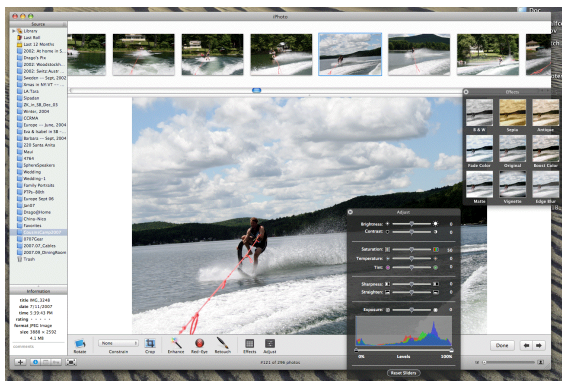
31

## Application: Maya



32

## Application: iPhoto



33

## Application: MODE (1992)



34

## What's different?

- \* ...about multimedia applications (i.e., not covered in off-the-shelf SW textbooks)?
- \* Media as data model
- \* Architecture of complex interactive software
- \* Input/output modes, devices
- \* Multi-threaded UI, processing, streaming IO
- \* User interface (task, widgets)
- \* Content databases or streaming
- \* Networking: content, control, and interaction
- \* Programming techniques (?)

35

## Question

- \* So, how do we get there?
- \* Look at theoretical and practical issues
- \* Learn about APIs and tools for developing media-rich applications (graphics, audio, stored data, networking, interactivity)
- \* Separate what's truly hard from already-answered technological questions.

36

## The Game Plan

- \* CS theory
  - \* Programming languages
  - \* Software tools (compilers, development tools)
- \* Software engineering technology
  - \* Structured and object-oriented techniques
  - \* Analysis and design
- \* Advanced programming techniques
  - \* Real-time, non-linear, multi-threaded, multi-host, multi-platform, multi-user
- \* Multimedia APIs
  - \* Snd, image, threads, controllers

— MEDIA ARTS & TECHNOLOGY PROGRAM

37

## A Brief Multimedia History

- \* Pre-Digital Age
- \* Late 1890s – Radio was introduced
- \* Early 1900s – Motion picture introduced
- \* 1906 – Color photography made practicable
  - \* <http://www.niepce.com/pagus/pagus-inv.html>
- \* 1940s – Television
- \* 1945 – Vannevar Bush, memex “As We May Think”
  - \* <http://www.theatlantic.com/unbound/flashbks/computer/bush.htm>
- \* 1960s – Ted Nelson, Xanadu, “a universal instantaneous hypertext publishing network”
- \* 1967 – Nicholas Negroponte formed  
M  
I  
T Architecture Machine Group (later in 1985 MIT Media Lab opens)

— MEDIA ARTS & TECHNOLOGY PROGRAM

38

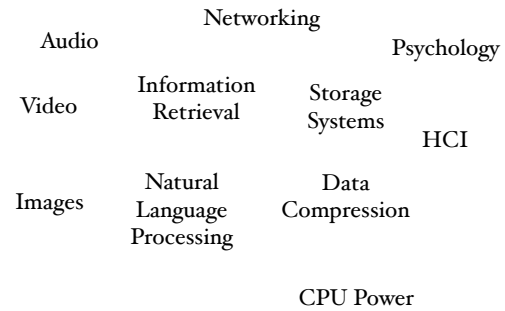
## A Brief Multimedia History

- \* 1983 – Audio CD introduced
- \* 1987 – RCA's David Sarnoff Labs' announce Digital Video Interactive
- \* 1989 – Tim Berners-Lee proposed the World Wide Web to CERN
- \* 1989 – Fraunhofer receives german patent for mp3
- \* 1991 – Motion Picture Experts Group (MPEG)
- \* 1993 – NCSA Mosaic (first browser)
- \* 1994 – Netscape; creation of World Wide Web Consortium (W3C)
- \* 1995 – JAVA for platform-independent application development
- \* 1996 – PNG (Portable Network Graphics)
- \* 1997 – HTML 4.0
- \* 1998 – XML 1.0
- \* 1999 – XSLT 1.0 and Xpath 1.0
- \* 2001 – MPEG-7, JPEG 2000, SVG
- \* 2002 – intellectual property and JPEG 2000

— MEDIA ARTS & TECHNOLOGY PROGRAM

39

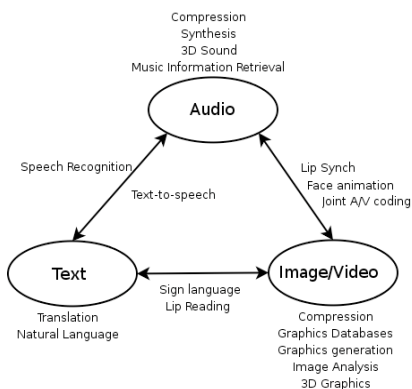
## Multimedia Topics



— MEDIA ARTS & TECHNOLOGY PROGRAM

40

## Multi-media



— MEDIA ARTS & TECHNOLOGY PROGRAM

41

## Elements of a Multimedia System

- \* Person-to-person communication



- \* User interface provides mechanisms for users to interact with each other and creates the multimedia signal
- \* The Transport layer is the channel in the traditional S&W sense
- \* Ex: teleconferencing, videophones, shared workspaces, distant learning...

— MEDIA ARTS & TECHNOLOGY PROGRAM

42

## Elements of a Multimedia System

### \* Person-to-machine communication



- \* The user creates multimedia information and “stores” it in a machine or the user retrieves information already existing in a machine
- \* Ex: broadcast video, interaction with archival collections, iPod...

MEDIA ARTS & TECHNOLOGY PROGRAM

43

## Properties of Multimedia Systems

### \* Combination of Media

- \* Some authors require one continuous (time-dependent) and one discrete (time independent) media

### \* Independence

- \* Different media should be arranged to enable independent processing and allow for different combinations

### \* Computer supported integration

- \* Computer support is a must and allows for high-level of integration and coherence

MEDIA ARTS & TECHNOLOGY PROGRAM

44

## Properties of Multimedia Systems

### \* Communication Systems

- \* A multimedia system should be able to communicate with another multimedia system or a remote user

### \* Digital

- \* It ensures many of the above as any kind of information can be represented independently as a series of 0's and 1's

### \* Interactive

- \* Must provide complete user interaction

MEDIA ARTS & TECHNOLOGY PROGRAM

45

## Classification of Media

### \* Media is an essential component of any

multimedia system. It can be classified along several criteria:

### \* Perception

- \* What perceptual sense is exploited?
- \* Note that although current systems mostly only exploit 2 senses ideally all of them should be involved

### \* Representation

- \* How is the multimedia information represented?
- \* E.g. ASCII code for text or PCM samples for audio

MEDIA ARTS & TECHNOLOGY PROGRAM

46

## Classification of Media

### \* Presentation

- \* What tools and devices are used to input and output the information?
- \* Paper, screen, speakers... for output
- \* Keyboard, mouse, microphone... for input

### \* Storage

- \* What is the format of the data carrier enabling storage?
- \* E.g. Paper, HD, CD, DVD...

MEDIA ARTS & TECHNOLOGY PROGRAM

47

## Classification of Media

### \* Transmission

- \* What is the information carrier for data transmission?
- \* E.g. Optical fiber, coaxial cable, air...

### \* Discrete/Continuous

- \* Discrete or Time independent
  - \* Data processing is not time critical
  - \* E.g. Text, graphics

### \* Continuous

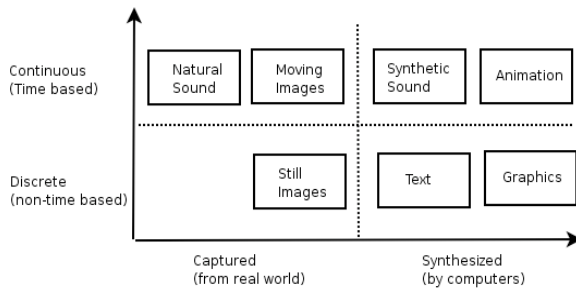
- \* Data processing is time critical

- \* In a combined multimedia system data processing is ALWAYS critical!

MEDIA ARTS & TECHNOLOGY PROGRAM

48

## Classification of Media



MEDIA ARTS & TECHNOLOGY PROGRAM

49

## Requirements of a MM Service

- \* From a user perspective:
  - \* Instant availability
  - \* Real-time information transfer and feedback
  - \* Constant online availability
  - \* Access from different points with different terminal configurations

MEDIA ARTS & TECHNOLOGY PROGRAM

50

## Requirements of a MM Service

- \* From a workstation perspective
  - \* High Processing speed to perform software-based real-time multimedia signal processing
  - \* Efficient Architecture to provide high-speed communication between the CPU and RAM or peripherals
  - \* High performance real-time Operating System.
  - \* High capacity and high-speed storage devices.
  - \* Well managed multimedia databases.
  - \* Efficient software tools with intuitive GUIs.

MEDIA ARTS & TECHNOLOGY PROGRAM

51

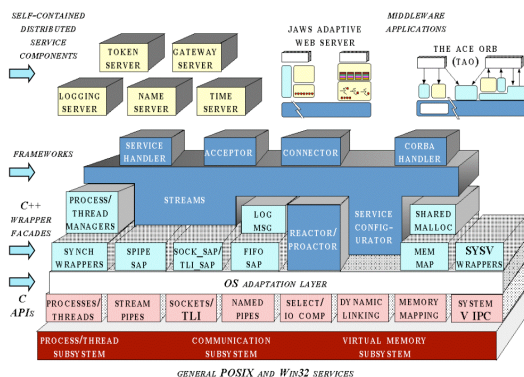
## Requirements of a MM Service

- \* From a network perspective
  - \* High speed and changing bit rates
  - \* Several virtual connections using same access
  - \* Synchronization of different information types
  - \* Standardized services

MEDIA ARTS & TECHNOLOGY PROGRAM

52

## Topic: Computer Science



MEDIA ARTS & TECHNOLOGY PROGRAM

53

## Readings 1

- \* Programming Languages
  - \* Overviews
  - \* Comparisons
  - \* Web languages
  - \* Language implementation

MEDIA ARTS & TECHNOLOGY PROGRAM

54

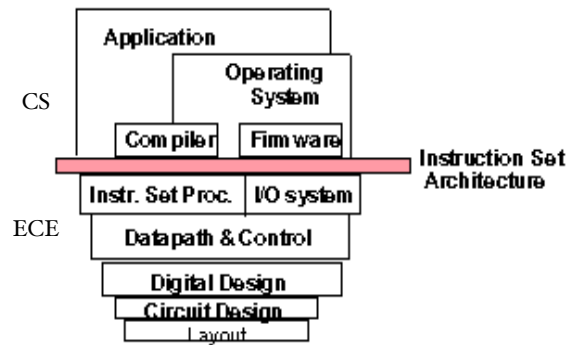
## What is CS/SE all about?

- \* Algorithms and data (behavior and state)
- \* Methodologies: object-oriented analysis, design, and programming
- \* Compilers and software tools
- \* Operating systems and run-time support
- \* Windowing systems and GUIs
- \* (LA-/WA-) Networks and data transmission

— MEDIA ARTS & TECHNOLOGY PROGRAM

55

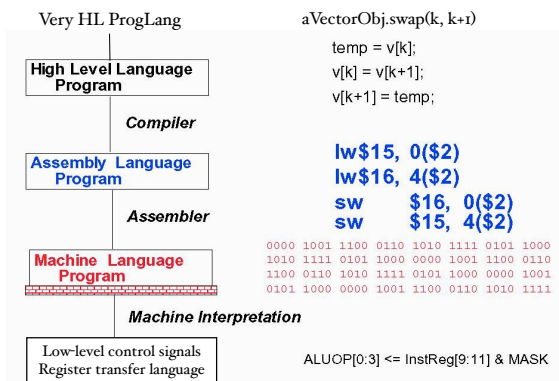
## What's the big picture?



— MEDIA ARTS & TECHNOLOGY PROGRAM

56

## How are computers programmed?

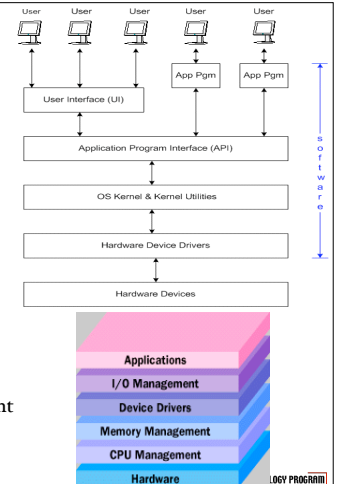


— MEDIA ARTS & TECHNOLOGY PROGRAM

57

## ProgTech: What's in an OS?

- \* Who manages the HW and multiple applications?
- \* CPU & I/O management (HW/FW)
- \* Memory/cache management
- \* Interrupt-handling, timers
- \* I/O drivers, DMA management
- \* APIs for OS interfaces (lots)



— MEDIA ARTS & TECHNOLOGY PROGRAM

58

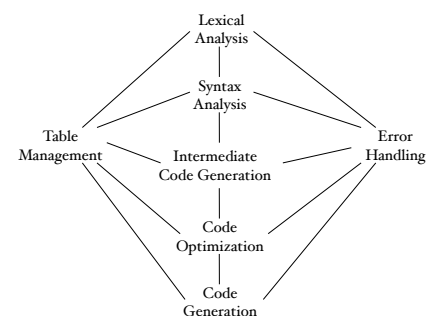
## What does a compiler do?

- \* Pre-process (process includes, macros, comments)
- \* Compile
  - \* Analysis: parse (read, tokenize) input stream; generate internal representation of code (grammar trees)
  - \* Optimize code in the abstract
  - \* Synthesize: generate target CPU (or virtual) assembler code
- \* Assemble
  - \* Optimize the native code
  - \* Translate to binary "obj" file
- \* Link
  - \* Linker combines object modules and external libraries
  - \* Make a single exec file and fill in the fcn jump-to addresses
  - \* OR make a smaller executable that relies on a start-up-time linker and dynamic libraries

— MEDIA ARTS & TECHNOLOGY PROGRAM

59

## Phases of a Compiler



- \* See "Dragon Book" (Aho/Ullman Compiler Design text)

— MEDIA ARTS & TECHNOLOGY PROGRAM

60

## What happens at run time?

- \* Program loader
- \* Start-up-time linker (for dynamic libraries)
- \* Run-time libraries, traps, system calls
- \* Event distribution framework
- \* Optionally:
  - \* Garbage collector
  - \* Interpreter read/eval/print loop
  - \* Network proxies and remote object broker
  - \* Expert system shell or unification engine

MEDIA ARTS & TECHNOLOGY PROGRAM

61

## The “Hello, world” C Example

- \* Used in K&R C Book
 

```
#include <stdio.h> // include, header
int main (int argc, char **argv) { // main fcn signature
 printf("Hello, world.\n"); // operation, print
} // end of main()
```
- \* Compile, assemble, link (gcc file.c; a.out)
- \* Execute in a shell
- \* What’s the run-time support?

MEDIA ARTS & TECHNOLOGY PROGRAM

62

## What types of programming languages are there?

- \* Plenty of adjectives:
  - \* Procedural vs. declarative (functional, logic, etc.)
  - \* Compiled vs. interpreted (meaningful?)
  - \* High-level vs. low-level (what determines?)
  - \* Compile-time vs. run-time types (untyped? 1 type?)
  - \* Sequential vs. non-sequential
  - \* Heuristic vs. imperative
  - \* Programming vs. scripting
  - \* General-purpose vs. domain-specific
  - \* Modeling/specification languages
- \* See readings on the prog. lang. literature, also SIGPLAN and TOPLAS

MEDIA ARTS & TECHNOLOGY PROGRAM

63

## What’s the History of Mainstream Programming Languages?

- \* 1950s: Assembly languages (an improvement over RTL), then “high-level” (abstract) languages -- LISP (symbolic), FORMula TRANslator (numeric)
- \* 1960s: Procedural -- ALGOL, PL/I, Simula
- \* 1970s: Structured procedural -- Pascal, Modula, B/C
- \* 1980s: Object-oriented -- Smalltalk, CommonLISP
- \* 1990s: Big messy hybrids -- C++, Java
- \* 1990s Scripting -- perl, php, python, ...
- \* Next?

MEDIA ARTS & TECHNOLOGY PROGRAM

64

## Historical Programming Languages

- \* Konrad Zuse developed the "Plankalkul" ca. 1945 in Germany
- \* Over 200 programming languages were developed between 1952 and 1972.
- \* Kinnersley’s list of influential languages:
 

<ul style="list-style-type: none"> <li>•1957 <u>F</u>ORTRAN</li> <li>•1958 ALGOL</li> <li>•1960 <u>L</u>ISP</li> <li>•1960 <u>C</u>OBOL</li> <li>•1962 <u>A</u>PL</li> <li>•1962 SIMULA</li> <li>•1964 <u>B</u>ASIC</li> <li>•1964 PL/I</li> <li>•1970 Prolog</li> <li>•1972 <u>S</u>malltalk</li> <li>•1975 <u>C</u></li> <li>•1975 Pascal</li> <li>•1975 Scheme</li> </ul>	<ul style="list-style-type: none"> <li>•1977 <u>O</u>PS</li> <li>•1978 CSP</li> <li>•1978 FP</li> <li>•1983 <u>A</u>da</li> <li>•1983 Parlog</li> <li>•1984 Standard ML</li> <li>•1986 C++</li> <li>•1986 Eiffel</li> <li>•1988 CLOS</li> <li>•1988 Mathematica</li> <li>•1988 Oberon</li> <li>•1989 HTML</li> <li>•1990 Haskell</li> </ul>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

MEDIA ARTS & TECHNOLOGY PROGRAM

65

## What’s out there (in use)?

- \* FORTRAN family: C, C++, Java, C#, MATLAB
- \* Object-oriented: Smalltalk-80, CommonLISP, SuperCollider, Oberon, ObjectiveC
- \* Functional: Haskell, ML, Scheme, Erlang
- \* Logic: Prolog, Mozart, Mercury
- \* Scripting: Python, JavaScript, shell, perl, php
- \* Mixes: O/P, L/F, O/L (see the readings)

MEDIA ARTS & TECHNOLOGY PROGRAM

66

## Yet Another Programming Language Comparison (G. van Rossum, 1999)

	Teachability	Ease of use	Expressiveness	Expert usability
C/C++	hard	hard	good	very good
Java	hard	hard	good	very good
JavaScript	easy	so-so	so-so	poor
Lisp	hard	hard	good	very good
Logo	easy	easy	okay	poor
LogoMation	easy	easy	okay	poor
Perl	hard	hard	very good	very good
Python	easy	easy	good	very good
Scheme	so-so	so-so	good	good
Smalltalk	so-so*	easy	good*	good*
Tcl	easy	so-so	so-so	so-so
Visual Basic	so-so	so-so	so-so	so-so

— MEDIA ARTS & TECHNOLOGY PROGRAM

67

## Dimensions and Goals

- \* Simplicity, minimalism
  - \* Uniformity of data
  - \* Uniformity of functionality
- \* Modular structure
- \* Separation of specification from implementation
- \* Types and safety
- \* Legacy code/skills
- \* Boot-strapping

— MEDIA ARTS & TECHNOLOGY PROGRAM

68

## Criteria for a “High-Level” Programming Language

- \* (After Aho&Ullman Dragon Book)
- \* Ease of understanding
- \* Naturalness
- \* Portability
- \* Efficiency of use
- \* Features that “facilitate reliable programming”
  - \* Data structures
  - \* Scope rules
  - \* Flow-of-control constructs
  - \* Subroutines

— MEDIA ARTS & TECHNOLOGY PROGRAM

69

## What’s not a programming language?

- \* Criteria: 1st-class functional extension, visible types, modules
- \* Data/content notation: HTML, XML
- \* Data type definition language: SQL, XML DTD
- \* Query language: SQL, HTML
- \* Specification language: IDL/ODL
- \* Scriptable application: Max, MusicN, MEL

— MEDIA ARTS & TECHNOLOGY PROGRAM

70

## What are the Elements of a Programming Language?

- \* Required
  - \* Operator/function execution
    - \* `3 + 4; sin(x)`
  - \* Control Flow
    - \* `if (bool) then {block} else {block}`
    - \* (What type is bool, block?)
  - \* Program structuring
    - \* Create new procedures
    - \* Collect these into abstract data types
  - \* Built-in data types (immediate/manifest) (compile-time/run-time)
    - \* `int, float, char*`

— MEDIA ARTS & TECHNOLOGY PROGRAM

71

## “Optional” (desirable) Elements

- \* Comments
  - \* Put descriptive text in code (`// this is C++`)
- \* Variable declaration
  - \* Named data stores (`int x;`)
- \* Variable assignment
  - \* Associate data/name (`x = 7;`)
- \* Modules, interfaces, abstract types
  - \* Separate specification and implementation
- \* Many others
  - \* Assertions, pre/post conditions, rules, exceptions, inheritance

— MEDIA ARTS & TECHNOLOGY PROGRAM

72

## Fine Points

- \* Grouping of type and operation (modules, ADTypes, objects)
- \* Extensible types
- \* Blocks/closures as objects?
- \* Is control flow extensible?
- \* Are operators and function-calls different?
- \* Operator overloading (arithmetic, accessing, etc.)
- \* Is the compiler available at run-time (code-gen in apps)

MEDIA ARTS & TECHNOLOGY PROGRAM

73

## What's a Scripting Language?

- \* General Features
  - \* Lightweight (simple?) syntax
  - \* Probably type-less (uniform [invisible] data model)
  - \* Easy (run-time) functional extension
- \* Many options
  - \* Extension language for an app.
  - \* Data/content notation, dynamic content
  - \* Scriptable application

MEDIA ARTS & TECHNOLOGY PROGRAM

74

## Language “Purity” and “Uniformity”

- \* Uniformity
  - \* Single data type
  - \* Single activation paradigm
  - \* Single module/ADT abstraction
- \* Flexibility, language extensibility
  - \* How much does the compiler know?
  - \* Is a program a data structure?
- \* Special objects
  - \* Booleans, Blocks
  - \* Contexts, StackFrames
  - \* Classes, Namespaces
- \* Examples: Smalltalk, C++

MEDIA ARTS & TECHNOLOGY PROGRAM

75

## What does the syntax mean?

- \* How to form simple and compound statements and expressions
  - \* How to call a function giving it arguments and storing the return value
  - \* How to separate multiple statements
  - \* How to describe control flow options
- \* Deals with special characters, punctuation, comments, indentation, coding style
- \* See examples given above

MEDIA ARTS & TECHNOLOGY PROGRAM

76

## The Lingo (Hell)

- \* Naming differences, e.g.,
  - \* C++        data member
  - \* Java       field
  - \* Smalltalk   instance variable
  - \* LISP, Self   slot
- \* Same with function, class, module, package...

MEDIA ARTS & TECHNOLOGY PROGRAM

77

## Data Types

- \* Dimensions
  - \* Type: storage format (simplest view, so the compiler knows how much space to reserve)
  - \* Interface: behavioral specification
  - \* Class: mix of the above, with possible inheritance relationships
- \* Abstract data type (ADT): like an interface
- \* Data vs. pointer vs. reference
- \* Naming and assignment
- \* Function signatures: arguments and return types (and exceptions)

MEDIA ARTS & TECHNOLOGY PROGRAM

78

## Data Examples

- \* Name only (in ST80, style guide suggests that name indicate both type and role)
  - \* | theExtentPoint | "Smalltalk"
  - \* var freq; // SuperCollider
- \* Name and type
  - \* int i; /\* C \*/
- \* New data type
  - \* typedef float sample;
  - \* sample \* sampPtr;
- \* Assignment
  - \* i = 3; let(i, 3), i := 3, i <= 3

— MEDIA ARTS & TECHNOLOGY PROGRAM

79

## Functionality

- \* Functions, behaviors, rules, facts
- \* Procedure vs. function
  - \* (Old) return value, side effects
  - \* Pass-by-value vs. pass-by-reference
- \* Messages and methods
  - \* O-O languages: message-to-method mapping done by receiver object
  - \* Late binding, enables polymorphism
- \* Grouping of functions
  - \* None (flat), modules, classes, namespaces

— MEDIA ARTS & TECHNOLOGY PROGRAM

80

## Behavior Examples

- \* Function signature: name, return value, arguments, exceptions raised
  - \* boolean doSomething(in string whatToDo, inout string whatToDoItTo) raises (BadDirective);
- \* Function header: external reference
- \* Function body: implementation
- \* Parameter passing
  - \* Pass by reference (pointer, shares 1 copy; called fcn. can change caller's copy)
  - \* Pass by value (called fcn. gets its own copy)
  - \* Return values

— MEDIA ARTS & TECHNOLOGY PROGRAM

81

## Example: C++ Method

```
// Load the run-time database from a flat file of FC structs
FeatureDB & AnalysisEngine :: load_runtime_data(char * filename) {
 FC_IO fc_io; // declare a FC_IO
 FILE * input = fopen(filename, "r"); // open a file
 printf("Reading database from file %s\n", filename);
 while(! feof(input)) { // loop reading file
 FCS * fcs = (FCS *) malloc(sizeof(FCS)); // alloc FCS storage
 fc_io.read_FC(fcs, input); // read data in
 mDatabase.push_back(fcs); // store in list member
 } // end of loop
 fclose(input); // close file
 return mDatabase; // return member
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

82

## Example: LISP Expert Rule

```
;; Example dynamics rule -- make metal music tighter
(defrule match-target-dynamics ; IF clause
 (genre (name Metal)) ; the song is heavy metal
 (target (dyn-range ?tdr)) ; set variable ?tdr to the
 ; target's dynamic range
 (song (dyn-range ?sdr)) ; set variable ?sdr
 (test (> ?tdr ?sdr)) ; if tdr > sdr
 (test (> ?sdr 0.0)) ; and sdr > 0
=> ;; THEN clause ; add a bit to the "tightness"
 (modify ?par (tightness ; math is ugly in LISP
 (+ ?tig (- (** (/ ?tdr ?sdr) 0.2) 1.0))))
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

83

## Example: Smalltalk-80 Method

```
AbstractFunction methodsFor: 'geometry' "class, category"
indexNearestX: anX "name, argument"
"Answer the receiver's point nearest the given x value."
"(LinSeg from: #((0@0) (1@1))) indexNearestX: 0.7"

1 ind 1 "declare a name"
ind := 1. "assign into it"
2 to: data size do: "loop through my array"
 [:i | ((data at: ind) x - anX) abs "linear search"
 < ((data at: i) x - anX) abs "if we're in the interval"
 ifTrue: [i - 1] "return index"
 ifFalse: [ind := i]]. "else just count"
^data size "else return max"
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

84

## Processing: Interpreters

- \* Interactive programs
  - \* Simplest programming examples: read, evaluate, print loops
  - \* Calculators, shells, etc.
  - \* Example: shell and shell script
- \* Interpreted languages
  - \* What are these?
  - \* How to define new functions
  - \* Load/store data, functions, or “workspace”
  - \* Examples: LISP\*, Smalltalk\*, PostScript (shell?)

— MEDIA ARTS & TECHNOLOGY PROGRAM

85

## Compiler/Interpreter Alternatives

- \* One-stage vs. multi-stage
- \* Translate to native code (e.g., all C/C++ compilers) vs. to a “virtual machine” language (e.g., Java, Smalltalk, Python, C#)
- \* Kinds of virtual machines (optimizations)
- \* Easy to write slow interpreters
- \* Fast ones use rapid turn-around compilers and (polymorphic inline) code-caching virtual machines (who knows what this means?)
- \* Cross-compilers (host and target platform are different, e.g., target has no tools)

— MEDIA ARTS & TECHNOLOGY PROGRAM

86

## Software Engineering

- \* Since the 1970s: structured programming
  - \* Data and implementation hiding are good (behavioral interfaces and ADTypes)
  - \* Separate specification (.h file) from implementation (.c file)
  - \* Use abstract data types with polymorphic behaviors (reusable algorithms)
  - \* Object-oriented technology is evolutionary rather than revolutionary..

— MEDIA ARTS & TECHNOLOGY PROGRAM

87

## Development Cycle Alternatives

- \* 1960s edit/compile/run loop
  - \* As in C, C++, Java
  - \* Code stored in files, editor and compiler are separate
  - \* Program data loaded at start-up time (each time)
- \* 1970s: Rapid turn-around interpreter (or incremental compiler)
  - \* LISP, Forth, Prolog, Smalltalk
  - \* Code in source code database or “workspace”
  - \* Objects in working memory or persistent “core”

— MEDIA ARTS & TECHNOLOGY PROGRAM

88

## What’s the Structure of a (FORTRAN-family) Program?

- \* Header
  - \* Includes/imports
- \* Declarations
  - \* Data
  - \* Function prototypes
- \* Functions/procedures
  - \* “Main” function (should be small)
- \* Examples: Hello world, CSL components

— MEDIA ARTS & TECHNOLOGY PROGRAM

89

## What’s an API

- \* Application programmer’s interface, i.e., what you use to get the job done
- \* In C, etc.
  - \* Pre-processor macros, header files, and libraries
  - \* Data structures and functions
  - \* Call-back fcn. signatures, exceptions
- \* In C++, etc.
  - \* Abstract and concrete classes

— MEDIA ARTS & TECHNOLOGY PROGRAM

90

## How to Read an API

- \* Look at:
  - \* Macros and constants (#defines)
  - \* Data structures (or class hierarchy)
  - \* Functions/methods and their signatures
  - \* Exceptions, error codes, data ranges, etc.
  - \* How to compile [.h file] and link [.a file]
  - \* (Private data and code?)
- \* Doc and examples
- \* Protocols and p-APIs
- \* IDL descriptions (where available)
- \* Principle of least astonishment!

MEDIA ARTS & TECHNOLOGY PROGRAM

91

## Example: UNIX/C String API

- \* Say man string to get the man page
- \* These have a standardized format
  - \* STRING(3) BSD Library Functions Manual
  - \* NAME -- strcpy, strcat, strchr, strcmp, strncmp, strcasecmp [...] - string specific functions
  - \* LIBRARY -- Standard C Library (libc) [object file]
  - \* SYNOPSIS -- #include <string.h> [header file]
  - \* ...function descriptions

MEDIA ARTS & TECHNOLOGY PROGRAM

92

## String Function Signatures

```
// copy src to dst; answer dst
char * strcpy(char *dst, const char *src);

// concatenate count chars of
// append onto s; answer s
char * strncat(char *s, const char *append,
 size_t count);

// answer ptr to first instance of c in s
char * strchr(const char *s, int c);
```

MEDIA ARTS & TECHNOLOGY PROGRAM

93

## (ANSI STL) C++ String API

- \* using namespace std;
- \* #include <string>
- \* Construction, conversion (to char\*)
- \* Length, size
- \* Array/vector operations
- \* Character accessing
- \* Searching
- \* String operators

MEDIA ARTS & TECHNOLOGY PROGRAM

94

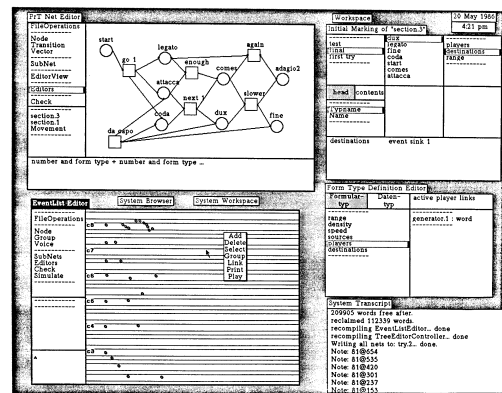
## Other API Examples

- \* UNIX math.h, socket.h, etc.
- \* BOOST libraries
- \* C++ STL
- \* PortAudio API header file
- \* FFTW API
- \* DirectX class libraries
- \* QuickTime API
- \* iTunes visual plug-in API
- \* GUI APIs (see below)
- \* Others...

MEDIA ARTS & TECHNOLOGY PROGRAM

95

## Topic: IDEs



MEDIA ARTS & TECHNOLOGY PROGRAM

96

## Readings 1B

- \* SW Development Tools & IDEs
  - \* Requirements
  - \* Compilers
  - \* IDEs
- \* APIs and libraries
- \* MM programming techniques

— MEDIA ARTS & TECHNOLOGY PROGRAM

97

## What tools do you need to develop software?

- \* Required
  - \* Code text editor
  - \* Compiler, interpreter, translator, or assembler
  - \* Linker, loader, run-time support
- \* Optional
  - \* Debugger, profiler (very useful)
  - \* Source code mgr, configuration/version system
  - \* GUI development tools
- \* These might all be packaged together in an integrated development environment (IDE)

— MEDIA ARTS & TECHNOLOGY PROGRAM

98

## IDE families

- \* TextEd + shell/make/cc + adb (1970s)
  - \* Linux, CygWin, Mac OS X
- \* Project + “integrated” (turbo) tools (1980s)
  - \* VisualStudio, XCode, KDevelop, FORTE, VisualCafe, Kawa, Eclipse, DevC++
- \* Incremental, rapid-turn-around IDE
  - \* Smalltalk, VisualAge/Java, CommonLISP, SuperCollider
  - \* Method-by-method compilation, late binding linker

— MEDIA ARTS & TECHNOLOGY PROGRAM

99

## How did Grandpa do it? Make and Makefiles

```
Declare an object file list (this is a comment)
OBJECT_FILES = CSL_Core.o Gestalt.o Variable.o
Declare some compiler flags
CCFLAGS = -O2 -I../CSL/Kernel -I../CSL/Sources
Add a rule or dependency (target, source, command)
CSL_Core.o: ../Kernel/CSL_Core.cpp
 $(CC) -c $(CCFLAGS) ../Kernel/CSL_Core.cpp
Add a “target” or executable
beep: $(OBJECT_FILES) Beep.o
 $(CC) $(OBJECT_FILES) Beep.o $(LD_FLAGS) -o beep
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

100

## Executing Make in a Shell

```
#make (type in shell command)
gcc -O2 -I../Kernel ../Kernel/CSL_Core.cpp
...
gcc -O2 -I../Kernel ../Tests/Beep_demo.cpp
gcc CSL_Core.o CGestalt.o [...] Beep_demo.o
 -L/usr/local/lib -lm -lportaudio
 -L/usr/lib/gcc/darwin/3.3 -lstdc++
 -o beep_demo
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

101

## Debugging

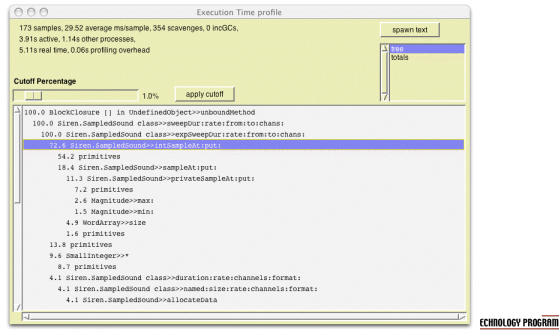
- \* Running an executable within a debugger allows you to:
  - \* control execution (breakpoints, single-stepping)
  - \* Examine system state just before/after an error (stack, variables)
- \* Examples: gdb, ddd, etc.
- \* Debuggers in IDEs
  - \* Issues, differences...

— MEDIA ARTS & TECHNOLOGY PROGRAM

102

## Profiling

- \* Run the program and “observe” where it spends its time; print out some meaningful representation of this



103

## Gprof Profiler Output

% cumulative	self	self	total	
time	seconds	seconds	calls	ms/call name
0.0	0.00	0.00	714	0.00 0.00 past end of text [2]
0.0	0.00	0.00	688	0.00 0.00 ___ZN3cs110Oscillator11next_bufferERNS_6BufferES2_<cycle 1> [33827]
0.0	0.00	0.00	688	0.00 0.00 ___ZN3cs111FrameStream11next_bufferERNS_6BufferES2_ [33828]
0.0	0.00	0.00	688	0.00 0.00 ___ZN3cs112SampleStream11next_bufferERNS_6BufferES2_ [33829]
0.0	0.00	0.00	688	0.00 0.00 ___ZN3cs113UnitGenerator13pull_controlsERNS_6BufferES2_ [33830]
0.0	0.00	0.00	688	0.00 0.00 ___ZN3cs14Sine16mono_next_bufferERNS_6BufferES2_ [33831]
0.0	0.00	0.00	688	0.00 0.00 ___ZN3cs16Phased14pull_frequencyERNS_6BufferES2_<cycle 1> [33832]
0.0	0.00	0.00	688	0.00 0.00 ___ZN3cs18Envelope16mono_next_bufferERNS_6BufferES2_ [33833]
0.0	0.00	0.00	514	0.00 0.00 ___ZN3cs111LineSegment16mono_next_bufferERNS_6BufferES2_ [33834]
0.0	0.00	0.00	356	0.00 0.00 ___ZN3cs16Buffer8set_sizeEj [33835]
0.0	0.00	0.00	350	0.00 0.00 ___ZN3cs16BufferD1Ev [33836]
0.0	0.00	0.00	350	0.00 0.00 ___ZN3cs16BufferD4Ev [33837]
0.0	0.00	0.00	344	0.00 0.00 ___ZN3cs12IO21print_time_statisticsEP7timevalS2_PIS3_S3_ [33838]
0.0	0.00	0.00	344	0.00 0.00 ___ZN3cs16BufferC1Eij [33839]
0.0	0.00	0.00	344	0.00 0.00 ___ZN3cs16BufferC4Eij [33840]

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

104

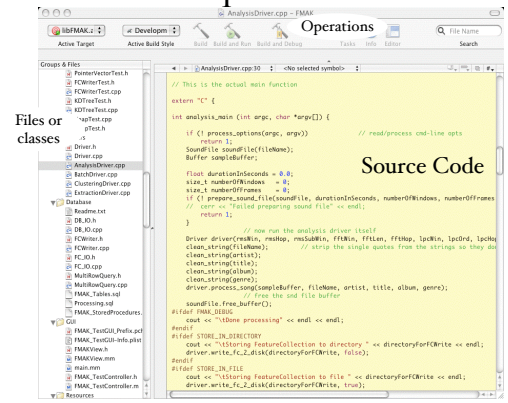
## Integrated Development Environments (IDEs)

- \* Project/file management
- \* File trees, libraries and frameworks
- \* Source code editing
- \* Syntax-aware code editor
- \* File/class browser and cross-referencing
- \* Compilation/linking
- \* Rapid-turn-around compiler (?)
- \* Execution/debugging
- \* Execution environment and I/O
- \* Breakpoints and single-stepping
- \* Packaging/deployment
- \* Source code management system interface
- \* Development vs. deployment compiler options

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

105

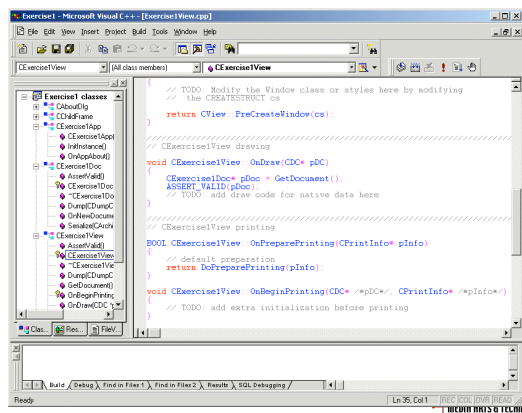
## IDE Example: Xcode Editor



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

106

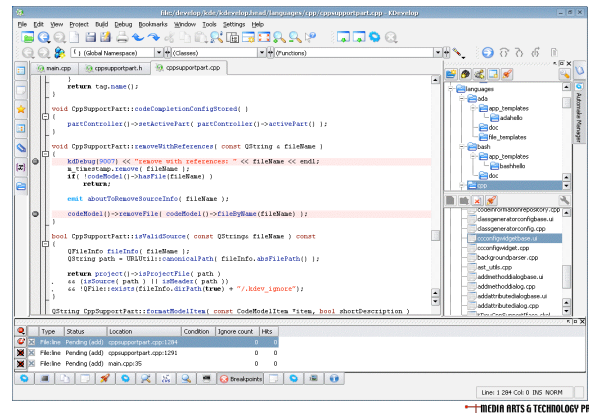
## IDE Example: MS Visual C++



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

107

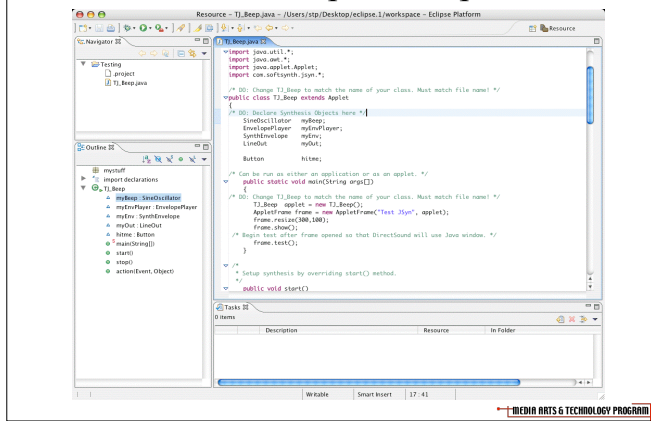
## IDE Example: KDevelop



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

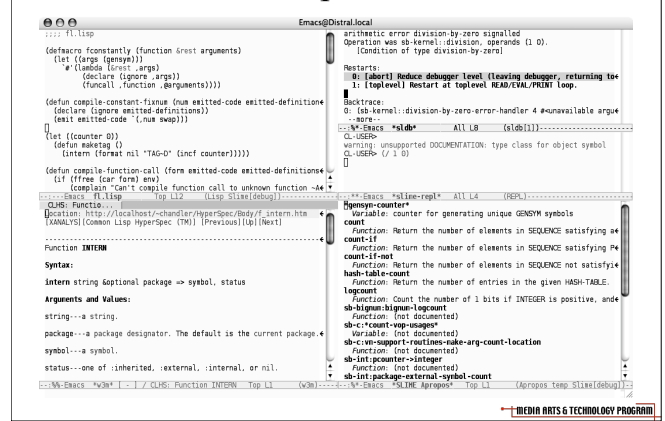
108

## IDE Example: Eclipse



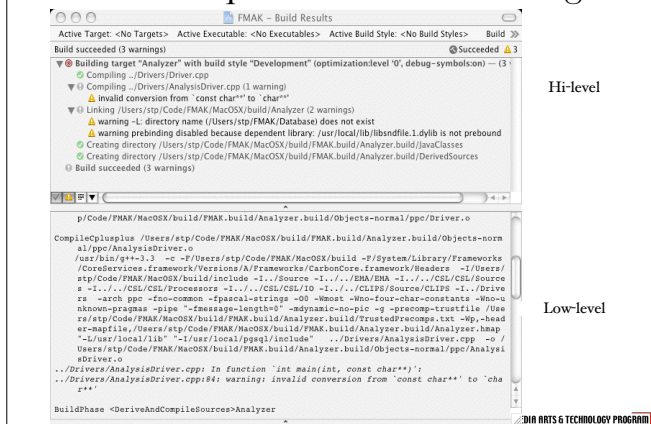
109

## IDE Example: XEmacs



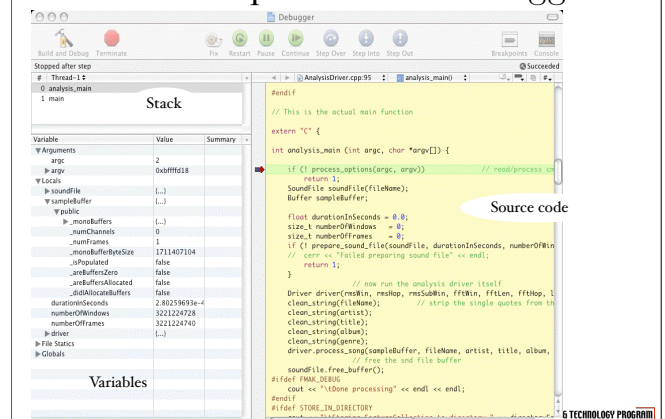
110

## IDE Example: Xcode Build Stages



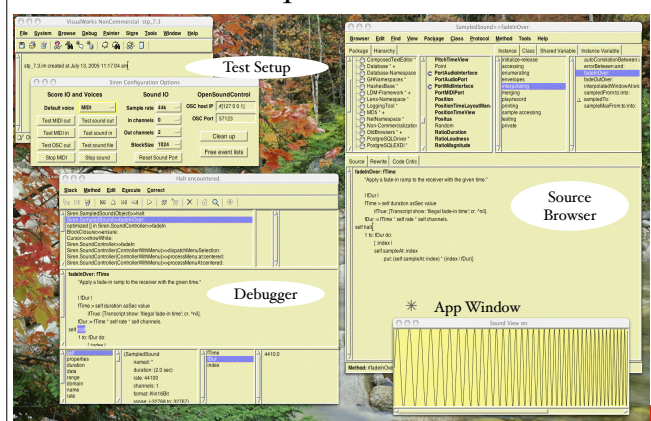
111

## IDE Example: Xcode Debugger



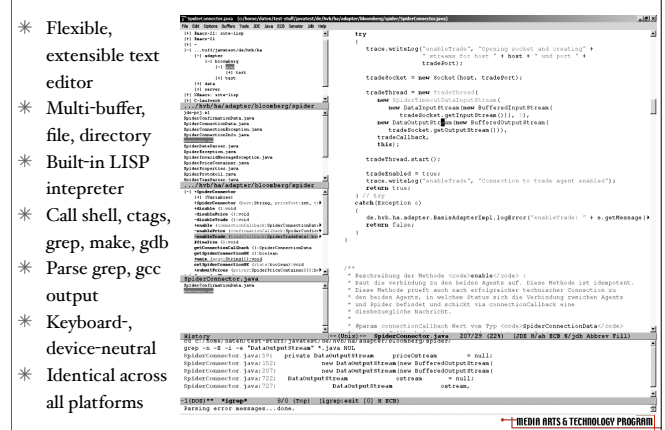
112

## IDE Example: Smalltalk-80



113

## In Praise of Emacs



114

## Installing a Linux Package

- \* Down-load a “tar-ball” (gzip-compressed tar archive); unpack it
  - \* `tar -xvfz tarball.tgz`
- \* Run configuration shell script
  - \* `./configure`
- \* Build target
  - \* `make`
- \* Install target
  - \* `sudo make install`
- \* Building distributions: `conf/make/package`

— MEDIA ARTS & TECHNOLOGY PROGRAM

115

## Learning a new IDE

- \* Target language (try not to change everything at once)
- \* Host/target processor/OS
- \* Compiler/linker options
  - \* Optimization, debugging, dynamic libraries
- \* Debugger operation: stack and variables
- \* Tool GUI style
  - \* Editor, project/compiler, debugger
- \* Package discipline
  - \* Project vs. makefile
  - \* New project setup (often the hardest part)

— MEDIA ARTS & TECHNOLOGY PROGRAM

116

## Project/Makefile Examples

- \* CSL Xcode project and makefile
- \* FFTW compilation
- \* Building the PortAudio project
- \* Wonder configure script, Qt project and makefile
- \* Other APIs
  - \* BOOST, JUCE, liblo, CSL

— MEDIA ARTS & TECHNOLOGY PROGRAM

117

## Other SW Tools, CASE

- \* Source code, version control
- \* Configuration management
- \* Test suites
- \* RDBMS APIs
- \* SWIG and IDLs
- \* XML I/O
- \* HTML page generation
- \* DPE, CORBA, server farms

— MEDIA ARTS & TECHNOLOGY PROGRAM

118

## Tool Availability

- \* Macintosh OS X: CREATE & MAT
  - \* Xcode/InterfaceBuilder, emacs/gcc/make
- \* MS-Windows: MAT class room
  - \* VisualStudio C++, DevC++, CygWin gcc, gdb
- \* Linux: MAT class room
  - \* Xemacs, gcc, gdb
  - \* Kdevelop
  - \* Others?
- \* All platforms
  - \* Eclipse
  - \* Smalltalk, LISP, others
- \* Your machines: which platform/tools?

— MEDIA ARTS & TECHNOLOGY PROGRAM

119

## Source Code Management

- \* Team programming (programming in the large) and source code management tools
  - \* DB issues: shared access, persistency
  - \* Easy if there's only 1 writer (rare case)
- \* Configuration management (CM)
- \* Releases and version handlers
- \* How it's done
  - \* CVS demo: check-out, alter, check-in
  - \* Fancier tools: Subversion
  - \* Even fancier tools: Store, Envy

— MEDIA ARTS & TECHNOLOGY PROGRAM

120

## SCCS: CVS

- \* Process: check out, modify, check in
  - \* # cvs checkout project
  - \* # cd project;
  - \* # vi source\_file.c
  - \* # make; test
  - \* # cvs update
  - \* # cvs commit
- \* Integration with IDEs
  - \* File-based IDEs
  - \* Transparent, shared source IDEs
  - \* File DB vs. Code DB

MEDIA ARTS & TECHNOLOGY PROGRAM

121

## Putting it all Together

- \* LSF (list sound files) example
- \* Header file
- \* Source file
- \* Makefile
- \* Compile/link process
- \* Execution/debugging

MEDIA ARTS & TECHNOLOGY PROGRAM

122

## What's Next?

- \* Coding exercises
  - \* Use a good C/C++/Java tutorial
- \* IDE comparisons (next week)
- \* Readings
  - \* OS/IDE background
  - \* Programming language comparisons
  - \* Adrian Freed programming pearls

MEDIA ARTS & TECHNOLOGY PROGRAM

123

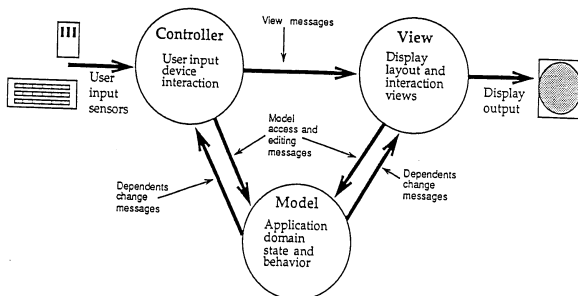
## To Do

- \* Brush up on programming
  - \* Look at C/C++/Java if unfamiliar
  - \* Try some scripting languages and interpreters (shell scripts, perl, ruby, python, etc.)
- \* Reading: languages, IDEs
- \* Install IDE(s) on your computer
- \* Make “hello world” work
  - \* In C, C++, and Java (at the least)
  - \* On Windows, Linux, and MacOS (with various IDEs and SW development tools)
- \* Use CVS
  - \* Public repositories
  - \* Test project

MEDIA ARTS & TECHNOLOGY PROGRAM

124

## MAT 201B Topic 2: Application Organization



MEDIA ARTS & TECHNOLOGY PROGRAM

125

## Application Organization

- \* Application architecture: models and patterns
- \* Interactive and real-time software
- \* Multi-threaded applications
- \* Databases and media content
- \* Special programming techniques
- \* Exercises

MEDIA ARTS & TECHNOLOGY PROGRAM

126

## Readings 2

- \* Software Architecture
  - \* The UNIX OS Model
- \* Multi-threaded Programming
  - \* POSIX Threads
- \* Databases
  - \* SQL Intro

MEDIA ARTS & TECHNOLOGY PROGRAM

127

## What will we be learning?

- \* How is a large program organized?
- \* How are processing and I/O controlled?
- \* What are the “legacy” APIs for I/O?
- \* What are design patterns for apps. and I/O?
- \* How are multi-threaded and real-time applications programmed?
- \* How can persistent/shared data be managed?
- \* What special programming techniques are used for media SW?

MEDIA ARTS & TECHNOLOGY PROGRAM

128

## Application Structures

- \* Application organization, control flow
  - \* Old-world: input/processing/output (AKA batch processing)
  - \* Interactive languages and command-line interpreters (read/eval/print loops)
  - \* WIMP model (windows, interaction, mouse-pointer -- event handling loops)
  - \* Other models

MEDIA ARTS & TECHNOLOGY PROGRAM

129

## Batch Processing (1950s-present)

- \* Non-interactive systems, still in common use (just look at your phone bill)
- \* Input/processing/output architecture (or load, modify, store in DB terms)
- \* Input media and mechanisms: punched card decks, lab sensors, database queries
- \* Kinds of processing (diverse)
- \* Output media: report generation, graph plots, database records inserted
- \* OS Scheduler schedules jobs to run (coarse-grained, context-switching rare); every process has the whole processor to itself; OS interface is a job control language (JCL)

MEDIA ARTS & TECHNOLOGY PROGRAM

130

## Time-sharing (1960s-present)

- \* IBM TSO (OS360 time-sharing option “tee-slow”)
- \* Every process thinks it has the whole processor (albeit a slower one)
- \* OS manages per-process resources
- \* Some tasks take input while running (strange, eh?)
- \* Context-switching is frequent (on the msec scale)
- \* Some processes have their own interactive read/eval/print loops
- \* Interactive command-line interpreter (shell) or real language interpreter (in addition to JCL)

MEDIA ARTS & TECHNOLOGY PROGRAM

131

## Command-line Interpreters

- \* UNIX shell
  - \* # wc PAIO.h -- command
  - \* 43 156 1111 PAIO.h -- output
- \* LISP listener
  - \* ; (DEFUN Reciprocal (n) (/ 1 n)) ; define function
  - \* ; Reciprocal ; response
  - \* ; (Reciprocal 5) ; call function
  - \* ; .2 ; result
  - \* ; (Reciprocal 2384)
  - \* ; .000419463

MEDIA ARTS & TECHNOLOGY PROGRAM

132

## The UNIX I/O Model

- \* IO devices look like files (character, block flavors) with functions for open(), close(), read(), write() (opt. Async IO), and ioctl()
- \* ioctl() takes command flags and variable arguments for device-specific control
- \* Error codes returned as errno; use perror() to print (no out-of-scope exceptions)
- \* Utility programs manipulate device “files” and read/write pointer data

— MEDIA ARTS & TECHNOLOGY PROGRAM

133

## UNIX I/O Model and API

- \* man open leads to
  - \* OPEN(2) BSD System Calls Manual
  - \* NAME -- open - open or create a file for reading or writing
  - \* SYNOPSIS -- #include <fcntl.h>
  - \* int open(const char \*path, int flags, mode\_t mode);
- \* UNIX special device files (major/minor device number) e.g., /dev/disk0s1 = 14/1 (try it!)
- \* Kernel has device tables with fcn ptrs to open/close/read/write/ioctl
- \* To see ioctls, see device man, e.g., man pty

— MEDIA ARTS & TECHNOLOGY PROGRAM

134

## Other I/O Models

- \* Macintosh toolbox or BeOS “managers”
- \* Object-oriented models
  - \* Smalltalk stream/accessor model
  - \* Design: buffer, content, stream, device
  - \* Java, Siren, STK, clm, Jsynth, MusicKit, ...
  - \* BeOS I/O objects
- \* Plug-in architectures
  - \* Browser plug-ins
  - \* Streaming app. plug-ins

— MEDIA ARTS & TECHNOLOGY PROGRAM

135

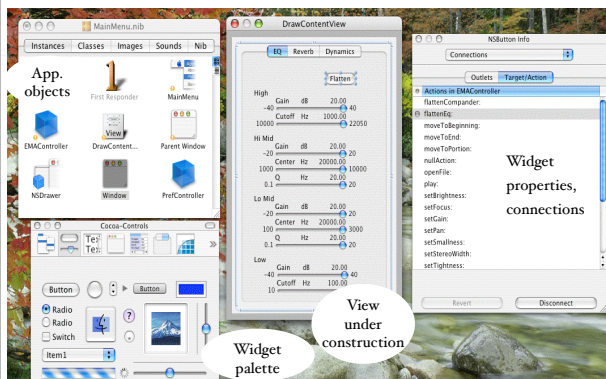
## GUIs: Event Handlers

- \* How do GUI-based programs work?
  - \* Applications register event handlers (EH, e.g., on-mouse-down-call-this-fcn, etc.)
  - \* Window manager process delegates user input events to application (window) processes
  - \* Each EH processes a certain kind of events (e.g., close window, menu/button, keyboard input)
  - \* The main program loop now just reads and delegates user input events
  - \* Modern tools allow you to add GUI widgets to a screen and assign message sends to them

— MEDIA ARTS & TECHNOLOGY PROGRAM

136

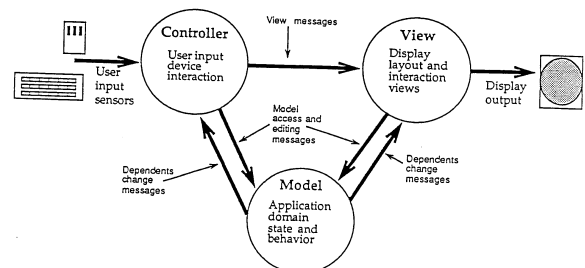
## IDE Tool: GUI Screen Painter



— MEDIA ARTS & TECHNOLOGY PROGRAM

137

## The Model/View/Controller (MVC) Design Pattern and Framework

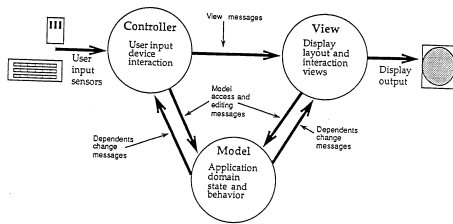


- \* Early 1980s Smalltalk apps., LISP flavors system
- \* MVC factoring for reuse

— MEDIA ARTS & TECHNOLOGY PROGRAM

138

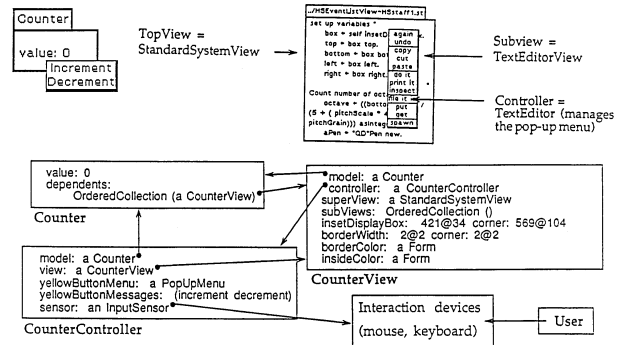
## MVC Message Flow



- \* User interacts with mouse or keyboard
- \* Input device's driver generates "event"
- \* OS and window system delegate event to an application process via a callback function
- \* (Your code) callback function changes model object's state or viewing mode, generates change broadcast message
- \* One or more view objects get update messages and redisplay

— MEDIA ARTS & TECHNOLOGY PROGRAM  
139

## Example: 1980s MVC Objects



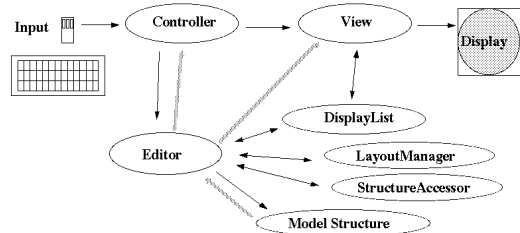
— MEDIA ARTS & TECHNOLOGY PROGRAM  
140

## Support for MVC

- \* All modern GUI frameworks have OO class libraries for MVC objects
- \* Models: classes for data structures, DB interfaces, I/O APIs, thread libraries, etc.
- \* Views: display objects, text rendering, etc.
- \* Controllers: event mapping, filtering
- \* Composite objects: GUI widgets, apps.
- \* See: Cocoa, DirectX, Xlib, QT, etc.

— MEDIA ARTS & TECHNOLOGY PROGRAM  
141

## Advanced MVC: Navigator (1980s)

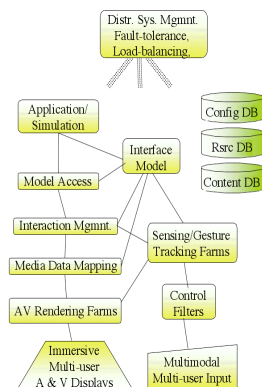


- \* More reusable parameterized classes
- \* Additional GUI design patterns

— MEDIA ARTS & TECHNOLOGY PROGRAM  
142

## Advanced MVC: DSCP

- \* Distributed Sensing, Computation, and Projection (**MVC** on steroids)
- \* Back-end application models are scientific/numerical/simulation
- \* Multimodal multiuser sensing/control and tracking/mapping farms
- \* Application = sensing/tracking policies + output data mappings
- \* Presentation/interaction via CNSI Sphere, LAN/WAN streaming
- \* Infrastructure uses CRAM mgmt
- \* DBs for configurations, resources, and media content (renderers)



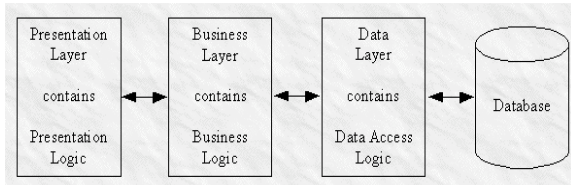
— MEDIA ARTS & TECHNOLOGY PROGRAM  
143

## Application Architecture

- \* 3-Parts/tiers
- \* Model/view/controller
- \* GUI, business rules, database
- \* Presentation may consist of GUI widgets or HTML page generation
- \* Control may be via widgets or in-coming page request delegation
- \* "Business" objects may be a simple DB mapping (e.g., catalog) or complex stateful application (e.g., auction)
- \* Data layer may be simple or complex

— MEDIA ARTS & TECHNOLOGY PROGRAM  
144

## Multi-tier Architecture



- \* Links may be function calls or remote method invocations (RMI)
- \* Any layer may be distributed or replicated
- \* Layers can be further refined (i.e., there may be > 3 tiers)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

145

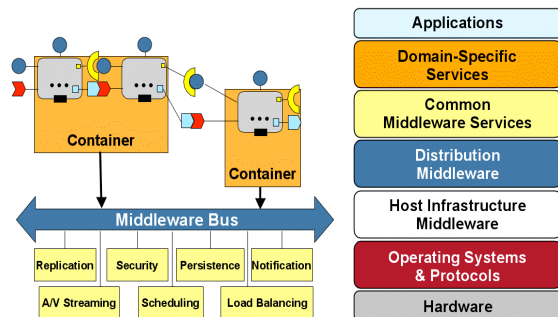
## Application Servers

- \* Front-end
  - \* Web request handling, distributed application, on-line transaction processing and reporting, content server, presentation layer (e.g., HTML page generation)
- \* Middleware
  - \* Transaction processing monitors, message queues, session management, rules, servlet engine
- \* DB Layer
  - \* Handled by persistent containers (beans), smart caches, object-relational mappers
- \* Any component may run on multiple hosts (for scalable performance, fault-tolerance)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

146

## Application Middleware (Patterns)

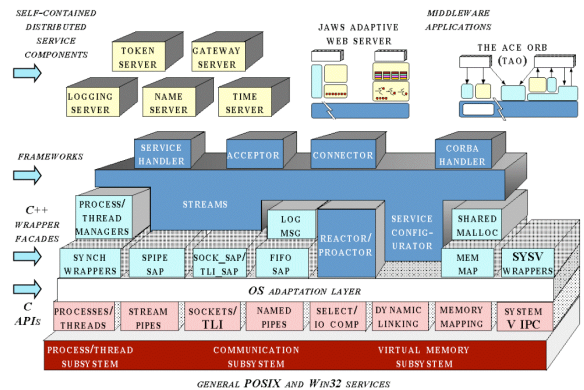


- \* Pattern- or service-oriented architecture (D. Schmidt)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

147

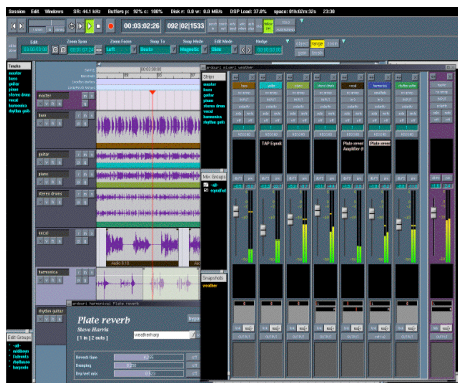
## ACE Architecture



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

148

## Topic: Prog. Techniques



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

149

## Programming: Call-backs

- \* Problem: you need to customize the behavior of some thread in a language that lacks blocks as objects (see prog. lang. features above)
- \* Solution: register a function (by pointer) to be called from some driver, scheduler, or event loop thread
- \* Examples
  - \* GUI event handlers: how to handle common events
  - \* PortAudio sound I/O API: how to create your own sound output routine

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

150

## Example: Call-back function

- \* API specifies signature (not name) of call-back
- \* Write your own fcn with that signature
- \* Register it for activation when appropriate (e.g., for the close box, do this, when you need a buffer of sound, do this)
- \* Example: PortAudio sound call-back signature
- \* `int PA_Callback(void *inputBuffer, void *outputBuffer, unsigned long framesPerBuffer, void *userData) {`
- \* `/* do something with the I/O buffers here */`
- \* `}`

— MEDIA ARTS & TECHNOLOGY PROGRAM

151

## Example: Register the Call-back Function

- \* PA API function to open an I/O stream (called from main function)

```
* err = Pa_OpenDefaultStream(
* &stream, /* writes into back stream pointer */
* 0, /* no input channels */
* 2, /* stereo output */
* paFloat32, /* 32 bit floating point output */
* 44100, /* sample rate */
* 256, /* frames per buffer */
* myPACallback, /* callback fcn name (fcn ptr) */
* &data); /* pass some data struct to callback */
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

152

## Modern main()

- \* Allocate any shared data
- \* Register call-back functions
- \* Fork background threads
- \* Start main event-processing loop
- \* Wait until killed (by a call-back function in response to the quit message)
- \* Example main() functions: Qt, CSL, etc.

— MEDIA ARTS & TECHNOLOGY PROGRAM

153

## Real-time Systems, Schedulers

- \* What if your program has some response-time constraints or scheduled activity?
  - \* Multiple threads, one a scheduler
  - \* How to manage thread priority?
- \* Schedulers
  - \* Time-sorted list of actions
  - \* How to insert into this efficiently?
  - \* Main loop: sleep to next event, then do it

— MEDIA ARTS & TECHNOLOGY PROGRAM

154

## Scheduler Example

- \* Simple animation playback program
- \* Main() opens input file, reads events into a list, closes file
- \* Loop until done:
  - \* Timer waits for next event
  - \* Process\_event() executes and resets timer

— MEDIA ARTS & TECHNOLOGY PROGRAM

155

## Real-time Prioritization

- \* The key is to be able to guess how long a given task will take and to be able to prioritize tasks
- \* Both of these can be difficult in general-purpose OS software due to virtual memory, caching, I/O buffering, networks, etc.
- \* Many MM applications require “easy real-time” performance (a few msec of latency jitter acceptable)

— MEDIA ARTS & TECHNOLOGY PROGRAM

156

## Multi-threaded Applications

- \* Writing them: a challenge
- \* Debugging them: a nightmare
- \* Processes vs. threads
  - \* Process: a thread of control and a protected address space
  - \* Threads are “lightweight process,” several in the same address space (shared memory)
  - \* Thread-change is assumed to be simpler/faster than process context switch
- \* The system is only executing one thread (in one process) at a time (per processor)

MEDIA ARTS & TECHNOLOGY PROGRAM

157

## The POSIX Thread Library

- \* Write a function with a specific control behavior (e.g., an endless loop with a delay or callback registry)
- \* Create a thread with the call
  - \* `int pthread_create(pthread_t * thread_ptr,`
  - \* `pthread_attr_t * attributes,`
  - \* `void *(*start_routine)(void *), void * args)`
- \* It shares all memory with the thread that created it (so watch out!)
- \* You can later check, kill, test it

MEDIA ARTS & TECHNOLOGY PROGRAM

158

## Support for Multi-threaded SW

- \* Thread synchronization
  - \* Shared data
  - \* Mutex, semaphores, condition variables
- \* Heavy-weight processes
  - \* Inter-process communication
    - \* Shared memory
    - \* Semaphores
    - \* Sockets and protocols (see below)
    - \* Remote procedure calls

MEDIA ARTS & TECHNOLOGY PROGRAM

159

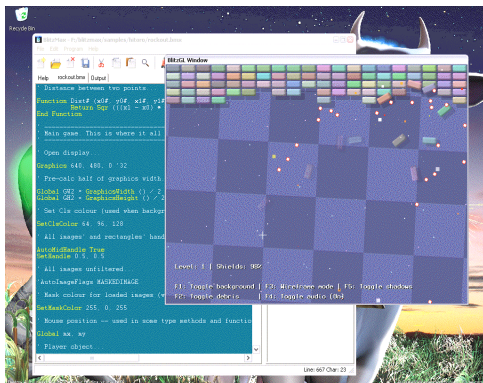
## Examples: Multi-threaded SW

- \* Producer/consumer applications
  - \* Writer/reader threads share data
  - \* Synchronization via shared data, mutex signals, semaphores, flags, etc.
- \* Interactive systems (e.g., CSL)
- \* Window systems
- \* In-class example

MEDIA ARTS & TECHNOLOGY PROGRAM

160

## Topic: Databases



MEDIA ARTS & TECHNOLOGY PROGRAM

161

## Databases

- \* How do we:
  - \* Handle persistent, shared, high-volume data?
  - \* Manage multiple concurrent access?
  - \* Provide data structure definition and query language?
- \* Database systems have tools for indexing, optimizing, replicating, backing up, etc.

MEDIA ARTS & TECHNOLOGY PROGRAM

162

## Database Technology

- \* Shared, controlled, persistent data
- \* Many options for file- and structure-based storage
  - \* ISAM: indexed sequential access method
  - \* Networked, relational, and object databases
- \* Indexing and caches (keys to good performance)
- \* Easy to do for simple cases
- \* Hard to make scale well (volume or throughput)
- \* Expensive to make robust/secure

MEDIA ARTS & TECHNOLOGY PROGRAM

163

## Example: Relational Databases

- \* Tabular-format data and the structured query language (SQL): RDB mgmnt. systems
- \* Create a table
  - \* CREATE TABLE Events (Time real, Duration real, Pitch real);
- \* Insert into a table
  - \* INSERT INTO Events (Time, Duration) VALUES (0.0, 1.0);
- \* Select from a table
  - \* SELECT (Time) FROM Events WHERE (Duration > 1.0);

MEDIA ARTS & TECHNOLOGY PROGRAM

164

## SQL Table (from CRAM)

```
CREATE TABLE NetworkNode (
 id serial, status TEXT, -- Table of LAN nodes
 name TEXT, fullname TEXT, -- A node's basic properties
 l_port INT, e_port INT, -- network listener ports
 -- Performance
 MIPS SMALLINT, MFLOPS SMALLINT,
 latency SMALLINT, jitter SMALLINT,
 -- Hardware
 sound TEXT CHECK (sound in ('None', 'Stereo', 'Multi_chan', 'Other')),
 graphics TEXT CHECK (graphics in ('None', 'Screen', 'Camera', 'HMD', 'Other')),
 haptic TEXT CHECK (haptic in ('None', 'Glove', 'Tracker', 'Mouse', 'MIDI', 'Other')),
 network TEXT CHECK (network in ('E10', 'E100', 'E1000', 'Other')),
 processor TEXT CHECK (processor in ('PowerPC', 'SPARC', 'MIPS', 'X86', 'Other')),
 -- Software
 OS TEXT CHECK (OS in ('MacOSX', 'Solaris', 'IRIX', 'Linux', 'MS-Windows', 'Other'))
);
What queries are possible?
```

MEDIA ARTS & TECHNOLOGY PROGRAM

165

## More Advanced DB Systems

- \* Problems with Relational (RDBMS)
  - \* Lots of design issues
  - \* Problems with unformatted data (BLOBs)
  - \* Problems with large single values (BLOBs)
  - \* Queries based on non-Boolean conditions?
  - \* Networks/webs of relationships (joins)?
- \* Alternatives
  - \* Object-oriented databases
  - \* Network databases
  - \* Others

MEDIA ARTS & TECHNOLOGY PROGRAM

166

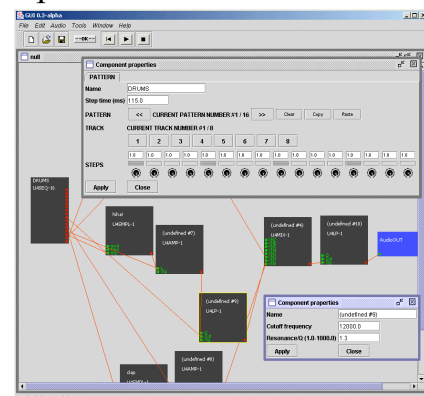
## Multimedia Content

- \* Media databases are difficult due to:
  - \* Large volume (sound/images/video)
  - \* Complex feature vectors (query on many properties)
  - \* Compute-intensive feature extraction (signal analysis)
  - \* Hierarchical (possibly remote) storage
  - \* Desire for streaming (high-throughput) access
- \* These will be components of many future applications in all domains!

MEDIA ARTS & TECHNOLOGY PROGRAM

167

## Topic: MM Data Processing



MEDIA ARTS & TECHNOLOGY PROGRAM

168

## Programming Techniques for Multimedia Applications

- \* Efficient media data processing
  - \* Coding for processing efficiency
  - \* Coding for I/O performance
  - \* Coding for portability
- \* Specifics
  - \* Numerics and code loops
  - \* Managing data volume and caching
  - \* Multiple data formats

MEDIA ARTS & TECHNOLOGY PROGRAM

169

## DSP and Media Processing

- \* General programming issues
  - \* Data structures and I/O
- \* Vector and block numerical processing
- \* Double-buffered I/O
  - \* Call-backs, async/blocking I/O
- \* Event- and I/O-driven programming
  - \* Control, GUIs
- \* Streams and DSP pipes

MEDIA ARTS & TECHNOLOGY PROGRAM

170

## Processor Issues

- \* Integer vs. floating-point performance and formats (endianness)
- \* Processor architecture
  - \* CISC vs. RISC vs. DSP issues
- \* Special-purpose processors
  - \* Synthesis/processing techniques
  - \* Rendering pipelines (see below)
  - \* HW support for FFT/IFFT
  - \* MM extensions to traditional instruction sets (MAT 200C)

MEDIA ARTS & TECHNOLOGY PROGRAM

171

## C/C++ Language Issues

- \* Data types: 8-64-bit ints, 32/64-bit floats
- \* Data structures: sample buffer [short \* samps], header, ...
- \* Operations: math, vector ops, storage
- \* Libraries and APIs
- \* Versions of C/C++/ObjC/Java/C++
- \* Platform portability

MEDIA ARTS & TECHNOLOGY PROGRAM

172

## Adrian Freed's Advice

- \* Use floats if feasible for sample math
- \* Use longs or doubles for time values
- \* Need > 16 bits for frequency values
- \* Use accuracy-preserving math
- \* Watch out for processors with slow integer performance (i.e., using f-p ALU)
- \* Watch out for processors with slow f-p performance (e.g., Pentium)
- \* Know your compiler and its options!

MEDIA ARTS & TECHNOLOGY PROGRAM

173

## Adrian Freed's Advice (2)

- \* Test the built-in library functions
- \* Use explicit casts where necessary (avoid if possible)
- \* Use parentheses for clarity of precedence
- \* Watch malloc() (performance problems with memory management)
- \* Use appropriate language features
  - \* Exceptions
  - \* Threads
- \* Know your compiler optimizations!
  - \* Register variables
  - \* for() {} vs. while() {}

MEDIA ARTS & TECHNOLOGY PROGRAM

174

## OSS Programming Guidelines

- \* Use API macros
- \* Check return error codes
- \* Don't assume initial conditions, set device parameters
- \* Be careful in interrupt call-backs
- \* Cast pointers and ioctl() arguments

— MEDIA ARTS & TECHNOLOGY PROGRAM

175

## Programming for Portability

- \* Several options
  - \* #ifdefs for all platforms
  - \* Macros for platform-specific operations
  - \* Use portability APIs (e.g., CygWin)
  - \* Obviously choose least-common denominator functionality and formats (sad)
- \* Examples
  - \* PortAudio, PortMIDI, LibSndFile
  - \* OpenGL, OpenSceneGraph
  - \* FFTW, LibRan, LibTSP

— MEDIA ARTS & TECHNOLOGY PROGRAM

176

## Performance Testing

- \* Benchmark Design
  - \* Be certain you're testing what you think you're testing
  - \* Gather enough data to extrapolate from
- \* Examples
  - \* int vs. float math in loops
  - \* Optimization options
  - \* Cache performance vs. buffer size
  - \* Accuracy vs. speed trade-offs
    - \* e.g., sin(x) vs. table look-up

— MEDIA ARTS & TECHNOLOGY PROGRAM

177

## Review

- \* Application models: I/O and MVC
- \* Call-backs and call-back APIs
- \* Schedulers and real-time interactivity
- \* Multi-threaded programming
- \* Database technology
- \* Programming techniques for multimedia

— MEDIA ARTS & TECHNOLOGY PROGRAM

178

## Exercises

- \* Write a read/eval/print loop program
- \* Use an IDE tool to build a GUI for it
- \* Write a trivial call-back function to play a sound via PortAudio
- \* Write a simple scheduler that can read a script/score file and play sounds or display images
- \* Write SQL for a simple table and some records related to your domain of interest
- \* Write fair cross-platform benchmarks of processor and memory performance

— MEDIA ARTS & TECHNOLOGY PROGRAM

179

## To Do

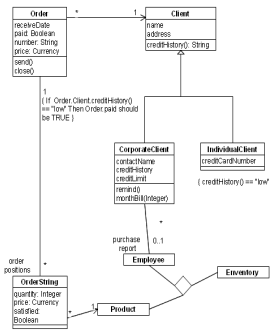
- \* Develop your programming skills (remember “programmer as reader”)
- \* Start actually doing the exercises by yourself (using all languages and platforms)
- \* Read ahead about OO technology

— MEDIA ARTS & TECHNOLOGY PROGRAM

180

## MAT 201B: Topic 3

# Software Engineering: Objects, Classes, Methodology and Exceptions



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

181

## Software Engineering

- \* Systems & Models
- \* OO technology
  - \* Objects, behaviors
  - \* Classes, relationships
- \* Analysis and design
  - \* Techniques, notations
  - \* OO Design Patterns
- \* Errors, exception-handling and robust programming

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

182

## What will we be learning?

- \* How do we describe large multimedia software systems?
- \* What's so important about object-oriented software technology
- \* What's a software analysis/design methodology, and do I need one?
- \* How does one handle exceptional conditions in programs with non-local goto structures?

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

183

## Systems, Models and Metamodels

- \* What is a system?
  - \* "a large collection of interacting functional units that together achieve a defined purpose" (IEEE group in System Science and Cybernetics).
  - \* "A system is a set of objects together with relationships between the object  
s  
and between their attributes." (Hall and Fagen, 1956)
  - \* "A system is anything that  
y  
ou have decided to regard as a system" (Nygard, 2001)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

184

## Systems, Models and Metamodels

- \* What is a model?
  - \* An abstract representation of a System
- \* Properties of models
  - \* A given system can be represented by different models
  - \* The model must yield the same results as the original system under the restrictions specified
  - \* It must have the "appearance" of the system
  - \* But it is a simplification of the system, leaving some things out and concentrating on others.
  - \* It can be exercised to predict the system's behaviour

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

185

## Systems, Models and Metamodels

- \* Object-orientation, systems and models
  - \* As mentioned, a system can be in general seen as a set of related objects (H&F, 1956) => OO is the best paradigm for representing systems
  - \* OO does not aim at describing the "real world" but at representing existing systems in the real world.
  - \* OO was in fact born as a system simulation artifact: Kristan Nygaard and the Simula language

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

186

## Systems, Models and Metamodels

- \* A metamodel is a “model of models”
  - \* Many related models may share some common constructs
  - \* These shared constructs may be represented by another model thus becoming a model of models.
  - \* The relation between a model and its metamodel is similar to the one existing between an object and a class.
- \* Metamodeling is very much related to the activity of “classifying classes”.

— MEDIA ARTS & TECHNOLOGY PROGRAM

187

## Frameworks generate Metamodels

- \* What is a framework
  - \* An OO software framework is defined as “a set of classes that embodies an abstract design for solutions to a family of problems”
  - \* Frameworks are large abstract applications in a particular domain that can be tailored for individual applications
  - \* A framework is a reusable software architecture comprising both design and code

— MEDIA ARTS & TECHNOLOGY PROGRAM

188

## Frameworks generate Metamodels

- \* Frameworks are constructed by discovering commonalities in a set of related applications
- \* These commonalities end-up becoming the framework itself and defining the domain metamodel (the pool of constructs that are valid for any model in a given domain).

— MEDIA ARTS & TECHNOLOGY PROGRAM

189

## 4MS Introduction

- \* Main hypothesis => any digital signal processing system can be efficiently modeled as a set of related objects.
- \* The metamodel presents a classification of objects into two main metaclasses: Processing and Processing Data
- \* A system can then be seen as a set of connected Processing objects that exchange Processing Data objects.
- \* Other secondary metaclasses exist for Connecting objects or Application-level objects.

— MEDIA ARTS & TECHNOLOGY PROGRAM

190

## Multimedia Processing Systems

- \* Although any system containing text, graphics, video, audio and music -- or any combination of the previous elements -- can be considered a Multimedia system we will restrict our domain to the subset of Multimedia Processing Systems.
- \* This means that these systems will be signal processing intensive (leaving out, for instance, static multimedia authoring systems).

— MEDIA ARTS & TECHNOLOGY PROGRAM

191

## Multimedia Processing Systems

- \* These are the most important features of any Multimedia Processing System (MMPS):
  - \* Data and Process separation:
    - \* multimedia consists of “multimedia data” and a “set of instructions”. In a MMPS it is important to be able to separate both concerns.
  - \* Stream oriented:
    - \* Multimedia processing systems work on streams of data.
    - \* These streams are made of atomic elements called the data tokens.
    - \* Most streams represent time varying signals so tokens have a more or less determinate time stamp.

— MEDIA ARTS & TECHNOLOGY PROGRAM

192

## Multimedia Processing Systems

- \* Most streams represent time varying signals so tokens have a more or less determinate time stamp.
- \* This means that a generic data flow has to be processed in a synchronous manner
- \* A generic multimedia system combines both continuous and discrete data.
- \* We consider the existence of non-temporal data irrelevant in the context of multimedia systems that are processing intensive. }.
- \* Multiple data:
  - \* by definition in a typical system we will have multiple data types and streams that will be processed, transformed and composed with.

MEDIA ARTS & TECHNOLOGY PROGRAM

193

## Multimedia Processing Systems

- \* Interaction:
  - \* most multimedia systems is that they are interactive: the system takes into account the user input in order to control or modify the execution flow.
  - \* Apart from the multiple data streams, we also have a separate event-driven control flow and this control flow may somehow influence the data flow.

MEDIA ARTS & TECHNOLOGY PROGRAM

194

## Multimedia Processing Systems

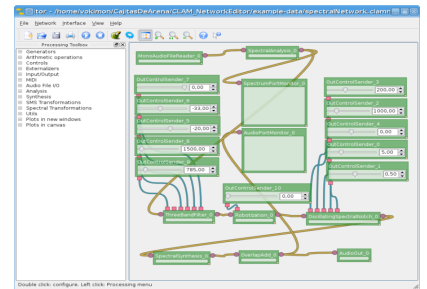
- \* System Complexity:
  - \* Even trivial MMPS can yield complex models containing multiple data types, streams and control paths. There is a clear need for encapsulation, detail hiding and composition at different levels.
- \* Software Orientation:
  - \* Computer-based processing is almost a necessity' for any multimedia system

MEDIA ARTS & TECHNOLOGY PROGRAM

195

## 4MS Network Metamodel

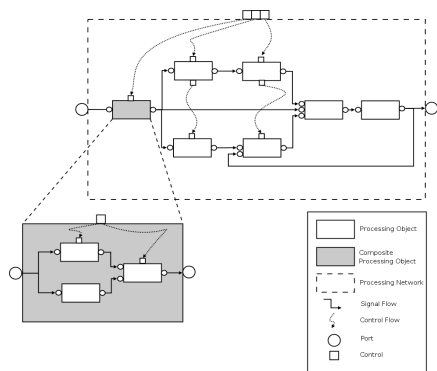
- 4MS Examples
  - The CLAM framework
  - The Spatialization framework by Jorge Castellanos
  - Framework for gesture recognition
  - Influence on CSL 4
- But... 4MS includes concepts already found in many previous environments



MEDIA ARTS & TECHNOLOGY PROGRAM

196

## 4MS Network Metamodel



MEDIA ARTS & TECHNOLOGY PROGRAM

197

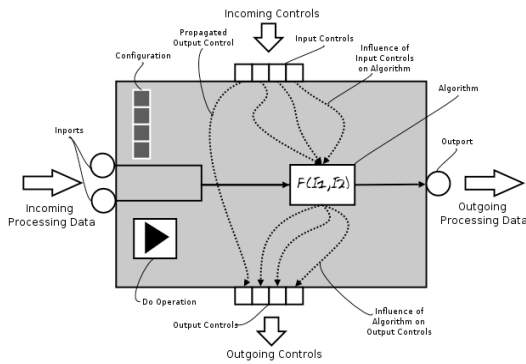
## Processing Objects (I)

- \* Processing object: OO encapsulation of a process.
- \* The concept of "Processing objects" has a clear equivalent in CLAM but also in similar environments:
  - \* Marsyas [Tzanetakis, 02]: transformations
  - \* OSW [Chaudhary, 99] & Kyma [Scaletti, 02]: transforms
  - \* Max [Puckette, 02]: objects
  - \* SndObj [Lazzarini, 00]: sound objects
  - \* CSL [Pope, 03], Siren [Pope, 07] & Aura [Dannenberg, 04]: unit generators
  - \* STK [Cook, 99]: instruments
  - \* FORMES [Rodet, 91]: processes

MEDIA ARTS & TECHNOLOGY PROGRAM

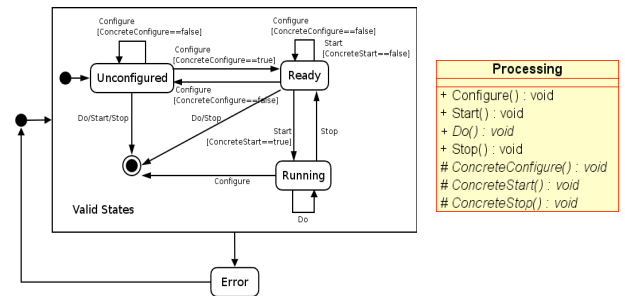
198

## Processing Objects (II)



MEDIA ARTS & TECHNOLOGY PROGRAM  
199

## Processing Object Life-cycle



Processing
+ Configure() : void
+ Start() : void
+ Do() : void
+ Stop() : void
# ConcreteConfigure() : void
# ConcreteStart() : void
# ConcreteStop() : void

MEDIA ARTS & TECHNOLOGY PROGRAM  
200

## Data Objects

- \* Data transmitted to and from Processing objects is encapsulated in Data objects.
- \* Data classes must offer (meta)services such as introspection, homogeneous interface, encapsulation, persistence, display facilities and composition.
- \* Processing Data implement the Payloads pattern [Manolescu, 97]
- \* Types of attributes in a Processing Data object:
  - \* Data Attributes: actual data container attributes
  - \* Value Attributes: auxiliary information related to data attributes (can be either Informative or Structural).

MEDIA ARTS & TECHNOLOGY PROGRAM  
201

## Data Objects and Controls

- \* Processing Data:
  - \* Complex data type
  - \* Semantic attached to the type
  - \* Are sent synchronously at a “predictable” rate.
  - \* In-band Partition pattern [Manolescu, 97]
- \* Controls:
  - \* Simple data types
  - \* No semantic attached to the type
  - \* Encapsulate events
  - \* Are sent asynchronously and instantaneously
    - \* may generate high traffic peaks
  - \* Out-of-band Partition pattern [Manolescu, 97]

MEDIA ARTS & TECHNOLOGY PROGRAM  
202

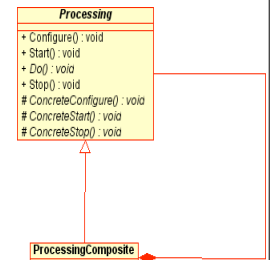
## Configuration

- \* The Configuration is used for setting the initial state of the processing object.
- \* It contains values for a Processing object’s non-execution variables,
  - \* Variable structural attributes that can only be changed when the Processing object is not in the Running state.
  - \* may also have initial values for non-structural attributes such as the controls.

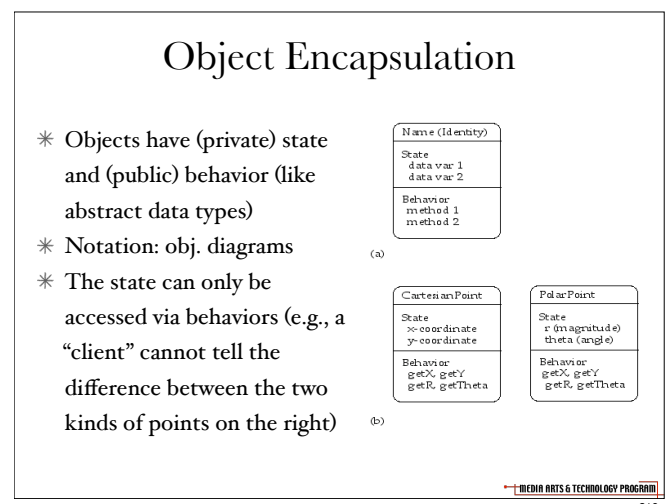
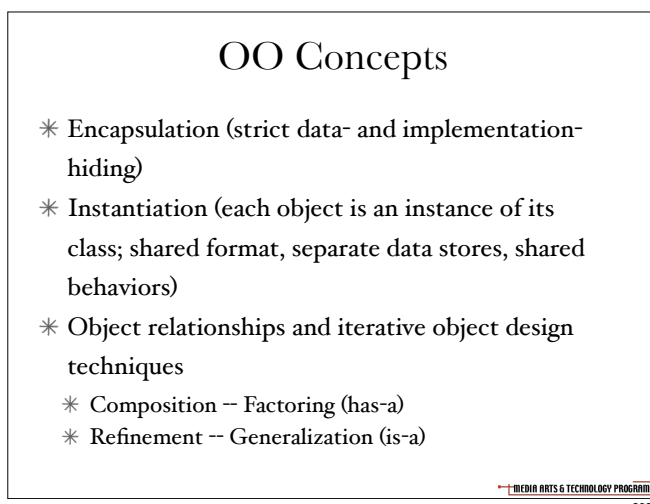
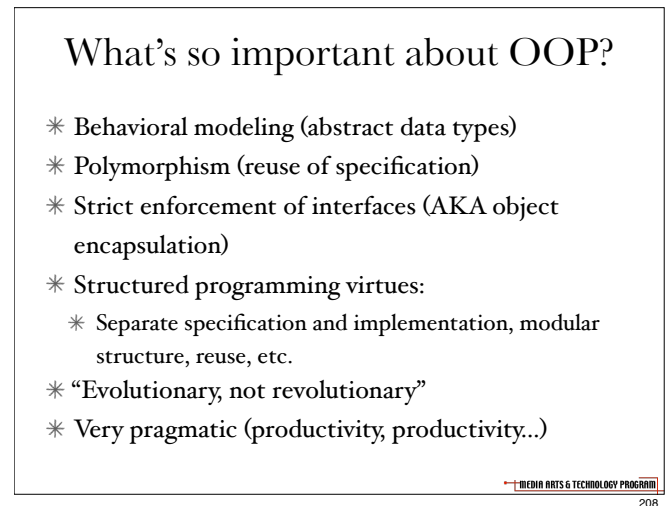
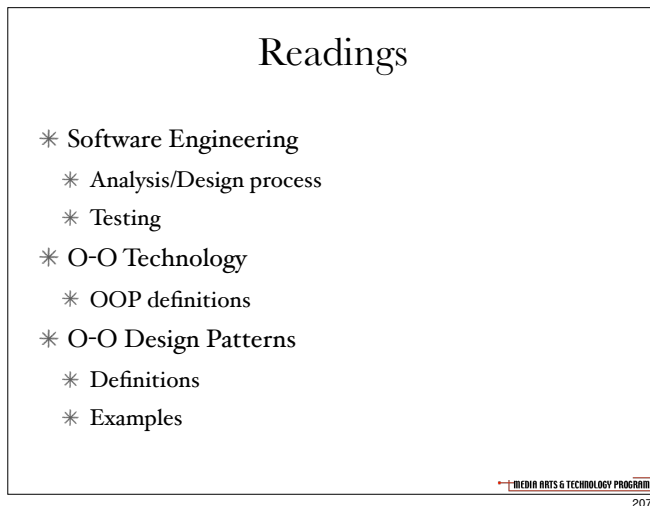
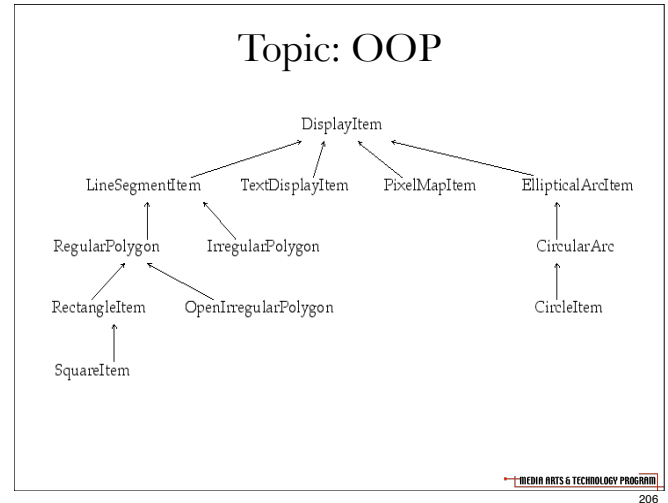
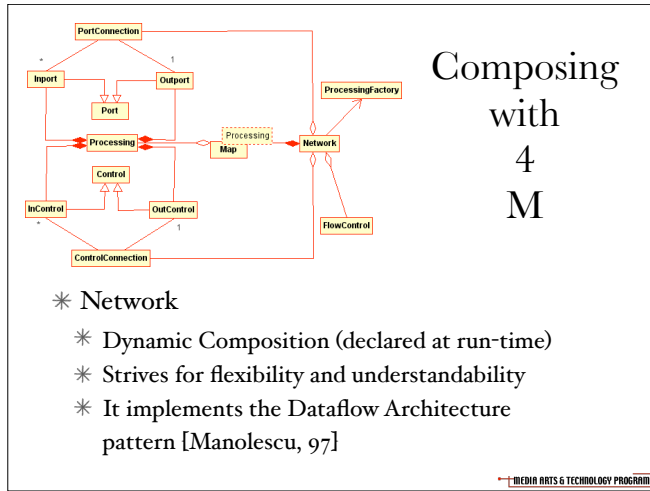
MEDIA ARTS & TECHNOLOGY PROGRAM  
203

## Composing with 4MS Objects (I)

- \* Processing Composite
  - \* Static Composition
  - \* position (declared at compile time)
  - \* Implements the Composite [Gamma, 95] and Adaptive Pipeline [Posnak, 96] patterns
  - \* Can be tuned for efficiency reasons
  - \* It is seen as a regular Processing object



MEDIA ARTS & TECHNOLOGY PROGRAM  
204



## Object Instantiation

- \* Each object is an instance of a class (e.g., the number 7 is an instance of Integer class)
- \* An object may have data components (elements, data members, instance variables, cached slots, etc.)
- \* These members are assumed to be objects themselves (composed objects)
- \* An object has a pointer to its class

MEDIA ARTS & TECHNOLOGY PROGRAM

211

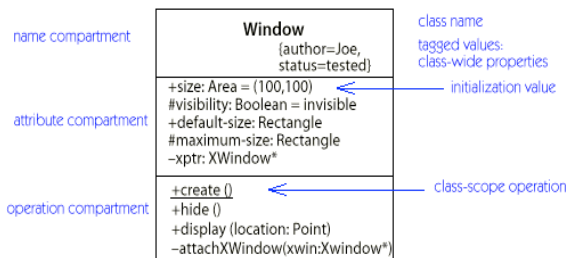
## Classes

- \* The class holds the template for the instance variables of the class (all Cartesian points have x/y coordinates)
- \* The class holds the methods (all points understand the same messages)
- \* Classes act as factories for instances (instance creation or constructor methods)
- \* Classes may have their own methods (static, constructors) and instance caches (Singletons, class pools)

MEDIA ARTS & TECHNOLOGY PROGRAM

212

## How a class is represented

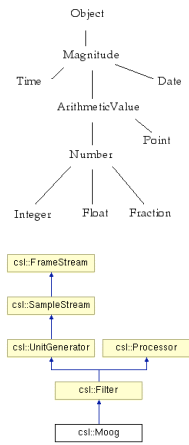


MEDIA ARTS & TECHNOLOGY PROGRAM

213

## Inheritance

- \* Classes can be related in a sub/super-class hierarchy
- \* A subclass inherits the state and behavior of its superclass
- \* Some languages support multiple superclasses (multiple inheritance), requires some way to disambiguate inheritance conflicts (same state or method from two parents)
- \* Classes may be abstract (e.g., you can't draw me a mammal)
- \* Inheritance promotes reuse of code
- \* Many examples in the literature (ST8o, Java, STL, OSG, etc.)



MEDIA ARTS & TECHNOLOGY PROGRAM

214

## Complex Inheritance

- \* Single- vs. Multiple-inheritance
- \* Sharing of state vs. behavior
- \* Sharing of specification vs. implementation
- \* Flavors of state/behavior composition
  - \* Superclass(es)
  - \* Interfaces (spec. only)
  - \* Traits (code only)
  - \* Species (2D inheritance)
- \* Mix-ins

MEDIA ARTS & TECHNOLOGY PROGRAM

215

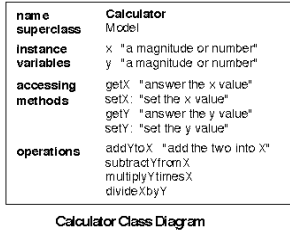
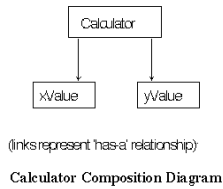
## Functions vs. Methods

- \* In C, the linker can tell exactly what function to call for any given activation (linker fills in function pointers, called early binding)
- \* In a C++ virtual method, the class of the receiver object determines (at run time) what method will be called (message-to-method mapping or method look-up, AKA late binding)
- \* This encourages reuse of specifications and polymorphism
- \* This also introduces some run-time overhead (which there are some fancy techniques to minimize)

MEDIA ARTS & TECHNOLOGY PROGRAM

216

## Data Structures: Composition

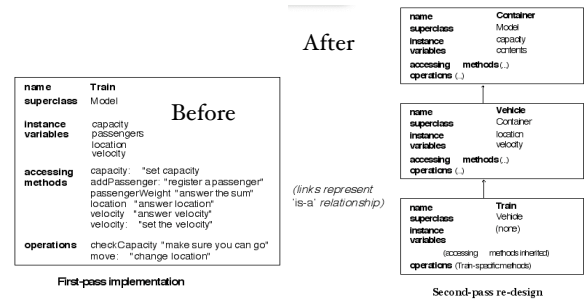


\* "Has-a" relationship (e.g., a point has-a X, has-a Y)

— MEDIA ARTS & TECHNOLOGY PROGRAM

217

## Decomposition or Factorization

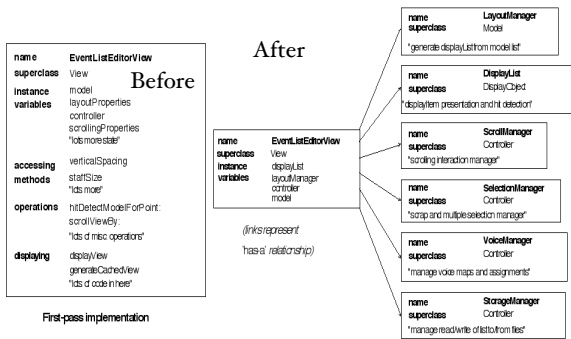


\* Smaller pieces are more reusable

— MEDIA ARTS & TECHNOLOGY PROGRAM

218

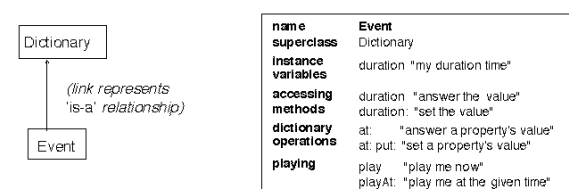
## Decomposition Example



— MEDIA ARTS & TECHNOLOGY PROGRAM

219

## Refinement or Subclassing

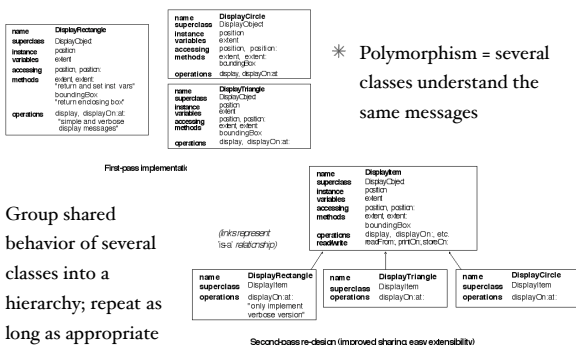


\* Inheritance means an "is-a" relationship between classes (e.g., the group of primates is a refinement of the group of mammals; Integer class is-a Number class)

— MEDIA ARTS & TECHNOLOGY PROGRAM

220

## Generalization or Abstraction



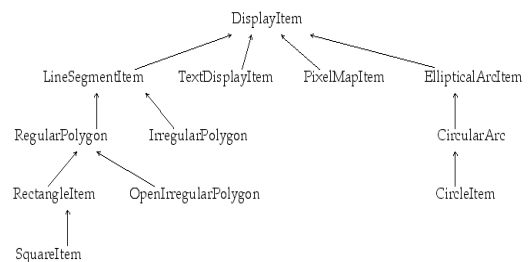
\* Group shared behavior of several classes into a hierarchy; repeat as long as appropriate

\* Polymorphism = several classes understand the same messages

— MEDIA ARTS & TECHNOLOGY PROGRAM

221

## A Possible Hierarchy of Geometrical Display Objects



\* Many alternatives; see real-world examples

— MEDIA ARTS & TECHNOLOGY PROGRAM

222

• MEDIA ARTS & TECHNOLOGY PROGRAM



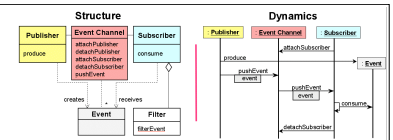
## OO Design Patterns

- \* “Generic designs” or example solutions
- \* History: Architecture, Gang-of-4 book
- \* Format: problem/scenario, components, configuration/relationships, advantages, when to use, notes
- \* Pattern categories
  - \* Creational
  - \* Structural
  - \* Behavioral
- \* Loads of examples, open-source versions

— MEDIA ARTS & TECHNOLOGY PROGRAM

229

## Design Pattern Example: MVC



- \* Abstract, when to use
  - \* Need reusable GUI widgets, multiple views, flexible control, etc.
- \* Components
  - \* MVC component class hierarchies
- \* Configuration, diagram
  - \* See above MVC and advanced MVC (Fig. is publish/subscribe)
- \* Notes
  - \* Limits of MVC: class explosion, issues

— MEDIA ARTS & TECHNOLOGY PROGRAM

230

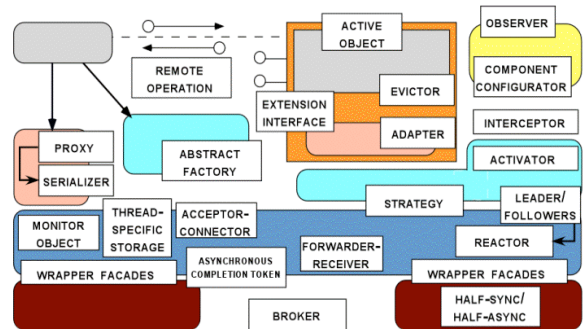
## Standard (Go4) Patterns

- \* Creational: Abstract Factory, Builder, Factory Method, Prototype, Singleton
- \* Structural: Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Proxy
- \* Behavioral: Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor

— MEDIA ARTS & TECHNOLOGY PROGRAM

231

## OO Design Pattern Relationships



- \* by D. Schmidt, POSA (who can interpret this?)

— MEDIA ARTS & TECHNOLOGY PROGRAM

232

## Advanced OO Design Patterns

- \* Application architecture: EJB and AppServer patterns
- \* Layered architectures
- \* Security
- \* Persistence
- \* Type objects
- \* See Schmidt, Manolescu, PLoP conferences

— MEDIA ARTS & TECHNOLOGY PROGRAM

233

## Application Architecture Patterns

- \* From Martin Fowler
- \* Base: Value Object, Gateway, Mapper, Money, Special Case, Plugin, Service Stub, Record Set
- \* Presentation: Page Controller, Front Controller, Template View, Transform View, Two-Step View, Application Controller
- \* Data Source Architectural/Behavioral: Table/Row Data Gateway, Active Record, Data Mapper, Unit-of-Work, Identity Map, Lazy Load
- \* Obj-Relational Structural/Mapping: Key Field, Embedded Value, Inheritance Mapper, Metadata Mapping, Query Object
- \* Session State: Client Session State, Server Session State, Database Session State.

— MEDIA ARTS & TECHNOLOGY PROGRAM

234

## Pattern, Idiom, and Rule

Type	Description	Examples
Idioms	Restricted to a particular language, system, or tool	Scoped locking
Design patterns	Capture the static & dynamic roles & relationships in solutions that occur repeatedly	Active Object, Bridge, Proxy, Wrapper Façade, & Visitor
Architectural patterns	Express a fundamental structural organization for software systems that provide a set of predefined subsystems, specify their relationships, & include the rules and guidelines for organizing the relationships between them	Half-Sync/Half-Async, Layers, Proactor, Publisher-Subscriber, & Reactor
Optimization principle patterns	Document rules for avoiding common design & implementation mistakes that degrade performance	Optimize for common case, pass information between layers

(by D. Schmidt, POSA)

MEDIAS ARTS &amp; TECHNOLOGY PROGRAM

235

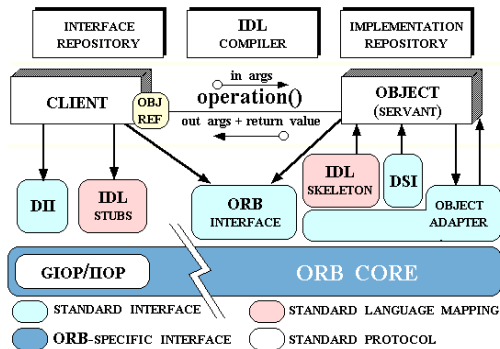
## API, Library, Framework

- \* API, ADT (data structs, fncs)
  - \* UNIX IO
- \* Class hierarchy (shared state/behavior)
  - \* C++ std Collections
  - \* Smalltalk Magnitude/Collection/Stream classes
- \* Framework (models, life-cycles)
  - \* CLAM C++ framework
  - \* Siren snd/music framework

MEDIAS ARTS &amp; TECHNOLOGY PROGRAM

236

## Topic: SW Methodology



MEDIAS ARTS &amp; TECHNOLOGY PROGRAM

237

## Programming in the Large

- \* What's difference between a development project requiring 6 staff weeks and one requiring 6 staff years?
- \* PITL issues
  - \* Analysis/design methodology: A/D processes and notations
  - \* Support for team programming (implies both methodology and tool issues)
  - \* Formal requirements capture/analysis
  - \* Formal testing at several levels

MEDIAS ARTS &amp; TECHNOLOGY PROGRAM

238

## Software Analysis & Design

- \* Analysis: figuring out what to build (policies, rules)
- \* Design: figuring out how to build it (mechanisms, implementations)
- \* Many techniques, tools, notations, and process models over the generations (this is a very big deal if it's an expensive project)

MEDIAS ARTS &amp; TECHNOLOGY PROGRAM

239

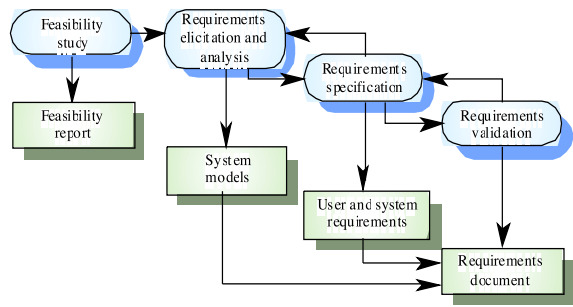
## Software Analysis & Design II

- \* Risk management: being certain about what you don't know may be more valuable than what you do know.
- \* Tools for computer-aided software engineering (CASE) (e.g., notation editors)
- \* Also available in an OO flavor! (big surprise)

MEDIAS ARTS &amp; TECHNOLOGY PROGRAM

240

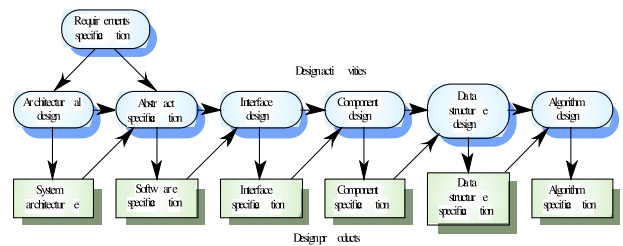
## Requirements Engineering Process



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

241

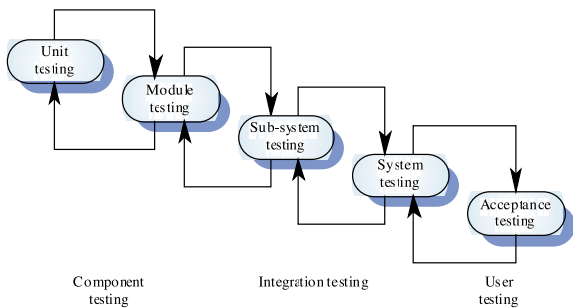
## Software Design



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

242

## The Testing Process



Component testing

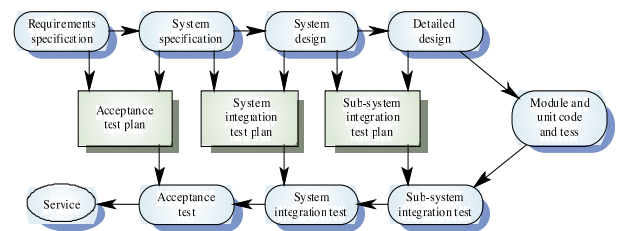
Integration testing

User testing

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

243

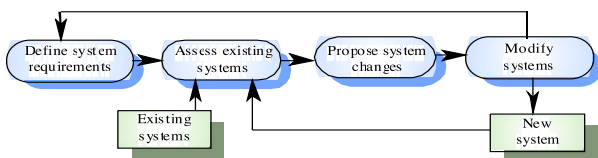
## Testing through the Process



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

244

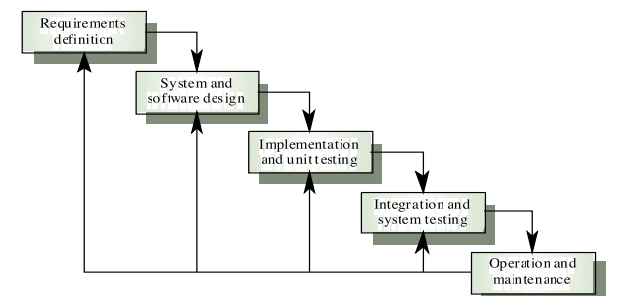
## Software Evolution



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

245

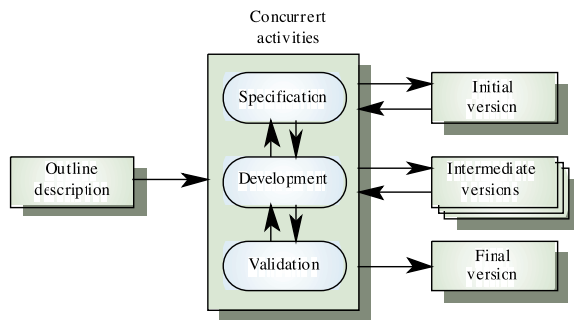
## Waterfall Process Model (NOT)



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

246

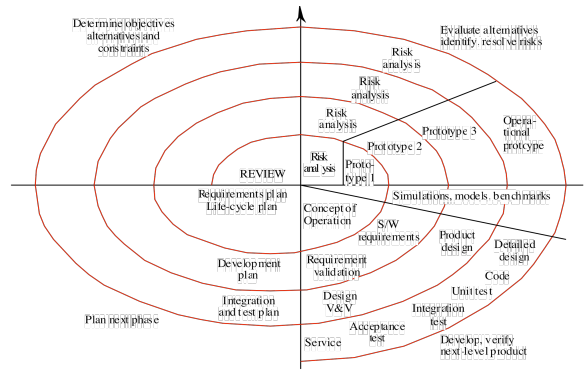
## Evolutionary Model



— MEDIA ARTS & TECHNOLOGY PROGRAM

247

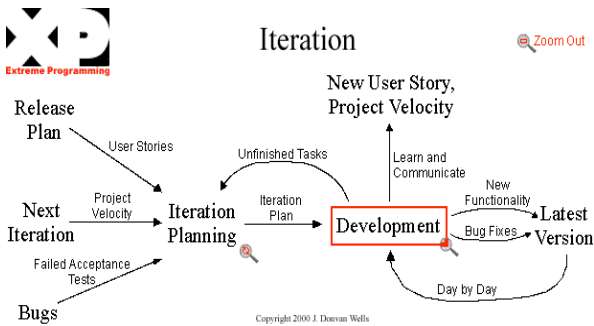
## Spiral Development



— MEDIA ARTS & TECHNOLOGY PROGRAM

248

## Iteration in the XP Process



— MEDIA ARTS & TECHNOLOGY PROGRAM

249

## Analysis Steps

- \* Start with user requirements
  - \* Capture these in some semi-formal language (scenarios, scripts, screens)
- \* Determine necessary functionality (object behaviors)
  - \* Use a graphical notation (CRC cards)
  - \* Capture temporal flow (sequence diagrams, etc.)
- \* Group them into classes
  - \* Cross-check for coverage
  - \* Iterate over naming, language design, and partitioning
- \* Group classes into hierarchies
  - \* Generalization, etc.
- \* Iterate until "done"
- \* Deliverable may be a domain model (special case)

— MEDIA ARTS & TECHNOLOGY PROGRAM

250

## Design Steps

- \* Determine system architecture and components (subsystems)
  - \* Partitioning, layering, coupling/cohesion
- \* Specify interfaces between subsystems
  - \* APIs and specifications
  - \* Design for reuse (assuming a given class library)
- \* Refine until satisfied you can plan the implementation (and estimate the cost [LOE])
  - \* All components specified to a given level of detail

— MEDIA ARTS & TECHNOLOGY PROGRAM

251

## Generations of SE Methods

- \* Ad-hoc methods
- \* Structured (SASD) methodology
  - \* Focus on specifications and sub-system partitioning
  - \* Client/server architecture
- \* DB-oriented (IE) techniques
  - \* Table formats and screen design
- \* Several flavors of OO methods and notations
  - \* Booch, Coad/Yourdon, OBA

— MEDIA ARTS & TECHNOLOGY PROGRAM

252

## OOA and OOD Methods

- \* Object behavior analysis
  - \* Iterative, incremental life-cycle and process model
  - \* Strict focus on behavior rather than state
  - \* Application, domain, process, or enterprise models
- \* Many modeling approaches
  - \* Type as class
  - \* Process as class
  - \* Interface as class
- \* OOD techniques exploit/encourage reuse

MEDIA ARTS & TECHNOLOGY PROGRAM

253

## OO Software Libraries

- \* Uniformity (Everything is an object!)
- \* Sophistication/coverage
- \* Generic vs. domain-specific
- \* Based on design patterns (see below)
- \* Tools for reuse
  - \* Xerox PARC “programmer as reader” project and the Smalltalk-80 tool set
- \* Two rules of reuse (these are tool, doc issues)
  - \* 1: If you can't find it, you can't reuse it.
  - \* 2: If you don't trust it, you won't reuse it.

MEDIA ARTS & TECHNOLOGY PROGRAM

254

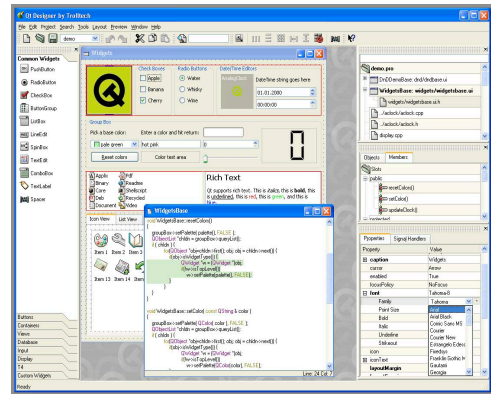
## Special Tools for OOP

- \* Class browsers
  - \* Built in to most IDEs
  - \* Often not used because they're funky and users think of code in files
- \* Refactoring tools
  - \* Rare (unfortunately)
- \* Good OO libraries
  - \* Rare, poorly standardized, few strict naming and design conventions found

MEDIA ARTS & TECHNOLOGY PROGRAM

255

## Topic: Control Flow



MEDIA ARTS & TECHNOLOGY PROGRAM

256

## Readings 3

- \* Errors and Exception Handling
  - \* C++ Tutorial: Exception Handling
  - \* COP 2334 Lecture Notes
- \* Language-specifics
  - \* Java
  - \* Smalltalk

MEDIA ARTS & TECHNOLOGY PROGRAM

257

## Control Flow, Error Handling and Exceptions

- \* Control flow
  - \* Statements that alter the linear program flow (goto, if/then, while/do, for...)
  - \* Assume some way of delineating blocks ({...})
  - \* Assembly language: jump-to-subroutine, branch-if-not-equal, etc.
- \* What to do in “exceptional” conditions
  - \* Jump to a “handler” function/block
  - \* Return invalid (out-of-band) data
  - \* Set some global error flag

MEDIA ARTS & TECHNOLOGY PROGRAM

258

## Goto Example from CSL

```
while (TRUE) { // endless loop
 xfer_len = read(aSocket, (char *) _buffer, num2read);
 if (xfer_len < 0) { // if read returns < 0 (error)
 perror("RemoteIO loop read"); // whine
 goto next_loop; // goto statement label
 }

 // handle input data here...

next_loop: // error handler -- ignore it
 continue; // continue with loop
}
```

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

259

## Set jump & long jump

- \* Frequent use of goto leads to “spaghetti code”
- \* In addition to goto, C provides “long jump” for non-local control transfers.
- \* The question is: how does one manage the call stack (and temp. objects in it) in this case? (See the various C-language setjmp/longjmp tutorials.)
- \* This will quickly get very messy...

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

260

## Handling Errors

- \* It is often useful to detect an error in a subroutine you call, e.g., a file protection error in a file I/O function.
- \* The most common solution is to return a numerical code whereby the value 0 normally means “no error.”
- \* See the C-language APIs we’ve discussed so far (man read, man malloc).

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

261

## Problems with Error Codes

- \* Uniformity: all functions must return the same kind of error codes
- \* Procedure vs. function: functions cannot return other kinds of values
- \* 1-Dimensionality: Integer error codes provide only one linear range for errors
- \* There are work-arounds for each of these issues, but it gets more complicated and less scalable

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

262

## Solution: Model Exceptions

- \* The OO answer: use an exception object to represent the exceptional condition or error
- \* An exception can be thrown by a piece of code that encounters some exceptional situation.
- \* The exception can be caught by another piece of code (anywhere up the call stack) that knows how to handle the given situation.
- \* This must be built into the language and run-time support libraries (i.e., it’s hard to add it after-the-fact).

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

263

## Spreadsheet Example

- \* The task:
  - \* Calculate a value for the current cell
  - \* If you get a divide-by-zero error, do this...
  - \* If you get a sqrt-of-a-negative-number error, do this...
  - \* If you get any other kind of math error, do this...

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

264

## Spreadsheet Example in Java

```
try {
 updateCell(row, col); // do a calculation
} catch (DivideByZero err) { // if x / 0
 // x/0 handler code goes here
} catch (SqrtOfANegative err) { // if sqrt(-1)
 // handler
} catch (Exception err) { // other exception
 // handler
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

265

## Catching Unrelated Exceptions

```
try {
 /* do something block */
} catch (ArrayIndexOutOfBoundsException e) {
 System.err.println("Index out of bounds: "
 + e.getMessage());
} catch (IOException e) {
 System.err.println(" IOException: "
 + e.getMessage());
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

266

## Exception Details

- \* Exceptions are members of a hierarchy (see examples below).
- \* Calling code can handle specific or general exceptions (concrete or abstract exc. classes).
- \* Outer-level functions (e.g., event-handler loops) generally handle high-level exceptions in a general way (whine, save data, exit).
- \* Inner functions often handle specific exceptions in more refined ways (substitute return values, clean up).

— MEDIA ARTS & TECHNOLOGY PROGRAM

267

## Advanced Exception Usage

- \* Special considerations for finally blocks and unwinding: clean-up examples.
- \* Exceptions should be declared in the signatures of methods that can throw them (in Java or IDL).
- \* This is an important part of the design of an object framework: what new kinds of exceptions can happen here?

— MEDIA ARTS & TECHNOLOGY PROGRAM

268

## Advanced Exceptions 2

- \* Exceptions in “advanced” languages (Smalltalk, LISP, ADA, ML, Eiffel, etc.)
- \* Unwind/protect blocks and finalization
- \* Handler collections and nesting
- \* Exceptions vs. semaphores
- \* Class vs. instance hierarchies and delegation vs. inheritance
- \* Restarting wedged functions
- \* Other uses of exceptions (schedulers, interrupts, etc.)

— MEDIA ARTS & TECHNOLOGY PROGRAM

269

## Final Blocks and Clean-up

- \* Example: case where a file needs to be closed no matter what:
 

```
try {
 * write(...); // operation that might throw exc.
 * out.close(); // normal case: close the file
 * } catch (SomeExceptionClass e) {
 * out.close(); // duplicate code here
 * }
 * Rewrite this with a finally block at the end
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

270

## Problems and Issues

- \* Many Java/C++ frameworks do not define and throw fine-grained exceptions (see examples below).
- \* The treatment of sequential handlers is different between C++, Java, etc.
- \* C++ exceptions are generally not special objects but strings or ints (leading to switch statements, yuck!)

— MEDIA ARTS & TECHNOLOGY PROGRAM

271

## Exception Hierarchy Examples

```

class java.lang.Exception
 class java.lang.ClassNotFoundException
 class java.lang.CloneNotSupportedException
 class java.lang.IllegalAccessException
 class java.langInstantiationException
 class java.lang.InterruptedException
 class java.lang.NoSuchFieldException
 class java.lang.NoSuchMethodException
 class java.lang.RuntimeException
 class java.lang.ArithmeticException
 class java.lang.ArrayStoreException
 class java.lang.ClassCastException
 class java.lang.IllegalArgumentException
 class java.lang.IllegalThreadStateException
 class java.lang.NumberFormatException
 class java.lang.IllegalMonitorStateException
 class java.lang.IndexOutOfBoundsException
 class java.lang.ArrayIndexOutOfBoundsException
 class java.lang.StringIndexOutOfBoundsException
 class java.lang.NegativeArraySizeException
 class java.lang.NullPointerException
 class java.lang.SecurityException
 class java.lang.UnsupportedOperationException

class java.lang.Error
 class java.lang.LinkageError
 class java.lang.ClassCircularityError
 class java.lang.ClassFormatError
 class java.lang.UnsupportedClassVersionError
 class java.lang.ExceptionInInitializerError
 class java.lang.IncompatibleClassChangeError
 class java.lang.AbstractMethodError
 class java.lang.IllegalAccessError
 class java.lang.InstallationError
 class java.lang.NoSuchFieldError
 class java.lang.NoSuchMethodError
 class java.lang.UnsatisfiedLinkError
 class java.lang.VerifyError
 class java.lang.ThreadDeath
 class java.lang.VirtualMachineError
 class java.lang.InternalError
 class java.lang.OutOfMemoryError
 class java.lang.StackOverflowError
 class java.lang.UnknownError

```

- \* In Java

— MEDIA ARTS & TECHNOLOGY PROGRAM

272

## Exception Hierarchy Examples

```

exception
 bad_alloc (thrown by new)
 bad_cast (thrown by dynamic_cast when fails with a referenced type)
 bad_exception (thrown when an exception doesn't match any catch)
 bad_typeid (thrown by typeid)
 logic_error
 ... domain_error
 ... invalid_argument
 ... length_error
 ... out_of_range
 runtime_error
 ... overflow_error
 ... range_error
 ... underflow_error
 ios_base::failure (thrown by ios::clear)

```

- \* In C++ (add-on libraries such as Rogue Wave or BOOST augment this)

— MEDIA ARTS & TECHNOLOGY PROGRAM

273

## Exception Hierarchy Examples

- \* In Smalltalk (with several additional packages loaded)

```

Category Hierarchy
Object
 GenericException
 Exception
 AllocationFailure
 ArithmeticError
 DomainError
 ZeroDivide
 RangeError
 UnorderedNumbersError
 BranchLimit
 ByteSwappedError
 CannotResumeError
 CannotReturnError
 CantResumeError
 ClassConstructionError
 CodeSimulationError
 CodeStorageError
 ColorError
 CompilationError
 CTypeError
 DuplicateBindingsError
 ExceededLimitsError
 ExternalAccessFailed
 ExternalDatabaseException
 AuthenticationFailure
 ConnectionException
 ConnectionNotOpen
 DuplicateIndexesError
 DynamicSQLError
 ExternalDatabaseLibraryInaccessible
 ExternalDatabaseResumableException
 NoSuchConnection
 RequiredEnvironment
 RequiredPassword
 RequiredUsername
 UnableToReconnect
 UnableToFreeResource
 IndexConflict
 InvalidConnectionState

Category Hierarchy
InvalidSettingValueError
InvalidStoreSettingDataError
MemberNotFoundError
MenuAugmentationError
MessageNotUnderstood
NamespaceFailureSignal
NativeError
PostgreSQLException
PostgreSQLError
PostgreSQLErrorResponse
PostgreSQLUnexpectedMessage
PostgreSQLUnsupportedFeature
PostgreSQLLoggingError
NoMatchingFontError
NoModificationError
ConstantBindingModificationError
NonCharacterError
NotFoundError
IndexNotFound
BindingNotFoundError
KeyNotFoundError
NonIntegerIndexError
ValueNotFoundError
NFCMapError
ObsoleteClassError
ObsoleteError
ObsoleteOperation
OsInaccessibleError
OsInvalidArgumentError
OsNeedRetryError
OsNotification
OsTransferFailure
ParseError
PrimitiveFailure
PrintTimeoutError
QueueOverflowError
ReflectionError
RestartRequest

```

— MEDIA ARTS & TECHNOLOGY PROGRAM

274

## Review

- \* Object-oriented concepts and implementation
- \* Iterative techniques: factoring and generalization
- \* OO analysis and design techniques
- \* Errors and exceptions: exception handling in modern languages

— MEDIA ARTS & TECHNOLOGY PROGRAM

275

## Coding Example

- \* OO class hierarchies (analyze, use, refine, develop)
- \* Use UML tools
- \* Add exc. handling to an existing application
- \* Divider class in Java
  - \* Version 0 (no mention of exceptions)
  - \* Version 1: no handlers
  - \* Updates: add handlers
    - \* Missing cmd-line arguments, missing input file, math error, output file creation, other cases?

— MEDIA ARTS & TECHNOLOGY PROGRAM

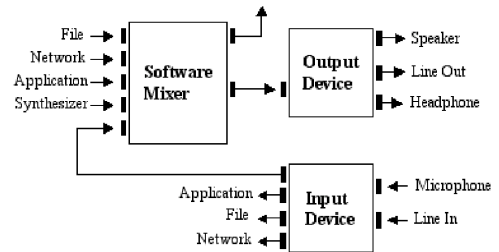
276

## What's Next?

- \* Topic 4: Sound/music
  - \* Readings 4
- \* Coding
  - \* Audio Apps
  - \* Audio APIs

## MAT 201B: Topic 4

### Computing with Sound and Music Data



## Computing with Sound and Music Data

- \* Music/sound representation, notation, interchange formats
- \* Sound file formats, history, data structures, examples
- \* Tagged or chunked file formats
- \* Sound (stream and file) I/O APIs

## Readings 4

- \* A Child's Garden of Sound File Systems
- \* PortAudio Tutorial, P. Burke
- \* References
  - \* Platform-specific
  - \* Cross-platform
  - \* Plug-in APIs

## What will we be learning?

- \* How does one represent music or sound (theory and practice)?
- \* What's the fascinating history of sound storage formats and file systems?
- \* How do sound APIs work?
- \* How do I program sound IO on Windows, Mac, and Linux?
- \* What are the popular cross-platform sound APIs?

## Music Representation, Notation and Interchange

- \* History
  - \* Score, sketch, tabulature
  - \* 1960s: Music input languages (MILs), compositional algorithms
  - \* 1970s: AI/KR and music representation
  - \* 1980s: Music theory, psychology, and MR-grammars, nets, rules, GAs, etc.
  - \* Abstractions used in MILs

## Classes of Systems

- \* In the literature
  - \* Music input languages
  - \* Music programming languages
  - \* Music representation models
  - \* Music interchange protocols

MEDIA ARTS & TECHNOLOGY PROGRAM

283

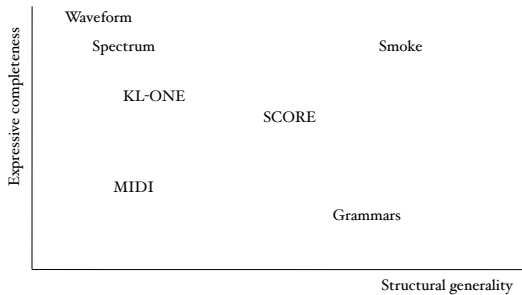
## Music Representation, Notation, and Interchange

- \* Music vs. sound vs. score
- \* Structure and content
  - \* Compositional algorithm as representation
- \* Performance scripts
  - \* Traditional score, screenplay
- \* Derived analysis
  - \* Spectrograms and reader's scores

MEDIA ARTS & TECHNOLOGY PROGRAM

284

## “Dimensions of Representation” (from Wiggins et al.)



MEDIA ARTS & TECHNOLOGY PROGRAM

285

## Sound File Formats

- \* History, Background
- \* OS, HW, and API considerations
- \* Sampling rates
- \* Sample formats and resolutions
- \* Headers and header-less formats
- \* Makes for thrilling reading!

MEDIA ARTS & TECHNOLOGY PROGRAM

286

## Sampling and Quantization

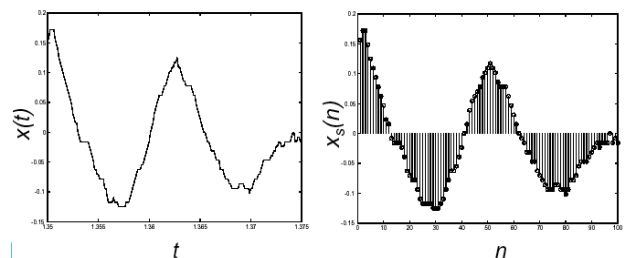
The **digitization** of a signal  $x(t)$  comprises two steps:  $x(t) \rightarrow x_s(n) \rightarrow x_q(n)$

- **Sampling:**  $x_s(n) = x(nT)$  where
  - $T$  is the **sampling period**
  - $F = 1/T$  is the **sampling frequency**
  - The inverse transformation ( $x_s(n)$  to  $x(t)$ ) is called **interpolation**
- **Quantization:**  $x_q(n) = Q(x_s(n))$ 
  - $Q(\cdot)$  is a **rounding function** which maps the values of  $x_s(n)$  into  $N$  levels ( $\Delta$  is the **quantization step**)
  - Typically,  $N = 2^{N_b}$ , therefore we need  $N_b$  bits to represent one quantized sample.

MEDIA ARTS & TECHNOLOGY PROGRAM

287

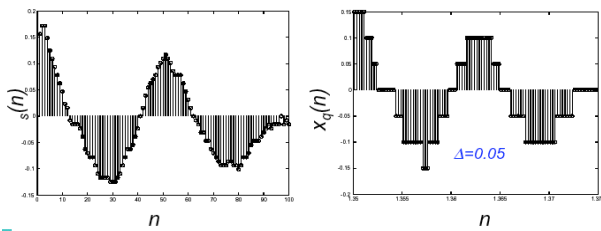
## Sampling Example



MEDIA ARTS & TECHNOLOGY PROGRAM

288

## Quantization Example



MEDIA ARTS & TECHNOLOGY PROGRAM

289

## The Wonderful History of Sound File Formats

- \* Tape-based interchange (1960s)
  - \* SWSS and D/A in multiple stages
- \* Disks and sound file systems (1970s)
  - \* Special-purpose CCSS (cylinder-contiguous)
  - \* General-purpose SFS on UNIX FS
- \* Large-scale sound storage (1980s)
- \* Streaming audio over LAN/WAN (1990s)
- \* Compressed formats and perceptual coding (MP3, AAC) (1990s)

MEDIA ARTS & TECHNOLOGY PROGRAM

290

## Sound File Systems

- \* Issues
  - \* Throughput, file size, mark-up, compression, multiple formats
- \* UNIX special-device model
  - \* `open()`, `close()`, `read()`, `write()`, `ioctl()`
  - \* Special directory operations
- \* OO Model
  - \* Directory, Filename, `IOStream` objects

MEDIA ARTS & TECHNOLOGY PROGRAM

291

## The Play/Record Program

- \* Steps, functions
  - \* Command parsing, set-up
  - \* File I/O
  - \* Buffering
  - \* Driver and scheduling
  - \* Interrupt servicing
- \* The Sound I/O API
  - \* OS-, HW- and format-dependent

MEDIA ARTS & TECHNOLOGY PROGRAM

292

## Sampling Rates

- \* Early: 10 kHz, 12-bit samples (determined by 1960s DAC HW)
- \* See table in Pope & van Rossum
- \* Nyquist and 44 KHz
- \* Up-sampling conversion and reconstruction filters
- \* Modern rates: 5500 - 44100 - 192000 Hz
  - \* Common divisors of high frequencies
  - \* Common multiples of powers of two or other sampling frequencies

MEDIA ARTS & TECHNOLOGY PROGRAM

293

## Sample Formats, Resolutions

- \* 8-bit compressed - 24-bit linear - 32- or 64-bit floating-point (or higher)
- \* Format: linear,  $\mu$ -law, floating-point
- \* Resolution: 4-64-bits, relationship to computer word size
- \* Perception and limen testing (70s, 90s)
- \* Linearity, monotonicity, residual noise of contemporary playback systems
- \* Current 24-bit is overkill (straight wire has noise around 20-bits)

MEDIA ARTS & TECHNOLOGY PROGRAM

294

## Sampled Sound Format Examples

Speech/Audio Type	Frequency Range	Sampling Rate	Bits/Sample	Uncompressed Bit Rate
Narrowband Speech	200-3200 Hz	8 kHz	16	128 kb/s
Wideband Speech	50-7000 Hz	16 kHz	16	256 kb/s
CD Audio	20-20000 Hz	44.1 kHz	16 x 2 channels	1.41 Mb/s

\* Others?

— MEDIA ARTS & TECHNOLOGY PROGRAM

295

## Sound Compression

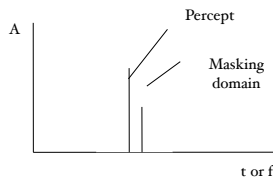
- \* Criteria: effective / real-time / artifact-free: pick any 2
- \* Linear/exponential:  $\mu$ -law (- A-law)
  - \* Non-linear quantization, look-up tables
  - \* 4:1 -- OK for speech only
- \* Statistical: Huffman
  - \* File dictionaries (compress, zip, StuffIt)
  - \* Generally ineffective for sound (try it!)

— MEDIA ARTS & TECHNOLOGY PROGRAM

296

## Compression Models

- \* LPC model (subtractive synthesis)
  - \* CELP, ADPCM, MPEG
  - \* Very effective (100:1), complex, OK for speech
- \* Masking model
  - \* MPEG, MP3
  - \* Effective, complex
  - \* Has artifacts



— MEDIA ARTS & TECHNOLOGY PROGRAM

297

## Sound Formats and Bandwidths

- \* Surround audiophile HiFi (8 @ 24/96) - 18 M b/s
- \* “CD-quality” (2 @ 16/44) - 1.4 M b/s
  - \* Losses (rel. to 24/96) easily heard over good systems
- \* MP3/AAC (DCT, masking) - 128 k b/s (VBR)
  - \* Debatable losses relative to “CD-quality” over “mid-fi” systems for  $\geq 128$  k b/s bandwidth
  - \* Obvious losses for lower bit-rates
- \* RealAudio (CELP) - 14 k b/s
  - \* Monophonic, “telephone-quality”
- \* Best speech compression - 1.5 k b/s

— MEDIA ARTS & TECHNOLOGY PROGRAM

298

## The Sound File

- \* Header vs. header-less
  - \* Type implicit in name (no header)
    - \* xxx.au, xxx.snd
  - \* Format described in file header (or resource fork)
- \* Fixed-format vs. “chunked” formats
  - \* File starts with a single data structure followed by data
  - \* File as a list of self-describing data chunks

— MEDIA ARTS & TECHNOLOGY PROGRAM

299

## The Sound File Header

- \* “Magic number” — normally 32-bit (or 4-char string), identifies file type
- \* Basic parameters
  - \* Sampling rate
  - \* Format/resolution
  - \* Number of channels
- \* Sample data block (typically 1)
- \* Other data
  - \* Comment, text
  - \* Pitch
  - \* MIDI data
  - \* Loop points

— MEDIA ARTS & TECHNOLOGY PROGRAM

300

## The NeXT/Sun File Header (1986)

```
typedef struct { /* .snd file header struct */
 int magic; /* magic number SND_MAGIC */
 int dataLocation; /* offset/pointer to the data */
 /* (header can be >28 bytes) */
 int dataSize; /* number of bytes of data */
 int dataFormat; /* the data format code */
 int samplingRate; /* the sampling rate */
 int channelCount; /* the number of channels */
 char info[4]; /* optional text information */
} SNDSoundStruct;
```

MEDIA ARTS & TECHNOLOGY PROGRAM

301

## Extended Fixed Headers

- \* MOD, AVR, SndDesigner
  - \* Sample Loop points (begin/end)
  - \* Pitch, MIDI key
  - \* MIDI keyboard split, multiple samples
- \* IRCAM/BICSF, EBICSF
  - \* MaxAmp, ampl. envelopes
  - \* Pitch envelopes, LPC
  - \* Processing scripts
  - \* Cues, virtual files

MEDIA ARTS & TECHNOLOGY PROGRAM

302

## Tagged Formats: AIFF/WAV

- \* File is 2 or more self-describing “chunks”
- \* Each chunk has an 8-byte header, followed by a structure whose interpretation depends on the chunk type
- \* Chunk = ID, Size, Data
  - \* struct ChunkHeader {
    - \* int32 ckID; // char[4] = unique type identifier
    - \* int32 ckSize; // size of chunk in bytes
  - \* };
- \* File: FORM, COMM, DATA, etc.
- \* See examples in readings

MEDIA ARTS & TECHNOLOGY PROGRAM

303

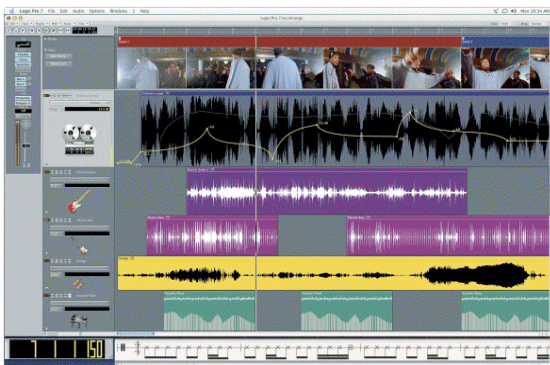
## Advanced: SDIF

- \* Berkeley/IRCAM spectral data interchange format
- \* Supports, sound, STFT spectral data, tracked pitch contours, LPC data, feature vectors, etc.
- \* All data is time-stamped matrices with type keys

MEDIA ARTS & TECHNOLOGY PROGRAM

304

## Sound APIs



MEDIA ARTS & TECHNOLOGY PROGRAM

305

## Sound I/O Hardware

- \* Old days: custom D/A converters with bus interface and device driver
- \* Now: “sound card” with
  - \* D/A and A/D interface
  - \* Digital audio I/O (e.g., S/P-DIF)
  - \* MIDI I/O
  - \* Programmable DSP processor

MEDIA ARTS & TECHNOLOGY PROGRAM

306

## Sound IO APIs

- \* Platform-specific
  - \* MS-Windows: DirectX, DirectSound
  - \* Macintosh: CoreAudio, CoreMIDI
  - \* Linux: ALSA, OSS, Jack, LADSPA
- \* Cross-platform
  - \* JavaSound
  - \* PortAudio
  - \* Sndlib



307

## 3 Dimensions of Sound APIs

- \* How sample buffers are handled
  - \* Sample blocks (short \* samples[])
  - \* Opaque pointers (void \* data)
  - \* I/O stream object
- \* I/O Model
  - \* UNIX model (open/close/read/write/ioctl)
  - \* Sync vs. async I/O (rare)
  - \* Call-back registry (common)
- \* API structure
  - \* Procedural
  - \* Object-oriented



308

## DirectX, DirectSound

- \* Merge of DirectMusic (+ Producer, Composer), DirectSound, Direct3D
- \* Current: DirectX 9.0
- \* Extensions: Downloadable Sounds (DLS), DirectX Media Objects, Direct3D
- \* File and streaming API, DSP pipe-line and plug-ins
- \* DirectSound buffers, synthesizers, segments and containers



309

## DirectMusic vs. DirectSound

The following table summarizes the functionality offered by the two APIs.

Functionality	DirectMusic	DirectSound
Play WAV sounds	Yes	Yes
Play MIDI	Yes	No
Play DirectMusic Producer segments	Yes	No
Load content files and manage objects	Yes	No, but some support in sample code
Control musical parameters at run time	Yes	No
Manage timeline for cuing sounds	Yes	No
Use downloadable sounds (DLS)	Yes	No
Set volume, pitch, and pan of individual sounds	Yes, through DirectSound API	Yes
Set volume on multiple sounds (audiopaths)	Yes	No
Implement 3-D sounds	Yes, through DirectSound API	Yes
Apply effects (DHOs)	Yes, through DirectMusic Producer content or DirectSound API	Yes
Chain buffers for mix-in (send) effects	Yes, through DirectMusic Producer content	No
Capture WAV sounds	No	Yes
Implement full duplex	No	Yes
Capture MIDI	Yes	No
Control allocation of hardware buffers	No	Yes



310

## DirectSound APIs

- \* IDirectSound8 -- Used to create buffer objects and set up environment.
- \* IDirectSound3DBuffer8 -- Used to retrieve and set parameters that describe the position, orientation, and environment of a sound buffer.
- \* IDirectSound3DListener8 -- Used to retrieve and set parameters that describe a listener's position, orientation, and listening environment.
- \* IDirectSoundBuffer8 -- Used to manage sound buffers.
- \* IDirectSoundCapture8 -- Used to create sound capture buffers.
- \* IDirectSoundCaptureBuffer8 -- Sound capture buffers.
- \* IDirectSoundFullDuplex8 -- Represents a full-duplex stream.
- \* IDirectSoundFXEcho8 -- (Plug-ins...) used to set and retrieve effect parameters on a buffer that supports echo.
- \* IDirectSoundNotify8 -- Sets up notification events for a playback or capture buffer.
- \* IKsPropertySet -- Enables drivers to provide extended capabilities that can be used without API extensions.



311

## DirectSound Example: PlaySound

### Functions & Globals

```
INT_PTR CALLBACK MainDlgProc(HWND hDlg, UINT msg,
 WPARAM wParam, LPARAM lParam);
VOID OnInitDialog(HWND hDlg);
VOID OnOpenSoundFile(HWND hDlg);
HRESULT OnPlaySound(HWND hDlg);
HRESULT PlayBuffer(BOOL bLooped);
VOID OnTimer(HWND hDlg);
VOID EnablePlayUI(HWND hDlg, BOOL bEnable);

CSoundManager * g_pSoundManager = NULL;
CSound * g_pSound = NULL;
BOOL g_bBufferPaused;
```



312

## PlaySound.cpp: OnOpenSoundFile()

```
// Setup the OPENFILENAME structure
OPENFILENAME ofn = { sizeof(OPENFILENAME), hDlg, NULL,
 TEXT("Wave Files*.wav\\All Files*.*"), NULL,
 0, 1, strFileName, MAX_PATH, NULL, 0, strPath,
 TEXT("Open Sound File"),
 OFN_FILEMUSTEXIST|OFN_HIDEREADONLY, 0, 0,
 TEXT(".wav"), 0, NULL, NULL };

...

// Load the wave file into a DirectSound buffer
if(FAILED(hr = g_pSoundManager->Create(&g_pSound, strFileName,
 0, GUID_NULL)))
```



313

## DirectSound Examples

- \* See MDSN Web site
  - \* [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9\\_c/directx/html/directsound.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/directx/html/directsound.asp)
- \* 3D sound
- \* SoundFX
- \* Devices and control
- \* Producer, Composer



314

## Mac OSX CoreAudio

- \* Old Mac World(s)
  - \* SoundManager, MIDIManager
  - \* ASIO, OMS, FreeMidi
- \* Mac OSX (Darwin)
  - \* Mach/BSD base
  - \* Apple Frameworks
  - \* Classic support
  - \* QuickTime 6



315

## CoreAudio Structure

- \* Hardware Abstraction Layer
- \* AudioUnits
- \* Audio Toolbox
- \* QuickTime
- \* SoundManager/Classic
- \* C API with C++, ObjC, and Java wrappers



316

## CoreAudio HAL

- \* AudioHardware calls
  - \* List audio IO devices
  - \* Get/set device properties (property names are #define'd)
  - \* AudioDevice structures and API call-back functions
  - \* Device property listener functions
  - \* AudioStreams and IO



317

## Example: Get the Device List

```
UInt32 theSize;
theStatus = AudioHardwareGetPropertyInfo
 (kAudioHardwarePropertyDevices, &theSize, NULL);
theNumberDevices = theSize / sizeof(AudioDeviceID);
theDeviceList =(AudioDeviceID*) malloc (theNumberDevices
 * sizeof(AudioDeviceID));
UInt32 theSize = theNumberDevices * sizeof(AudioDeviceID);
theStatus = AudioHardwareGetProperty
 (kAudioHardwarePropertyDevices,
 &theSize, theDeviceList)
```



318

## IO Process Call-back Functions

```
(*AudioDeviceIOProc) (AudioDeviceID inDevice,
 const AudioTimeStamp* inNow,
 const AudioBufferList* inInputData,
 const AudioTimeStamp* inInputTime,
 AudioBufferList* outOutputData,
 const AudioTimeStamp* inOutputTime,
 void* inClientData);
```

Register an IO Process on a device

```
AudioDeviceAddIOProc(AudioDeviceID inDevice,
 AudioDeviceIOProc inProc, void* inClientData);
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

319

## AudioUnits (Plug-ins)

- \* Components that have sources and destinations
- \* AUs can be connected into AUGraphs
- \* AUs have properties and a property API
- \* AUs have IO call-backs
- \* Call-backs and the “pull” model

— MEDIA ARTS & TECHNOLOGY PROGRAM

320

## AU AudioStreams

```
struct AudioStreamBasicDescription {
 Float64 mSampleRate; // the native sample rate
 UInt32 mFormatID; // the specific encoding type
 UInt32 mFormatFlags; // flags specific to each format
 UInt32 mBytesPerPacket; // the number of bytes in a packet
 UInt32 mFramesPerPacket; // the number of frames in each packet
 UInt32 mBytesPerFrame; // the number of bytes in a frame
 UInt32 mChannelsPerFrame; // the number of channels per frame
 UInt32 mBitsPerChannel; // the number of bits in each channel
};
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

321

## Example: JMC’s Sine Generator

```
OSStatus appIOProc (AudioDeviceID inDevice, const AudioTimeStamp*
 inNow, const AudioBufferList* inInputData, const AudioTimeStamp*
 inInputTime, AudioBufferList* outOutputData, const AudioTimeStamp*
 inOutputTime, void* appGlobals) {
 appGlobalsPtr globals = appGlobals;
 int i;
 double phase = gAppGlobals.phase;
 double amp = gAppGlobals.amp;
 double pan = gAppGlobals.pan;
 double freq = gAppGlobals.freq * 2. * 3.14159265359
 / globals->deviceFormat.mSampleRate;
 int numSamples = globals->deviceBufferSize
 / globals->deviceFormat.mBytesPerFrame;
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

322

## Sine Demo, part 2

```
// assume floats for now...
float *out = outOutputData->mBuffers[0].mData;
for (i=0; i<numSamples; ++i) {
 float wave = sin(phase) * amp;
 phase = phase + freq;
 *out++ = wave * (1.0-pan);
 *out++ = wave * pan;
}
gAppGlobals.phase = phase;
return (kAudioHardwareNoError);
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

323

## CoreAudio Tools, Examples

- \* CODECs
- \* Services
- \* Java
- \* HAL
- \* Public code
- \* CSL, SC2, other CREATE projects

— MEDIA ARTS & TECHNOLOGY PROGRAM

324

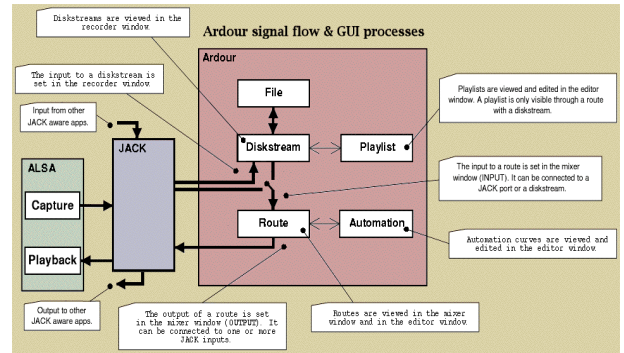
## Sound on Linux

- \* UNIX/Linux sound APIs
  - \* /dev/audio and /dev/dsp
  - \* OSS on \*NIX (now obsolete)
  - \* ALSA (A=Advanced)
  - \* LADSPA (plug-ins)
  - \* Jack (Audio Connection Kit) low-latency server
  - \* RT-Linux kernel patch (not necessary on all releases)
  - \* Audio/specific Linux releases
    - \* PlanetCCRMA and AGNULA
- \* Support for sound cards -- see ALSA support matrix at <http://www.alsa-project.org/alsa-doc>

MEDIA ARTS & TECHNOLOGY PROGRAM

325

## Ardour Block Diagram



MEDIA ARTS & TECHNOLOGY PROGRAM

326

## ALSA SndFile Playback

```
#include <stdio.h>
#include <stdlib.h>
#include <alsa/asoundlib.h>

main (int argc, char *argv[]) {
 int i;
 short buf[128];
 snd_pcm_t *playback_handle;
 snd_pcm_hw_params_t *hw_params;
 snd_pcm_open (&playback_handle, argv[1],
 SND_PCM_STREAM_PLAYBACK, 0);
 snd_pcm_hw_params_malloc (&hw_params);
 snd_pcm_hw_params_any (playback_handle, hw_params);
 snd_pcm_hw_params_set_format (playback_handle,
 hw_params, SND_PCM_FORMAT_S16_LE);
```

MEDIA ARTS & TECHNOLOGY PROGRAM

327

## ALSA Playback (2)

```
snd_pcm_hw_params_set_rate_near (playback_handle,
 hw_params, 44100, 0);
snd_pcm_hw_params_set_channels (playback_handle,
 hw_params, 2);
snd_pcm_hw_params (playback_handle, hw_params);
snd_pcm_hw_params_free (hw_params);
snd_pcm_prepare (playback_handle);

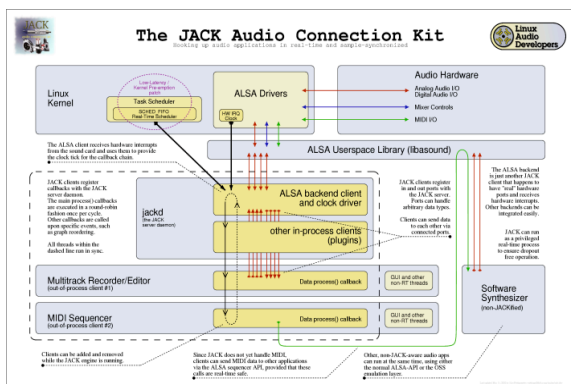
for (i = 0; i < 10; ++i)
 snd_pcm_writei (playback_handle, buf, 128);

snd_pcm_close (playback_handle);
}
```

MEDIA ARTS & TECHNOLOGY PROGRAM

328

## JACK



MEDIA ARTS & TECHNOLOGY PROGRAM

329

## Linux Audio References

- \* Linux audio books
- \* Large-scale tools
  - \* Ardour (DAW)
  - \* Rosegarden (notation editor)
- \* Planet CCRMA
  - \* <http://ccrma.stanford.edu/planetccrma/software>
  - \* Special Linux release
  - \* Many tools ported and supported
  - \* Mailing lists, etc.
  - \* See also AGNULA distribution

MEDIA ARTS & TECHNOLOGY PROGRAM

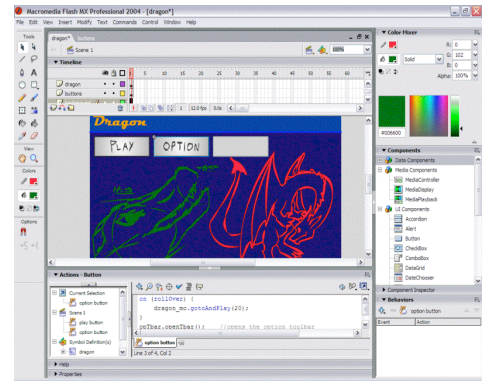
330

## Ardour OpenSource DAW



331

## Topic: Cross-Platform Audio



332

## PortAudio

- \* Phil Burk (SoftSynth) and Ross Benecia, 2000 (based on an open design process started at UCSB)
- \* Music-dsp mailing list (<http://shoko.calarts.edu/~glmrboy/musicdsp/music-dsp.html>)
- \* Media\_api mailing list ([media\\_api@create.ucsb.edu](mailto:media_api@create.ucsb.edu), [http://www.create.ucsb.edu/mailman/listinfo/media\\_api](http://www.create.ucsb.edu/mailman/listinfo/media_api))
- \* PortAudio:
  - \* <http://www.portaudio.com/>
  - \* <http://www.cs.cmu.edu/~music/portmusic/>
- \* Ports to Mac OS, MS Windows, Lunix, SGI, etc.
- \* Relation to PortMIDI effort

333

## PortAudio

## Call-back functions and streams

```
#include "portaudio.h"

// call-back function signature
typedef int (PortAudioCallback)(
 void *inputBuffer, void *outputBuffer,
 unsigned long framesPerBuffer,
 PaTimestamp outTime, void *userData);

// usage example

err = Pa_Initialize();
if(err != paNoError)

 printf("PortAudio error: %s\n", Pa_GetErrorText(err));
```

334

## PortAudio Example: Sawtooth

```
static int paSawtoothCallback(void *inputBuffer, void *outputBuffer, unsigned
 long framesPerBuffer, PaTimestamp outTime, void *userData) {
 paTestData *data = (paTestData*) userData; // cast data struct ptr
 float *out = (float*) outputBuffer; // cast output buf ptr

 for(unsigned i = 0; i < framesPerBuffer; i++) {
 *out++ = data->left_phase; // left (output = phase value)
 *out++ = data->right_phase; // right output
 data->left_phase += 0.01f; // increment phase
 if(data->left_phase >= 1.0f) data->left_phase -= 2.0f; // wrap around
 data->right_phase += 0.03f; // different freq. on the right
 if(data->right_phase >= 1.0f) data->right_phase -= 2.0f;
 }
 return 0; // return value
}
```

335

## PortAudio Streams

```
err = Pa_OpenDefaultStream(
 &stream, // passes back stream pointer */
 0, // no input channels */
 2, // stereo output */
 paFloat32, // 32 bit floating point output */
 44100, // sample rate */
 256, // frames per buffer */
 0, // # buffers, 0 => use minimum */
 paSawtoothCallback, // specify our custom callback */
 &data); // pass our data to callback */

err = Pa_StartStream(stream);
if(err != paNoError) goto error;
```

336

## PortAudio Details

- \* Stream control
- \* Device and CPU queries
- \* Timestamps
- \* Blocking I/O option
- \* PortAudio Tutorial



337

## PortAudio Examples

- \* `pa_tests/pa_devs.c` = print a list of available devices
- \* `pa_tests/pa_minlat.c` = determine minimum latency for your machine
- \* PortAudio ring modulator source



338

## Excerpt from `patest_record.c`

```
static int recordCallback(void *inputBuffer, void *outputBuffer,
 unsigned long numFrames, PaTimestamp outTime, void *userData) {
 SAMPLE *rptr = (SAMPLE*)inputBuffer;
 for(l = 0; l < numFrames; l++) {
 *wptr++ = *rptr++, // copy 2 samples
 *wptr++ = *rptr++, // to elsewhere
 }
}

In main()
err = Pa_OpenStream(&stream,
 Pa_GetDefaultInputDeviceID(),
 data.samplesPerFrame, // stereo input
 PA_SAMPLE_TYPE, NULL, paNoDevice, 0,
 PA_SAMPLE_TYPE, NULL, S_RATE, 1024, // frames / buf
 0, paClipOff, recordCallback, &data);
while(Pa_StreamActive(stream)) // loop until done
 Pa_Sleep(100);
err = Pa_CloseStream(stream); // then close
```



339

## Building and Extending PortAudio

- \* Making the library (see tutorial)
- \* Making the tests
- \* Integration with a program (see CSL)
- \* Porting to a new platform (see advanced notes)



340

## Sndlib Introduction

- \* CCRMA/Stanford, Bill Schottstaedt, 1990-present
- \* C API for `clm` and other packages (CREATE auralizer)
- \* Supports SGI (either audio library), NeXT, Sun, Be, OSS or ALSA (Linux and others), Mac, OS-9, -X, HPUNIX, LinuxPPC, and MS Windows
- \* Blocking I/O model (UNIX I/O)



341

## Sndlib Ports and Functionality

System	SndSine	SndInfo	Audinfo	SndPlay	SndRecord	CLM
NeXT 68k	ok	ok	ok	ok	ok	ok
NeXT Intel	ok	ok	ok	interruptions	runs (*)	ok
SGI old and new AL	ok	ok	ok	ok	ok	ok
OSS (Linux et al)	ok	ok	ok	ok	ok	ok
Be	ok	ok	not written	not written	not written	ok
Mac	ok	ok	ok	ok	ok	ok
Win95	ok	ok	ok	ok	not written	ok
Sun	ok	ok	ok	ok	interruptions	ok
HPUNIX	untested	untested	untested	untested	untested	untried
LinuxPPC	ok	ok	ok	ok	untested (**)	ok
ALSA	ok	ok	ok	ok	ok	ok
Mac OS-X	ok	ok	not written	not written	not written	ok

(\*) I can't find a microphone.

(\*\*) Last I looked, recording was still not supported in this OS.



342

## Sndlib Source Files

- \* io.c (read and write sound file data)
- \* headers.c (read and write sound file headers)
- \* audio.c (read and write sound hardware ports)
- \* sound.c (provide slightly higher level access to the preceding files)
- \* sndlib.h (header for the preceding files)
- \* sndlib2scm.c and sndlib-strings.h (tie preceding into Guile)
- \* clm.c and clm.h (Music V implementation)
- \* clm2scm.c, vct.c and vct.h (tie clm.c into Guile)
- \* old-sndlib.h (old names)

— MEDIA ARTS & TECHNOLOGY PROGRAM

343

## Sndlib Data I/O Functions

```
int mus_sound_open_input (const char *arg)
int mus_sound_open_output (const char *arg, int srates, int chans,
 int data_format, int header_type, const char *comment)
int mus_sound_reopen_output (const char *arg, int type,
 int format, int data_loc)
int mus_sound_close_input (int fd)
int mus_sound_close_output (int fd, int bytes_of_data)
int mus_sound_read (int fd, int beg, int end, int chans,
 MUS_SAMPLE_TYPE **bufs)
int mus_sound_write (int fd, int beg, int end, int chans,
 MUS_SAMPLE_TYPE **bufs)
int mus_sound_seek (int fd, long offset, int origin)
int mus_sound_seek_frame (int fd, int frame)
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

344

## Sndlib Examples

- \* sndinfo.c
- \* sndplay.c
- \* sndrecord.c
- \* CLM and Guile interfaces
- \* Sndlib and the CREATE auralizer
- \* Building sndlib

— MEDIA ARTS & TECHNOLOGY PROGRAM

345

## Excerpt from sndinfo.c

```
mus_sound_initialize();
if (mus_file_probe(argv[1])) {
 date = mus_sound_write_date(argv[1]);
 srates = mus_sound_srates(argv[1]);
 chans = mus_sound_chans(argv[1]);
 samples = mus_sound_samples(argv[1]);
 comment = mus_sound_comment(argv[1]);
 length = (float)samples / (float)(chans * srates);
 loops = mus_sound_loop_info(argv[1]);
 type = mus_sound_header_type(argv[1]);
 format = mus_sound_data_format(argv[1]);
}
fprintf(stdout, "%s\n srates: %d\n chans: %d\n length: %f\n",
 argv[1], srates, chans, length);
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

346

## CREATE Auralizer audio\_io.c

```
void setup_DAC() {
 if (mus_sound_initialize() < 0)
 while("Error: unable to initialize sndlib", 1);
 audio_port = mus_audio_open_output(MUS_AUDIO_DEFAULT,
 sample_rate, 2, MUS_COMPATIBLE_FORMAT, 0);
 ...
}
void output_loop() {
 while (1) {
 // write sample buffer to the DAC
 // This will block until the write's done.
 mus_audio_write(audio_port,
 (char *) (xf_bufptr[xf_bufnum]), buf_len);
 ...
 }
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

347

## Compiling sndplay on Linux

- \* cc -c io.c -O -DLINUX
- \* cc -c audio.c -O -DLINUX
- \* cc -c headers.c -O -DLINUX
- \* cc -c sound.c -O -DLINUX
- \* cc sndplay.c -o sndplay -O -DLINUX audio.o io.o headers.o sound.o -lm
- \* Or
- \* ld -r audio.o io.o headers.o sound.o -o sndlib.a
- \* cc sndplay.c -o sndplay -O -DLINUX sndlib.a -lm

— MEDIA ARTS & TECHNOLOGY PROGRAM

348

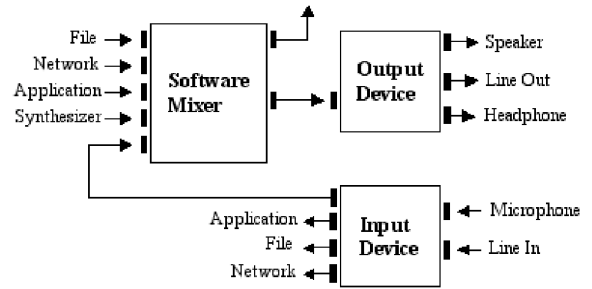
## JavaSound

- \* Low-level object-oriented API for sound and MIDI (Java Media Framework (JMF) is higher-level API)
- \* Intended to support many capabilities and application domains
- \* See API object javadoc file
- \* See Java presentation slides

MEDIA ARTS & TECHNOLOGY PROGRAM

349

## JavaSound Objects



MEDIA ARTS & TECHNOLOGY PROGRAM

350

## JavaSound Packages

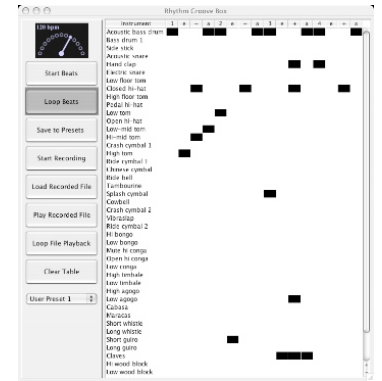
- \* Central packages
  - \* `javax.sound.sampled`
  - \* `javax.sound.midi`
- \* Service provider interfaces
  - \* `javax.sound.sampled.spi`
  - \* `javax.sound.midi.spi`

MEDIA ARTS & TECHNOLOGY PROGRAM

351

## JavaSound Examples

- \* MAT 240A student code
- \* On-line example code



MEDIA ARTS & TECHNOLOGY PROGRAM

352

## LibSndFile

- \* Cross-platform C-language API
- \* UNIX-like API (open, close, read, write)
- \* Uses `SNDFILE` and `SF_INFO` data structs
- \* Open files in a variety of formats (all popular formats supported)
- \* Read/write various sample formats with automatic conversion (e.g., open a WAV and read 32-bit float buffers)
- \* See numerous examples
  - \* <http://www.mega-nerd.com/libsndfile>

MEDIA ARTS & TECHNOLOGY PROGRAM

353

## DASP Plug-in APIs

- \* VST, AudioUnits, JACK, TDM, DirectSnd
- \* Look like PortAudio
  - \* Register call-back function that gets sample ptrs and #-frames
- \* Differences
  - \* GUI methods (use parameter struct?)
  - \* Return label array, range, default, ...
  - \* Return getter/setter functions
  - \* (Optional) Create window, load images
- \* Plug-in Instruments
  - \* Read MIDI, generate samples

MEDIA ARTS & TECHNOLOGY PROGRAM

354

## Review

- \* Music/sound representation, notation, interchange
- \* Sound file formats, history, data structures
- \* Tagged or chunked file formats
- \* Sound I/O APIs
- \* CODE
  - \* CSL PAIO, JackIO, CAIO, VSTIO

MEDIA ARTS & TECHNOLOGY PROGRAM

355

## To Do

- \* Sound I/O API demos
- \* Play/record programs
- \* Streaming I/O to disk
- \* Cross-platform APIs
- \* Audio plug-in APIs

MEDIA ARTS & TECHNOLOGY PROGRAM

356

## What's Next?

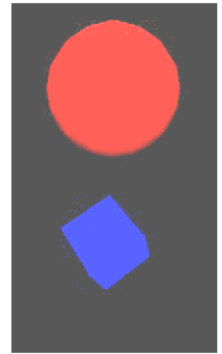
- \* Graphics readings
- \* Audio coding
- \* Final project planning

MEDIA ARTS & TECHNOLOGY PROGRAM

357

## MAT 201B: Topic 5

Computing with  
Still and  
Dynamic  
Images



MEDIA ARTS & TECHNOLOGY PROGRAM

358

## MAT 201B: Topic 5

- \* Computer graphics (CG) introduction and background
- \* Geometry and models of images and scenes
- \* CG representations and languages
- \* CG API examples

MEDIA ARTS & TECHNOLOGY PROGRAM

359

## Readings 5

- \* Graphics File Formats
- \* Overview of OpenGL
- \* Common Image File Formats
- \* Compression, Encoding and Graphics Files
- \* Parsing and Writing QuickTime Files

MEDIA ARTS & TECHNOLOGY PROGRAM

360

## What will we be learning?

- \* What are the theoretical foundations of computer graphics?
- \* What are the most common and useful representations and storage and interchange formats for graphical data?
- \* What are the standard declarative and procedural representations for computer graphics?

— MEDIA ARTS & TECHNOLOGY PROGRAM

361

## Computer Graphics

- \* Background
  - \* Models of images and scenes
  - \* Geometry for computer graphics
- \* CG description languages
- \* CG API examples

— MEDIA ARTS & TECHNOLOGY PROGRAM

362

## Math Topics for Image Representation: Geometry

- \* Coordinate systems (dimensions and quantization of space)
- \* Transformations (descriptions of motion, rotation, scaling, shearing, etc.)
- \* Basic algorithms (line-drawing, polygon-filling, tiling, etc.)
- \* Rendering techniques (Z-plane sorting, hidden surface removal, shading, etc.)

— MEDIA ARTS & TECHNOLOGY PROGRAM

363

## Models of Visual Data

- \* Scene vs. image
- \* Object vs. model
- \* Shape vs. contour
- \* Surface vs. texture
- \* Motion vs. frames
- \* Color vs. pixel value

— MEDIA ARTS & TECHNOLOGY PROGRAM

364

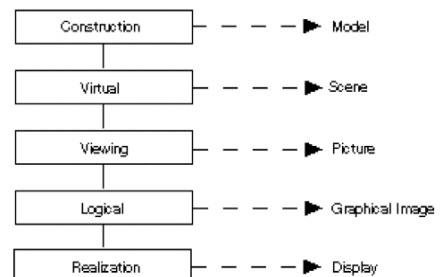
## Image Representation

- \* Models of images and scenes
- \* Structured graphics and raster graphics
- \* Scene descriptions and rendering
- \* Models for dynamic images
- \* Standard formats

— MEDIA ARTS & TECHNOLOGY PROGRAM

365

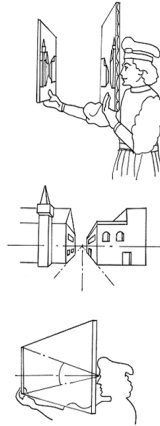
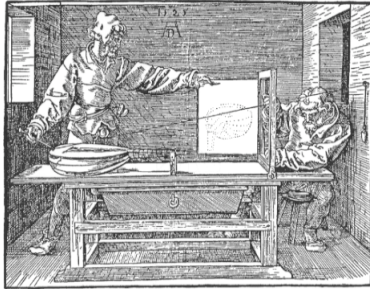
## Computer Graphics Reference Model



— MEDIA ARTS & TECHNOLOGY PROGRAM

366

## Perspective Machines



MEDIA ARTS & TECHNOLOGY PROGRAM

367

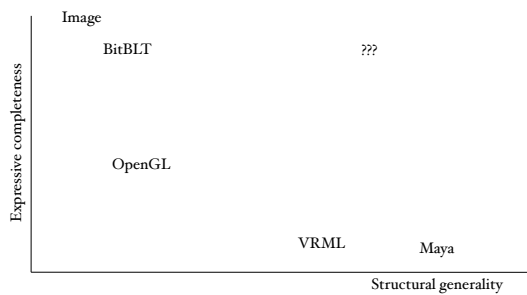
## Model Categories

- \* 2-, 3-D graphical models
  - \* Object vs. scene model
- \* Vector drawing descriptions and languages
- \* Bitmap operational scripts
- \* Page description languages
- \* Rendering API commands

MEDIA ARTS & TECHNOLOGY PROGRAM

368

## Wiggins/Smail Graph Again



MEDIA ARTS & TECHNOLOGY PROGRAM

369

## CG Detail Topics

- \* 2D and 3D spatial models
- \* Cartesian and polar
- \* Vectors, planes, solids
- \* Color, color models and color tables
- \* Textures and mapping
- \* Color and spatial dithering

MEDIA ARTS & TECHNOLOGY PROGRAM

370

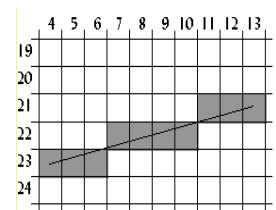
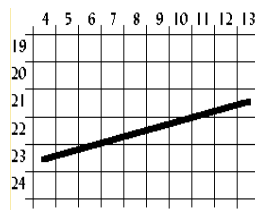
## Geometry for Computers

- \* Mostly Cartesian x@y
- \* Origin in upper-left (as scanned)
- \* Origin in lower-left (1st quadrant)
- \* Where are the points?
  - \* Mid-pixel
  - \* Some corner
- \* Line-drawing algorithms and stroke thickness

MEDIA ARTS & TECHNOLOGY PROGRAM

371

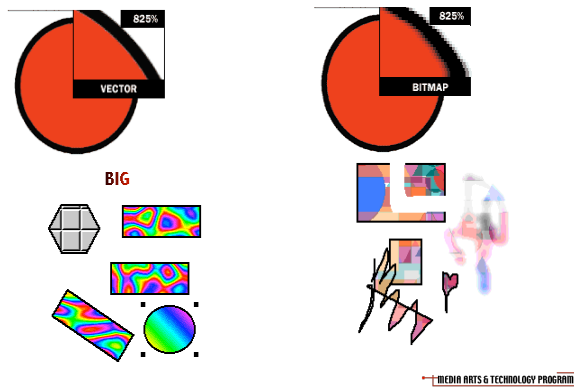
## Line-Drawing



MEDIA ARTS & TECHNOLOGY PROGRAM

372

## Vectors (Draw) vs. Rasters (Paint)



373

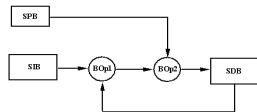
## Representation

- \* Structured graphics display lists
- \* Bitmaps, pixel-maps
- \* Semantics of operations
  - \* Drawing operations
    - \* Scale, order, copy, stroke
  - \* Painting operations
    - \* Mask, fill, erase

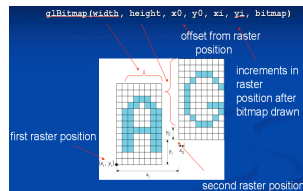
MEDIA ARTS &amp; TECHNOLOGY PROGRAM

374

## BitBLT Operations



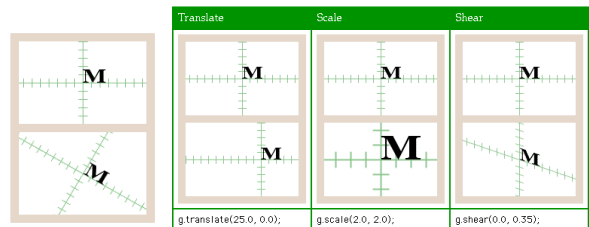
- \* Memory-to-memory copy with shift/mask
- \* Resulting destination pixels are some logical operation of the source, mask, alpha channel, and current destination pixel.
- \* Very simple to do;
- \* Very hard to do fast



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

375

## Transformations



```
Graphics2D g = (Graphics2D)graphics;
...
g.rotate(30.0 * Math.PI / 180.0);
g.setFont(new Font("Serif", Font.BOLD, 24));
g.drawString("M", 0, 0);
```

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

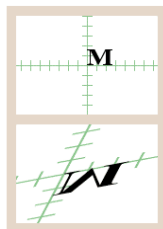
376

## Transformations

```
Graphics2D g = (Graphics2D)graphics;

g.scale(2.5, -1.5);
g.translate(-10.0, 0.0);
g.shear(0.5, 0.15);
g.rotate(10.0 * Math.PI / 180.0);

g.setFont(new Font("Serif", Font.BOLD, 24));
g.drawString("M", 0, 0);
```



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

377

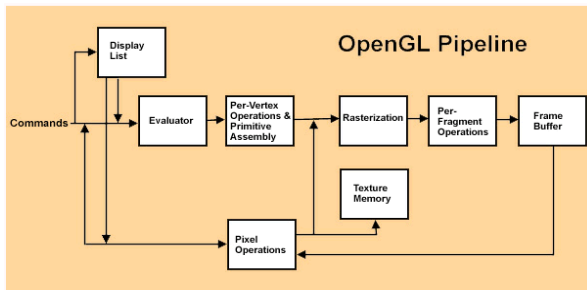
## Rendering and Coordinate Systems

- \* Object Space: coordinates in which an object is defined.
- \* World Space: global system independent of the camera.
  - The interface moves objects from the object space to the world space.
- \* Camera Space: coordinate system with the viewpoint at the origin and the direction of the view along the Z axis.
- \* Screen Space: Normalized after the scene is projected onto a viewing plane.
- \* Raster Space: A two-dimensional space in which the top left corner of the top left pixel of the output image lies at (0,0) and pixel corners lie at non-negative locations.

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

378

## The Rendering Pipeline



MEDIA ARTS & TECHNOLOGY PROGRAM

379

## Rendering Primitives (OpenGL)

Points	individual points	
Lines	pairs of vertices interpreted as individual line segments	
Polygons	boundary of a shape, convex polygons	
Triangles	triple of vertices interpreted as triangles	
Quads	quadruple of vertices interpreted as four-sided polygons	
Line Strip	series of connected line segments	
Line Loop	series of lines, with a segment added between last and first vertex	
Triangle Strip	linked strip of triangles	
Triangle Fan	linked fan of triangles	
Quad Strip	linked strip of quadrilaterals	

MEDIA ARTS & TECHNOLOGY PROGRAM

380

## Structured Graphics Essentials

- \* Primitives
  - \* Line, arc, polygon, text, image
- \* Transformations
  - \* Move, scale, rotate
- \* Grouping and containment
- \* Window composition as display list
  - \* Window, header, layout, application pane
- \* Examples: QuickDraw, Xlib

MEDIA ARTS & TECHNOLOGY PROGRAM

381

## Raster/Image Graphics

- \* (i: It's the input, stupid)
- \* Formats: pixel-maps
- \* Operations
  - \* Draw, fill,
  - \* BitBlt (Bit-wise block logical transfer)
    - \* Source, destination, operation
- \* Structure extraction
  - \* Edge-detection
  - \* Region-building

MEDIA ARTS & TECHNOLOGY PROGRAM

382

## Images, Masks, Opacity and the Alpha Channel

- \* Image combination
- \* "Opaque forms" and masking
- \* The stencil-paint model
- \* Mixed Formats
- \* Apple PICT
- \* Adobe PostScript

MEDIA ARTS & TECHNOLOGY PROGRAM

383

## Alpha Channel

Example: use alpha channel to mask image (1-bit opacity mask)

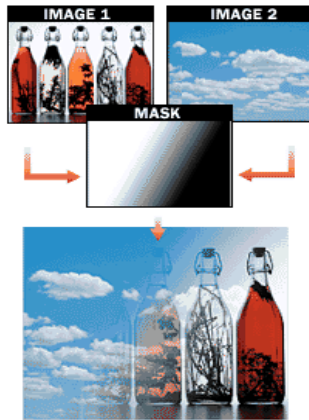


MEDIA ARTS & TECHNOLOGY PROGRAM

384

## Masking

Example: use alpha channel to cross-fade (multi-bit opacity mask)

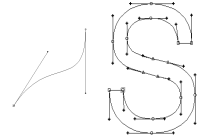
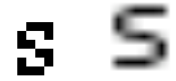


MEDIA ARTS & TECHNOLOGY PROGRAM

385

## Fonts

- \* Raster and stroked fonts
- \* Kerning (inter-character spacing)
- \* Character processing
  - \* Rotation, scaling, shadow, outline
- \* Font storage
- \* Character formats
- \* Kerning tables
- \* Anti-aliasing



MEDIA ARTS & TECHNOLOGY PROGRAM

386

## Color Models

- \* RGB Color
  - \* Red, green, blue -- additive (CRT)
  - \* Values may be ints or floats
- \* CMY Color
  - \* Cyan, magenta, and yellow -- subtractive (gels)
- \* CMY and RGB colors are complementary



MEDIA ARTS & TECHNOLOGY PROGRAM

387

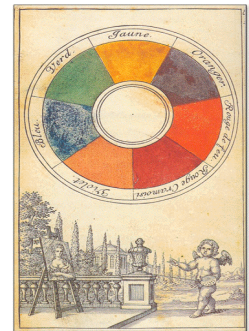
## Color Models 2

### HSV Color

Hue, saturation, and value  
 struct HSVColor {  
   float hue; // Fraction of circle,  
             // red at 0  
   float sat; // 0-1, 0 for gray,  
             // 1 for pure color  
   float val; // 0-1, 0 for black,  
             // 1 for most intensity  
 };

### HSL Color

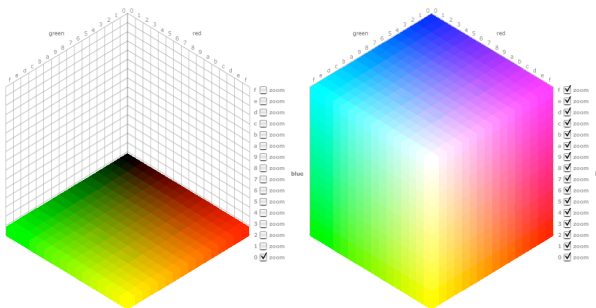
Hue, saturation, and lightness  
 Value = 0-1, 0 for black, 1 for white



MEDIA ARTS & TECHNOLOGY PROGRAM

388

## The Color Cube



<http://www.morecrayons.com/palettes/webSmart/colorcube.php>

MEDIA ARTS & TECHNOLOGY PROGRAM

389

## Color Maps

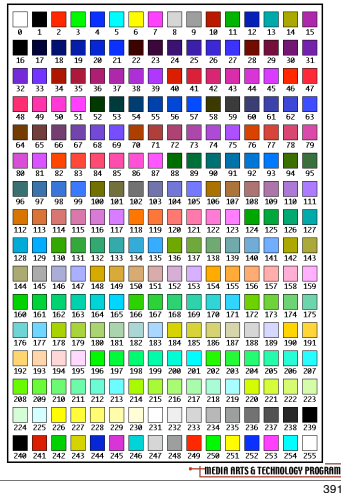
- \* Map between an image's fixed (low) number of colors and a medium's (larger) number.
- \* Size is typically a power of two (pixel depth of image)
- \* There may be one per window or per screen
- \* There are several standards

MEDIA ARTS & TECHNOLOGY PROGRAM

390

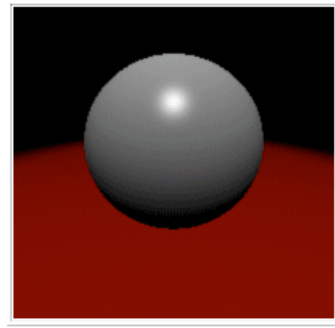
## 256-Place Color Map

(Most humans can differentiate approx. 250 colors.)



391

## Topic: Graphics Formats



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

392

## Graphics Representations

- \* Scene Description (modeling)
- \* Computer display (presentation)
- \* Page description (printing)
- \* Animation (key frame rendering)
- \* Issues
  - \* Units
  - \* Colors
  - \* Structure/pixels

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

393

## Storage and Interchange Formats

### Simple Pixmap Structures

```
int x, y, d, *pixel_data;
```

### Sun RAS format

```
struct rasterfile {
 int ras_magic;
 int ras_width;
 int ras_height;
 int ras_depth;
 int ras_length;
 int ras_type;
 int ras_maptype;
 int ras_maplength;
};
```

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

394

## BMP FILE FORMAT

- \* Images with 1, 4, 8, or 24 bits per pixel
- \* Stored by scan line, bottom to top (!!)
- \* Two incompatible versions of this format, one introduced with OS/2 & Windows 3.0
- \* Resolution-independent (!!)
- \* Hierarchical color map (!!)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

395

## BMP File Header (MS-C Format)

Data	Description
WORD Type;	File type. Set to "BM."
DWORD Size;	Size in DWORDs of the file
DWORD Reserved;	Reserved. Set to zero.
DWORD Offset;	Offset to the data.
DWORD headerSize;	Size of rest of header. Set to 40.
DWORD Width;	Width of bitmap in pixels.
DWORD Height;	Height of bitmap in pixels.
WORD Planes;	Number of Planes. Set to 1.
WORD BitsPerPixel;	Number of bits per pixel.
DWORD Compression;	Compression. Usually set to 0.
DWORD SizeImage;	Size in bytes of the bitmap.
DWORD XPixelsPerMeter;	Horizontal pixels per meter.
DWORD YPixelsPerMeter;	Vertical pixels per meter.
DWORD ColorsUsed;	Number of colors used.
DWORD ColorsImportant;	Number of "important" colors.

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

396

## TIFF - Tagged Image File Format

- \* Basic chunked format
- \* Four TIFF Classes have been defined:
  - \* Class B for bilevel (1-bit) images
  - \* Class G for grayscale images
  - \* Class P for palette color images
  - \* Class R for RGB full color images

— MEDIA ARTS & TECHNOLOGY PROGRAM

397

## GIF

- \* Chunked format
- \* LZW (statistical) compression
- \* Unisys/Compuserve ownership (\$5000 per program that generates GIFs)
- \* Move to PNG!

— MEDIA ARTS & TECHNOLOGY PROGRAM

398

## GIF89a File Structure (chunks)

- \* GIF89a HEADER
- \* LOGICAL SCREEN DESCRIPTOR BLOCK
- \* GLOBAL COLOR TABLE
- \* optional NETSCAPE APPLICATION EXTENSION BLOCK
- \* a stream of graphics, each graphic being composed of:
  - \* optional GRAPHIC CONTROL BLOCK (one for each IMAGE)
  - \* a single IMAGE DESCRIPTOR or PLAIN TEXT BLOCK which can include an optional LOCAL COLOR TABLE for an image and the actual IMAGE or TEXT data table
- \* GIF TRAILER ends the series of images

— MEDIA ARTS & TECHNOLOGY PROGRAM

399

## Animated GIF

- \* Series of GIFs with delays
- \* Compressor does inter-frame relative differentials
- \* Recognizes
  - \* Fades
  - \* Panning
  - \* Scaling
  - \* Other transformations

— MEDIA ARTS & TECHNOLOGY PROGRAM

400

## PNG: Portable Network Graphics

- \* Supports lossless compression
- \* Supports up to 48-bit color or 16-bit gray scale images
- \* Full alpha channel and transparency support
- \* Palette-mapped images to 256 colors
- \* Streamability and progressive display
- \* Hardware and platform independence
- \* W3C Standard

— MEDIA ARTS & TECHNOLOGY PROGRAM

401

## JPEG

- \* JPEG: Joint Photographic Experts Group
- \* JBIG: Joint Bi-level Image experts Group (fax)
- \* Lossy image compression scheme for images to be viewed by humans
- \* Better for natural scenes than text
- \* Trade-offs between compression, loss, and speed (pick any 2 again)

— MEDIA ARTS & TECHNOLOGY PROGRAM

402

## JPEG Formats

- \* JFIF (JPEG File Interchange Format)
  - \* Low-end, pixels only
  - \* Standard on the Web
- \* TIFF/JPEG, aka TIFF 6.0
  - \* High-end, includes annotations
  - \* Used in graphics programs

— MEDIA ARTS & TECHNOLOGY PROGRAM

403

## Structured Graphics Formats

- \* Declarative/Procedural
- \* QuickDraw and PICT format
- \* PostScript format as a language
- \* Hierarchy, naming, ordering
- \* Renderers

— MEDIA ARTS & TECHNOLOGY PROGRAM

404

## Struct. Graphics History

- \* IFIPS (1973) formed two committees to come up with a standard graphics API
  - \* Graphical Kernel System (GKS) – 2D but contained good workstation model
  - \* Core – Both 2D and 3D
- \* GKS adopted as ISO and later ANSI standard (1980s)
  - \* GKS not easily extended to 3D (GKS-3D)
  - \* Far behind hardware development

— MEDIA ARTS & TECHNOLOGY PROGRAM

405

## Struct. Graphics History (2)

- \* Programmers Hierarchical Graphics System (PHIGS)
  - \* Arose from CAD community
  - \* Database model with retained graphics (structures)
- \* X Window System
  - \* DEC/MIT effort, based on Stanford's V and W
  - \* Client-server architecture with low-level graphics
- \* PEX combined the two
  - \* Not easy to use (all the defects of each)

— MEDIA ARTS & TECHNOLOGY PROGRAM

406

## SGI & GL

- \* Silicon Graphics (SGI) revolutionized the graphics workstation by implementing the rendering pipeline in hardware (1982)
- \* To use the system, application programmers used a library called GL
- \* With GL, it was relatively simple to program three dimensional interactive applications

— MEDIA ARTS & TECHNOLOGY PROGRAM

407

## OpenGL

- \* The success of GL lead to OpenGL (1992), a platform-independent API:
  - \* Easy to use
  - \* Close enough to the hardware to get excellent performance
  - \* Focus on rendering
  - \* Omitted windowing and input

— MEDIA ARTS & TECHNOLOGY PROGRAM

408

## OpenGL/GLUT Example

```
#include <glut.h>
void mydisplay(){
 glClear(GL_COLOR_BUFFER_BIT);
 glBegin(GL_POLYGON);
 glVertex2f(-0.5, -0.5);
 glVertex2f(-0.5, 0.5);
 glVertex2f(0.5, 0.5);
 glVertex2f(0.5, -0.5);
 glEnd();
 glFlush();
}
int main(int argc, char** argv) {
 glutCreateWindow("simple");
 glutDisplayFunc(mydisplay);
 glutMainLoop();
}
```

Generate a square on a solid background



MEDIA ARTS & TECHNOLOGY PROGRAM

409

## PostScript Programming/Page-description Language

- \* Derived from Xerox INTERPRESS
- \* Based on FORTH
- \* Stencil-paint imaging model
- \* All positions are arbitrary-precision floats
- \* All paths are splines
- \* Context and transformations
- \* Paths and clipping
- \* Outline Fonts

MEDIA ARTS & TECHNOLOGY PROGRAM

410

## Examples: Draw a box

```
%!
%% Draws a one square inch box and inch in
%% from the bottom left

%% Convert inches -> points (1/72 inch)
/inch {72 mul} def

newpath
1 inch 1 inch moveto
2 inch 1 inch lineto
2 inch 2 inch lineto
1 inch 2 inch lineto
closepath
stroke
showpage
```

MEDIA ARTS & TECHNOLOGY PROGRAM

411

## Use a box as a clipping region

```
%!
% operator box: xcoord ycoord box - Creates one inch box
/box {
 newpath
 moveto
 72 0 rlineto
 0 72 rlineto
 -72 0 rlineto
 closepath
} def
/Times-Roman findfont 30 scalefont setfont
gsave % Save the old clip path
72 72 box % Set up our box
gsave % Don't allow box to be
 stroke % lost after stroke
grestore % Restore the box path
clip % Clip to the box
60 60 moveto
(This is Times-Roman clipped to a box) show
grestore % Get the clip path back
```

MEDIA ARTS & TECHNOLOGY PROGRAM

412

## Rotation and Transformation

```
gsave
100 450 translate % Set the origin
45 rotate % Rotate by 45 degrees
0.5 1 scale % Scale coordinates
0 0 box stroke % Draw box
75 0 moveto
(Everything) show
grestore
```

MEDIA ARTS & TECHNOLOGY PROGRAM

413

## EPS & PDF Formats

- \* File obeys Adobe PostScript Document Structuring Conventions (DSC)
- \* May include 1 or more pages
- \* May contain a low-resolution thumbnail
- \* Compressed PS code and immediate images

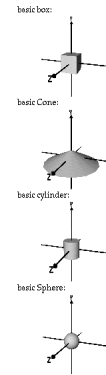
MEDIA ARTS & TECHNOLOGY PROGRAM

414

## VRML

- \* History
  - \* SGI, VRML 1.0, Web3D and VRML 97
- \* Worlds and coordinate spaces
- \* Transforms
- \* Light sources
- \* Hyperlinks and WWW
- \* Object behaviors and scripting

## (Platonic) VRML Primitives



Syntax for a box:

```
Box {
 size 2 2 2 # field SFVec3f
}
```

Syntax for a cone:

```
Cone {
 bottomRadius 1
 Height 3
}
```

Syntax for a cylinder:

```
Cylinder {
 radius 3
 height 10
}
```

Syntax for a sphere:

```
Sphere {
 radius 10
}
```

## VRML Example

```
#VRML V2.0 utf8
Transform {
 children [
 NavigationInfo { headlight FALSE } # We'll add our own light
 DirectionalLight { # First child
 direction 0 0 -1 # Light illuminating the scene
 }
 Transform { # Second child - a red sphere
 translation 3 0 1
 children [
 Shape {
 geometry Sphere { radius 2.3 }
 appearance Appearance {
 material Material { diffuseColor 1 0 0 } # Red
 }
 }
]
 }
]
}
```



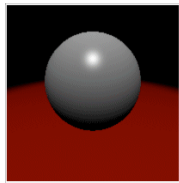
## VRML Example, cont'd

```
Transform { # Third child - a blue box
 translation -2.4 .2 1
 rotation 0 1 1
 children [
 Shape {
 geometry Box {}
 appearance Appearance {
 material Material { diffuseColor 0 0 1 } # Blue
 }
 }
] # end of children for world
}
```

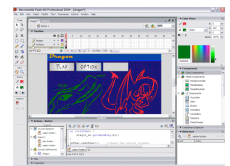
See <http://www.vrml.org/technicalinfo/specifications/vrml97/part1/exampleD.2.wrl>

## RenderMan Example

```
LightSource "spotlight" 1 "from" 0 20 -10 "to" 0 0 0
 "intensity" 500 "coneangle" 0.250
AttributeBegin
 Surface "plastic"
 Color [1 1 1]
 Sphere 1 -1 1 360
AttributeEnd
AttributeBegin
 Translate 0 -1 0
 Surface "plastic"
 Color [1 0 0]
 Polygon "P" [-10 0 -10 -10 0 10 10 0 10 10 0 -10]
 "s" [0 0 1 1]
 "t" [0 1 1 0]
AttributeEnd
```



## Flash



- \* Macromedia (now Adobe) streaming and storage format
- \* Supports both raster and vector representations
- \* Supports video and audio
- \* Manages animation
- \* User input can be managed via actionscript and widgets (see below)
- \* Stored with SWF ("swiff") file format (compact, streamable)

## Flash Tools

- \* Macromedia Flash suite
- \* Open-source readers/renderers: FLIRT, GPLFlash, OpenLaszlo and JSFL, JavaScript-to-Flash API (see OSFlash.org)
- \* Plug-in renderers for most web browsers
- \* Stand-alone execution engine

MEDIA ARTS & TECHNOLOGY PROGRAM

421

## Scalable Vector Graphics: SVG

Text- (XML-) based

Blessed by the W3C and Adobe (open spec.)

Not as flexible or compact as Flash

SVG Example (draw a box)

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="500" height="500">
 <rect x="111" y="78" width="187" height="160"
 style="fill:rgb(192,192,255);
 stroke:rgb(0,0,128);
 stroke-width:1"/>
</svg>
```

MEDIA ARTS & TECHNOLOGY PROGRAM

422

## Image/Video Format Standards

Format name	Lines/frame	Pixels/line	Bits/Pixel
FAX	2200	1700	1
VGA	480	640	8
XVGA	768	1024	24

Format name	Lines/frame	Pixels/line	Frames/s	Interlaced	SS scheme	Aspect ratio
CIF	288	352		N	4:2:0	4:3
QCIF	144	176		N	4:2:0	4:3
SOQIF	128	96		N	4:2:0	4:3
SIF-525	240	352	30	N	4:2:0	4:3
SIF-625	288	352	25	N	4:2:0	4:3
NTSC	486	720	29.97	Y	4:2:2	4:3
PAL	576	720	25	Y	4:2:2	4:3
HDTV	720	1280	59.94	N	4:2:0	16:9
HDTV	1080	1920	29.97	Y	4:2:0	16:9

- **Example:** what is the bit-rate for interlaced HDTV format?  
 $N_l = 1080$  lines per frame,  $N_p = 1920$  pixels per line,  $R_F = 29.97$  frames/s  
frames per second, 12 bits per pixels (luminance + subsampled chrominances):  $N_l N_p R_F \cdot 12 = 745,749,504$  bits/s.

MEDIA ARTS & TECHNOLOGY PROGRAM

423

## Other formats

- \* ...so many to choose from
  - \* 1100-page book (<http://www.oreilly.com/catalog/gffcd>)
  - \* Web references
- \* QuickTime
- \* New image formats
  - \* Geographical data
- \* New OO DisplayList formats
  - \* VR

MEDIA ARTS & TECHNOLOGY PROGRAM

424

## File Format Convertors

- \* All can import and save many formats
- \* GraphicConverter -- also has many transformations: scaling, dithering, etc.
- \* Xv
- \* Debabelizer
- \* Netpbm
- \* many others

MEDIA ARTS & TECHNOLOGY PROGRAM

425

## Container Formats

- \* An extrapolation from tagged formats
- \* Allow mixed use of audio, video, text streams, other content media
- \* Generally support many formats and codecs (as supported by renderers)
- \* Examples
  - \* QuickTime
  - \* AVI/WMV/MOVI

MEDIA ARTS & TECHNOLOGY PROGRAM

426

## AVI Movie File Format

### Specialization of RIFF

'RIFF' (4 byte file length) 'AVI '  
 // file header (a RIFF form)  
 'LIST' (4 byte list length) 'hdrl'  
 // list of headers for AVI file  
 The 'hdrl' list contains:  
 'avih' (4 byte chunk size) (data)  
 // the AVI header (a chunk)  
 'strl' lists of stream headers for each stream (audio, video, etc.) in the AVI file.  
 For an AVI file with one video and one audio stream:  
 'LIST' (4 byte list length) 'strl'  
 // video stream list (a list)

MEDIA ARTS & TECHNOLOGY PROGRAM

427

## AVI Video Stream

The video 'strl' list contains:  
 'strh' (4 byte chunk size) (data)  
 // video stream header (a chunk)  
 'strf' (4 byte chunk size) (data)  
 // video stream format (a chunk)  
 'LIST' (4 byte list length) 'strl'  
 // audio stream list (a list)  
 The audio 'strl' list contains:  
 'strh' (4 byte chunk size) (data)  
 // audio stream header (a chunk)  
 'strf' (4 byte chunk size) (data)  
 // audio stream format (a chunk)  
 'JUNK' (4 byte chunk size) (data - usually all zeros)  
 // an OPTIONAL junk chunk to align on 2K byte boundary  
 'LIST' (4 byte list length) 'movi'  
 // list of movie data (a list)  
 The 'movi' list contains the actual audio and video data.

MEDIA ARTS & TECHNOLOGY PROGRAM

428

This 'movi' list contains one or more ...  
 'LIST' (4 byte list length) 'rec '  
 // list of movie records (a list)

'#wb' (4 byte chunk size) (data)  
 // sound data (a chunk)  
 '##dc' (4 byte chunk size) (data)  
 // video data (a chunk)  
 '##db' (4 byte chunk size) (data)  
 // video data (a chunk)

A 'rec ' list contains the audio/video data for a single frame.

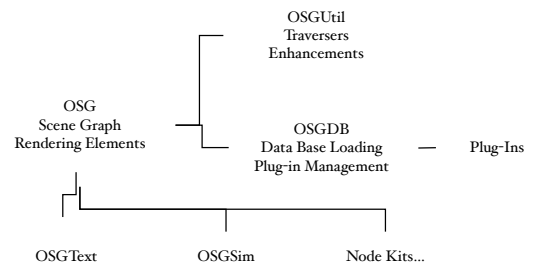
'#wb' (4 byte chunk size) (data)  
 // sound data (a chunk)  
 '##dc' (4 byte chunk size) (data)  
 // video data (a chunk)  
 '##db' (4 byte chunk size) (data)  
 // video data (a chunk)

## MOVI List

MEDIA ARTS & TECHNOLOGY PROGRAM

429

## Topic: Graphics APIs



MEDIA ARTS & TECHNOLOGY PROGRAM

430

## Graphics APIs

- \* Cross-platform
  - \* Java2D/3D
  - \* OpenGL
- \* Platform-specific
  - \* MS-Windows: DirectX
  - \* Macintosh: Quartz
  - \* Linux: X

MEDIA ARTS & TECHNOLOGY PROGRAM

431

## 2D vs. 3D Graphics

- \* Page description vs. scene description
  - \* Display list vs. scene graph
  - \* Fill pattern vs. texture
  - \* Lighting and viewing in 3D
- \* 2D primitives vs. 3D primitives
- \* Rendering issues
- \* GUI issues and behaviors
- \* Frame animation vs. VE (update/cull/draw)

MEDIA ARTS & TECHNOLOGY PROGRAM

432

## Java2D

- \* Geometry
- \* Graphics and Graphics2D objects
- \* Shape objects
- \* Operations on Graphics2D

— MEDIA ARTS & TECHNOLOGY PROGRAM

433

## Java Graphics Packages

- \* java.awt (context, paint)
- \* java.awt.color (profiles)
- \* java.awt.font (attributes, metrics, measurers, layouts)
- \* java.awt.geom (2D transform, shapes)

— MEDIA ARTS & TECHNOLOGY PROGRAM

434

## Basic Geometry

- \* Coordinates are between pixels
- \* 2D affine transforms provided
- \* Shape interface (Area): bounds, containment
- \* Lines: x/y, geometrical functions, drawing
  - \* Float/double subclasses
- \* Rectangular shapes: frame
  - \* Arc, ellipse, rectangle, etc.
- \* Others
  - \* General path, cubic/quadratic splines

— MEDIA ARTS & TECHNOLOGY PROGRAM

435

## Graphics (Context) Object

- \* Has:
  - \* The component (medium) object on which to draw.
  - \* A translation origin for rendering and clipping
  - \* The current clip shape
  - \* The current color
  - \* The current font
  - \* The current logical pixel operation function (XOR or Paint)
  - \* The current XOR alternation color

— MEDIA ARTS & TECHNOLOGY PROGRAM

436

## Graphics (Context) Object

- \* Does:
  - \* Getters/setters for all aspects
  - \* Draw (line, arc, image, chars, etc.)
  - \* Fill (shape)
  - \* Constructor takes no args
- \* It's abstract, Graphics2D has all the working implementations!

— MEDIA ARTS & TECHNOLOGY PROGRAM

437

## Basic Usage

```
public void paintComponent(Graphics g) {
 super.paintComponent(g);
 Graphics2D aG2D = (Graphics2D) g;
 // do something with aG2D
 Ellipse2D.Double circle = // NB: scoped class name
 new Ellipse2D.Double(x, y, diam, diam);
 aG2D.fill(circle);
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

438

## Graphics2D Objects

- \* Extend Graphics: formalize shape (stroking), text (glyph vectors), and image (with transforms) operations
- \* Aspects: paint, font, stroke, transform, composite, clip
- \* Rendering hints and antialiasing renderers

MEDIA ARTS & TECHNOLOGY PROGRAM

439

## Java2D Examples

- \* Zip file from O'Reilly
- \* Marty Hall's tutorial examples
- \* Import `javax.swing.*`, `java.awt.*`, `java.awt.geom.*`
- \* Extend `JComponent` or `JPanel`
- \* Implement
  - \* `public static void main(String[] args)`
  - \* `public void paintComponent(Graphics g)`

MEDIA ARTS & TECHNOLOGY PROGRAM

440

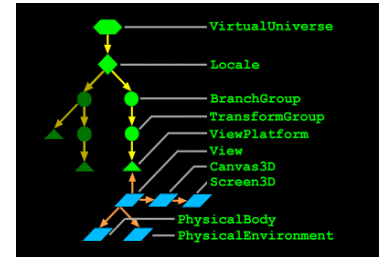
## Also in AWT (see later)

- \* `BorderLayout`: composition of components (n/e/s/w)
- \* `Window`: on-screen container
- \* `Frame`: top-level window
- \* `Menu` (bar, component, item)
- \* `Panel` (container)
- \* `Scrollbar`

MEDIA ARTS & TECHNOLOGY PROGRAM

441

## Java 3D



- \* 3D scene graphs and content
  - \* View branch: controls
  - \* Content branch: lights, shapes

MEDIA ARTS & TECHNOLOGY PROGRAM

442

## The javax.media.j3d Package

- \* `Canvas`, `Screen3D`, `BoundingBox`, `Clip`
- \* `VirtualUniverse`, `SceneGraphObject`, `Node`, `Leaf`, `Raster`, `LOD switch`
- \* `Point/Line/Triangle/QuadArray`
- \* `Font3D`, `Text3D`
- \* `Texture`, `Material`
- \* `Point/Directional/Ambient-Light`, `Fog`
- \* `Point/Cone/Background-Sound`, `Soundscape`

MEDIA ARTS & TECHNOLOGY PROGRAM

443

## Related Java3D Packages

- \* `com.sun.j3d.utils.geometry`
  - \* `Box`, `Cone`, `Cylinder`, `Sphere`, etc.
- \* `com.sun.j3d.utils.universe`
  - \* `Viewer`, `SimpleUniverse`
- \* Audio utilities, engines
- \* Applet utilities
- \* Loaders, parsers
- \* Behavior utilities: input

MEDIA ARTS & TECHNOLOGY PROGRAM

444

## Java3D Hello World

```
public class Hello3d {
 public Hello3d() {
 SimpleUniverse uni = new SimpleUniverse();
 BranchGroup group = new BranchGroup();
 group.addChild(new ColorCube(0.3));
 uni.getViewingPlatform().setNominalViewingTransform();
 uni.addBranchGraph(group);
 }
 public static void main(String[] args) {
 new Hello3d();
 }
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

445

## Advanced Java3D Examples

- \* Scene construction
- \* Textures and materials
- \* Lighting
- \* Viewing
- \* Java3D as GUI
- \* Behaviors
- \* Virtual environments and games

— MEDIA ARTS & TECHNOLOGY PROGRAM

446

## OpenGL

- \* Low-level graphics API
- \* Shapes are collections of vertices
- \* Points, lines, polygons, images
- \* Display lists can be stored and parameterized
- \* Built for C/C++; bindings also for Java, Tcl, Ada, Smalltalk, and FORTRAN
- \* Mesa is open-source implementation
- \* Window-system and GUI-neutral

— MEDIA ARTS & TECHNOLOGY PROGRAM

447

## OpenGL draw() Function

```
void draw(void) {
 glColor3f(1.0, 1.0, 1.0); // set color
 glBegin(GL_POLYGON); // create square
 glVertex2f(-0.5, -0.5);
 glVertex2f(-0.5, 0.5);
 glVertex2f(0.5, 0.5);
 glVertex2f(0.5, -0.5);
 glEnd(); // end square
 glFlush(); // display it
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

448

## OpenGL/TK main()

```
void main(int argc, char **argv) {
 tkInitWindow("Hello World")
 glClearColor(0.0, 0.0, 0.0, 0.0);
 tkExposeFunc(expose); // user-defined
 tkReshapeFunc(reshape);
 tkDisplayFunc(draw); // our draw fcn
 tkMouseDownFunc(mouse_d); // e.g.
 tkExec();
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

449

## OpenSceneGraph

- \* A C++ API built on OpenGL for
  - \* Scene management
  - \* Graphics rendering optimization
- \* Cross-platform
- \* Multi-threaded, distributed, multi-user
- \* Windowing-system-agnostic
- \* Open source
- \* Single display performance
- \* Multi display scalability

— MEDIA ARTS & TECHNOLOGY PROGRAM

450

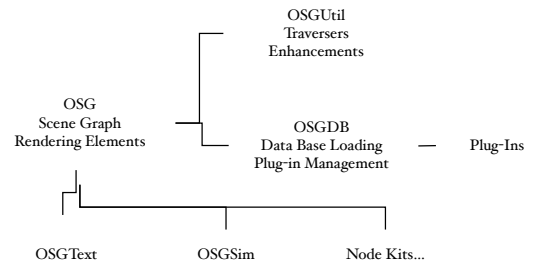
## Core OSG Classes

- \* Helper classes: memory management, maths classes
- \* `osg::Nodes`: the internal nodes in the scene graph
- \* `osg::Drawables`: leaves of the scene graph
- \* `osg::State*`: encapsulate OpenGL state
- \* Traversers/visitors: traversing and operations on the scene

— MEDIA ARTS & TECHNOLOGY PROGRAM

451

## OSG Functional Groups



— MEDIA ARTS & TECHNOLOGY PROGRAM

452

## OpenSceneGraph Example

```

osgProducer::Viewer viewer;
osg::Group* root = new osg::Group();
osg::Geode* pyrGeode = new osg::Geode();
osg::Geometry* pyrGeom = new osg::Geometry();
pyrGeode->addDrawable(pyrGeom);
root->addChild(pyrGeode);
osg::Vec3Array* pyrVert = new osg::Vec3Array;
pyrVert->push_back(osg::Vec3(0, 0, 0));
// add more here...
pyrGeom->setVertexArray(pyrVert);

```

— MEDIA ARTS & TECHNOLOGY PROGRAM

453

## OpenSceneGraph Example (2)

```

viewer.setUpViewer(osgProducer::Viewer
 ::STANDARD_SETTINGS);
viewer.setSceneData(root);
viewer.realize();
while(!viewer.done()) {
 viewer.sync();
 viewer.update();
 viewer.frame();
}

```

— MEDIA ARTS & TECHNOLOGY PROGRAM

454

## OpenSceneGraph Applications

- \* 3D rendering from various (i.e., all known) formats
- \* Virtual environments (immersion, simulation)
- \* Visualization (algorithmic worlds)
- \* Gaming platforms (behaviors, distribution)

— MEDIA ARTS & TECHNOLOGY PROGRAM

455

## Other OpenGL Wrappers

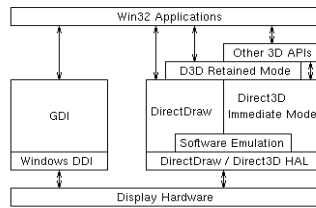
- \* OpenGL-based APIs
  - \* QuartzGL
  - \* OglExt
  - \* Graphic card APIs
  - \* VRML-to-OpenGL
- \* Other language bindings
  - \* JOGL
- \* OpenGL and Direct3D
  - \* OGRE
  - \* DirectX OpenGL Wrapper

— MEDIA ARTS & TECHNOLOGY PROGRAM

456

## Graphics on MS-Windows

- \* Direct3D
  - \* HAL, HEL
- \* System.Drawing
  - \* Geometry
  - \* Primitives
  - \* Graphics (context) class
  - \* Color
  - \* Font, Text, Measurers



— MEDIA ARTS & TECHNOLOGY PROGRAM

457

## Windows Graphics in C#

```
using System.Drawing;
public class Hello:Form {
 public Hello() {
 this.Paint += new PaintEventHandler(f1_paint);
 }
 private void f1_paint(object s,PaintEventArgs e) {
 Graphics g = e.Graphics;
 g.DrawRectangle(new Pen(Color.Pink, 3), 20, 20,
 150, 100);
 g.DrawString("Hello C#", new Font("Verdana", 20),
 new SolidBrush(Color.Tomato), 40, 40);
 }
 public static void Main() {
 Application.Run(new Hello());
 }
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

458

## MacOS Graphics

- \* Aqua GUI
- \* Application frameworks
  - \* Carbon
  - \* Cocoa
- \* Quartz graphics
  - \* QuickDraw heritage
  - \* PDF for documents
  - \* OpenGL & QuickTime for scenes

— MEDIA ARTS & TECHNOLOGY PROGRAM

459

## Quartz Example (PathDemo)

```
void rectangles(CGContextRef gc, NSRect rect) {
 int i;
 int w = rect.size.width;
 int h = rect.size.height;
 for (i = 0; i < 20; i++) {
 CGContextSetRGBFillColor(gc, 1, 0, 0, 0);
 CGContextFillRect(gc, CGRectMake(
 rand()%w, rand()%h,
 rand()%w, rand()%h));
 }
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

460

## OpenGL Example in ObjectiveC

```
- (void) drawRect: (NSRect) rect {
 glViewport(0, 0, (GLsizei) rect.size.width,
 (GLsizei) rect.size.height);
 GLfloat clear_color[4] = { 0.0f, 0.0f, 0.0f, 0.0f };
 clear_color[color_index] = 1.0f;
 glClearColor(clear_color[0], clear_color[1],
 clear_color[2], clear_color[3]);
 glClear(GL_COLOR_BUFFER_BIT
 + GL_DEPTH_BUFFER_BIT
 + GL_STENCIL_BUFFER_BIT);
 [[self openGLContext] flushBuffer];
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

461

## Linux/UNIX Graphics

- \* X history (V, W, 1986: X10 std)
- \* X11R6 graphics library
- \* X Protocol
  - \* Graphical operations
  - \* Window services
  - \* Input handling
  - \* Text and fonts
- \* X GUI (L&F) & application toolkits
  - \* Interviews, TK, Motif, OpenLook, Andrew

— MEDIA ARTS & TECHNOLOGY PROGRAM

462

## X Example Functions

```
Display *dsp = XOpenDisplay(NULL);
Window win = XCreateSimpleWindow(dsp, ...);
XMapWindow(dsp, win);
GC gc = XCreateGC(dsp, win, 0, NULL);
XSetForeground(dsp, gc, black);
XDrawLine(dsp, win, gc, 10, 10, 190, 190);
XSelectInput(dsp, win, ButtonPressMask | ButtonReleaseMask);
XEvent evt;
do {
 XNextEvent(dsp, &evt); // calls XFlush()
} while(evt.type != ButtonRelease);
```

MEDIA ARTS & TECHNOLOGY PROGRAM

463

## Other Graphics Frameworks

- \* Many others
  - \* CORE, GKS
  - \* QuickDraw
  - \* Hardware-specific (SGI, G-card)
  - \* Language-specific (LISP, Smalltalk, etc.)
- \* See also
  - \* Video APIs
  - \* VR systems
  - \* Gaming platforms

MEDIA ARTS & TECHNOLOGY PROGRAM

464

## Review

- \* Geometry for computer graphics
- \* History and background
- \* Representations and file formats
- \* APIs for CG

MEDIA ARTS & TECHNOLOGY PROGRAM

465

## To Do

- \* Skills with graphical utilities (draw, paint, text, file format conversion, etc.)
- \* Compile and extend CG demos of cross-platform and platform-specific APIs
- \* Try higher-level APIs such as OSG

MEDIA ARTS & TECHNOLOGY PROGRAM

466

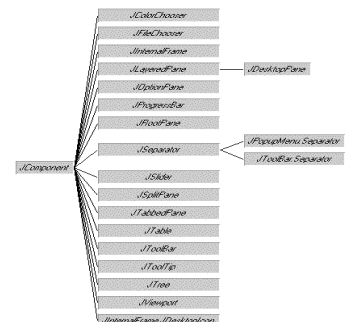
## Coding

- \* Using graphics APIs
  - \* OpenGL/GLUT
  - \* Qt
  - \* IB/Aqua
  - \* Swing
- \* Multi-output graphics
  - \* Render to screen/page
- \* Image/model import/export

MEDIA ARTS & TECHNOLOGY PROGRAM

467

## MAT 201B: Topic 6



APIs and Tools for Interaction  
and GUI Construction

MEDIA ARTS & TECHNOLOGY PROGRAM

468

## APIs and Tools for GUIs

- \* GUI frameworks and support
- \* Window systems and composition
- \* Widget sets and hierarchies
- \* GUI tools and GUIDEs
- \* Cross-platform APIs and tools

— MEDIA ARTS & TECHNOLOGY PROGRAM

469

## Readings 6

- \* QT 3.3 Whitepaper
- \* Eclipse Tutorial
  - \* Basic SWT Widgets
- \* References
  - \* Platform-specific
  - \* Cross-platform
  - \* Language-specific

— MEDIA ARTS & TECHNOLOGY PROGRAM

470

## What will we be learning?

- \* How does an operating system manage applications and their windows?
- \* How is a look and feel implemented?
- \* What tools are there for building interactive WIMP-style applications?
- \* What cross-platform support exists?

— MEDIA ARTS & TECHNOLOGY PROGRAM

471

## GUI Frameworks

- \* Interactive applications
- \* Graphics for GUIs
- \* GUI class hierarchies
- \* GUI tools
- \* GUI APIs

— MEDIA ARTS & TECHNOLOGY PROGRAM

472

## Interactive Applications

- \* Batch model: input, process, output
- \* Simulation model: update system, display view
- \* Interpreter model: shell, read/eval/print loop
- \* Interactive model: event processors and/or processing thread(s) (MVC)
- \* Browser content delivery: request, DB I/O, page generation
- \* Application delivery
  - \* Application platform (window system, browser)
  - \* Control/event delegation model

— MEDIA ARTS & TECHNOLOGY PROGRAM

473

## Graphics for GUIs

- \* Window systems and screen mgmnt
- \* Hierarchy and containment: part-whole component trees of widgets and layout info
- \* Graphical constraints and layout
  - \* Layout frames -- e.g., inset 33% + 3pixels
  - \* Auto-align/distribute

— MEDIA ARTS & TECHNOLOGY PROGRAM

474

## Window Systems

- \* Manage screen real estate
  - \* Display list of overlapping windows
  - \* Propagation of “damage” events
- \* Handle and delegate input events
  - \* List of controllers and ordering
- \* May or may not be linked to a single look and feel and GUI API (often is)
- \* May or may not be considered part of (inseparable from) the OS (often is)

— MEDIA ARTS & TECHNOLOGY PROGRAM

475

## GUI Framework

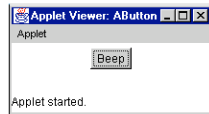
- \* Look and feel of default widgets
- \* Application architecture, MVC model
- \* API for developers
  - \* Widget (control) set
  - \* Event-handling (call-back, signal) framework

— MEDIA ARTS & TECHNOLOGY PROGRAM

476

## A Simple Example (in Java)

```
import java.awt.Button;
import java.applet.Applet;
```



```
public class AButton extends Applet {
 public void init() {
 Button beepButton = new Button("Beep");
 /* this-> */ add(beepButton);
 beepButton.addActionListener(new Beeper());
 }
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

477

## Java Event Handler

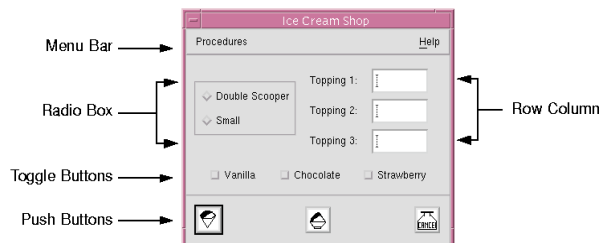
```
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.Component;
```

```
public class Beeper implements ActionListener {
 public void actionPerformed(ActionEvent event) {
 Component c = (Component)event.getSource();
 c.getToolkit().beep();
 }
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

478

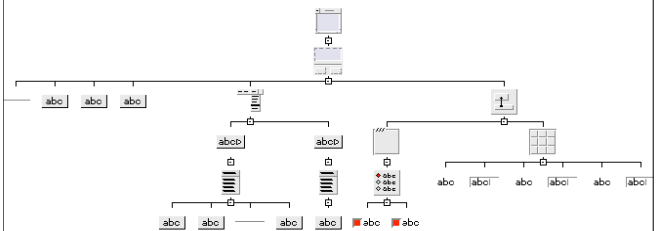
## More Complex Example (Motif)



— MEDIA ARTS & TECHNOLOGY PROGRAM

479

## Composition of the GUI



- \* Widget instance composition hierarchy

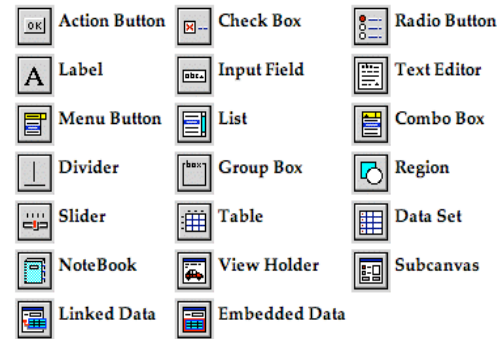
— MEDIA ARTS & TECHNOLOGY PROGRAM

480

## GUI Class Hierarchies

- \* Model/View/Controller design pattern
  - \* Model is your code
  - \* View/controller pair often merged into:
    - \* Widget (reusable, parameterized), and
    - \* Event handler (your code)
- \* Container class
  - \* Building node/leaf hierarchies
- \* Pane/view class
  - \* Rectangle with a parent link
- \* Event class (?)

## Widget Set Example (Smalltalk/Wrapper)



## Categories of Widgets

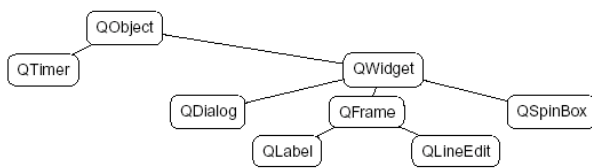
- \* Containers/Hierarchy: pane, window
- \* Inputs: button, button group, selection list, slider, read/write text field, menu item, menu
- \* Outputs: string, gauge, read-only text field
- \* Composites: radio button group, table, menu bar, MDI
- \* Grouping: outline, notebook, layout frame, scroller, splitter, pull-out

## Categories of Widgets (2)

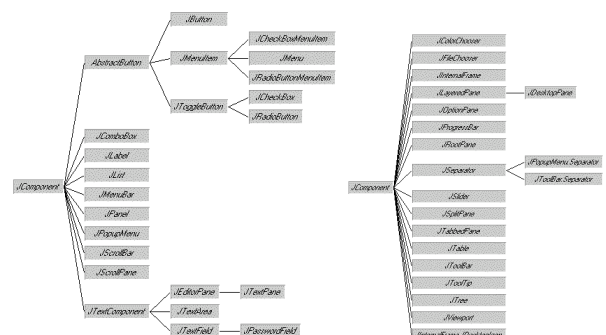
- \* Meta-control: scroll bar (panning view), subview tab, notebook widget
- \* Visual: label, line, box, image
- \* Dialogs: FillInTheBlank, modal, non-blocking, warning notifier, file selector
- \* Complex: list view, tree view, table view
- \* Default apps: text editor, draw, spreadsheet, document model
- \* Advanced: drag'n'drop handling, embedded application, HTML, XML integration

## Simple Widget Hierarchy (Qt)

- \* Basic widget classes



## Widget Hierarchy (Swing)



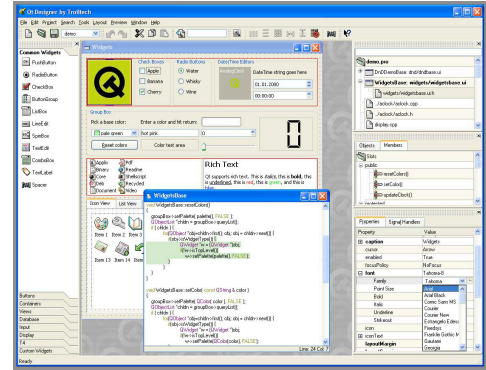
## Widget Hierarchy (GTK, Partial)

- GFC::Gdk::Widget
  - GFC::Gdk::Calendar
  - GFC::Gdk::Container
    - GFC::Gdk::Bin
      - GFC::Gdk::Alignment
      - GFC::Gdk::Button
        - GFC::Gdk::ColorButton
        - GFC::Gdk::FontButton
        - GFC::Gdk::ToggleButton
          - GFC::Gdk::CheckButton
          - GFC::Gdk::RadioButton
      - GFC::Gdk::ComboBox
        - GFC::Gdk::ComboBoxEntry
          - GFC::Gdk::ComboBoxEntryText
          - GFC::Gdk::ComboBoxText
      - GFC::Gdk::EventBox
      - GFC::Gdk::Expander
      - GFC::Gdk::Frame
        - GFC::Gdk::AspectFrame
      - GFC::Gdk::HandleBox
      - GFC::Gdk::Item
        - GFC::Gdk::MenuItem
          - GFC::Gdk::CheckMenuItem
          - GFC::Gdk::RadioMenuItem
          - GFC::Gdk::ImageMenuItem
          - GFC::Gdk::SeparatorMenuItem
          - GFC::Gdk::TearOffMenuItem
- GFC::Gdk::Box
  - GFC::Gdk::ButtonBox
    - GFC::Gdk::HButtonBox
    - GFC::Gdk::VButtonBox
  - GFC::Gdk::HBox
    - GFC::Gdk::StatusBar
  - GFC::Gdk::VBox
    - GFC::Gdk::ColorSelection
    - GFC::Gdk::FileChooserWidget
    - GFC::Gdk::FontSelection
    - GFC::Gdk::GammaCurve
  - GFC::Gdk::Fixed
  - GFC::Gdk::Layout
  - GFC::Gdk::MenuShell
    - GFC::Gdk::Menu
    - GFC::Gdk::MenuBar
  - GFC::Gdk::Notebook
    - GFC::Gdk::HPaned
    - GFC::Gdk::VPaned
  - GFC::Gdk::Socket
  - GFC::Gdk::Table
  - GFC::Gdk::TextView
  - GFC::Gdk::ToolBar
  - GFC::Gdk::TreeView

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

487

## Topic: GUI Tools & IDEs



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

488

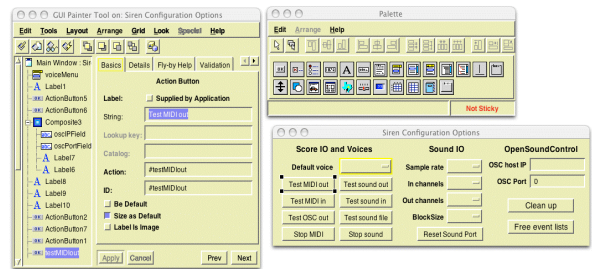
## GUI Development Tools

- \* Visual layout
  - \* Widget selection, placement
  - \* Visual constraints and scaling behavior
- \* Widget property dialogs
  - \* Naming, callbacks, order, visual properties, keyboard behavior, tabbing
- \* Linking widgets to the model
- \* Code generation or resource linking

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

489

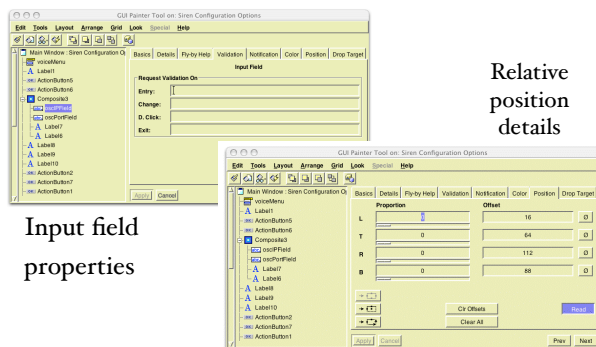
## GUI Builder Tool (VisualWorks Smalltalk, Wrapper Fmwrk)



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

490

## Widget Property Sheets (VisualWorks)



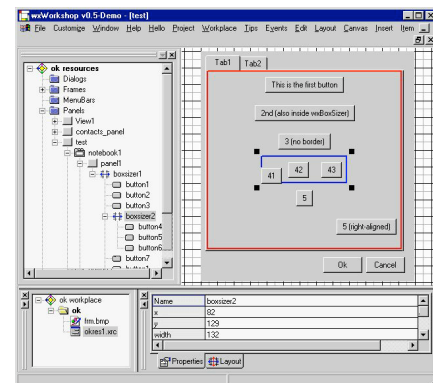
Relative position details

Input field properties

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

491

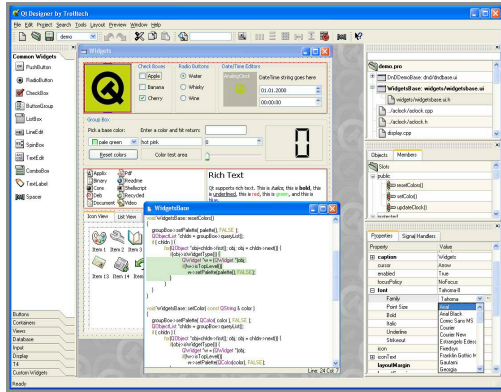
## GUI Builder (WxWorkshop)



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

492

## GUI Builder (Qt Designer)



493

## Differences Between GUIDEs

- \* Level of integration into the IDE
  - \* High: VisualStudio, VisualWorks
  - \* Separated: InterfaceBuilder
- \* Ease of reuse or integration of existing GUIs
- \* Some generate code; some generate opaque resource files
- \* Provision for composition of GUIs
- \* Enforcement of layout guidelines
- \* How (if) call-back stubs are generated

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

494

## What's a Look and Feel?

- \* Windows
  - \* Borders, shading
  - \* Border decorators, buttons, scrollers
  - \* Subview composition and editing
- \* Widgets
  - \* Widget graphics and activation
  - \* Event mapping, kbd shortcuts
- \* Controller feel: statefulness, gestures, feedback

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

495

## GUI Guidelines

- \* Standard look
  - \* Widget placement, dialog construction
  - \* View layout, panes, editing
- \* Standard Feel
  - \* Dialog process flow
  - \* Application menu bar
  - \* Keyboard command shortcuts
  - \* Feedback about system activity

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

496

## Advanced GUI Topics

- \* Application composition and GUI reuse
- \* Composite widgets: notebooks, view holders, tabbed views, MDI
- \* Tables and data set views
- \* HTML page generation, Flash-as-GUI
- \* Managing sessions and transaction state machine

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

497

## Custom Widget Look Examples

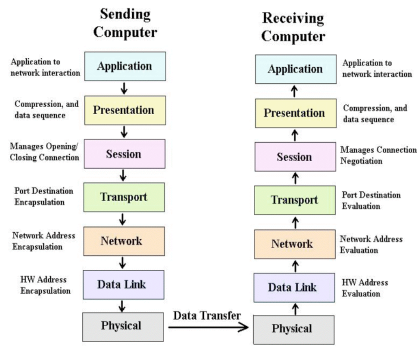
- \* Rotary knobs  
& linear sliders  
for VST



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

498

## Topic: GUI APIs



MEDIA ARTS & TECHNOLOGY PROGRAM

499

## GUI APIs

- \* Cross-platform C/C++
  - \* Qt
  - \* Fast Light Toolkit (FLTK)
  - \* WxWidgets
  - \* Flash (not really an API)
  - \* GLV/MINT
- \* Platform-specific
  - \* Windows (C#), Mac (Objective-C), Linux
- \* Language-specific (all cross-platform)
  - \* Java AWT/Swing
  - \* Smalltalk, LISP, Python, etc.

MEDIA ARTS & TECHNOLOGY PROGRAM

500

## The Qt GUI API and Library

- \* Cross-platform (Windows, Mac, Linux, PDA, phone, embedded)
- \* C++-oriented (with other language mappings)
- \* Model of signals and slots (requires special preprocessor)
- \* Designer GUIDE and other tools
- \* Large widget set

MEDIA ARTS & TECHNOLOGY PROGRAM

501

## Hello World in Qt

```
#include <qapplication.h>
#include <qpushbutton.h>

int main(int argc, char **argv) {
 QApplication a(argc, argv);
 QPushButton hello("Hello world!", 0);
 hello.resize(100, 30);
 a.setMainWidget(&hello);
 hello.show();
 return a.exec();
}
```

MEDIA ARTS & TECHNOLOGY PROGRAM

502

## WxWidgets

- \* Like Qt (cross-platform, C++ and other languages, tools, etc.)
- \* More complex than Qt
- \* Event/callback input model (no preprocessor)
- \* Document, diagram models

MEDIA ARTS & TECHNOLOGY PROGRAM

503

## Hello World in WxWidgets

```
#include "wx/wx.h"
#include "HelloWorldApp.h"

IMPLEMENT_APP(HelloWorldApp) // macro

bool HelloWorldApp::OnInit() {
 wxFrame *frame = new wxFrame((wxFrame*) NULL,
 -1, "Hello World");
 frame->CreateStatusBar();
 frame->SetStatusText("Hello World");
 frame->Show(TRUE);
 SetTopWindow(frame);
 return true;
}
```

MEDIA ARTS & TECHNOLOGY PROGRAM

504

## GUIs in Flash

- \* Flash for windows, Flash widgets
- \* Actionscript programming
- \* MacroMedia/Adobe Flash MX tools
- \* OpenSource tools
  - \* FLIRT
  - \* Mtasc
  - \* Enflash app. framework



505

## Hello World in Actionscript

```
import mx.controls.Button;

class helloworld {
 function helloworld(){
 var myBut:Button = _root.createClassObject(Button,
 "button2", 5, {label:"Push me!"});
 myBut._x = 300;
 myBut._y = 100;
 myBut.addEventListener("click", buttonclicked);
 }
 private function buttonclicked(event : Object) : Void {
 mx.controls.Alert.show("hello world", "title");
 }
}
```



506

## Platform-specific GUI APIs

- \* Windows
  - \* MFC, DirectX
- \* Mac
  - \* Cocoa, Carbon
- \* Linux
  - \* X, Motif, GTK, etc.



507

## GTK Hello World

```
#include <gfc/main.hh>
#include <gfc/gtk/window.hh>
#include <gfc/gtk/widgetsignals.hh>
using namespace GFC;

class HelloWorld : public Gtk::Window,
 protected Gtk::WidgetSignals {
protected:
 virtual bool on_delete_event(const Gdk::EventAny& event);
 void on_hello();
public:
 HelloWorld();
 ~HelloWorld();
};
```



508

## GTK Example (2)

```
HelloWorld::HelloWorld() : Gtk::WidgetSignals(this) {
 set_border_width(10);
 Gtk::Button *button = new Gtk::Button("Hello World");
 button->sig_clicked().connect(sigc::mem_fun(this,
 &HelloWorld::on_hello));
 add(*button);
 button->show();
}

int main (int argc, char *argv[]) {
 init(&argc, &argv);
 HelloWorld window;
 window.sig_destroy().connect(sigc::ptr_fun(&GFC::Main::quit));
 window.show();
 run(); // main event handler loop
}
```



509

## Advanced API Topics

- \* Internationalization
  - \* Separation of text messages from GUI
  - \* Support for 16-bit fonts
  - \* Support for non-L2R text
- \* Automatic generation of GUIs
  - \* By code, from XML, from a DB, etc.
- \* Pluggable look and feel (see example)
- \* Special user accessibility
  - \* Display (visual, spoken)
  - \* Input (keyboard, other media)



510

## Related GUI Topics

- \* 2D/3D graphics output
- \* Multi-modal input
  - \* Gesture sensors (haptics)
  - \* Sound rendering (text-to-speech)
  - \* Speech commands, recognition
  - \* Video input, computer vision
- \* Multi-user GUIs (remote event stream, Kansas)
- \* Multi-screen GUIs (desktop mgmnt, pointer, focus)
- \* Plug-ins for Web GUIs (GotoMyPC, VNC)

MEDIA ARTS & TECHNOLOGY PROGRAM

511

## Review

- \* GUI frameworks
- \* GUI development tools
- \* GUI APIs
- \* Advanced topics

MEDIA ARTS & TECHNOLOGY PROGRAM

512

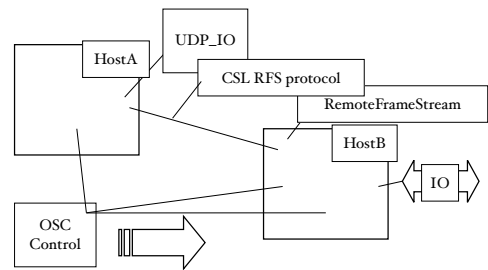
## Exercises

- \* Build GUIs by hand
  - \* Platform-specific APIs
  - \* Cross-platform APIs
- \* Using GUIDEs
  - \* Platform-specific tools
  - \* Cross-platform tools
- \* Writing custom widgets
  - \* Graphic output
  - \* Event handling and input
- \* Application analysis

MEDIA ARTS & TECHNOLOGY PROGRAM

513

## MAT 201B: Topic 7 Distributed Applications and Network Programming



MEDIA ARTS & TECHNOLOGY PROGRAM

514

## Distributed Applications and Network Programming

- \* Distributed processing and client/server architecture
- \* Standards for distributed programming (CORBA and RMI)
- \* IP networking basics
- \* Network programming: sockets and protocols
- \* Protocols for multimedia streaming

MEDIA ARTS & TECHNOLOGY PROGRAM

515

## Readings 7

- \* Internet: The Big Picture
- \* Network Layers
- \* TCP/IP Networking Protocols
- \* Beej's Guide to Network Programming
- \* Remote Method Invocation: Java RMI
- \* References
  - \* Protocol API docs

MEDIA ARTS & TECHNOLOGY PROGRAM

516

## What will we be learning?

- \* How do distributed applications work?
- \* What do you need to write a client/server program?
- \* How does one do low-level network programming and streaming?
- \* What protocols are relevant to media computing?

— MEDIA ARTS & TECHNOLOGY PROGRAM

517

## What happens if...

- \* Your application requires massive compute resources (> 1 CPU)?
- \* Your application must be fault tolerant?
- \* Your application is by nature geographically distributed?
- \* ...multi-host distributed processing: client/server (c/s) or peer-to-peer (p2p) application architecture

— MEDIA ARTS & TECHNOLOGY PROGRAM

518

## Distributed Processing

- \* To support distributed or client/server processing, one needs:
  - \* An external data representation (XDR or serialization methods) for passing data (structs or objects) between machines
  - \* Support for a remote procedure call (RPC) or remote method invocation (RMI) between machines, with processing of passed-in arguments, return values, and exceptions

— MEDIA ARTS & TECHNOLOGY PROGRAM

519

## Early Distributed Processing (1980s)

- \* XDR is a simple serialization API (hand-written or table-driven) for passing “data” over TCP or UDP socket streams
- \* RPC is managed by a daemon (msg. request brokers) implementing a proprietary network protocol over IP sockets with function look-up tables in the servers
- \* Client/server and 3-tier architectures
- \* Reuse of legacy applications
- \* Some scalable/fault-tolerant applications

— MEDIA ARTS & TECHNOLOGY PROGRAM

520

## 1990s Distributed Object Computing (DOC)

- \* RPC daemons become object request brokers (ORBs) and the Common ORB Architecture (CORBA) is born
- \* XDR serialization methods become standardized cross-language, cross-platform marshalling/unmarshalling libraries and IP inter-ORB protocols (GIOP, IIOP)
- \* Field of DOC middleware flourishes
- \* Provides:
  - \* Performance/throughput scalability and load-balancing
  - \* Redundancy and fault-tolerance, clean fail-over
  - \* Distributed/replicated databases, safe, high-bandwidth

— MEDIA ARTS & TECHNOLOGY PROGRAM

521

## DOC Architectures

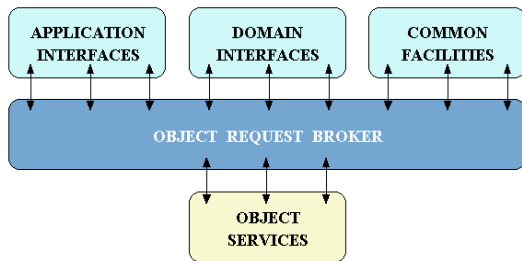
- \* Client/server and component architectures
- \* N-tier solutions: MVC++ and design patterns
- \* Distributed GUIs and multi-user apps
- \* Reuse of legacy software (app. wrappers, screen scrapers, RDBMS front-ends)
- \* Web access to applications, app. servers
- \* DOC services (see below)
- \* Transaction, message queue, logging systems
- \* Tools, databases for managing DOC SW

— MEDIA ARTS & TECHNOLOGY PROGRAM

522

## OMG CORBA Introduction

- \* Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA)

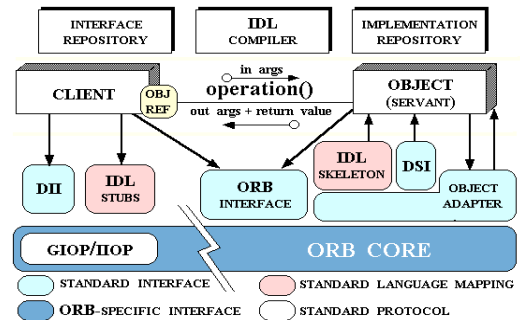


— MEDIA ARTS & TECHNOLOGY PROGRAM

523

## CORBA Message Flow

- \* Client says: `return_val = server_obj_ref.operate(inout args);`



— MEDIA ARTS & TECHNOLOGY PROGRAM

524

## CORBA IDL

- \* CORBA interface description language (IDL)
  - \* IDL as a specification (not implementation) language similar to C++ (without method bodies)
  - \* Has modules, interfaces, operations, inheritance
  - \* Core types, structures, sequences, user-defined types and classes
  - \* Operation signatures (in, inout params, return values, exceptions)
  - \* Sophisticated exception handling
  - \* Bindings for all OOPs
  - \* Cross-compilers generate code stubs

— MEDIA ARTS & TECHNOLOGY PROGRAM

525

## CORBA IDL Example

### Example from the CORBA A/V spec

```
interface StreamEndPoint : PropertyService::PropertySet {
 void stop(in flowSpec the_spec) raises (noSuchFlow);
 void start(in flowSpec the_spec) raises (noSuchFlow);
 void destroy(in flowSpec the_spec) raises (noSuchFlow);
}
```

Other examples (e.g., ATON DPE)

— MEDIA ARTS & TECHNOLOGY PROGRAM

526

## Writing and Running CORBA Apps

- \* Define interfaces in IDL
- \* Compile these into (C++, Java, ST, ...) stubs
- \* Implement services (i.e., write the app. glue)
- \* Compile/link server programs
- \* Run-time: how to clients/servers find out about one another?
  - \* Cmd-line args with hostname/port
  - \* Database of connectivity (as in CRAM)
  - \* CORBA services: naming (WP), trading (YP)

— MEDIA ARTS & TECHNOLOGY PROGRAM

527

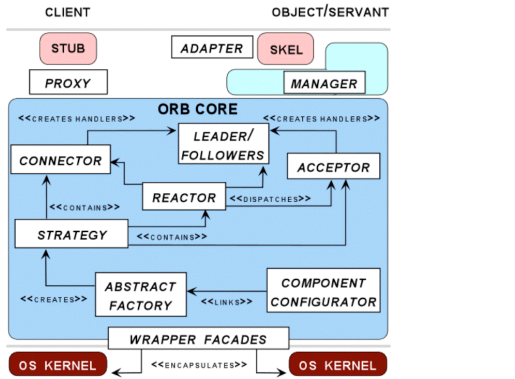
## CORBA Distributed Processing

- \* Reliable blocking uni-cast message-passing
- \* Deferred synchronous or asynchronous communications
- \* Client/server or peer-to-peer relationships
- \* App. managers for setting up repositories, running life-cycles
- \* Domain-specific frameworks and add-on specs.
- \* Can be used in secure environments (GIOP and IIOP)
- \* Optimized ORBs for speed, size, OS, language, etc.

— MEDIA ARTS & TECHNOLOGY PROGRAM

528

## ORB Design Patterns



529

## CORBA Services (Utilities)

- \* Naming: white pages DB (find service by name)
- \* Trading: yellow pages DB (find service by behaviors or features)
- \* Interface repositories: service catalog DB
- \* Logging: centralized logging sink and DB
- \* Life-cycle: manage server creation/migration
- \* Publish/subscribe event services and message queues

530

## CORBA Interoperability and Protocols

STANDARD CORBA PROGRAMMING API			
ORB MESSAGING COMPONENT	GIOP	GIOP Lite	ESIOIP
ORB TRANSPORT ADAPTER COMPONENT	IIOP	VME-IOP	ATM-IOP
TRANSPORT LAYER	TCP	VME DRIVER	RELIA- BLE SEQUENCED AAL5
NETWORK LAYER	IP	DRIVER	ATM
PROTOCOL CONFIGURATIONS			

\* D. Schmidt, CORBA Tutorial

531

## CORBA Plus/Minus

- \* Disadvantages:
  - \* Complex, messy, expensive, fragile
  - \* Advanced services (naming service, trader, interface repository) required for most non-trivial applications
- \* Still, it's used in many large-scale distributed applications (telecomm, financial services, medicine, defense, all web svcs.)
- \* ORBs: Orbix, e\*ORB, JacORB, omniORB, TAO, MICO, Orbit, Visibroker, SmalltalkBroker

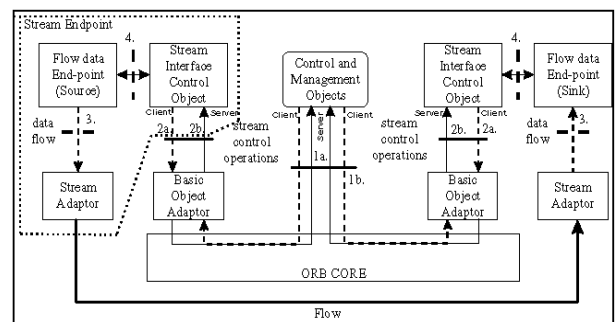
532

## CORBA A/V Streams

- \* OMG SIGs
  - \* Medicine, transportation, telecomm, ...
- \* A/V Streaming Architecture
  - \* A/V Flow/stream connections
  - \* Flow devices
  - \* (Virtual/real) MM-devices and drivers
  - \* Endpoint properties, sources/sinks
  - \* Control objects
  - \* In-band and out-of-band (OOB) communication

533

## CORBA A/V Stream Architecture



534

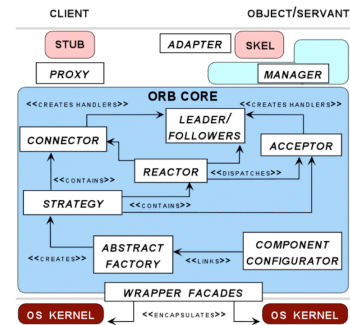
## Modern Distributed Processing: CORBA Descendants

- \* Current ORBs
  - \* Real-time/concurrent (TAO), lightweight (MICO)
  - \* Dynamic networks and services (DPE, CRAM)
  - \* CORBA app. mgmt. tools (life-cycle, POA)
- \* Java RMI
  - \* Like CORBA for Java-only (JVM runs ORB)
  - \* Classes implement the Serializable interface
- \* XML+RPC (SOAP)
  - \* XDR is XML (advantages and disadvantages)
  - \* Simple protocol for RPC
  - \* Huge time/space overhead
- \* COM/DCOM/WDNA
- \* (Ent)JavaBeans, DST/ObjWeb

MEDIA ARTS & TECHNOLOGY PROGRAM

535

## TAO ORB Design Patterns (D. Schmidt)

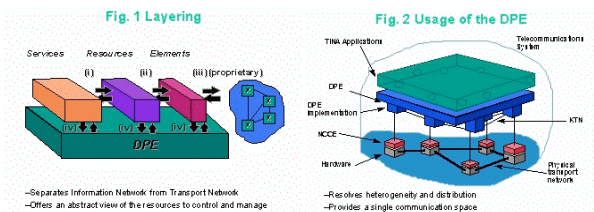


MEDIA ARTS & TECHNOLOGY PROGRAM

536

## Running Large DOC Systems

- \* TINA-C architecture for networks
- \* DPE Distributed Processing Environment model and design patterns

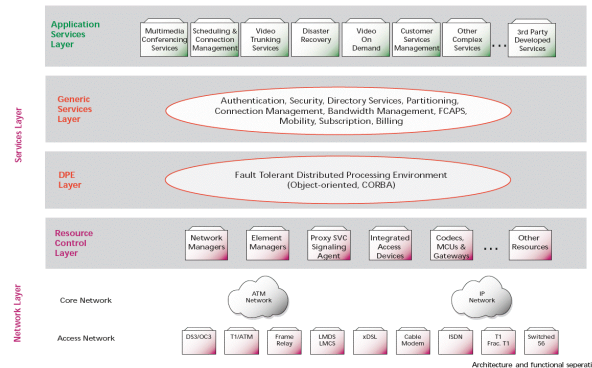


MEDIA ARTS & TECHNOLOGY PROGRAM

537

## Comprehensive TINA-C Architecture

Starvision SCMS 25000

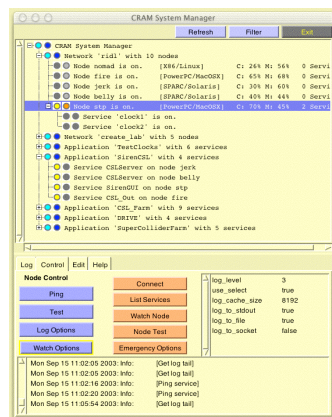


MEDIA ARTS & TECHNOLOGY PROGRAM

538

## DOC Tools

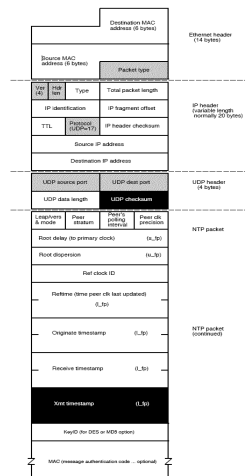
- \* Service, app., and network DBs
- \* Management and start-up: tool or scripts
- \* Monitoring: tool, profiler, fail-over
- \* Example: UCSB CRAM DBs and tools



MEDIA ARTS & TECHNOLOGY PROGRAM

539

## Topic: Networking



MEDIA ARTS & TECHNOLOGY PROGRAM

540

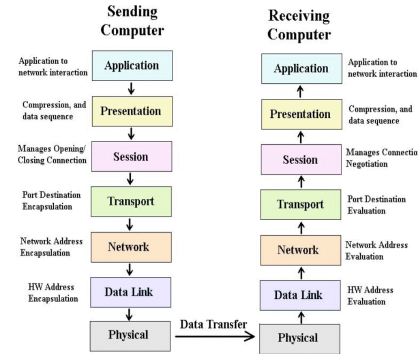
## Data Networking

- \* Connecting computers via networks
- \* Representations of data and messages
- \* Trade-offs between mapping layers, compactness, APIs, and protocols
- \* How to program at the TCP/IP and UDP/IP level (actually useful)
- \* What are RFCs and who cares?

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

541

## Data Networks: OSI Model



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

542

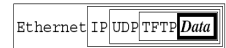
## OSI Model: 7 Layers

- \* **Physical:** connection via wire, fiber, radio, etc.
- \* **Data link:** low-level packet transfer method, e.g., 802x Ethernet
- \* **Network:** inter-host message routing and circuits, e.g., IP protocol
- \* **Transport:** data stream sequencing and error detection, e.g., TCP or UDP protocols
- \* **Session:** client model state and transactions
- \* **Presentation:** data types and semantics for intermedia mapping
- \* **Application:** user services and components

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

543

## TCP/IP (DOD) 4-Layer Model



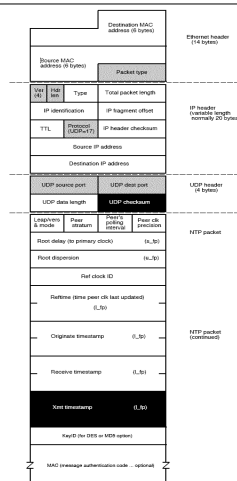
- \* **Link layer:** (= OSI's physical + data link) Ethernet, PPP, SLIP
- \* **Network layer:** IP, ARP, RARP
- \* **Transport layer:** TCP, UDP, ICMP, IGMP
- \* **Application layer:** (session, presentation, application) FTP, SMTP, DHCP, POP, IMAP, Telnet, HTTP, NFS, SNMP, ...

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

544

## What does this mean?

- \* Network packets have multi-level headers with data for each of the layers and protocols
- \* Example: structure of a network time protocol (NTP) packet



545

## TCP/IP Function Groups

- \* Packaging and low-level control: IPv4, IPv6, ATM
- \* Transport: TCP (stream), UDP (datagram)
- \* Network management: SNMP, ICMP, ARP
- \* Host management: RARP, DHCP, BOOTP
- \* Info. System: LDAP, Rendezvous, NIS
- \* File-sharing: NFS, AppleShare, Samba
- \* Mail: SMTP, POP, IMAP
- \* Multicasting: IGMP
- \* Routing: BGP, RIP, OSPF
- \* WWW: HTTP, RTSP

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

546

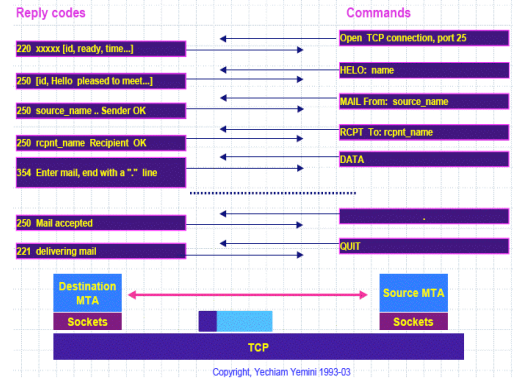
## TCP vs. UDP

- \* Stateful stream protocol vs. stateless datagram protocol
- \* TCP
  - \* Remembers connections, re-sends packets if dropped, guarantees packet order
  - \* Has much more overhead (esp. on a busy network)
  - \* Still cannot guarantee quality of service (QoS)
- \* UDP
  - \* Is connectionless (no remembered packet state)
  - \* Can drop any packet any time (apps. must do own reliable transfer)
  - \* Has much lower overhead than TCP
  - \* Applications can choose to provide some of the features of TCP on top of UDP (examples...)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

547

## SMTP Protocol Email Example



MEDIA ARTS &amp; TECHNOLOGY PROGRAM

548

## Network Programming: Sockets

- \* What do you need?
  - \* An address
  - \* A connection
  - \* An agreement about the message format
- \* What's a socket?
  - \* A UNIX-like "virtual device" (see above)
  - \* An integer that stands for an OS resource connected to a "port," specified with a host IP address and OS port number

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

549

## Addressing Sockets

- \* Each computer connected to a network has a 32-bit identifier for each LAN/WAN hardware interface (written as 4 bytes in "dot notation" e.g., 128.111.92.3)
- \* On a computer, multiple programs can open (OS-managed and routed) "ports" on a network interface (see the table in the file /etc/services)
- \* To connect from the outside therefore, one needs the host's IP address and the port number

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

550

## OK, So how's it programmed?

### Socket address structures

```
// Internet address (a structure for historical reasons)
struct in_addr {
 unsigned long s_addr; // that's 4 bytes
};

// Socket address (Internet address + port number)
struct sockaddr_in {
 short int sin_family; // Address family (AF_INET)
 unsigned short int sin_port; // Port number
 struct in_addr sin_addr; // 32-bit Internet address
 unsigned char sin_zero[8]; // Same size as struct
};
```

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

551

## How do you process these?

- \* See Beej's examples
  - \* Include the platform-specific files
  - \* Create and zero-out data structures
  - \* Set INET-format address (in host order)
  - \* Set or find port number
  - \* Call socket() to create the accessor
  - \* Check for error return code (< 0)

MEDIA ARTS &amp; TECHNOLOGY PROGRAM

552

## Putting Sockets Together

- \* Binding a socket to an IP/port address
- \* Listening for connection on it
- \* Connecting to a remote host from another
- \* Accepting a connection
- \* Once you're on: send() and recv()  
or sendto() and recvfrom()
- \* The telephone analogy
- \* See client/server examples (CSL and others)

— MEDIA ARTS & TECHNOLOGY PROGRAM

553

## Fancier Usage: select()

- \* Normal UNIX I/O model: blocking r/w on each file
- \* Bad idea with unreliable sockets
- \* Use the select() system call with multiple file descriptors, multi-try read and a time-out loop
- \* Give it a list of read and write file descriptors to monitor, and a time-out duration
- \* It returns when there's anything to read or write
- \* See examples (CRAM); all real apps use this

— MEDIA ARTS & TECHNOLOGY PROGRAM

554

## Use select() in a sentence

```
while(TRUE) {
 // Endless loop
 // Set up the file descriptor lists
 FD_ZERO(&read_fds); // clear read_fds
 FD_ZERO(&write_fds); // clear write_fds
 FD_SET(mysockfd, &read_fds); // set bit to read the OSC socket
 // Do the "select" call on socket
 if (select(sockfd + 1, &read_fds, &write_fds, (fd_set *)0,
 (struct timeval *)0) < 0) {
 perror("OSC: bad select"); // if select reported an error
 goto quit; // whine and exit
 }
 if(FD_ISSET(mysockfd, &read_fds)) // if there's a message coming in
 receive_OSC_packet(sockfd); // Call receive function
 // handle other input streams?
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

555

## Reliable Socket I/O

- \* Which transport to use: UDP vs. TCP
- \* Ubiquitous error detection (return values)
- \* Handling of open/connect errors
- \* Handling of partial reads/writes
  - \* Loops
- \* Handling of I/O errors
  - \* Reconnection
- \* See code examples

— MEDIA ARTS & TECHNOLOGY PROGRAM

556

## Using the Higher-level Protocols

- \* Libraries, APIs, class hierarchies for application-layer protocols
- \* QkSoft SMTP objects
- \* VisualWorks protocol classes
- \* Len Holgate's C++ socket server classes
- \* Chilkat ftp server classes
- \* ...many more

— MEDIA ARTS & TECHNOLOGY PROGRAM

557

## So what's the problem?

- \* Standard protocols are AOK for mail and ftp, but are not really suitable for multimedia applications
- \* Unreliable or unpredictable
- \* No guarantees of throughput (or QoS)
- \* No way to reserve bandwidth
- \* No run-time monitoring of performance
- \* Many limits (e.g., packet size)

— MEDIA ARTS & TECHNOLOGY PROGRAM

558

## Other Solutions

- \* Backbone networks don't use Ethernet
  - \* ATM, SONET networking
  - \* FireWire networking
- \* These networks don't use TCP/IP
  - \* ATM protocols and IP-on-ATM (MPOA)
  - \* Programming ATM makes brain surgery look easy!
- \* What better protocols are there for the IP world?

MEDIA ARTS & TECHNOLOGY PROGRAM

559

## Advanced Media Protocols

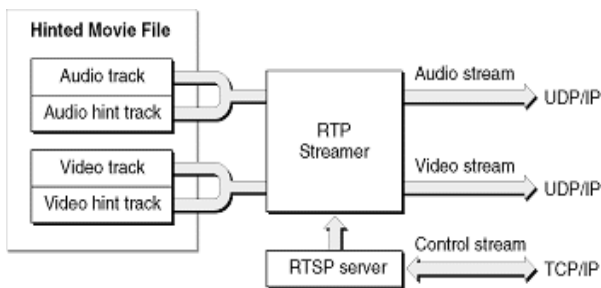
- \* Real-Time Protocol (RTP): transport protocol for the delivery of real-time data, including streaming media
- \* Real-Time Control Protocol (RTCP): part of RTP that helps with synchronization and QoS management
- \* Real Time Streaming Protocol (RTSP): control protocol for initiating and directing delivery of streaming multimedia from media servers, the "Internet VCR remote control protocol"

MEDIA ARTS & TECHNOLOGY PROGRAM

560

## Media Streaming Servers

- \* Example: QuickTime Server



MEDIA ARTS & TECHNOLOGY PROGRAM

561

## 1-to-Many Networking

- \* Up to now, we've spoken of 1-to-1 connections
- \* Many applications (teleconferencing, video-on-demand) want 1-to-many "broadcasting" or even many-to-many "multicasting"
- \* Multicast is for this; it's built into many modern operating systems
- \* There is a range of IP addresses reserved for multicast groups
- \* Routers recognize this
- \* See Linux "Multicast over TCP/IP HOWTO"

MEDIA ARTS & TECHNOLOGY PROGRAM

562

## How to use Multicast Networking

```
// Simple calls based on setsockopt()
struct ip_mreq {
 // IP multicast address of group
 struct in_addr imr_multiaddr;
 // local IP address of interface
 struct in_addr imr_interface;
};

// set this option on some socket
setsockopt(socket, IPPROTO_IP,
 IP_ADD_MEMBERSHIP,
 &mreq, sizeof(mreq));
```

MEDIA ARTS & TECHNOLOGY PROGRAM

563

## Other Networking Topics

- \* Gee, one really could have a whole course on this topic...
- \* Loads more issues
  - \* Services
  - \* Protocols
  - \* Streaming and QoS
  - \* Better multicast group management

MEDIA ARTS & TECHNOLOGY PROGRAM

564

## Review

- \* Distributed processing and client/server architecture
- \* Standards for distributed programming (CORBA and RMI)
- \* Network programming: sockets and protocols
- \* Protocols for multimedia streaming

MEDIA ARTS & TECHNOLOGY PROGRAM

565

## Exercises

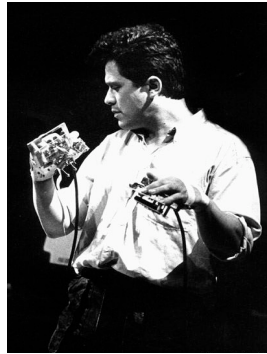
- \* CORBA anyone?
  - \* Download MICO to start
- \* Use JRMI or SOAP
- \* Low-level socket tasks
  - \* Simple socket I/O
  - \* Build a simple XDR/RPC framework
- \* IP Protocols: use RTP, etc.
- \* Streaming using Helix or QuickTime streaming server

MEDIA ARTS & TECHNOLOGY PROGRAM

566

## MAT 201B: Topic 8

### Control Protocols and Sensor I/O



MEDIA ARTS & TECHNOLOGY PROGRAM

567

## Control Protocols and Sensor I/O

- \* Control vs. Content
- \* MIDI
- \* Open Sound Control
- \* Sensor input and drivers
- \* Control APIs

MEDIA ARTS & TECHNOLOGY PROGRAM

568

## Readings 8

- \* PortMIDI Header file
- \* OpenSoundControl Paper
- \* References
  - \* LibLo
  - \* Other MIDI
  - \* CUI Course

MEDIA ARTS & TECHNOLOGY PROGRAM

569

## What will we be learning?

- \* What's so different about control?
- \* What's MIDI, who cares?
- \* What's better than MIDI for control?
- \* How do you program apps that use MIDI, OSC, USB game controllers, etc?

MEDIA ARTS & TECHNOLOGY PROGRAM

570

## Control vs. Content

- \* Media content
  - \* Constant-rate streaming
  - \* Large-ish fixed-size buffers
  - \* LAN-based apps generally have small number of sources/sinks
- \* Control
  - \* Is bursty and unpredictable
  - \* Packets are small and variable size
  - \* We often want multiple sources and sinks streaming on a LAN

— MEDIA ARTS & TECHNOLOGY PROGRAM

571

## MIDI

- \* Musical Instruments Digital Interface (1983)
- \* Simple low-level protocol implemented over low-cost p2p network
- \* Isochronous communication, (mostly) stateless receiver
- \* Most commands are 2 or 3 bytes
  - \* Note-on/pitch/volume: 0x90, 0x60, 0x70
- \* Also used for lighting, motor control, etc.
- \* Implemented over USB, etc.
- \* Terrible design, ubiquitous in music apps.

— MEDIA ARTS & TECHNOLOGY PROGRAM

572

## MIDI Details

- \* Keyboard-centric paradigm: note/key/velocity
- \* Weak models: pitch as key number, amplitude as velocity, timbre as channel
- \* Extensible via system-exclusive (SYSEX) messages
- \* Continuous control as controller updates (easy to soak up all bandwidth)
- \* Limited to 16 channels/voices, short cable runs, 31 kHz bandwidth
- \* Voice-stealing policies

— MEDIA ARTS & TECHNOLOGY PROGRAM

573

## MIDI Messages

- \* Basic Commands

Command	Meaning	# parameters	param 1	param 2
0x80	Note-off	2	key	velocity
0x90	Note-on	2	key	velocity
0xA0	Aftertouch	2	key	touch
0xB0	Continuous controller	2	controller #	controller value
0xC0	Patch change	2	instrument #	
0xD0	Channel Pressure	1	pressure	
0xE0	Pitch bend	2	lsb (7 bits)	msb (7 bits)
0xF0	(non-musical commands)			

- \* Extended Commands

command	meaning	# param variable
0xF0	start of system exclusive message	
0xF1	MIDI Time Code Quarter Frame (Sys Common)	
0xF2	Song Position Pointer (Sys Common)	
0xF3	Song Select (Sys Common)	
0xF4	???	
0xF5	???	
0xF6	Tune Request (Sys Common)	
0xF7	end of system exclusive message	0
0xF8	Timing Clock (Sys Realtime)	
0xFA	Start (Sys Realtime)	
0xFB	Continue (Sys Realtime)	
0xFC	Stop (Sys Realtime)	
0xFD	???	
0xFE	Active Sensing (Sys Realtime)	
0xFF	System Reset (Sys Realtime)	

— MEDIA ARTS & TECHNOLOGY PROGRAM

574

## MIDI APIs

- \* APIs for low-level MIDI I/O in most modern OSs (some still based on serial driver APIs)
- \* Poll driver for pending input (since it's bursty), or register a event handler call-back
- \* Schedule (small or large) messages for output
- \* Driver-level mapping to multiple interfaces and devices (virtual devices and patch bays)

— MEDIA ARTS & TECHNOLOGY PROGRAM

575

## MIDI on MS-Windows

```
HMIDIOUT device; // device for sending MIDI output
union {
 unsigned long word;
 unsigned char data[4];
} message;
message.data[0] = 0x90; // MIDI note-on message
message.data[1] = 60; // Key number (60 = middle C)
message.data[2] = 100; // Key velocity (100 = loud)

midiOutOpen(&device, 0, 0, 0, CALLBACK_NULL);
midiOutShortMsg(device, message.word);
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

576

## MIDI API Functionality

- \* Query attached devices (OS data structs) and virtual device mappings
- \* Manage SYSEX up/down-loads
  - \* Voice editor/librarians
  - \* Sample dump I/O
- \* Store configurations (virtual studio)
- \* MIDI file I/O
- \* API structs: ports, devices, controllers

MEDIA ARTS & TECHNOLOGY PROGRAM

577

## PortMIDI

- \* Related to PortAudio
- \* Open/close functions, no call-backs
- \* Blocking read, waits for input
- \* Non-blocking poll checks for input
- \* Fcns to get HW props, device names, etc.
- \* See portmidi.h and CSL examples
- \* See also other MIDI APIs (CoreMIDI, Linux / dev/midi and /dev/sequencer)

MEDIA ARTS & TECHNOLOGY PROGRAM

578

## Using MIDI

- \* Soft-synths: QTMusic, RAX, DirectX, DLS
- \* MIDI in web browsers
- \* MIDI in Flash
- \* SW-only solutions
- \* Code examples: Playing over built-in soft-synth

MEDIA ARTS & TECHNOLOGY PROGRAM

579

## MIDI++

- \* Rich literature on “The dysfunctions of MIDI”
  - \* Limits on # channels, continuous control
  - \* Low-level models of pitch, amplitude, etc.
  - \* Timing inaccuracies, low throughput
  - \* Note on/off event model
- \* Many attempts to make it better (4xMIDI, ZIPI)
  - \* Separate control from event triggering
  - \* Flexible models of pitch, amplitude, timbre
  - \* Scalable continuous controls
  - \* Run on top of a faster, wider-area network

MEDIA ARTS & TECHNOLOGY PROGRAM

580

## Open Sound Control (OSC)

- \* UCB/CNMAT, derived from ZIPI
- \* Based on UDP/IP (or other fast transport)
- \* Every parameter of every instrument has a unique address
- \* Commands fit into this hierarchy
- \* Servers assumed to have schedulers

MEDIA ARTS & TECHNOLOGY PROGRAM

581

## OSC Messages

- \* Addresses: UNIX file name style with wild cards
  - \* /guitar/2/string/4/frequency      -- parameter address
  - \* /oscillator/\*/phase                -- multiple parameters
- \* Command may be address + data, or just the address
  - \* /guitar/2/string/4/freq 440.0      -- set a variable
  - \* /guitar/2/stop                      -- send a command
- \* Data: int, float, string, blob
- \* Bundles: several commands to take place at the same time can be bundled together (scheduled by server)

MEDIA ARTS & TECHNOLOGY PROGRAM

582

## OSC Features

- \* Open-ended, dynamic, URL-style symbolic naming scheme (needs good implementation)
- \* Numeric and symbolic arguments to messages
- \* Pattern-matching language to specify multiple targets of a single message ("set all strings to quiet")
- \* High resolution time tags
- \* "Bundles" of messages whose effects must occur simultaneously (schedule-ahead)
- \* Query system to dynamically find out the capabilities of an OSC server and get documentation (discovery)

— MEDIA ARTS & TECHNOLOGY PROGRAM

583

## Using the OSC API Functions

- \* Writing OSC
  - \* Open a socket on a port
  - \* Format a message to write
  - \* See gesture sensor code
- \* Reading OSC
  - \* Straightforward call-back API
  - \* Listen for messages on a port
  - \* Register call-back functions at OSC addresses (variables or commands)
  - \* Main select() loop
  - \* See CSL instrument driver

— MEDIA ARTS & TECHNOLOGY PROGRAM

584

```
int sendP5OSC(struct p5glove_data *info, void *htmsocket) {
 OSCTimeTag tag;
 OSCbuf buf;
 OSC_initBuffer(&buf, 1024, bufferForOSCBuf);
 OSC_openBundle(&buf, tag);

 OSC_writeIntArg(&buf, (info->buttons & P5GLOVE_BUTTON_B) >> 1);
 // ...
 OSC_writeIntArg(&buf, z);
 OSC_closeBundle(&buf);
 if(TRUE != SendHTMSocket(htmsocket, OSC_packetSize(&buf),
 OSC_getPacket(&buf))) {
 printf("Failed Send Packet \n");
 return -1;
 }
 return 0;
}
```

## Writing OSC (P5Glove)

— MEDIA ARTS & TECHNOLOGY PROGRAM

585

## Reading OSC (CSL driver)

```
while(morePackets) {
 buf = OSCPacketBufferGetBuffer(pb);
 n = recvfrom(sockfd, buf, capacity, 0, &(ra->cl_addr),
 &(ra->clilen));
 if (n > 0) {
 int *sizep = OSCPacketBufferGetSize(pb);
 *sizep = n;
 OSCAcceptPacket(pb);
 if (time_to_quit) return;
 } else {
 OSCFreePacket(pb);
 morePackets = 0;
 }
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

586

## OSC Implementations

- \* CNMAT reference code
  - \* Requires low-level UNIX networking
  - \* Needs better search/parsing for good performance with a large address space
  - \* Needs a router for large distributed systems
- \* LibOSC++ wrappers
- \* LibLO C API (<http://liblo.sourceforge.net>)
- \* Implemented by many interesting packages (SuperCollider, Max/MSP, Reaktor, Siren, CSL, Bidule, Traktor, etc.)

— MEDIA ARTS & TECHNOLOGY PROGRAM

587

## Sensors and Input Drivers

- \* Other sensors
  - \* Serial port inputs: loop to parse messages send out some other protocol (see FlockOfBirds examples)
  - \* USB devices: same (see P5-to-OSC interface)
  - \* Camera input: frame-grabbers and video feature extraction

— MEDIA ARTS & TECHNOLOGY PROGRAM

588

## CSL GestureSensors

```
class GestureSensor { // Abstract CSL/OSC with gesture sensor
public:
 void * mData; // my data array (typically a float *)
 char * mCmd; // my OSC command name (without the '/')
 char * mTypeString; // my OSC type string, e.g., "ffff"
 // OSC_CALL_BACK * mReaderFcn; // my OSC call-back function
 // (NB: not a class method)

 // Constructor
 GestureSensor(char * name, char * types, OSC_CALL_BACK * callback);
 // get my data
 // Subclasses add accessors like getX/Y/Z/Roll/Fingers...
 virtual void * get_data() { return mData; };
 // parse an OSC msg. into my array according to my type string
 virtual status parse_OSC_packet(char * typeString, void * args);
};
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

589

## Reading from a GestureSensor

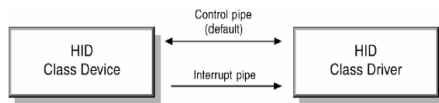
```
// This is forked as a loop to update global data from the Ebeam input
void map_ebeam(EBeam & ebeam) {
 float x = ebeam.get_x();
 float y = ebeam.get_y();
 if ((x == 0.0) || (y == 0.0)) return;
 int xQuadrant = x > 1800 ? 0 : 1; // get the current quadrant
 int yQuadrant = y > 850 ? 0 : 1;
 float xVal = x / 1800 - xQuadrant; // and position within quadrant
 float yVal = y / 850 - yQuadrant; // now in the range 0-1
 int index = xQuadrant * 2 + yQuadrant;
 gBaseFreq[index] = (xVal + 1) * gSBaseFreq[index]; // store somewhere
 gFreqRange[index] = (yVal + 1) * gBaseFreq[index];
 // printf("EB: %5.1f @ %5.1fn", x, y);
}
```

— MEDIA ARTS & TECHNOLOGY PROGRAM

590

## USB Human Interface Device (HID)

- \* Packet format for HIDs
- \* Refined in the physical interface device (PID) spec. for interfaces with feedback (force joysticks, steering wheels, etc.)
- \* Mouse, keyboard, gaming controller, etc.
- \* See P5-to-OSC driver example



— MEDIA ARTS & TECHNOLOGY PROGRAM

591

## Clients for Controllers

- \* Data mapping
  - \* Range scaling and offset
  - \* Smoothing and throttling
- \* Gesture recognition
  - \* Limit detection and feature extraction
  - \* Matching to a gesture vocabulary
- \* See CSL instrument classes
- \* Writing OSC servers

— MEDIA ARTS & TECHNOLOGY PROGRAM

592

## Review

- \* Control vs. content
- \* MIDI protocol
- \* Open Sound Control
- \* Other control interfaces

— MEDIA ARTS & TECHNOLOGY PROGRAM

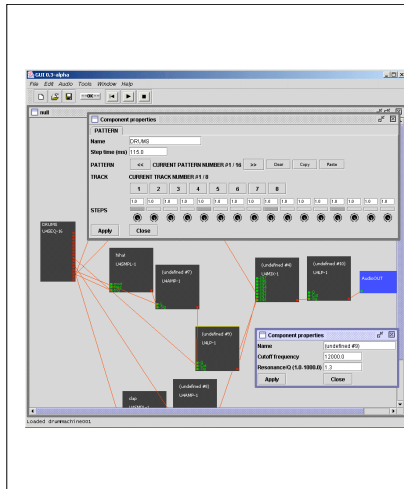
593

## Coding

- \* PortMIDI
- \* Platform-MIDI
- \* OSC - LibLo
- \* HID, CUI
- \* CSL GestureSensors
- \* Other APIs

— MEDIA ARTS & TECHNOLOGY PROGRAM

594



## MAT 201B: Topic 9

### Domain-specific Application Frameworks

MEDIA ARTS & TECHNOLOGY PROGRAM

595

## Domain-specific Application Frameworks

- \* Authoring, MM data management
- \* Performance systems
- \* Gaming platforms
- \* Virtual Environments
- \* Network-based content delivery
- \* Exercises: application analysis
- \* Course review

MEDIA ARTS & TECHNOLOGY PROGRAM

596

## What will we be learning?

- \* What kinds of frameworks and class libraries are available for specific (media-related) application domains?
- \* How would I write one myself?

MEDIA ARTS & TECHNOLOGY PROGRAM

597

## Categories of Frameworks

- \* Procedural API and data structures
- \* Domain model, class library
- \* Generic customizable application
- \* Scripting language with domain-specific libraries
- \* Method, notation, design patterns
- \* Tool kit, GUIDE
- \* GUI support, widgets

MEDIA ARTS & TECHNOLOGY PROGRAM

598

## Building Authoring Tools

- \* Content creation
  - \* Data capture
  - \* Control input
- \* Content data management
  - \* File storage
  - \* Browsing
  - \* Versioning
- \* Composition, compositing
- \* Rendering/performance
- \* Import/export formats

MEDIA ARTS & TECHNOLOGY PROGRAM

599

## Performance Systems

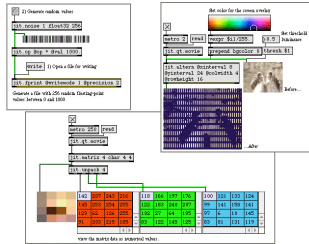
- \* Sound/Music
  - \* OO models of signals, sources, and processors
  - \* Handling of I/O, drivers, and scheduling
  - \* CLAM: C++ Library for Audio & Music
    - \* DSP Objects with ports
  - \* CSL: CREATE Signal Library
    - \* FrameStreams, IO, pull-model, OSC
  - \* JavaSynth
    - \* Core methods as native code
  - \* SuperCollider
    - \* Smalltalk dialect with DSP \* thread libraries
  - \* PureData

MEDIA ARTS & TECHNOLOGY PROGRAM

600

## Performance Systems

- \* Image/video
  - \* Modeling of scenes and images
  - \* Output API (OpenGL)
  - \* Input and control
- \* Basics
  - \* Java3D
  - \* OpenSceneGraph
  - \* Blender
- \* Animation frameworks
  - \* Alice, Maya
- \* VJ Tools
  - \* Jitter, NATO

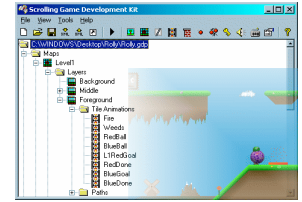
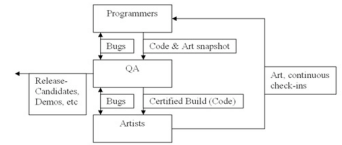


MEDIA ARTS & TECHNOLOGY PROGRAM

601

## Gaming Platforms

- \* Many web sites
  - \* gamedev.net
  - \* gamasutra.com
  - \* animationarena.com
- \* Game design
  - \* Characters, narrative, artwork
- \* Support APIs
  - \* Rendering
  - \* Interaction
  - \* Multi-user support



MEDIA ARTS & TECHNOLOGY PROGRAM

602

## Game Design

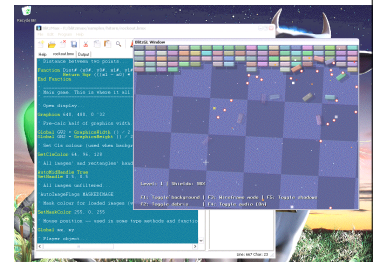
- \* Good starting points
  - \* Game Programming 101 (gamedev.net)
  - \* See Chris Charabaruk's design docs
- \* Characters and behaviors
- \* User model and psychology as a game character
- \* Model of goals and narrative
- \* The role of AI

MEDIA ARTS & TECHNOLOGY PROGRAM

603

## Game Development APIs

- \* A sampling:
  - \* Java Mobile Info. Device Profile (MIDP)
  - \* Microsoft Game API (GAPI)
  - \* Scrolling Game Development Kit
  - \* SDL
  - \* CrystalSpace3D
  - \* SecurePlay
  - \* pygame
  - \* BlitzMax
  - \* GameX, MINT



604

## Example: Simple DirectMedia Layer

- \* Cross-platform high-level C/C++ API for:
  - \* Video (OpenGL or frame buffer)
  - \* Audio I/O
  - \* Event input (Kbd, mouse, joystick, game controller)
  - \* Window/screen management
  - \* Threads
  - \* Timers
  - \* File I/O
  - \* CD-ROM input
- \* Used in many games, though not a complete gaming platform
- \* See the on-line SDL tutorials and examples

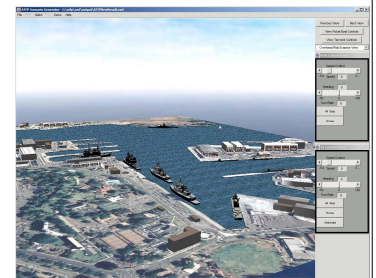


MEDIA ARTS & TECHNOLOGY PROGRAM

605

## Virtual Environments

- \* How different from gaming systems?
  - \* 3D models with behaviors
  - \* Different I/O
  - \* Level of immersion
  - \* Different user model
- \* VR vs. VE
  - \* Simulation base
  - \* Augmented reality

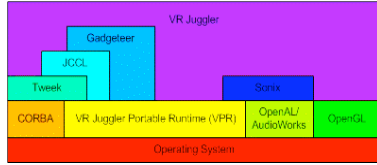


MEDIA ARTS & TECHNOLOGY PROGRAM

606

- \* OpenSceneGraph
- \* Blender
- \* WorldToolKit
- \* X3D, X3D
- \* VRUT
- \* DRIVE
- \* EON
- \* Syzygy
- \* FreeVR
- \* VR Juggler
- \* Meme
- \* VRPN

## VE Tools



MEDIA ARTS & TECHNOLOGY PROGRAM  
607

## Network-based Applications

- \* Web applications
  - \* Content delivery (old HTML model)
  - \* Dynamic content (CGI + RDBMS model)
  - \* Streaming servers (content on demand)
  - \* Applets (thin client: desktop, hand-held)
  - \* Distr. games (own protocol)
- \* New application models
  - \* On-line communities
  - \* Flexible content management (e.g., blogs)
  - \* Collaborative work over the Web

MEDIA ARTS & TECHNOLOGY PROGRAM  
608

## Support for Net Applications

- \* Dynamic HTML generation
  - \* DHTML
  - \* Template engines
  - \* Servlets/ASP
- \* Support APIs
  - \* E-commerce
  - \* Request delegation, load balancing
  - \* Applets and distributed behavior

MEDIA ARTS & TECHNOLOGY PROGRAM  
609

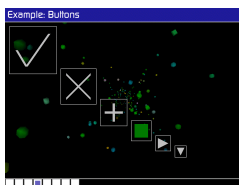
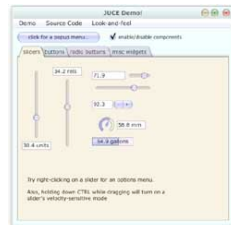
## Models of User State

- \* Role/model
- \* Community member
- \* Simulacrum
- \* System vs. user narrative
- \* User I/O/control media
- \* User network connection

MEDIA ARTS & TECHNOLOGY PROGRAM  
610

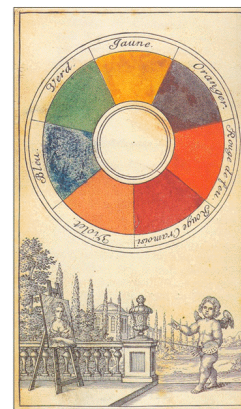
## All-purpose MM Frameworks

- \* The Future (generic/specific)
- \* JUCE
  - \* Audio, MIDI, graphics, GUI, network, XML, crypto,
  - \* Exhaustive, Literate, Coherent, Multi-platform, Modern
- \* GLV/MINT
  - \* DSP, audio, video, control, distribution



611

## Topic: Review



MEDIA ARTS & TECHNOLOGY PROGRAM  
612

## Final Review: Applications

- \* Analyze GUIs and design of interesting media-rich applications
- \* Data models, behaviors
- \* Domain frameworks
- \* Design patterns used
- \* Persistent storage and databases
- \* Networking interfaces
- \* Control input

MEDIA ARTS & TECHNOLOGY PROGRAM

613

## Application Examples

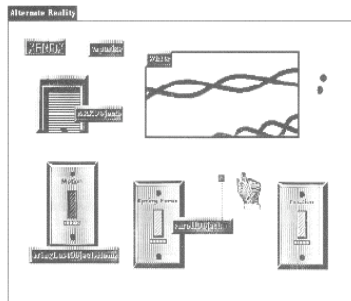
- \* See lecture 1 examples
  - \* Models, architecture, widgets
- \* Analysis of existing applications
- \* Analysis/design of new application/tool paradigms
- \* Where are the new application and tool models?

MEDIA ARTS & TECHNOLOGY PROGRAM

614

## Novel Application Models: ARK

- \* Alternate Reality Kit (ARK) (Smith & Ungar, 1986)
- \* Direct manipulation of software simulation
- \* Visual programming language
- \* Multi-user virtual world

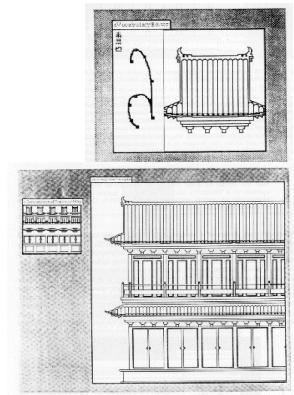


MEDIA ARTS & TECHNOLOGY PROGRAM

615

## Application Model: CDTK

- \* Chinese Temple Design Toolkit (Makkuni, 1987)
- \* Map gestures to features of vocabulary elements
- \* Design grammars for combination of elements
- \* Manage libraries of refinement processes

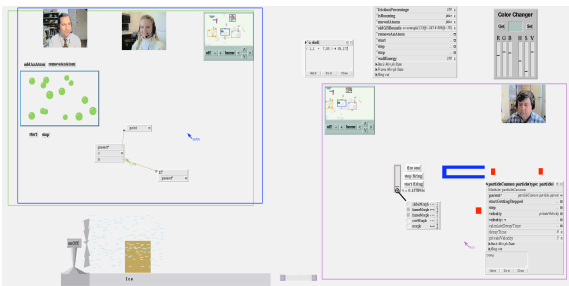


MEDIA ARTS & TECHNOLOGY PROGRAM

616

## Application Model: Self/Kansas

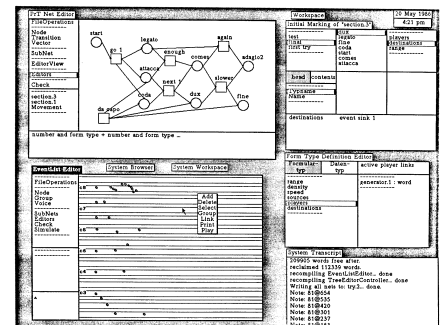
- \* OO development environment with no tools or windows, "only objects on the screen"
- \* Multi-user shared workspace



617

## Application Model: DoubleTalk

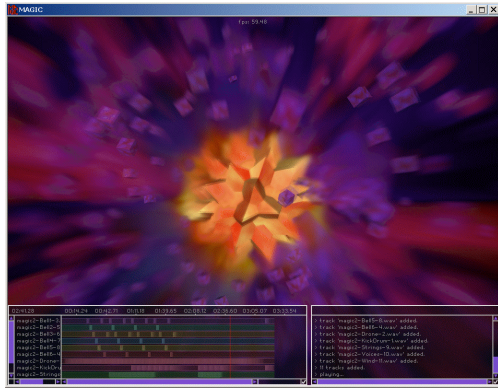
- \* Use logic-marked Petri nets for music composition
- \* Net and token editors
- \* Checkers and simulators



MEDIA ARTS & TECHNOLOGY PROGRAM

618

## Application Model: MAGIC



\* Eric Newman's "Synaesthetic Editor"

MEDIA ARTS & TECHNOLOGY PROGRAM

619

## Brain-storming New Models

- \* Applications
  - \* Sound: how to get away from tape recorder, score, cue sheet models?
  - \* Image: the same?
  - \* Programming: where's the visual programming language?
- \* Widgets
  - \* New MVC components for new models of state and interaction

MEDIA ARTS & TECHNOLOGY PROGRAM

620

## Conclusion/Review

- \* MAT 201B: Everything they didn't teach you in Java/C++ class...
- \* Where do you go from here?
  - \* As app developer, extender, or user
  - \* As tool developer...
- \* What's in store in the future?
  - \* New: media, interaction, networks, control, app. models

MEDIA ARTS & TECHNOLOGY PROGRAM

621

## Not Included in Package

- \* Advanced programming
  - \* C++/Java techniques (annotations, doc)
  - \* Other languages
- \* ProgInTheLarge
  - \* Team tools, process tools
- \* Media data and APIs
  - \* Other media, applications
- \* CS Techniques
  - \* AI, DB, OS, net
- \* Loads more to discover

MEDIA ARTS & TECHNOLOGY PROGRAM

622

## Acknowledgments

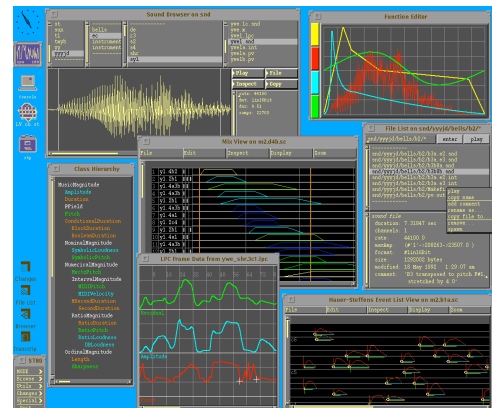
- \* Much material lifted from
 

Xavier Amatriain, Roger Dannenberg, Julius O. Smith,  
George Tzanetakis, Adrian Freed, Douglas Schmidt,  
Matt Wright and others

MEDIA ARTS & TECHNOLOGY PROGRAM

623

## Thank You!

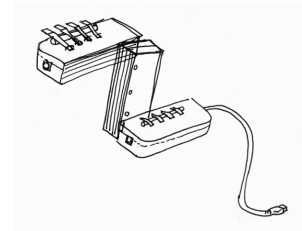


MEDIA ARTS & TECHNOLOGY PROGRAM

624

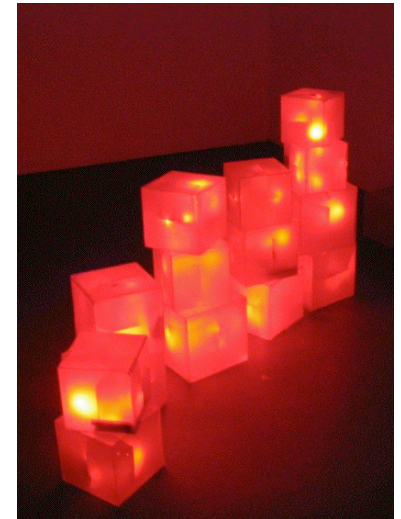
## **MAT 594O - Sensors and Interfaces** **for Media Art (Fall, 2008)**

The MAT Sensors and Interfaces course explores the use of multimedia sensor technologies and embedded microcontroller systems for interactive environments/installations and responsive artwork/performance systems. We will start with an introduction to the theories of space in art and human-computer interaction (HCI), and proceed to an in-depth analysis of current art-HCI technologies. Students will acquire experience that will allow them to research and develop custom sensor systems in their own work. The course will introduce the principles and operation of many sensor techniques and discuss applications in gestural human-computer interfaces, new musical instruments, the visual and spatial arts, engineering, science, and other areas of interest. It will cover the design of computer interface systems including analog/digital electronics and human factors/interaction styles. This is an applied theory course with laboratories and student projects that can potentially continue as independent research in following quarters. Output modalities will include interactive visuals, music/sound, and motion control/robotics.



### **Course Topics**

- Space & gestural interaction
- Human-computer interfaces: ergonomics & haptics
- Emerging interface technologies
- Transducers, sensors, signal capture & conditioning
- Pressure, position, optical, inertial, capacitive, and ultrasonic sensing techniques
- Sensor applications and signal conditioning electronics
- Microcontrollers & interfaces -- communication protocols, signal processing, feature extraction, and mapping schemes



### **Instructors**

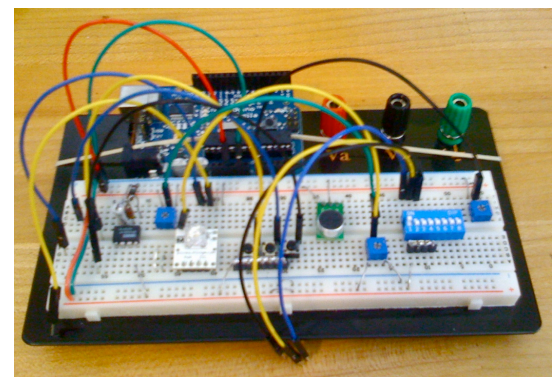
- Stephen T. Pope (stp@mat.ucsb.edu), Matthew Wright, and Amichi Amar

### **Meeting time and place**

- Tues/Thurs 5:00 - 6:50 PM
- Music 2215 or South Hall 4340

### **Electronic Resources**

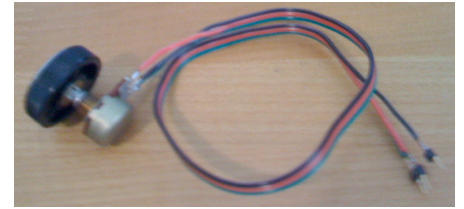
- Course Web Site: <http://www.mat.ucsb.edu/594O>
- Email Mailing List  
See <http://www.mat.ucsb.edu/mailman/listinfo/594O>



# MAT 594O Reader Table of Contents

## **Introduction/Course Notes**

DanO's Media Interface Technology Course Overview  
Cornell ECE 4760 Final Projects  
NYU Sensor Research Page  
MIT 6.270 Course Notes TOC  
NIME 2007/2008 Proceedings TOCs



## **Theory of Space and Gesture**

Wikipedia: Space & Time, Narrative, Interactive Art, Installation Art  
Spatio-Temporal Perspectives: A new way for cognitive enhancement: A. Goppold  
The Spatial Narrative: Animation and Art Installation: L. M. Pepi  
Why Gesture? (Slides): Matthew Turk  
A Manifold Interface for Kinesthetic Notation in High-Dimensional Systems: Insook Choi

## **Electronics**

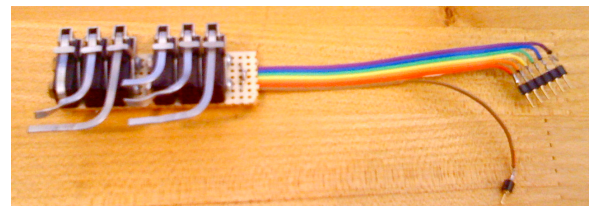
Princeton CS 436 Electronics Notes  
Electronics Demonstrations List: [www.falstad.com/circuit](http://www.falstad.com/circuit)  
Input/Data Acquisition System Design for Human Computer Interfacing: W. Putnam and R. B. Knapp  
Electronics & Schematics for Robots: Society of Robots  
How to build your first robot: Society of Robots  
How to bias an Op-amp: MIT MAS.836

## **Human-Computer Interfaces (HCI)**

Physical Interfaces in the Electronic Arts: Bert Bongers  
Less is More (More or Less): Uncommon Sense and the Design of Computers: Bill Buxton  
How Bodies Matter: Five Themes for Interaction Design: S. Klemmer, B. Hartmann & L. Takayama  
Reflective Physical Prototyping through Integrated Design, Test, and Analysis: B. Hartmann, S. Klemmer

## **Transducers and Sensors**

Arduino: Interfacing with Hardware (I/O device list)  
Wikipedia List of Sensors and Transducers  
Phidgets.com Analog Sensors catalog  
Sensors and Actuators: Society of Robots  
NYU ITP Sensor Reports  
Directory of Sources for Input Devices (excerpt): Bill Buxton  
Wikipedia: Accelerometer  
Aduino and Accelerometer: Near Future Labs  
ITP 3-Axis Accelerometer Notes  
Wikipedia: Piezoelectric Sensor, ITP Piezo input Circuitry  
Wikipedia: Solenoid  
Actuators - Solenoids: Society of Robots  
Multi-touch Systems I have Known and Loved: Bill Buxton  
SparkFun SerLCD V2.5 Notes



## **Microcontrollers**

Wikipedia: Microcontroller

Parallax Book TOCs (PDFs in code archive)

What is a Microcontroller?

Understanding Signals

Smart Sensors and Applications

Robotics with the Boe-Bot

Introduction to the Arduino Board, Arduino Hardware Index, Arduino Diecimila

Arduino/Wiring Language Reference, Example List, Libraries, Extensions

Tutorial: Getting Started with Arduino: Massimo Banzi

The CREATE USB Interface: Where Art Meets Electronics: Dan Overholt

d.tools Hardware Resources

Velleman USB Interface Board Data Sheet

Open Hardware Projects List

## **Applications**

Time Factors in Interface Design for Augmenting Human Intellect: A. Goppold

Musical Interaction Design with the CREATE USB Interface: Teaching HCI with CUIs instead of GUIs: Dan Overholt

The Acoustic, the Digital, and the Body: Thor Magnusson, Enrike Hurtado Mendieta

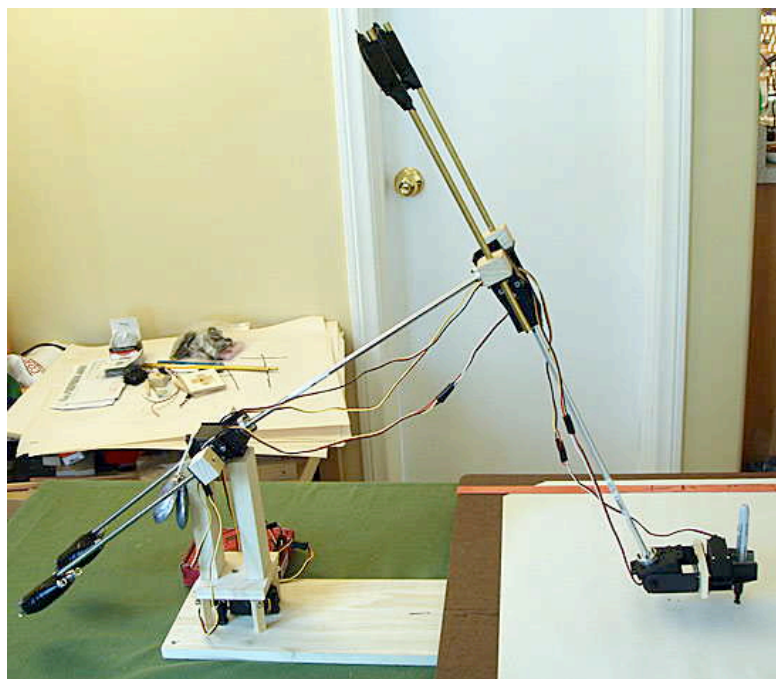
Musical Performance Practice on Sensor-based Instruments: Atau Tanaka

Defining a control standard for easily integrating haptic virtual environments with existing audio/visual systems: Stephen Sinclair and Marcelo M. Wanderley

uOSC: The Open Sound Control Reference Platform for Embedded Devices: Andy Schmeder and Adrian Freed

The Multimodal Music Stand: Bo Bell, et al.

A Force Sensitive Multi-touch Array Supporting Multiple 2-D Musical Control Structures: David Wessel, Rimas Avizienis, and Matthew Wright



## MAT 594O Code Archive

- Wiring/Arduino SW
- CUI Frameworks
- d.tools
- References & doc.
- MicroOSC
- Phidgets SW

▼ Arduino	⊕	39.4 MB
▶ Accel_to_Serial_stp	⊕	178.2 KB
▶ AnalogInput_stp	⊕	142.2 KB
▶ ASCIITable_stp	⊕	166.2 KB
▶ Blink_stp	⊕	137.3 KB
▶ BlinkM	⊕	3.6 MB
▶ Firmata-2.0beta1	⊕	4.5 MB
▶ inp_graph	⊕	2.5 KB
▶ MD_Arduino	⊕	719.6 KB
▶ Mic-Testing	⊕	33.3 KB
▶ ofxCppGlue	⊕	29.8 MB
▼ arduino-0011	⊕	175.5 MB
▶ drivers	⊕	410.4 KB
▶ examples	⊕	602.8 KB
▶ hardware	⊕	170.6 MB
▶ Metro	⊕	153.8 KB
▶ reference	⊕	838.8 KB
Arduino 11.app	⊕	2.9 MB
▶ CUI	⊕	8.3 MB
▶ d-tools	⊕	140.2 KB
▼ Documents	⊕	514.2 MB
▶ Courses	⊕	18 MB
▶ CPUs	⊕	66.6 MB
▶ Electronics	⊕	249.6 MB
▶ NIME	⊕	5.6 MB
▶ Output	⊕	3.1 MB
▶ Sensors	⊕	8.5 MB
▶ Software	⊕	47.7 MB
▶ Theory	⊕	114.8 MB
▶ PhidgetsSW	⊕	2.9 MB
▶ Processing 0148	⊕	35.8 MB
▼ uosc-1.02	⊕	5.8 MB
▶ doc	⊕	4.9 MB
▶ hex	⊕	289.2 KB
▶ max	⊕	64.5 KB
▶ py	⊕	56.1 KB
▶ win	⊕	7 KB
Boot Down.app	⊕	449.3 KB
a.out	⊕	12.6 KB
uosc-translate.c	⊕	4.1 KB
▶ Wiring 16	⊕	115.6 MB

**MEDIA ARTS & TECHNOLOGY PROGRAM**

# MAT 594O: Sensors and Interfaces for Media Art

Formerly "Media Interface Technology"

UCSB, Fall, 2008

Instructors

Stephen Pope, Matt Wright, Amichi Amar

<http://www.mat.ucsb.edu/594O>



1

## Course Objectives

- Understand the role of space and human interaction in dynamic art
- Survey human-computer interface (HCI) theory & design techniques
- Combine physical transducers & microcontrollers into sensor systems
- Develop and refine concrete interactive prototype applications



2

## Topics

- Space & gestural interaction
- Human-computer interfaces: ergonomics & haptics
- Transducers, sensors, signal capture & conditioning
- Microcontrollers & interfaces
- Application projects



3

## Logistics

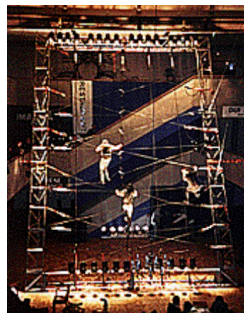
- Place: Music 2215 or SouthHall 4430
- Time: Tu/Th 5:00 - 6:50 PM
- Materials: book, code, hardware kit
- Web: <http://www.mat.ucsb.edu/594O>
- Email: <http://www.mat.ucsb.edu/mailman...>
- Grading: class participation, quizzes, presentations, projects



4

## Course Format

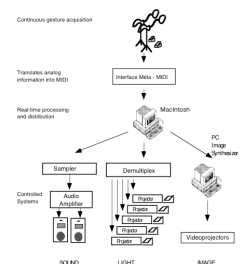
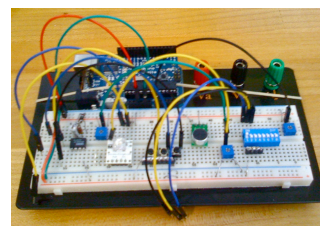
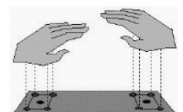
- Lectures
- Seminar Presentations
- Lab Sessions
- Group Projects
- Individual Projects



5

## Results

- Tools, Instruments, Installations



6

## Prerequisites

- Conceptual: ideas for interactive art
- Physics: forces & waves, acoustics
- Electronics: components and wires
- Mechanical: use of hand tools
- Software: programming fundamentals



7

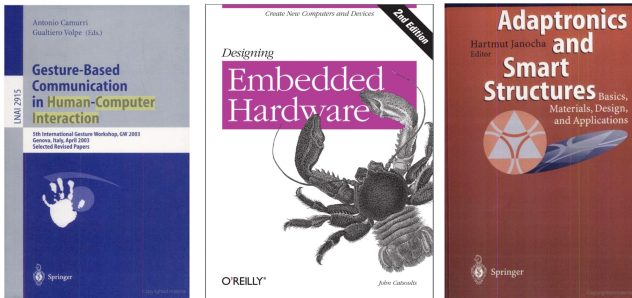
## Course Materials

- Books
- Readings
- Web
- Hardware sources
- Software sources
- Content sources



8

## References



9

## Books: General

- IRCAM Gesture Sensor Readings (PDFs)
- Brenda Laurel: The Art of Human-Computer Interface Design (cheap used)
- John Catsoulis: Designing Embedded Hardware (O'Reilly) (PDF)



10

## Books: Sensors

- Jacob Fraden: AIP Handbook of Modern Sensors
- Ramon Pallas-Areny and John G. Webster: Sensors and Signal Conditioning
- John Brignell & Neil White: Intelligent Sensor Systems
- H.R. Everett: Sensors for Mobile Robots



11

## Other Publications

- Sensors Magazine - Free!
- Circuit Cellar - Best EE-hacker magazine
- NASA Tech Briefs
- IEEE Sensors Journal
- Make (makezine)
- Robotics (botmag)
- Servo Magazine (servomagazine)



12

## Events

- Sensors Expo
- IEEE Sensors Conference
- Transducers Conference
- ACM UIST (UI SW tech, since 1988)
- NIME (on-line proceedings)
- ICMC, ISEA, ArtBot (applied)



13

## Sites

- SensorWiki: [www.sensorwiki.org](http://www.sensorwiki.org)
- <http://www.sensorsportal.com>
- <http://www.sensorsmag.com>
- <http://www.cs.cmu.edu/~chuck/robotpg/robofaq/10.html>
- <http://www.nime.org>
- <http://www.billbuxton.com/InputSources.html>
- <http://www.allaboutcircuits.com/>
- See links on course home page



14

## Hardware Source Sites

- SparkFun, Wulfdan
- Robotshop, Parallax
- Phidgets, Lego
- Microcontrollershop
- Digikey, Mouser
- Newark, Allied
- Marvac (local)

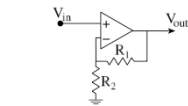


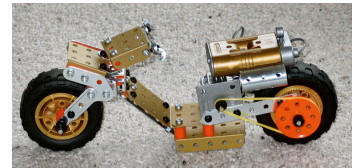
Figure 17: Non-Inverting Amplifier



15

## Software Sources

- See links from Course site
- Arduino
- Wiring
- CUI
- uOSC
- Robotics sites
- BASIC Stamp resources



16

## MAT 594O Code Archive

- Wiring/Arduino SW
- CUI Frameworks
- d.tools
- References & doc.
- MicroOSC
- Phidgets SW

▼ Arduino	39.4 MB
▶ Accel_to_Serial_stp	178.2 KB
▶ AnalogInput_stp	142.2 KB
▶ ASCTable_stp	166.2 KB
▶ Blink_stp	137.3 KB
▶ BlinkM	3.6 MB
▶ Firmata-2.0beta1	4.5 MB
▶ inp_graph	2.5 KB
▶ MD_Arduino	713.6 KB
▶ Mic-Testing	33.3 KB
▶ ofxCppGlue	29.8 MB
▼ arduino-0011	175.5 MB
▶ drivers	410.4 KB
▶ examples	602.8 KB
▶ hardware	170.6 MB
▶ Metro	153.8 KB
▶ reference	838.8 KB
▶ Arduino 1.1.app	2.9 MB
▶ CUI	8.3 MB
▶ d-tools	140.2 KB
▶ Documents	514.2 MB
▶ Courses	18 MB
▶ CPUs	66.6 MB
▶ Electronics	249.6 MB
▶ NIME	5.6 MB
▶ Output	3.1 MB
▶ Sensors	8.5 MB
▶ Software	47.7 MB
▶ Theory	114.8 MB
▶ PhidgetsSW	2.9 MB
▶ Processing 0148	35.8 MB
▼ uosc-1.02	5.8 MB
▶ doc	4.9 MB
▶ hex	289.2 KB
▶ max	64.5 KB
▶ py	56.1 KB
▶ win	7 KB
▶ Boot Down.app	449.3 KB
▶ a.out	12.6 KB
▶ uosc-translate.c	4.1 KB
▶ Wiring 16	115.6 MB



17

## Outline

1. Space & gesture
2. Electronics
3. Human-computer interfaces
4. Transducers & sensors
5. Microcontrollers and interfaces
6. Application integration



18

## 1. Space & Gestural Interaction

### Readings

- Art, narrative, space

### Demos

- Bill Buxton's Encyclopedia on Input Sources
- Bill Verplank's controllers reel

### Lecture: Spatial interaction with media

### Exercises

- Instrument, tool, object, installation



19

## 2. Electronics Introduction

### Readings:

- Electronics tutorials
- Sensor introductions



### Demos

- falstad.com demo applets

### Lecture: Circuits and Sensors

### Exercises

- Basic Bread-boarding
- Simple circuits

20

## 3. Human-computer Interfaces

### Readings

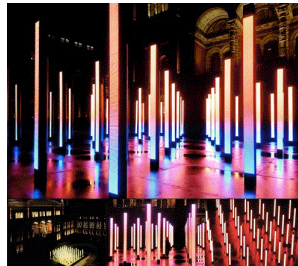
- HCI Design
- Tools & Methods for HCI

### Demos

- HCI design survey: the best & the worst

### Lecture: HCI for the Arts

### Exercises



21

## 4. Transducers & Sensors

### Readings

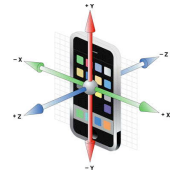
- Arduino, Phidgets
- Buxton Sensor References
- Wikipedia
- Data sheets

### Demos

### Lecture: Sensors and their Signals

### Exercises

- Arduino/CUI Examples
- Using op-amps



22

## 5. Microcontrollers and Interfaces

### Readings

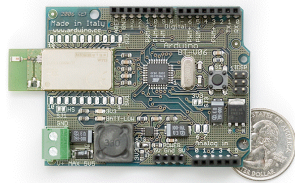
- Arduino, CUI, Wiring, Parallax Doc
- Microcontroller data sheets

### Demos

### Lecture: Single-chip Systems: Circuits and Software

### Exercises

- Arduino basics
- Using Wiring & Arduino



23

## 6. Application Integration

### Readings

- Application Examples, Surveys
- Supporting technologies
- Local efforts

### Demos

### Lecture: Putting it all together

### Exercises

- Client/server glue code with Max, CSL, SC, etc.

24

## Not Covered Here

- Content, intention in art
- Sensor design, physics
- Assembly-language programming
- "Deeper" electronics (> op-amps)
- Mechanical construction
- MAT 240E: MIDI/OSC
- MAT 201B: Computing with Media Data



25

## Course Hardware

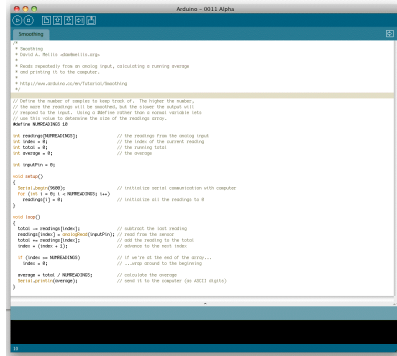
- Basic Tools, Components, Bread-board
  - Soldering, mechanical tools
  - Bread-board requirements & components
- Sensor/Microcontroller kits
  - Arduino (many flavors)
  - CREATE User Interface (CUI)
  - Wiring
  - Others
- See shopping lists below



26

## Course Software

- Clients
  - Wiring
  - C/C++/Java
- Servers
  - CSL, GLV (native)
  - OSC or MIDI
  - Max, Jitter, Pd
  - SuperCollider



27

## Why MAT 594O?

- Inspiration: new interaction media
  - "The most beautiful thing we can experience is the mysterious. It is the source of all true art and science." –Albert Einstein
- Implementation: assumptions
  - Interaction revolution – possibilities
  - Small, low-cost sensors easily available
  - Moore's Law – WAN + distr. sensors
  - What will come after keyboard and mouse...
  - Sensors are permeating everything



28

## Context: Relatives

- Related Courses Elsewhere/Elsewhen
  - UCSB "Media Interface Technology" course
  - Stanford/CCRMA HCI Workshops
  - MIT, Cornell, Columbia, USC, UC, Princeton
- UCSB Courses
  - MAT 200A: Art & Technology
  - MAT 200B: Music & Technology
  - MAT 201B: Computing with Media Data
  - MAT 240A-F: Digital Audio Programming



29

## The Big Context Question

- Setting the stage: what is the field?
  - Interactive systems
  - Media arts applications
  - Physical inputs (sensors = electronic nerves)
  - Physical outputs (actuators = electronic muscles)
- Related fields
  - Robotics, "Adaptronics," Meccano
  - Gaming systems, circuit-bending
  - Automotive, medical, defense electronics
  - Mobile/wearable/ubiquitous computing
  - Industrial design



30

## Context: StateOfTheArt(s)

- Art piece, instrument, tool, appliance
- Sensors in appliances
- Mobile devices
- Digital hub
- Wireless media
- Merge of art, game, education, expression, communication, exploration



Figure 3. August Black playing *El Lechero*.

31

## Jumping Ahead

- Bongers Art HCI paper
- Buxton sensor survey
- Arduino getting started doc
- ICMC06 CUI apps paper

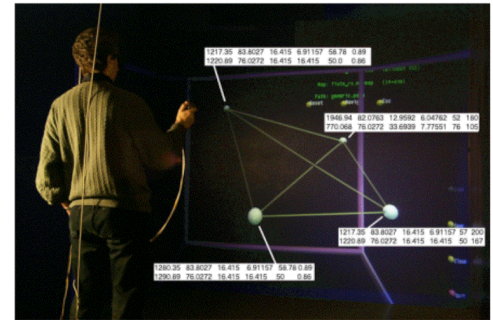
32



33

## Topic 1

- Space & Gestural Interaction



34

## Space & Gestural Interaction

- Readings
  - Wikipedia: Space & Time, Narrative, Interactive Art, Installation Art
  - Spatio-Temporal Perspectives: A new way for cognitive enhancement
  - The Spatial Narrative: Animation and Art Installation: L. M. Pepi
  - A Manifold Interface for Kinesthetic Notation in High-Dimensional Systems
- Demos
  - Bill Buxton's Encyclopedia on Input Sources
  - Bill Verplank's controllers reel
- Lecture: Spatial interaction with media
- Exercises
  - Instrument, tool, object, installation

35

## Spatial Interaction with Media

- Ontology: time and space
- Epistemology: perception and the links to time and space
- Cognition and self-awareness in space
- Information as state-change
- Media: sensory data vs. content

36

## Historical Context

- 50 yrs: electronic, automatic, programmed signal processing, computers
- 500 yrs: book printing, mechanical processing of written materials
- 5.000 yrs: history of world civilizations, writing, the alphabet: 2500 yrs
- 50.000 yrs: pictorial- / artefact- patterns of Homo Sapiens Sapiens
- 500.000 yrs: tool- / fire- / ritual- / language- patterns of Homo Sapiens
- 5.000.000 yrs: gestics- / sound- / tool- patterns of anthropoids
- 50.000.000 yrs: behavioral pattern transmission of mammals and birds
- 500.000.000 yrs: metazoa (multicellular organisms), Eukaryotes about 1 gig-a.
- 5.000.000.000 yrs: age of the earth, chemical-biological evol., Prokaryotes 3.5g
- 15.000.000.000 yrs: "Big Bang", age of the universe

- Spatio-gestural interaction vs. symbolic language



37

## So, what is art?

- "the use of skill and imagination in the creation of aesthetic objects, environments, or experiences that can be shared with others." (EB)
- "It is now taken for granted that nothing which concerns art can be taken for granted any more." (TA)



38

## Non-motivated Functions of Art

- Basic human instinct for harmony, balance, rhythm
  - Imitation, then, is one instinct of our nature. – Aristotle
- Experience of the mysterious
  - The most beautiful thing we can experience is the mysterious. It is the source of all true art and science." –Albert Einstein
- Expression of the imagination
- Universal communication
- Ritualistic and symbolic functions



39

## Motivated Functions of Art

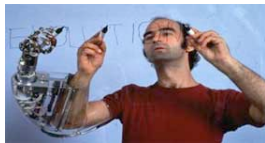
- Communication
  - [...] artefacts or images with symbolic meanings as a means of communication." –Steve Mithen
- Entertainment
- The Avante-Garde--Art for political change
- Art for psychological and healing purposes
- Art for social inquiry, subversion and/or anarchy
- Art for propaganda or commercialism



40

## Media-mediated Art

- Interactive art
  - "Spectator/participant" has agency
- Installation art
  - Takes into account the "viewer's" entire sensory experience
- Relation to immersive environments?
- Relation to Gesamtkunstwerk?



41

## Art-as-Theater

Element	In Drama	In Human-Computer Activity
Action	The whole action being represented. The action is theoretically the same in every performance.	The whole action, as it is collaboratively shaped by system and user. The action may vary in each interactive session.
Character	Bundles of predispositions and traits, inferred from agents' patterns of choice.	The same as in drama, but including agents of both human and computer origin.
Thought	Inferred internal processes leading to choice: cognition, emotion, and reason.	The same as in drama, but including processes of both human and computer origin.
Language	The selection and arrangement of words; the use of language.	The selection and arrangement of signs, including verbal, visual, auditory and other nonverbal phenomena when used semiotically.
Melody (Pattern)	Everything that is heard, but especially the melody of speech.	The pleasurable perception of pattern in sensory phenomena.
Spectacle (Enactment)	Everything that is seen.	The sensory dimensions of the action being represented: visual, auditory, kinesthetic and tactile, and potentially all others.

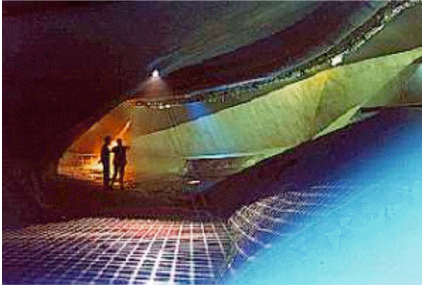
- Brenda Laurel's 6 elements from NMR



42

## Space and Media

- The dream (?): Virtual space with multi-modal content projection and interaction



43

## Space and Gesture

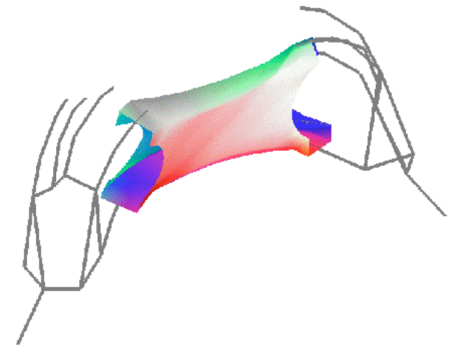
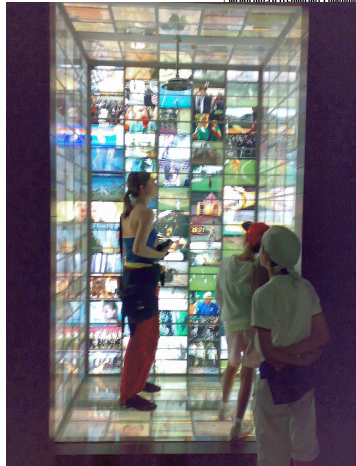


Fig. 13. Example of the sheet clamped to the index and thumb tips of both hands.

44

Ubiquitous  
sensing,  
computation,  
artistic  
content and  
presentation



45

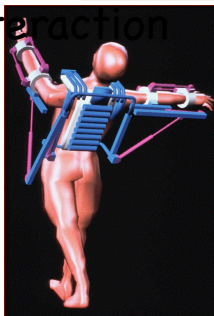
## Space, Time and Perception

- "the visual perception of space [is tied to] to prior bodily experience" - Alhacen, 1021
- The body is the ultimate instrument of all our external knowledge, whether intellectual or practical, experience [is] always in terms of the world to which we are attending from our body. — Michael Polanyi

46

## Human Spatial Interaction

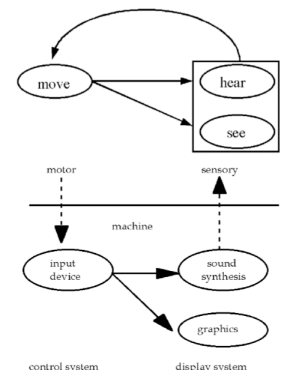
- Loop:
  - Observation
  - Orientation
  - Decision
  - Action
- "Conceptual Navigation"
- Semiotics/semantics perspective of spatio-temporal symbols



47

## Feedback and Sensation

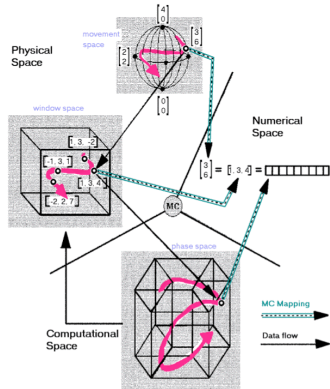
- We learn to interact through constant (multi-) sensory feedback (i.e., hand-eye coordination)



48

## Numerical Spaces

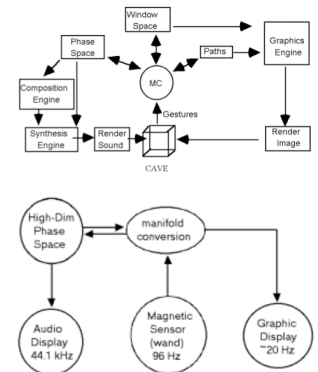
- Manifolds as functions operating on spatial mappings in different spaces
- Human orientation space != machine space



49

## Manifold Controller (I. Choi)

- Gestures mapped to medium-specific numerical spaces
- Output to image & sound projection in a CAVE



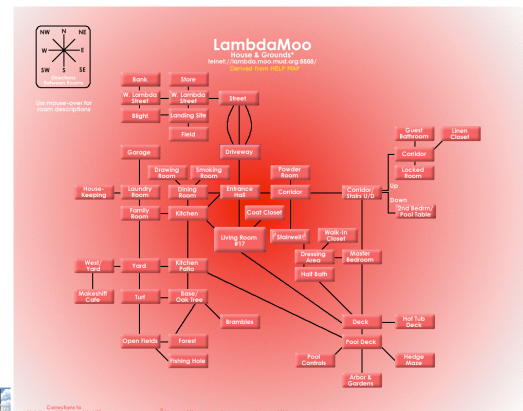
50

## Guidelines (from CU)

- Make the interaction intuitive, avoiding instructions exterior to the depicted world
- Use simple clear metaphors for interaction and navigation
- Expand/transgress familiar models by translating them into the digital world
- Allow a visible continuity between cause and effect
- Provide devices for focusing attention on crucial areas (hot spots)
- Define the placement, role and extent of control of the author and reader
- Determine whether the narrative is to be arranged in time or in space or in a combination of both

51

## Space and Maps



52

## What is a Gesture?

- "a motion of the body that contains information" (Kurtenbach & Hulteen, 1990)
- Waving good-bye, beckoning with your finger, sign languages - yes
- Typing on a keyboard, hand-writing - no
- Using a game controller - ??

53

## Gesture Recognition (M Turk)

- Theory level
  - What is a gesture? What is a gesture event?
  - What is the desired output?
  - What is the context of the gesture?
- Representation/Algorithm level
  - What features are important/relevant?
  - How to compute the features?
  - How to represent temporal aspects of gesture?
  - What specifically does recognition entail?
- Implementation
  - Are the sensors adequate?
  - Which classification technique is best? How much training needed?
  - How to initialize? Is it fast enough?

54

## MT's SOTA in GR

- My claim is that we have not done a good job in clarifying the "thinking vs. building" distinction, nor in keeping Marr's levels separate
- As a result, there is no underlying theory of gesture recognition. We're just applying computer vision and pattern recognition to various, loosely-related tasks.



55

## Gesture Theory

- What does it mean to do gesture recognition?
  - Just classification? ("Gesture #32 just occurred")
  - Semantic interpretation? ("He is waving goodbye")
- What is a gesture?
  - Blinking? Scratching your chin? Jumping up and down? Smiling? Skipping?
- What is the purpose of gesture?
  - Communication? Getting rid of an itch? Expressing feelings?
- What is the context of gesture?
  - A conversation? Signaling? General feedback? Control?
  - How does context affect the recognition process?



56

## Human Gestures

- Humans can produce up to 700,000 different physical signs (M. Pei)
- 250,000 facial expressions (Birdwhistell)
- 5000 hand gestures (Krout)
- People gesture when talking on the phone (!)
- Blind people often gesture when talking to each other



57

## GKS Logical Device Model

- Locator (input x/y position)
- Stroke (series of positions)
- String (text input)
- Valuator (scalar value)
- Choice (menu options)
- Pick (picture options)
- (GKS/ISO 1983)



58

## K&H'90 Gesture in HCI

- Taxonomy/dimensions
  - Orientation
  - Connection
  - Pointing/grouping
  - Function
  - Physicality
- Aspects of gesture-tracking



59

## How do we get there from here?

- Art and HCI - similar considerations?
- Spatial interaction with media content
- Interactive objects as art, as media
- Immersive space as delivery mechanism
- Kay's "personal dynamic medium" for the arts



60

## Introduction (MIT Course)

- Interaction revolution - possibilities exploding
  - Small, low-cost sensors easily available to measure nearly everything...
- Moore's Law makes processors capable of meaningfully exploiting the data in real time.
- Low barriers to entry - easy to try things
  - Deaf and blind computers...
  - What will come after keyboard and mouse...
  - You can't realize your vision for the future of interactivity by buying a card and plugging it in...
- Sensors are permeating everything
  - From toys to automobiles to smart homes
  - From Burglar alarms to Ubiquitous Computing



61

## Sensors

- Sensor modes are intrinsically synaesthetic
- Use physics and constraints to couple a measured quantity into an unknown
  - Temperature can infer wind velocity (heat loss)
  - Displacement can infer:
    - Pressure (with a spring:  $F = kx$ )
    - Volume of fluid in a tank ( $V = Ah$ )
    - Velocity (measurements at 2 times:  $v = dx/dt$ )
    - Temperature (thermometer level)
    - Angle from vertical (displacement of a bubble)
- Measurements are used with a mathematical model to derive other parameters
  - Estimation and Kalman Filtering, etc.



62

## Next Steps

- Reflections on narrative and space in artistic content
- Reflections on artistic considerations in HCI design
- Roles of related media
  - Cinema, animation
  - Installation art
  - Computer games
  - On-line communities



63

## Review

- Space and gesture as components of consciousness and interaction
- Maps of space
- Mappings of spatial data
- Definitions of art



64

## What's next?

- Electronics readings
  - DanO's 594O course notes
  - Perry Cook's electronics intro
  - Putnam & Knapp Sensor Electronics
  - MIT sensors course
- Exercises



65

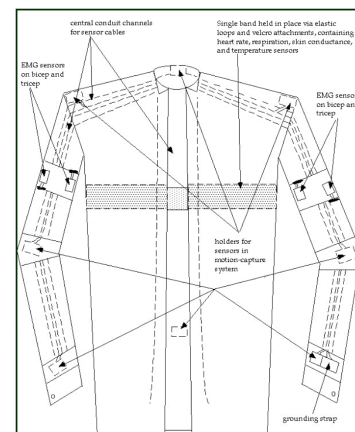


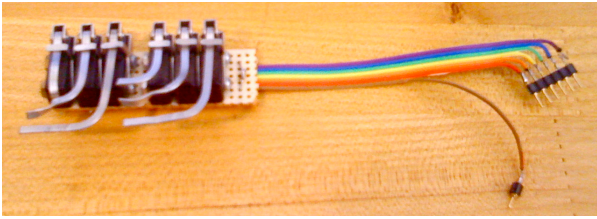
Fig. 1. Integration of physiological sensors into wearable form.



66

## Topic 2

### Electronics Introduction



67

## Electronics Introduction

### Readings:

- DanO's 594O course notes
- Perry Cook's electronics intro
- Putnam & Knapp Sensor Electronics
- MIT sensors course
- Other notes - op-amps, AllAboutCircuits.com

### Lecture: Circuits and Sensors

### Exercises

- Basic Bread-boarding
- Simple circuits

68

## Electronic Goals

- Learn to construct and modify input signal conditioning circuits and simple composite system circuits
- Not deep EE or math
- Combine of-the-shelf components on bread-boards with microcontrollers

69

## Circuits and Sensors

- Circuits and components
- Sub-systems and re-use
- Input, processing, output
- Constructing circuits
- Tools and supplies
- Circuit prototyping

70

## Mechanical/Electrical Analogs

Table 3.4. Mechanical, Thermal, and Electrical Analogies

MECHANICAL	THERMAL	ELECTRICAL	
MASS $M$ $F = M \frac{dv}{dt}$	CAPACITANCE $C$ $Q = C \frac{dT}{dt}$	INDUCTOR $L$ $V = L \frac{di}{dt}$	CAPACITOR $C$ $i = C \frac{dV}{dt}$
SPRING $k$ $F = k \int v dt$	CAPACITANCE $C$ $T = \frac{1}{C} \int Q dt$	CAPACITOR $C$ $V = \frac{1}{C} \int i dt$	INDUCTOR $L$ $i = \frac{1}{L} \int V dt$
DAMPER $b$ $F = bv$	RESISTANCE $R$ $Q = \frac{1}{R} (T_2 - T_1)$	RESISTOR $R$ $V = Ri$	RESISTOR $R$ $i = \frac{1}{R} V$

71

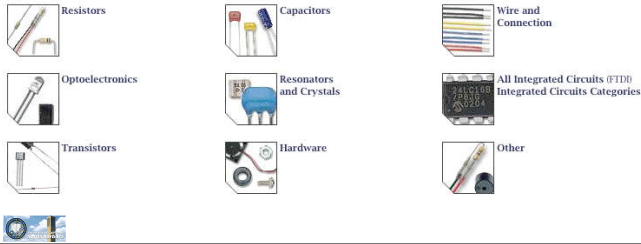
## Circuits and Components

- Flashlight circuit
- Voltage dividers: Ohm & Kirchoff
- Capacitors and Inductors
- Tubes and Transistors
- Operational Amplifiers
- Sensors: physical-to-electrical transducers (I or V sources)

72

## Kinds of Components

- Passive, active components
- Component, connector, transducer
- Discrete, IC



73

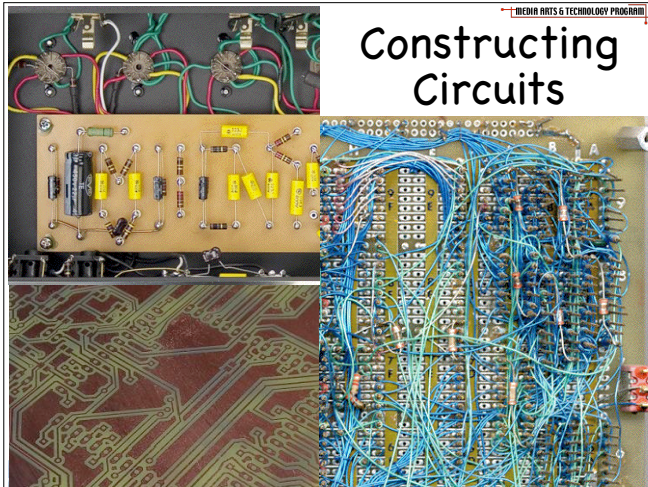
## Constructing Circuits

- Point-to-point wiring, soldering
- Wire-wrap boards
- Bread-boards
- Printed circuits
- ASICs



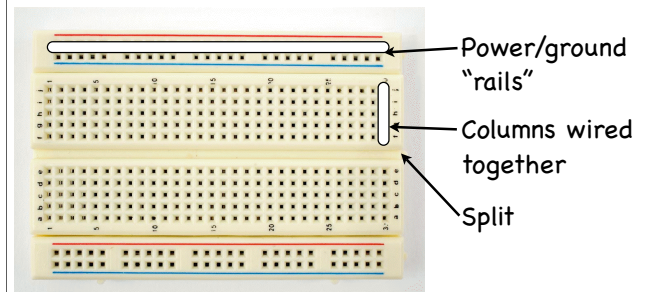
74

## Constructing Circuits



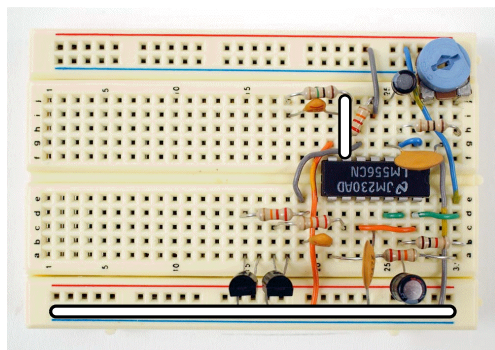
75

## Prototyping Bread-board



76

## Bread-board with Partial Circuit



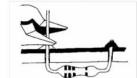
77

## Soldering

Make sure the solder joints are cone-shaped and shiny



Trim excess leads as close as possible to the solder joint



- Soldering tools
- Wire-to-wire
- Wire-to-binding post
- Wire-to-PCB trace
- Soldering guidelines



78

## Building Circuits

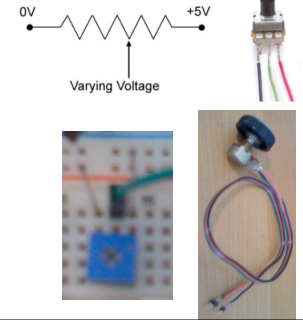
- From electronic circuit diagrams to actual boards: Wiring tutorial
- Reusable sub-systems
  - Input signal conditioning: signal scale/offset
  - Filters
  - Digital logic
  - Output processing: drivers, relays



79

## Making Circuits

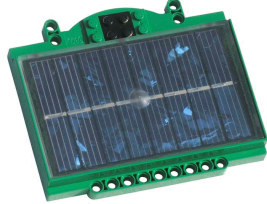
- Simplest circuit: voltage divider
- Simple op-amp circuits
- Using bread-boards



80

## Electronics Play

- Exercises
  - Basic bread-boarding
  - Simple circuits
    - Voltage dividers
    - Using switches
    - Off-board connectors
    - Power supplies
  - Soldering
  - Using op-amps in circuits



81

## Shopping List

- Basic CPU (Arduino, CUI, Wiring)
- Bread-board(s)
- Power supply
- Wires, connectors (jumpers, ribbon, I/O)
- Resistors, caps, op-amps, pots
- Inputs: switches, sensors (camera/mic)
- Outputs: LEDs, motors, relays, actuators
- Digital: USB/MIDI/IP



82

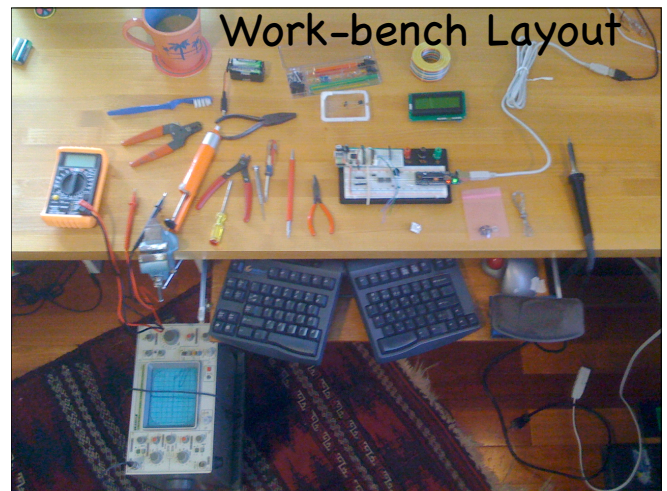
## Tools

- Screw drivers
- Pliers, clippers
- Soldering iron
- Vise, holders
- Wire, cables



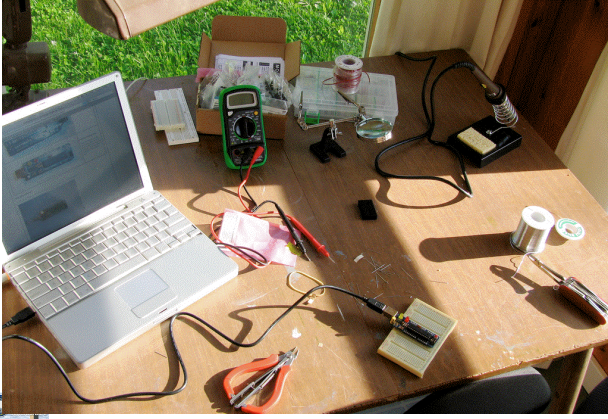
83

## Work-bench Layout



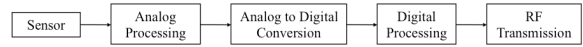
84

## Work-bench Layout 2



85

## Input Signal Conditioning



- Sensor/transducer
  - Possibly (frequently) multi-stage
- Analog signal processing
  - Scale/offset
  - Filter, de-noise
- A-to-D conversion
- Digital processing



86

## Sensor/Actuator/Control Circuits

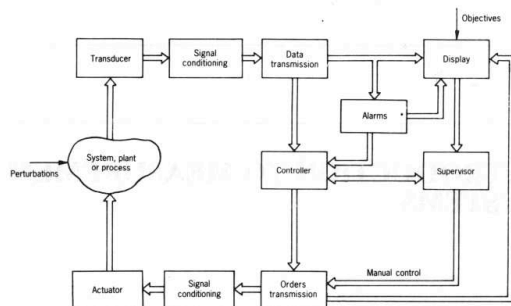


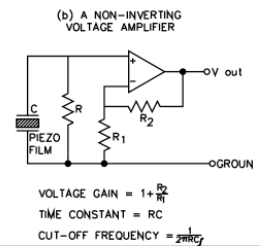
FIGURE 1.1 General structure of a measurement and control system.



87

## Input Circuits

- Op-amp circuits
  - Inverting/noninverting
  - Signal scale (gain) vs. offset (bias)



88

## Active Filters

- Using a capacitor in the feedback (low-pass) or the input path (hi-pass)

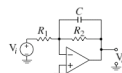


Figure 22: Single Pole Low-Pass Filter

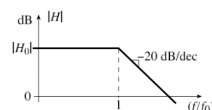


Figure 23: Frequency Response of Single Pole Lowpass Filter

$$f_0 = \frac{1}{2\pi R_1 C}$$

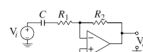


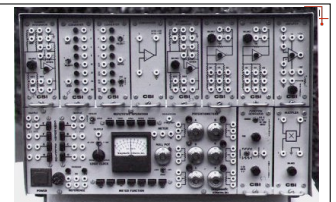
Figure 24: Single Pole High-Pass Filter



89

## Electronics Lab

- Discrete circuits
- Op-amps in circuits
- Sensor signal processing
- Integration
  - Power supply
  - Sensor input signal conditioning
  - Off-board connections



90

## Review

- Electronics for sensor input signal conditioning
- Simple circuits
- Using op-amps
- Filters and analog DSP



91

## What's Next?

- More electronics: microcontrollers
- Topic 3: HCI design for the arts
  - Readings
- Using microcontroller kits
- Applications



92



93

## Topic 3

### • Human-computer Interfaces

	Time	Space	Weight
Slashing	Quick	Flexible	Strong
Gliding	Sustained	Direct	Light
Pressing	Sustained	Direct	Strong
Flicking	Quick	Flexible	Light
Wringing	Sustained	Flexible	Strong
Dabbing	Quick	Direct	Light
Punching	Quick	Direct	Strong
Floating	Sustained	Flexible	Light

Eight Basic Efforts.



94

## Human-computer Interfaces

- Readings
  - Physical Interfaces in the Electronic Arts: Bert Bongers
  - Less is More (More or Less): Uncommon Sense and the Design of Computers: Bill Buxton
  - How Bodies Matter: Five Themes for Interaction Design: S. Klemmer, B. Hartmann & L. Takayama
  - Reflective Physical Prototyping through Integrated Design, Test, and Analysis: B. Hartmann, S. Klemmer
- Lecture: HCI for the Arts
- Exercises
  - Sensors and Interaction



95

## HCI for the Arts

- HCI Background
- HCI Context
- HCI Challenges
- HCI in the arts



96

## HCI Background

- Used to be MMI, now HCI
- For the arts: HAI?
- Relation to ergonomics, human-factors engineering, usability, user experience
- Originally dictated by hardware
- The HCI bottleneck and Moore's law
- Kay's definition of the application



97

## Evolution of User Interfaces

<u>When</u>	<u>Implementation</u>	<u>Paradigm</u>
1950s	Switches, punched cards	None
1970s	Command-line interface	Typewriter
1980s	Graphical UI (GUI)	Desktop
2000s	Perceptual UI (PUI)	Natural interaction

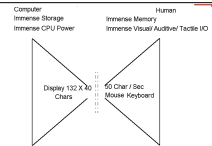
- After Turk



98

## HCI Challenges

- Display bottleneck**
  - Compare the 1000-fold increase in RAM and HD capacity and similar increase in CPU processor power to almost standstill in display technology.
- WIMP bottleneck**
  - Keyboard access (for touch typists) about 10 times faster than mouse click (Buxton on deaf Napoleon)
  - Loss of shell scripting and macro programming due to WIMP paradigm



99

## HCI Data Paths

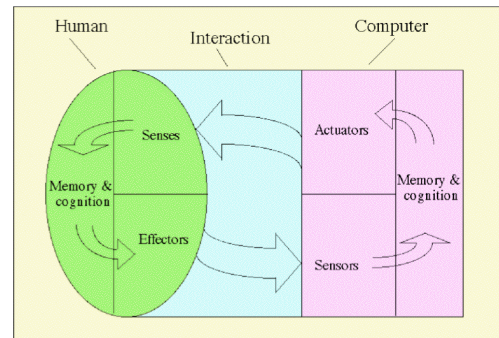


Fig. 1. Human-Machine Interaction.



100

## Multi-media, -sensory, -actuator

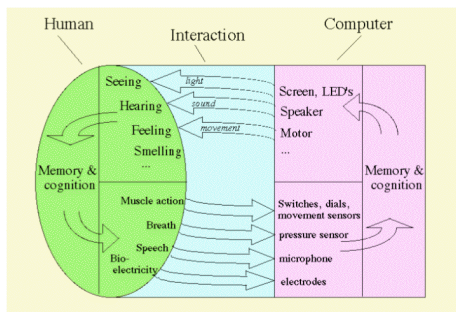


Fig. 2. Examples of modalities.



101

## Creator/Audience

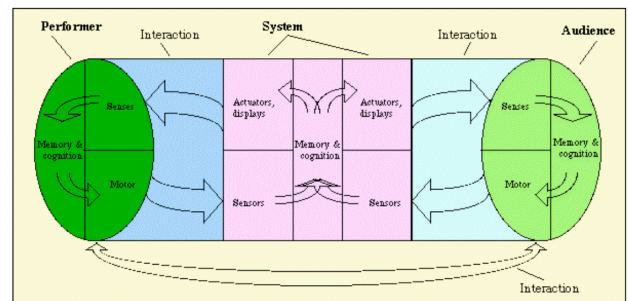


Fig. 7. The interactions between performer, system and audience.

- "Performer" and "audience" may be the same



102

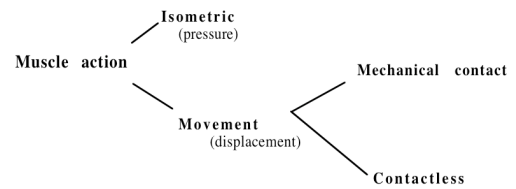
## OODA in HCI

### See above

- Physical inputs (sensors = electronic nerves)
- Physical outputs (actuators = electronic muscles)
- Interaction design guidelines
- Mappings between representations/spaces

103

## Action Taxonomies



### Categories of physical action

104

## Gesture Taxonomies



### Symbolic

- Conventional, context-independent, and typically unambiguous expressions (e.g., "OK" or peace signs)

### Deictic

- Gestures that point to entities, analogous to natural language deixis (e.g., "this is not that" or "put that there")

### Iconic

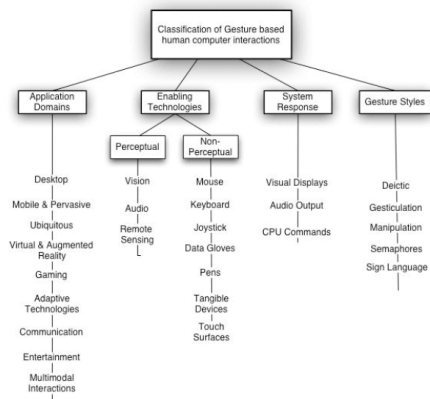
- Gestures that are used to display objects, spatial relations, and actions (e.g., illustrating the orientation of two robots at a collision scene)

### Pantomimic

- Gestures that display an invisible object or tool (e.g., making a fist and moving to indicating a hammer)

105

## Karam & Scheäfel's Categories



106

## Mapping Signal Flow

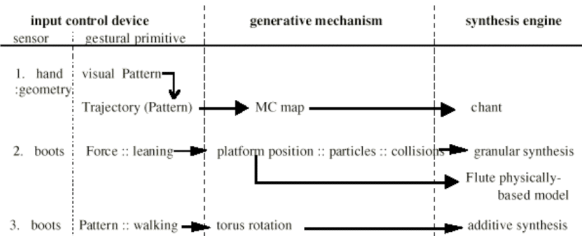


Fig. 15. Control Signal flow in experimental systems for real-time performances discussed in Section 4.

107

## HCI Design Processes

### From SW Engineering

- Rapid prototyping
- Incremental refinement
- User/customer involvement
- Focus on interfaces
- Libraries of reusable components
- CASE tools

108

## Five Themes for HCI Design

- 💡 Thinking through doing
- 💡 Performance
- 💡 Visibility
- 💡 Risk
- 💡 Thick practice
- 💡 (Klemmer, Hartmann & Takayama)



109

## HCI Design

- 💡 User level and user training
  - 💡 Novice vs expert interfaces
  - 💡 Learnability and surprise
- 💡 Customization and adaptation
- 💡 User sensing and input modalities
- 💡 Feedback media (haptic and otherwise)



110

## Advice for Novel HCI Design

- 💡 (See Laurel, ed.)
- 💡 Task analysis - user analysis
  - 💡 User requirements
  - 💡 User model space
  - 💡 Prototyping and user testing
- 💡 The HCI designer is normally not the typical user



111

## Generating New Ideas

- 💡 New uses for the object
- 💡 Adapt the object to be like something else
- 💡 Modify the object for a new purpose
- 💡 Magnify--add to the object
- 💡 Minimize--subtract from the object
- 💡 Substitute something similar
- 💡 Rearrange the data
- 💡 Reverse or transpose the information



112

## User Questions

- 💡 Goal-oriented: what can I do?
- 💡 Descriptive: what is this?
- 💡 Procedural: how do I do this?
- 💡 Interpretive: what does this mean?
- 💡 Navigational: where am I?
- 💡 (Laurel, ed. chapter on help systems)



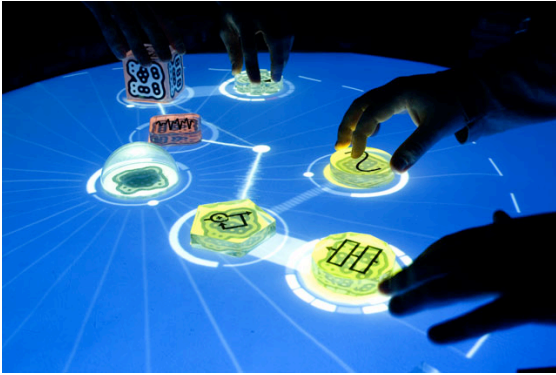
113

## Example: Air Traffic Control



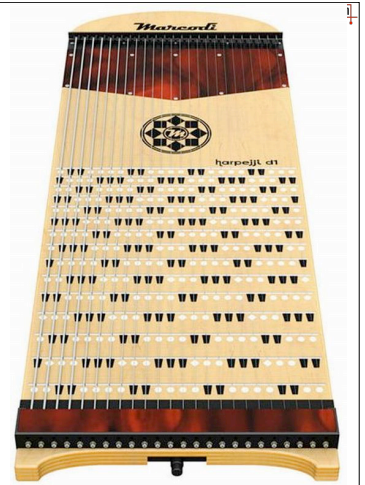
114

## Example: Table Interfaces



115

## Example: New Instruments (Harpejji)



116

## Example: String Tree



117

## HCI design for the Arts

- Tools vs instruments
  - Tool:
  - Instrument:
  - Exceptions:
- What's an Art HCI?
  - Obvious vs learnable vs unlearnable
  - Object vs system

118

## HCI Tools and Support

- HCI (GUI) software: GUIDEs
- Non-GUI HCI
  - Industrial design
  - Arts applications
- Custom HCI frameworks: d.tools

119

## Comment: Language Machines

- In the popular mythology the computer is a mathematics machine: it is designed to do numerical calculations. Yet it is really a language machine: its fundamental power lies in its ability to manipulate linguistic tokens—symbols to which meaning has been assigned. (Terry Winograd, SciAm, 1984)

120

## Review

- HCI & design & the arts
- Applications as new media
- Hyper-instruments
- Multi-modal augmented tools



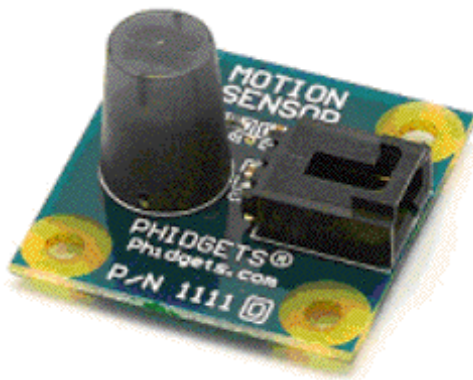
121

## What's Next?

- More HCI for the arts
- Sensors and user input
- Topic 4 Readings
  - Transducers & sensors
  - Sensor signal processing



122



123

## Topic 4

- Transducers & Signals



124

## Transducers & Signals

- Readings
  - Arduino: Interfacing with Hardware (I/O device list)
  - Wikipedia, Phidgets.com, Society of Robots Lists
  - NYU ITP Sensor Reports
  - Directory of Sources for Input Devices: B Buxton
  - Wikipedia: Accelerometer + applications
  - Wikipedia: Piezoelectric Sensor, Solenoid
  - Multi-touch Systems I have Known and Loved: Bill Buxton
  - SparkFun SerLCD V2.5 Notes
- Lecture: Sensors and their Signals
- Exercises
  - Arduino/CUI Examples



125

## Sensors and their Signals

- Sensor: physical signal to electrical signal transducer
  - Many kinds of physical signals
    - Position, pressure, temperature, wind, sound, moisture...
  - Several variables in electrical signals
    - Voltage vs current source
    - Signal scale/bias
    - AC signal frequency response
- Sensors in our daily lives
  - Car, kitchen, toy, environment



126

**Table 1.6. Stimulus**

Acoustic	Mechanical
Wave amplitude, phase, polarization	Position (linear, angular)
Spectrum	Acceleration
Wave velocity	Force
Other	Stress, pressure
Biological	Strain
Biomass (types, concentration, states)	Mass, density
Other	Moment, torque
Chemical	Speed of flow, rate of mass transport
Components (identities, concentration, states)	Shape, roughness, orientation
Other	Stiffness, compliance
Electric	Viscosity
Charge, current	Crystallinity, structural integrity
Potential, voltage	Other
Electric field (amplitude, phase, polarization, spectrum)	Radiation
Conductivity	Type
Permittivity	Energy
Other	Intensity
Magnetic	Other
Magnetic field (amplitude, phase, polarization, spectrum)	Thermal
Magnetic flux	Temperature
Permeability	Flux
Other	Specific heat
Optical	Thermal conductivity
Wave amplitude, phase, polarization, spectrum	Other
Wave velocity	
Refractive index	
Emissivity	
reflectivity, absorption	
Other	

## Categories of Stimulae

127

## Basic Physical Variables

Table 3.1. Classification of basic physical variables

	Flux ('through' variable)		Potential ('across' variable)	
	State	Rate	Rate	State
General (basic relationship)	$y$	$\dot{y} = \frac{dy}{dt}$	$\dot{x} = \frac{dx}{dt}$	$x$
Mechanical-translational	Momentum	Force	Velocity	Displacement
Mechanical-rotational	Angular momentum	Torque	Angular velocity	Displacement
Electrical	Charge	Current	Voltage	Flux linkages
Fluid flow	Volume	Flow rate	Pressure	-
Thermal	Heat	Heat flow rate	Temperature	-

128

## Transducer/Sensor Properties

Table 1.1. Specifications

Sensitivity	Stimulus range (span)
Stability (short and long term)	Resolution
Accuracy	Selectivity
Speed of response	Environmental conditions
Overload characteristics	Linearity
Hysteresis	Dead band
Operating life	Output format
Cost, size, weight	Other

129

## Sensors

- Relative to absolute references
- Relative to relative references
- Send/receive sensors
- Sense-related (or not)



Acceleration & Tilt



Color & Light



Compass & GPS



Object Detection



Temperature & Humidity



Pressure, Flex, & RPM

130

## Physical Transducers

Physical Property	position	rotary position	velocity	rotary velocity	isometric force	isotonic force	isometric rotary force	isotonic rotary force
discrete	key button	rotary switch						
infinite	fader touchscreen tracker	mod. wheel bend sensor rotary pot. abs. joystick	mouse	dial trackball	aftertouch isometric joystick (Finger-printR)	accelero-meter	isometric joystick (Finger-printR)	pitch-bend wheel spring-mounted joystick

Table 1. A categorization of transducers.

- Physical position/angle/force transducers: switch, pot, key...

131

## Common Sensors 1

- Air pressure sensors:** detects changes in air pressure resulted from opening doors and windows
- Capacitive:** detectors of human body capacitance
- Acoustic:** detectors of sound produced by people
- Photoelectric:** interruption of light beams by moving objects
- Optoelectric:** detection of variations in illumination or optical contrast in the protected area
- Pressure mat switches:** pressure-sensitive long strips used on floors beneath the carpets to detect weight of an intruder
- Stress detectors:** strain gauges imbedded into floor beams, staircases, and other structural components
- Switch sensors:** electrical contacts connected to doors and windows
- Magnetic switches:** a noncontact version of switch sensors
- Vibration detectors:** react to the vibration of walls or other building structures, also may be attached to doors or windows to detect movements
- Glass breakage detectors:** sensors reacting to specific vibrations produced by shattered glass

132

## Common Sensors 2

12. *Infrared motion detectors*: devices sensitive to heat waves emanated from warm or cold moving objects
13. *Microwave detectors*: active sensors responsive to microwave electromagnetic signals reflected from objects
14. *Ultrasonic detectors*: similar to microwaves except that instead of electromagnetic radiation, ultrasonic waves are used
15. *Video motion detectors*: video equipment which compares a stationary image stored in memory with the current image from the protected area
16. *Video face recognition system*: image analyzers that compare facial features with a database
17. *Laser system detectors*: similar to photoelectric detectors, except that they use narrow light beams and combinations of reflectors
18. *Triboelectric detectors*: sensors capable of detecting static electric charges carried by moving objects

133

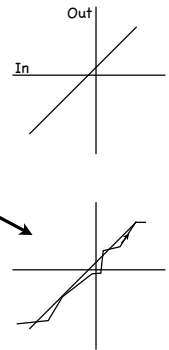
## Transducer Characteristics

### Ideal transducer

- Scale/offset parameters

### In the real world

- Linearity error
- Clipping range
- Zero-cross-over
- Hysteresis



134

## Integrated Sensors

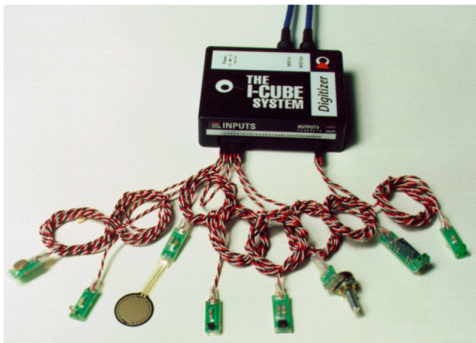


Fig. 2. The I-Cube System Digitizer and a few sensors. Courtesy Infusion Systems Ltd. [60]

135

## Position/Orientation

### Fixed-state

- Switches of all sorts

### Continuous

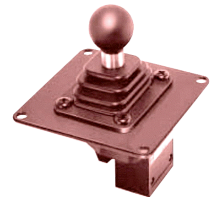
- Rotary or linear potentiometers
- Track-pads

### Magnetic/gravitational

- Absolute pos/ori

### CV-based sensing

- Multi-object sensing



136

## Switches



137

## P/O 2

### Non-contact proximity sensors

- Send-receive: ultra-sound, magnetic, RF

### Position vs motion: accelerometers

### Position vs orientation

- Gravity/magnetic

### Motion vs force: FSRs

### 2-point position sensors

- Tweezers
- Multi-touch

### Mixed-mode position/force sensors

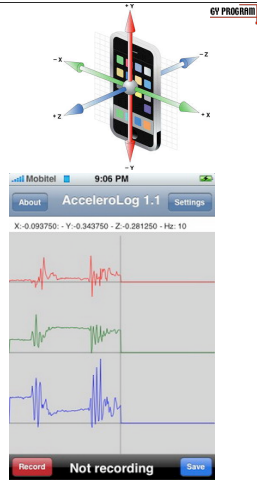
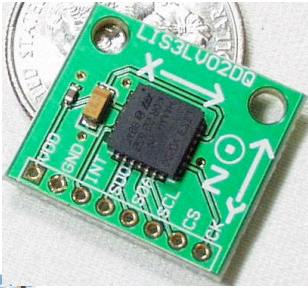
- Keyboard switch + after-touch



138

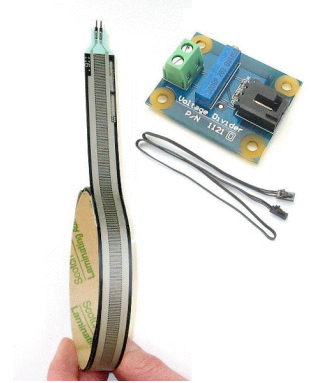
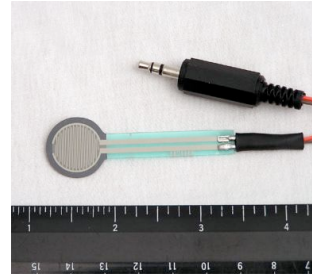
## Accelerometers

- Multi-sense or I2C
- Also GPSs



139

## FSRs



140

## Light Sensors

Table 1 - Comparison of Light Sensor Characteristics

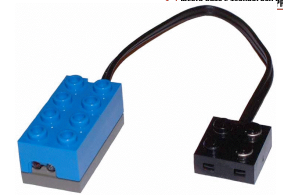
Electrical Characteristics	Photo-multiplier Tubes	Photo-diodes	Photo-transistors	CdS Photo-cells	Other Photo-conductors	Integrated Circuits	Hybrids	Sensor Electronic Assembly
Available Wavelengths (nm)	0.2-0.9	0.2-2.0	0.4-1.1	0.4-0.7	2-15	0.2-1.1	0.2-15.0	0.2-15.0
Performance-to-cost ratio	Fair	Good	Excellent	Excellent	Fair	Fair	Fair	Good
Sensitivity	Excellent	Very Good	Very Good	Very Good	Very Good	Very Good	Very Good	Very Good
Linearity	Good	Excellent	Good	Good	Good	Good	Good	Good
Ambient Noise Performance	Fair	Very Good	Very Good	Very Good	Very Good	Excellent	Excellent	Excellent
Dynamic Range	Very Good	Excellent	Very Good	Good	Good	Very Good	Very Good	Very Good
Stability	Very Good	Very Good	Good	Poor	Fair	Very Good	Very Good	Very Good
<b>Other Characteristics</b>								
Reproducibility	Fair	Excellent	Fair	Poor	Fair	Very Good	Very Good	Very Good
Cost	High	Low	Very Low	High	High	Medium	High	Medium
Ruggedness	Poor	Excellent	Excellent	Excellent	Good	Excellent	Very Good	Excellent
Physical Size	Large	Small	Small	Small	Small	Small	Medium	Medium
Ease of Customization	Poor	Easy	Fair	Fair	Poor	Poor	Poor	Fair
Cost of Customization	Very High	Low	Medium	Low	High	Very High	High	Medium
Lead time for Customization (weeks)	40	12	14	12	20	40	30	16

<http://www.engr.udayton.edu/faculty/jloomis/ece445/topics/egginc/tp4.html>

141

## Light Sensors

- Single-point
- Intensity/color
- Multi-pixel
- Camera control & scanning



142

## Visual Features

- Light presence, obstruction
- Color
- Direction, polarization
- 2-point light sensation
  - Optical encoder
  - Keyboard with velocity
- Image capture and computer vision

## Sound Sensors

- Microphones
- Audio analysis and feature extraction
- The semantics of music
- Speech understanding
- See MAT240F

## Other Media

- Temperature, atmospheric
- Biometrics (skin, blood, heart, EKG)
- Smell/chemical
- Air characteristics (humidity, pressure)
- Other EMR wavelengths
- Higher-level presence/motion detectors



145

## Multi-sensor Input Devices

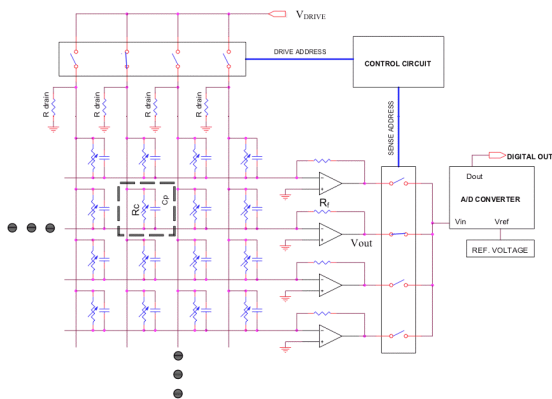


Most common



146

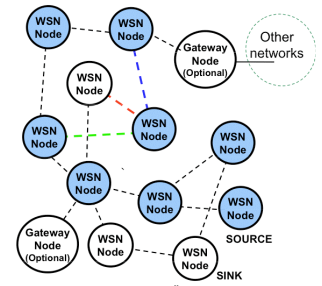
## Sensor Networks



147

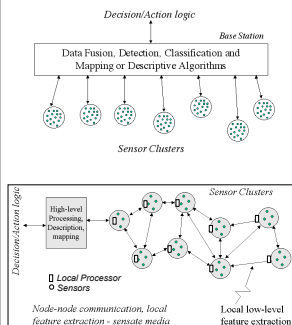
## Sensor Networks

- In a single device
  - E.g., switch networks
- Neighbor-aware devices
  - Wireless
  - Autonomous
  - RFID & WLAN



148

## Network Topologies



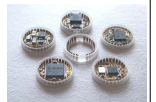
- Star topology
  - Assumes local processing
  - High rate possible
- P2P topology
  - Scale to high density
  - Feature extraction by local communication



149

## Sensor Net Support

- Berkeley Motes
  - Favors size and integration over modularity
  - Commercial Version from Crossbow (also moteiv)
  - Concentrates on ad-hoc networking application
- Philips SAND
  - Modular system for wireless sensing
  - Multiple panes with different functionality (IMU/ECG/DSP)
  - Networking via ZigBee
- Millennial Net
  - MIT ME Spinoff (lower-end)
- Ember
  - Media Lab Spinoff (higher-end)



150

## Sensors in a System

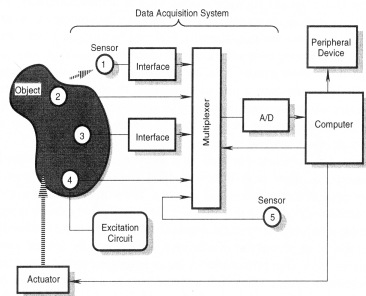


FIGURE 1.2. Positions of sensors in a data acquisition system. Sensor 1 is noncontact, sensors 2 and 3 are passive, sensor 4 is active, and sensor 5 is internal to a data acquisition system.

151

## Example: Radio Baton

- Dual 3D position sensors
- Use radio frequency signal with multiple antennae
- Microcontroller delivers dual X/Y/Z data

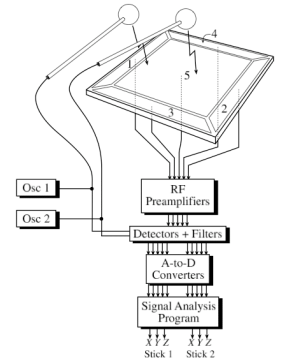


Figure 47: Block diagram of the Radio Baton system.

152

## Example: Lightning



Fig. 5. The Lightning™ II. Courtesy Buchla and associates [58].

153

## Example: CV for Hand Orientation

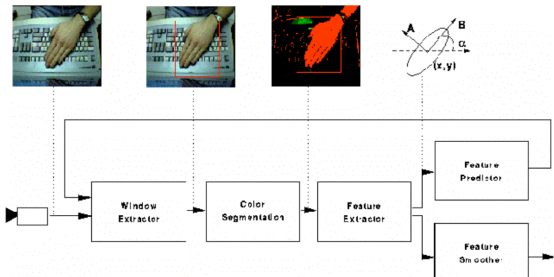


Fig. 22. Video-based pattern recognition of hand given initial region of interest and skin color calibration.

154

## Sensor Signal Processing

- HCI via sensors
  - Transducer
  - Signal conditioning
  - Digital data acquisition
  - Processing in software

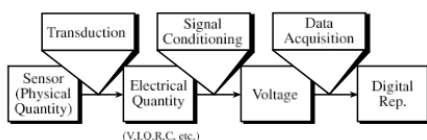


Figure 2: The Path from Human to Computer

155

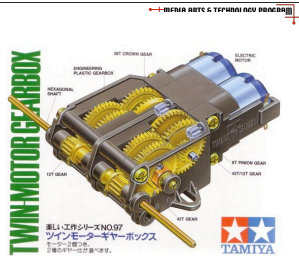
## Sensor Exercises

- Basics
  - Switches
  - Knobs
- Range/offset processing
- Sensor capture and display

156

## Outputs

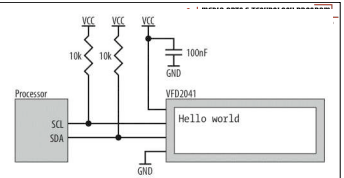
- Visual
- Audio
- Electrical
- Motion
- Relation to robotics



157

## Visual Output

- LEDs
- LCD text displays
- Graphics displays



158

## Audio Output

- Audio out
  - HW/SW synthesis
- MIDI out
  - Easy HW/SW
- OSC out
  - See uOSC
- Other methods

159

## Electrical Output

- Low-voltage, low-current
  - Use analog pins directly or out-board DAC
- Higher voltage
  - Use amplifiers and/or relays
- Higher current
  - Use switching transistors (motor examples)

160

## Physical Output

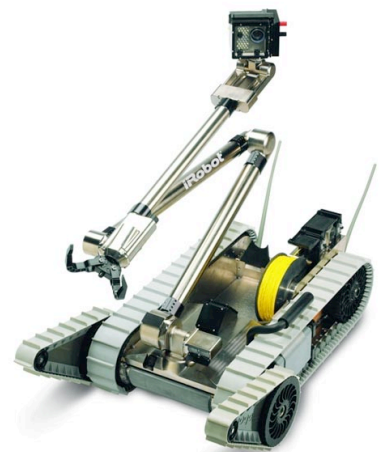
- Linear vs, angular motion
- Solenoids, actuators, and motors
- Robotics
- Haptic force feedback



161

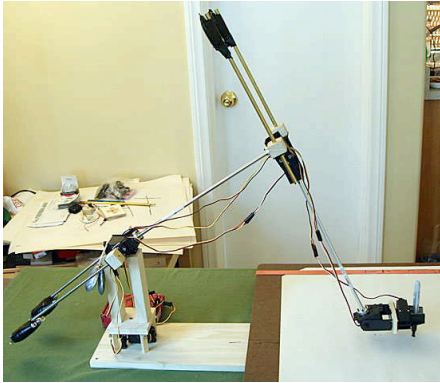
## General (US) Robotics

- DoD robot run by an iPhone app



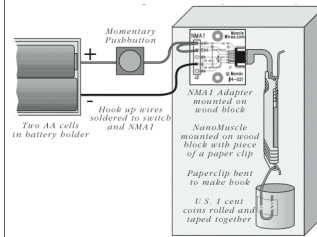
162

## Eric Newman's DrawBot



163

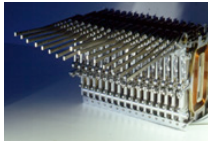
## Solenoids & Linear Actuators



164

## Sensor + Motor

- Haptic force feedback sensors



165

## Review

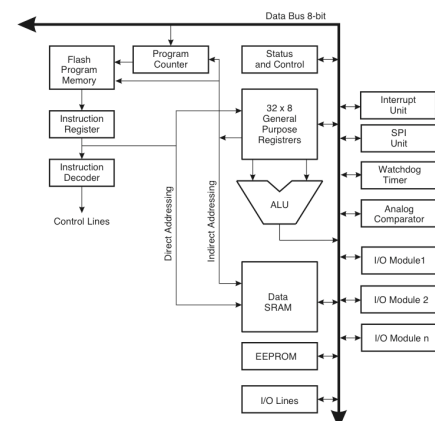
- Transducers & sensors
- Stimulae & physics
- Sensor conversion characteristics
- Composite/networked sensors
- Output signals and drivers

166

## What's Next?

- Microncontroller readings
  - Platforms & HW
  - Support SW
  - Attaching sensors
- Exercises combining HW & SW
- Sensor-integrated systems

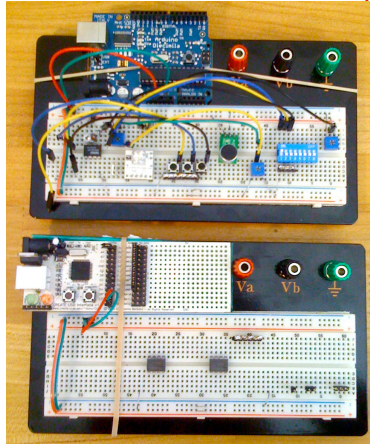
167



168

## Topic 5

### Microcontrollers and Interfaces



169

## Microcontrollers and Interfaces

### Readings

- Arduino overview & interfaces
- Wiring example list
- Microcontroller data sheets: PIC, Amtel, other

### Lecture: Single-chip Systems: Circuits and Software

### Exercises

- Arduino basics
- Using wiring with Arduino

170

## Single-chip Systems

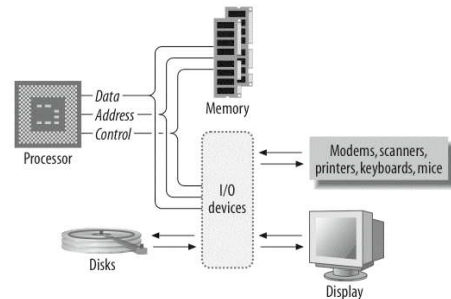
- Circuits and Software
- The CPU & core system
- IO ports & interfaces
- Control/programming

171

## Computer Architecture

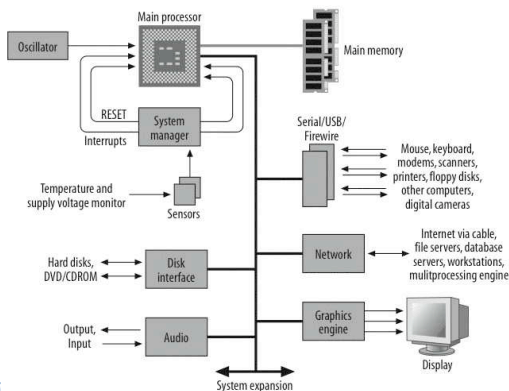
### Review: components of a computer

- CPU, memory, I/O, control, power...



172

## In more Detail

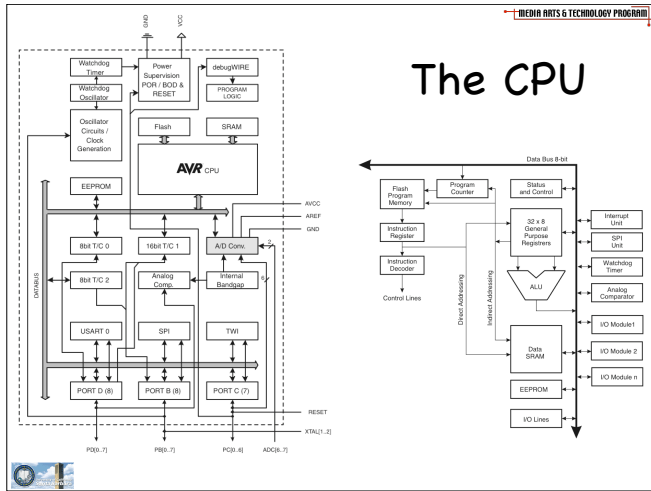


173

## The CPU

- Registers & Processing Units
- RAM and Memory Control
- Data Busses
- I/O Ports
- Internal View vs System View

174



175

## The System

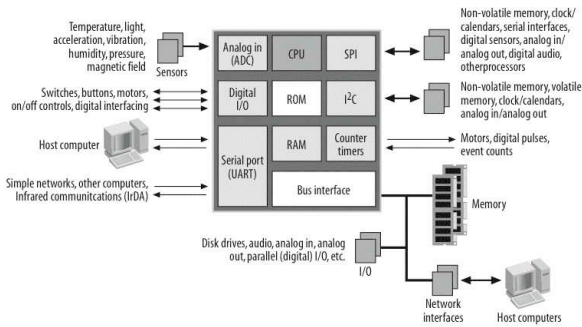
- Microcontroller CPU
- RAM (Instr/Data), ROM
- Power/reset
- Clock
- General A/D I/O ports
  - Opt, DAC and/or ADC
- Special serial I/O (USB)
- Programming Ports



176

## Embedded Systems

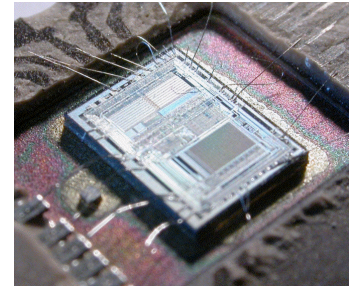
- Mostly the same components



177

## Micro-CPUs

- Slow
- Narrow
- No FPU
- Small RAM/ROM
- Timers
- Limited IO
- UARTs
- Simple SW monitors



The integrated circuit from an Intel 8742, an 8-bit microcontroller that includes a 12 MHz CPU, 128B RAM, 2kB EPROM, and I/O



178

## History: Intel 8051



Microcontroller information	
Type	Microcontroller
Manufacturing process	HMOS II
Data bus width	8 bit
Package	40-pin plastic DIP
Speed (MHz)	3.5 - 12
On-chip RAM (bytes)	128
On-chip ROM	4 KB masked ROM
On-chip peripherals	<ul style="list-style-type: none"> <li>Four 8-bit ports (32 I/O lines)</li> <li>Full-duplex serial port</li> <li>Two 16-bit timers/counters</li> <li>Clock oscillator</li> </ul>
Physical memory	128 KB (64 KB code and 64 KB data)
V core (V)	5 ± 10%
Min/Max operating temperature (°C)	-40 - 85
Max power dissipation (W)	0.69 (12 MHz)



179

## 8051-based Single-board Computer



180

## Modern CPUs for Microcontrollers

- Microchip PIC
- Amtel AVR
- TI MSP430
- Motorola 68k
- ARM family
- Intel i80XX
- DSPs

```
int max(int *array)
{
 char a;
 int maximum = -32768;

 for (a = 0; a < 16; a++)
 if (array[a] > maximum)
 maximum = array[a];

 return (maximum);
}
```

Table 15-1. Atmel's comparison of processor speed and efficiency

Processor	Compiled code size	Execution time (cycles)
AVR	46	335
8051	112	9,384
PIC16C74	87	2,492
68HC11	57	5,244

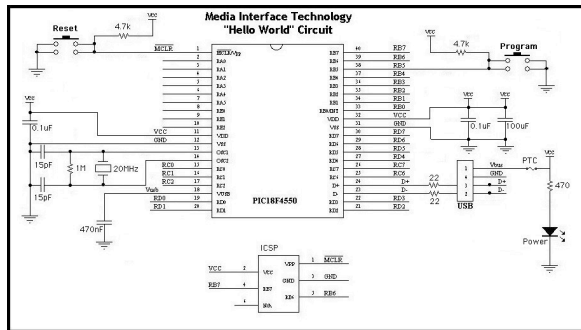
181

## Old-School SW

- Assembly
- FORTH-in-ROM
- BASIC-in-ROM
- UCSD Pascal system
- CPM monitor

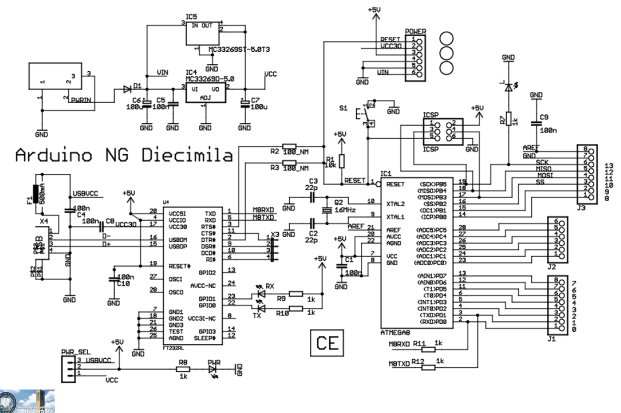
182

## Hello World HW (PIC)



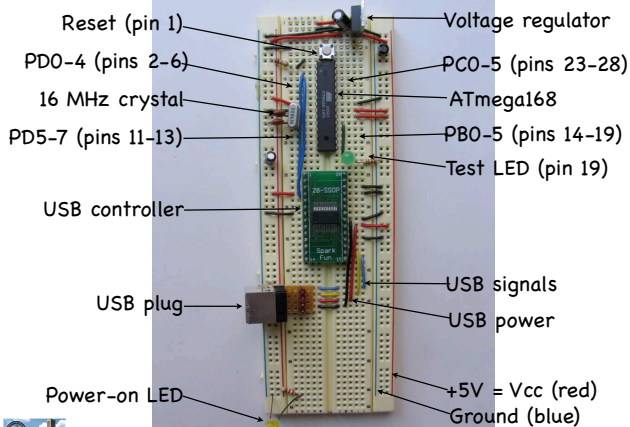
183

## Arduino Schematic Diagram



184

## Arduino Breadboard Details

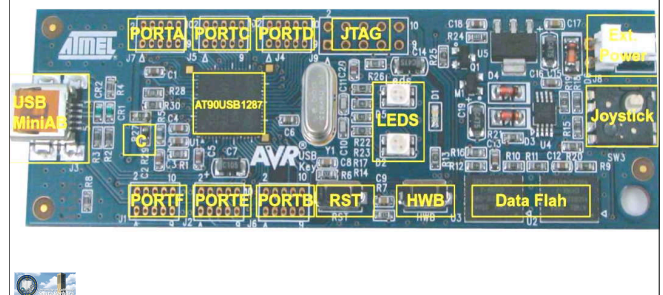


185

## Layout Example

- Amel AT90USB: CPU, RAM, A/D IO

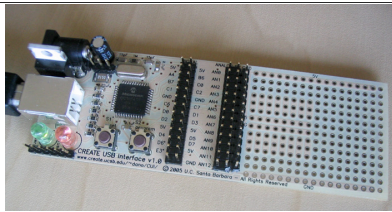
Figure 2-1. AT90USBKey Overview



186



## DanO's CUI

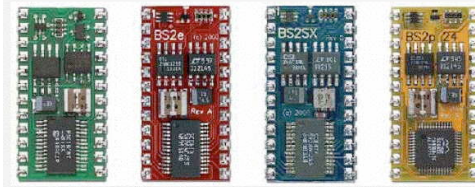


- PIC 18F4550 & USB
- 13 A/D inputs
- 18 general I/O ports
- On-board prototyping area



193

## Flavors of Stamps



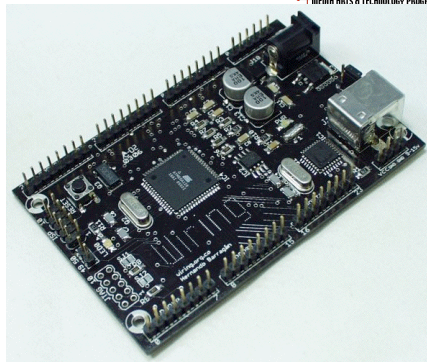
**Figure 1-3**  
BASIC Stamp  
Modules

*From Left to  
Right: BASIC  
Stamp 2, 2e,  
2SX, and 2p*



194

## Wiring

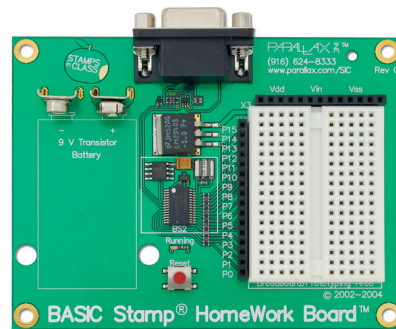


- Core & IO daughters



195

## BASIC Stamp

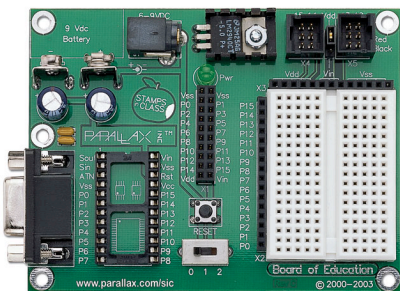


- Many flavors & configurations



196

## Parallax Boards



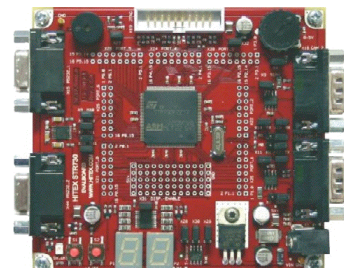
- Pluggable mother-daughter cards



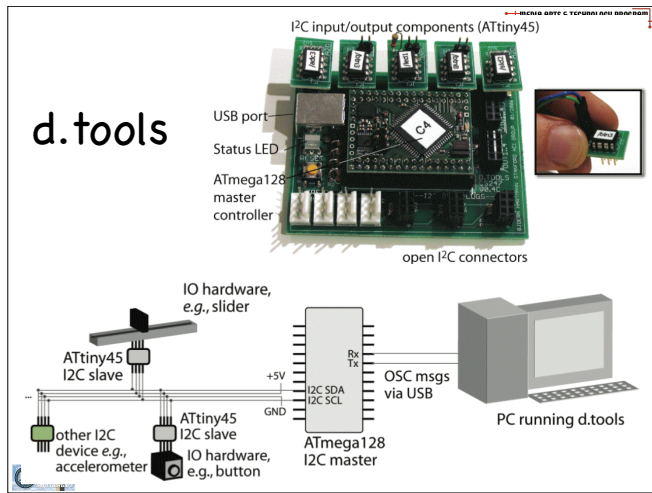
197

## ARM Systems

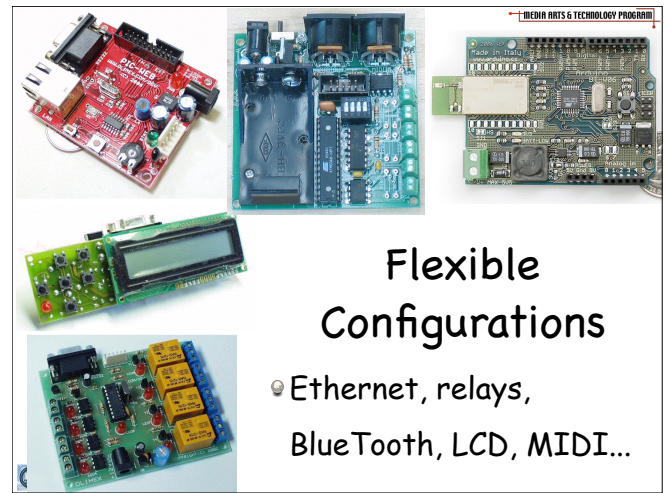
- In many appliances: phones, games, etc.



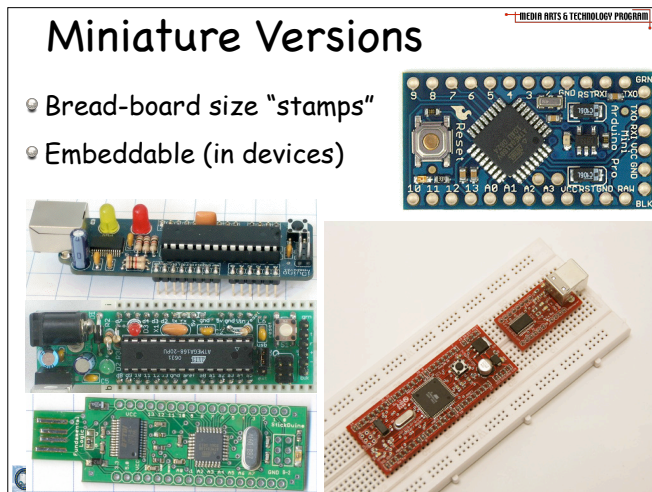
198



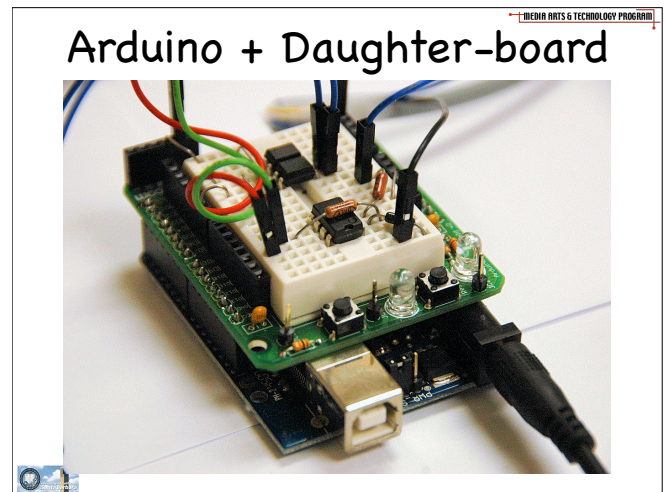
199



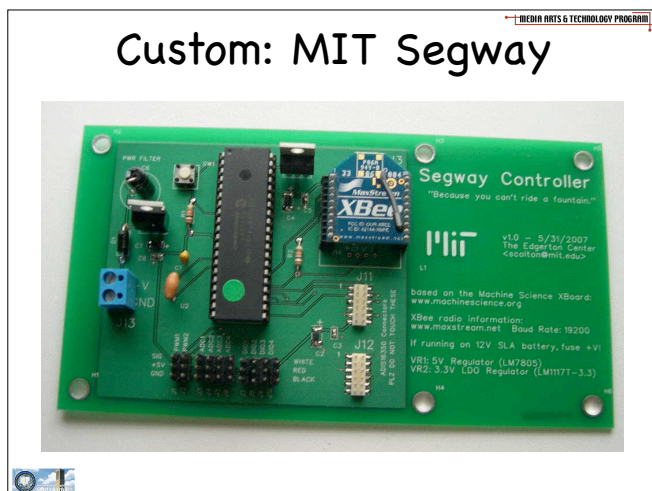
200



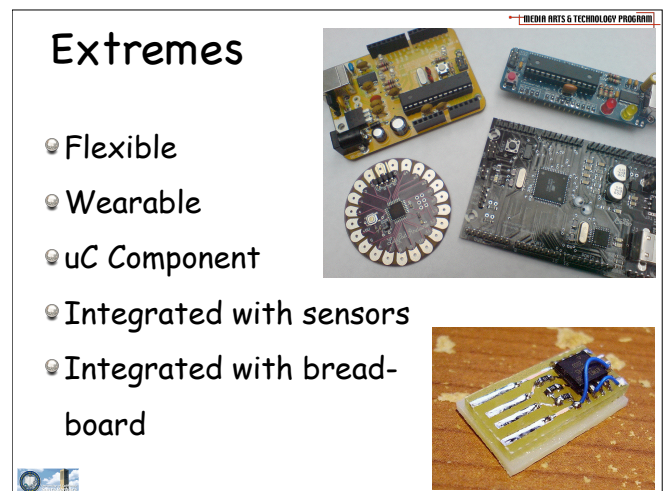
201



202



203

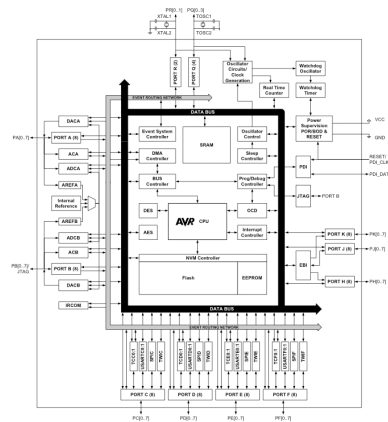


204

## "Hi-End" CPUs

- Faster
- More RAM/ROM
- More I/O

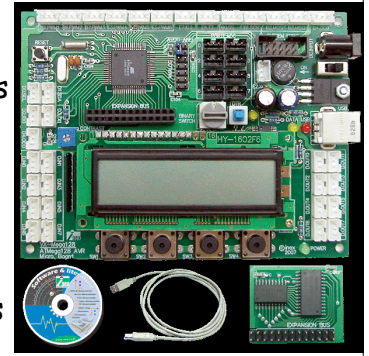
Figure 3-1. XMEGA A1 Block Diagram



205

## "Hi-End" Systems

- Hi-End CPU
- More peripherals (display, timers, ADC, buttons)
- Bigger board
- Daughter boards



206

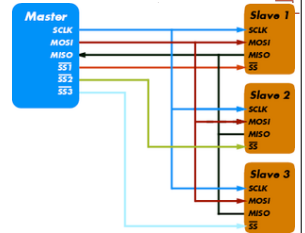
## Interfacing 1

- CPU power is 0/+5 low-current
- Devices deliver/expect higher V or I
- Input transformers, ADCs
- Output driver circuits, relays
- Serial protocols
  - IP, SLIP, MIDI, 802.11\*, BTooth, RF

207

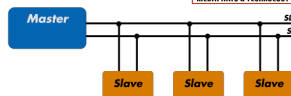
## Interfacing 2

- Serial UARTs
  - RS232, RS422, etc.
- I2C, SPI
  - Created To allow microcontrollers to off-load complex functionality to special purpose chips
  - Zillions of electronic devices use it in some fashion — from blinking LEDs to controlling stereo volume to airbag sensors to alarm clocks to tilt sensors



208

## I2C



- Only 2 wires from the CPU — SDA and SCL
- Works on a master/slave paradigm
  - One master, many slaves
- Each device has a unique address
- Wiring library provides easy programming
- But: Arduino's Atmega8 leaves little room for more than simple functionality

209

## I2C



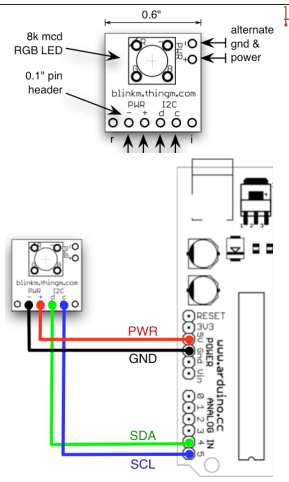
- When Issuing A Command, The Address Is Sent And The Appropriate Device "Pays Attention" To The Next Stream Of Data, Which Contains Information About What It Should Do — Send Back Information, Listen For Information, Turn Something On...
- i.e., each slave has a decoder/control microprocessor

210

## I2C Example: BlinkM RGB LED

### 4-wire interface

- Power, ground
- Data, clock



211

## Wiring for I2C

```
Wire.begin(); // set up I2C
Wire.beginTransmission(0x09);
 // talk to BlinkM 0x09
Wire.send('c'); // cmd 'c' == fade to color
Wire.send(0xff); // value for red channel
Wire.send(0xc4); // value for blue channel
Wire.send(0x30); // value for green channel
Wire.endTransmission(); // leave I2C bus
```

212

## Other Interconnection

- Ethernet
  - TCP/IP
  - OSC over USB
- Wireless
  - 802.11\* & RF
  - Bluetooth, Zigbee, CAN
- Other wired protocols
  - SensorML, TEDS
- Other media (?)

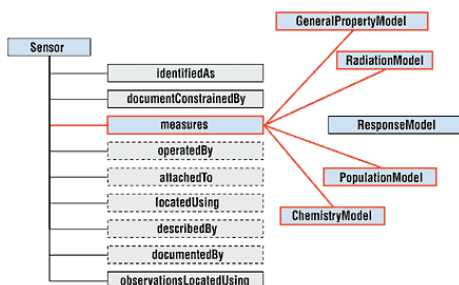
213

## Bluetooth (IEEE 802.15.1)

- Peripherals communicate with a single master
  - 7 nodes with 720 kbps total bandwidth
  - Power can be carefully managed
  - Some issues
    - 7 Slaves, 1 Master (although can nest subnets with shared node)
    - Takes 100's of msec(!) to shift between nodes in Bluetooth 1
- Many chip-sets and modules

214

## SensorML

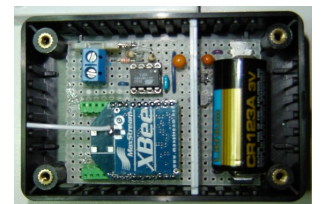
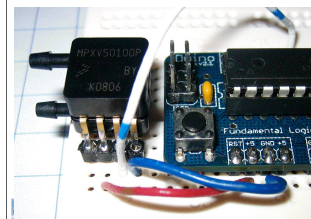
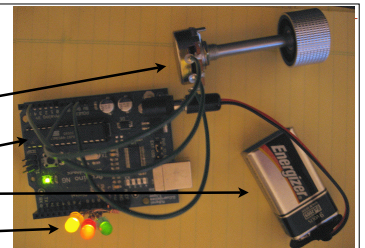


- Self-describing sensors
- Allows for auto-discovery

215

## Integration

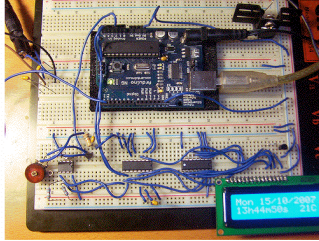
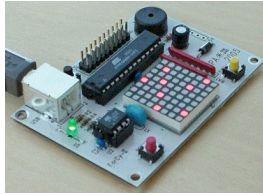
- Sensor input
- Microcontroller
- Power
- Outputs (?)



216

## Integrated CPU + I/O

- Support HW
- Interfaces
- Sensor signal conditioning



217

## Accessories

- See SparkFun, RobotShop, MarVac...



218

## Microcontroller Software

- Old-school (see above)
- Modern
  - Boot ROM monitor & loader
  - Loadable 2-stage routines - setup() & loop()
  - USB HID, configurable
  - MicroOSC, configurable
  - Web server (pages to configure device)
  - Java/Smalltalk VM in ROM
  - Mobile OS

219

## Software

- Low-level monitor in ROM
  - Reads programs from serial input
- API
  - Set-up call: initialize the HW
  - Call-back or timer loop: read/process/write
- Languages/IDEs
  - Wiring/Processing IDE
  - C/C++ with down-load (CUI)
  - BASIC-Stamp
  - Scripting languages
  - On-chip OSC, USB-HID broadcast

220

## Wiring Example

```
// Reading a potentiometer
// by BARRAGAN <http://barraganstudio.com>

int val;

void setup() {
 Serial.begin(9600); // sets serial port to 9600 Baud
}

void loop() {
 val = analogRead(0); // read analog input pin 0
 Serial.print(val, DEC); // prints the value read
 Serial.print(" "); // prints a space between numbers
 delay(100); // wait 100ms for next reading
}
```

221

## Wiring Example 2

```
// Controlling a servo position using a slider (variable resistor)
// by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>

#include <Servo.h>

Servo myservo; // create servo object to control a servo
int sliderpin = 0; // analog pin used to connect the slider
int sliderValue; // variable to read the value from the analog pin

void setup() {
 myservo.attach(0); // attach pin 0 to the object
}

void loop() {
 sliderValue = analogRead(sliderpin); // reads slider (0 - 1024)
 sliderValue = sliderValue/5.68; // scale for servo (0 - 180)
 myservo.write(sliderValue); // sets servo position
 delay(15); // waits for servo
}
```

222

## Wiring Example 3

```
// LCD print - by BARRAGAN <http://barraganstudio.com>
// creates a LiquidDisplay object with R/S, R/W and E
// on pins 8,9,10 and data pins on port 2
#include <LiquidCrystal.h>
LiquidCrystal myDisplay = LiquidCrystal(8,9,10,2);
int a = 0;

void setup() { /* nothing for setup */ }

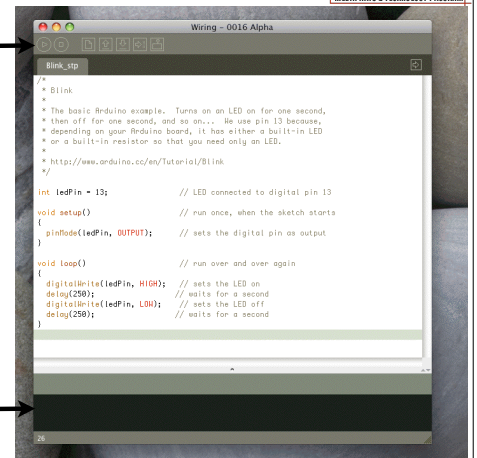
void loop() {
 myDisplay.clear();
 myDisplay.home();
 myDisplay.print("Variable a is: ");
 myDisplay.setCursor(16, 0);
 myDisplay.print(a);
 a = a + 1;
 delay(200);
}
```

223

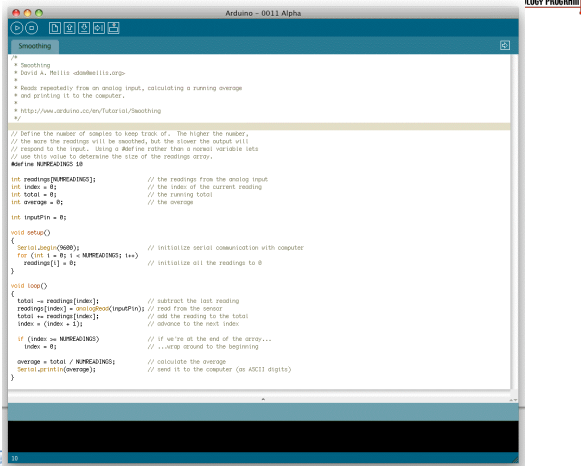
Operations  
menu bar

Wiring  
in  
Action

Serial  
read-out

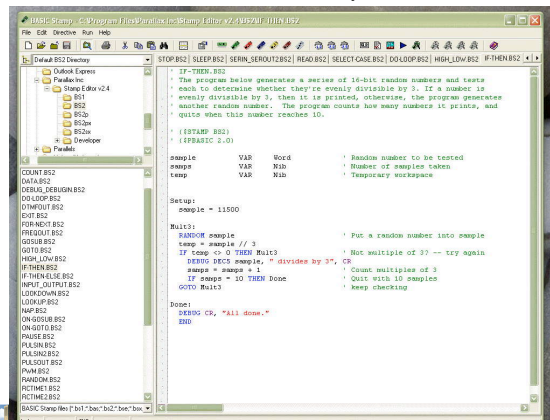


224



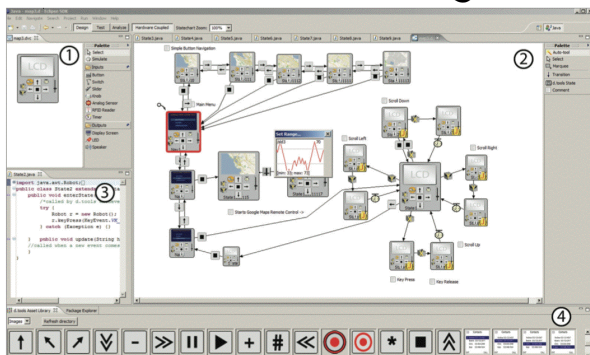
225

## BASIC Stamp IDE



226

## d.tools Authoring



State-chart editor-based

227

## Arduino/CUI/Wiring Examples

- Simple circuit + code examples
- Sensor signals and their conditioning
- Timing and output
- Coding for sensor input/output
  - USB HID
  - MicroOSC
  - MIDI

228

## MicroOSC

- Send OSC as SLIP over PIC serial port via USB to host
- Wrapper to decode SLIP to IP/OSC
  - in Max
  - C-language forwarder
  - In OSCPack, LibLo, etc.



229

## uOSC Port/Pin Address

```

/ra ffffff : generates/accepts port-report format
/o : individual pin control for /ra/o
/info : returns "dio", "adc", "pwm", "ttl"
/state : returns "input", "output", "reserved"
/set : accepts "low", 0, 0.0, or "high", 1, 1.0
/get : see /set
/1-5 (same as /ra/o)
/rb ffffffff : /rb port report (8 pins)
/o-7 (same as /ra/o)
/tx (same as /ra/o)
/rx (same as /ra/o)
/status
 /o : controls the yellow status LED
 /set : accepts "off", 0, 0.0, "on", 1, 1.0
 /get : return LED state
/1 : controls the red status LED
/osc/time/tick [t] : set the clock interval

```



230

```

/device
/platform : returns "Bitwacker", "CUI", etc.
/firmware : returns "uOSC 1.0"
/processor : returns "PIC18F2455 Rev. B4"
/ports : returns a list of port addresses
/pins : returns a list of pin addresses
/id : user-writeable 64-bit hex string
/save 1 : commits pin and module state to flash
/reset 1 : restore default state
/modules
/list : list available modules
/enable s : enable a module
/disable s : disable module
/pwm
/o : control the first hardware PWM
/rate f : rate in Hz
/duty f : duty cycle in [0.0-1.0]
/ttl
/o : control first hardware TTL
/open [baud, bits, stopbit]
/read : return string data
/write : write string data
/close
/usb
/stall : stall detected
/error : error detected

```

## uOSC Device Messages



231

## Review

- Microcontrollers
  - CPU
  - System configuration
- Systems
  - PIC, Amtel, other
- Software
  - Monitor, boot-loader
  - Processing loops



232

## Exercises

- Microcontroller programming
  - CUI HID
  - CUI uOSC
  - Arduino/Wiring
  - Arduino/Processing
  - Building clients



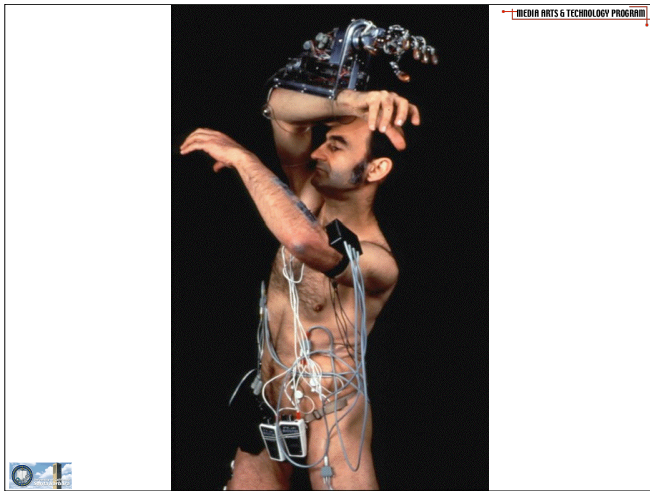
233

## What's next?

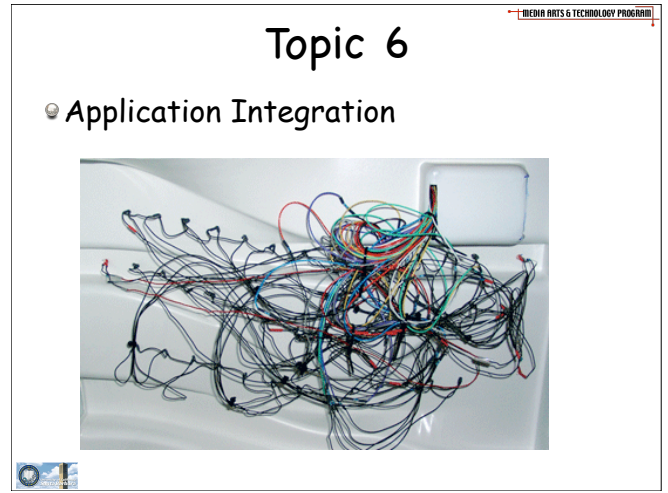
- Application integration
  - HCI application
  - Sensor design
  - Control system
  - Mapping, synthesis client



234



235



236

## Application Integration

- 💡 Readings
  - 💡 Musical Performance Practice on Sensor-based Instruments: Atau Tanaka
  - 💡 Neural Networks for Mapping Hand Gestures to Sound Synthesis Parameters: Paul Modler
  - 💡 Wiring, NYU Exhibitions
- 💡 Lecture: Putting it all together
- 💡 Exercises
  - 💡 Client/server glue code with Max, CSL, SC, etc.

237

## Putting it all Together

- 💡 Wiring, NYU, YLEM, ISEA, NIME Exhibitions
- 💡 Application categories
  - 💡 Tools
  - 💡 Instruments (musical, scientific)
  - 💡 Toys
  - 💡 Vehicles
  - 💡 Robots
  - 💡 Objets d'art
  - 💡 Novel HCI

238

## Application Frameworks

- 💡 Sensor systems
  - 💡 Interaction media
- 💡 MVC frameworks
  - 💡 Single vs composite model
  - 💡 Sensor as controller
- 💡 Pre-composed content & interaction
  - 💡 Delivery media

239

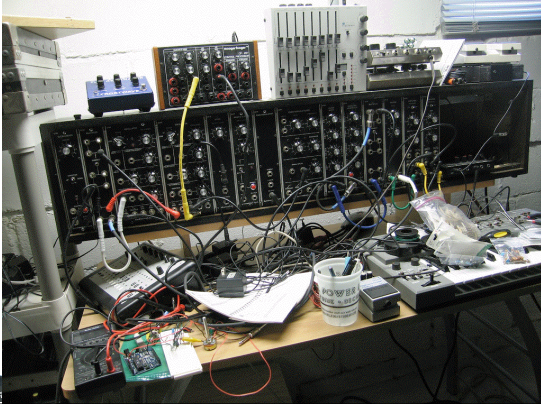
## Art in Galleries

- 💡 Positronic Neural Net by Russ Rubert

A photograph of a large, illuminated, abstract sculpture made of neon tubes, resembling a neural network, displayed in a gallery. The sculpture is composed of many glowing tubes connected in a complex, branching pattern.

240

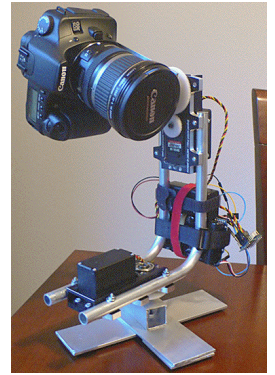
## Music Systems



241

## Robotics Applications

- Arduino-based pan/tilt camera/scanner



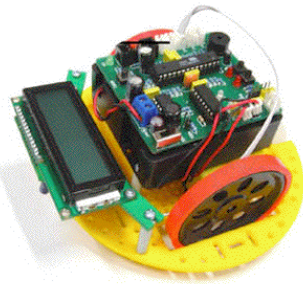
242

## Robotics Kits

- MicroCamp Robot Kit

### Package Contents

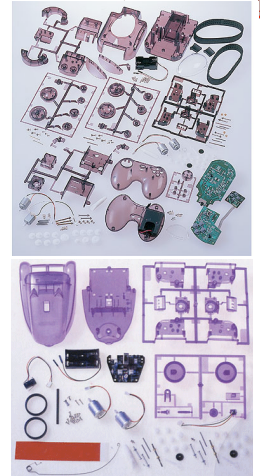
- Microcamp board with battery holder
- PX-400 AVR ISP Programmer
- Serial cable
- ZX-Switch, ZX-IR reflector and GP2D120 IR Sensor
- 16x2 LCD
- ER4 Remote Control
- Mechanical parts, CD-ROM and Documentation



243

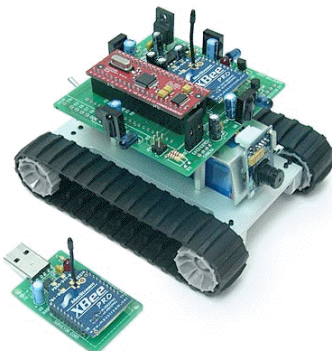
## Robotics Kits 2

- Sensors (multiple)
- Controller(s)
- Networking
- Motors, navigation
- Enclosure



244

## Makezine Surveillance Robot



245

## Wearable Tools

- Arduino turn signal jacket



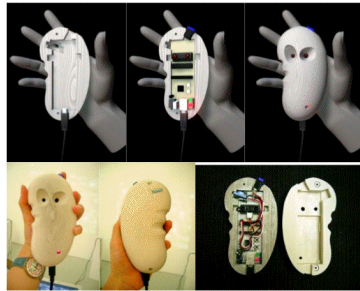
246

## 247

## 248

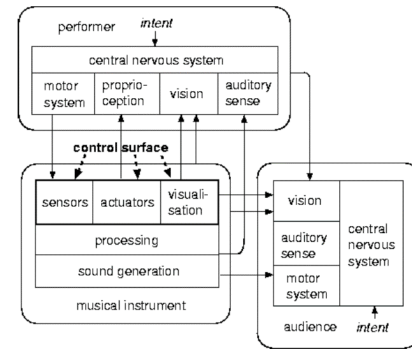
Figure 3. August Black playing *El Lechero*.

Figure 6. The author's own *BoingBoing* interface.



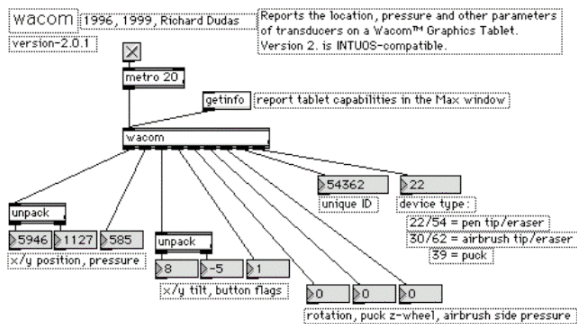
249

# Systems



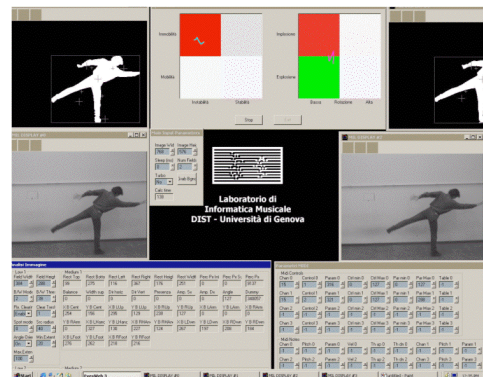
250

## Frameworks



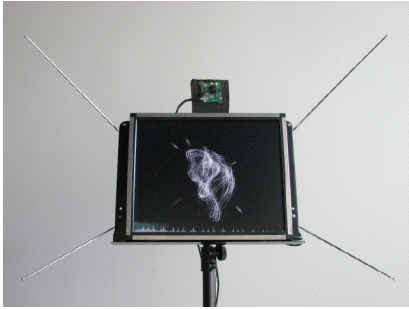
251

## Multi-modality



252

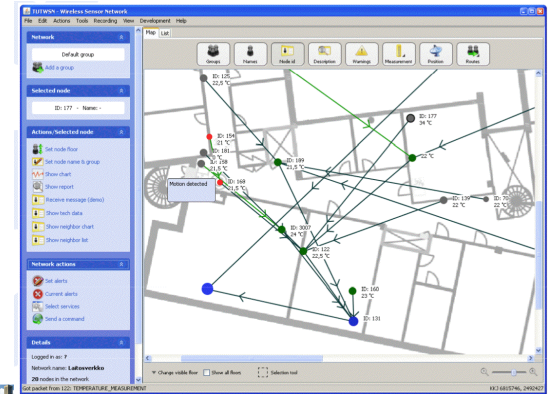
## MultiModal MusicStand



RF, CV, audio sensing, A/V output

253

## Sensor Network in Spaces



254

## Sensing/Speaking Space

User-aware spaces

Camera



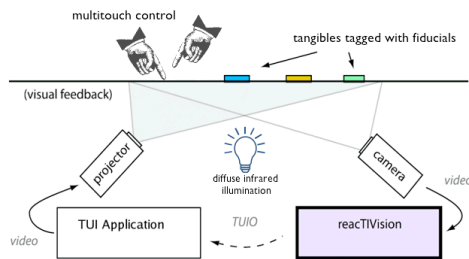
255

## Table Interfaces



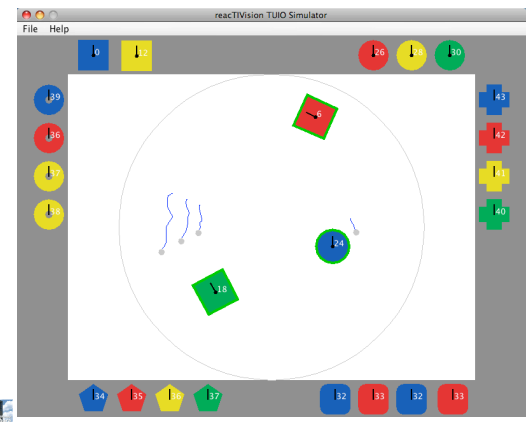
256

## ReacTable



257

## ReacTable TUIO SW



258

## ReacTable Apps



259

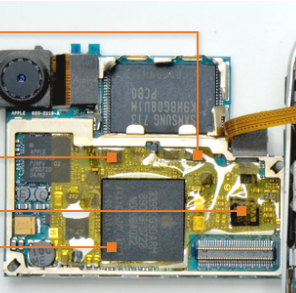
## Microsoft Surface



260

## iPhone Components (partial)

- A \$2.25
- B \$1.50
- C \$1.30
- D \$28.25



- A: motion sensor/accelerometer
- B: 24-bit RGB display
- C: Wolfson audio codec
- D: 620MHz ARM 1176JZF + 1GB of RAM
- Upper left: camera

261

## Asus Aura Concept Phone



- New sensors
- New IO streaming

262

## SCUBA Application

- Air pressure sensors, timers
- Wireless network
- In-mask display



263

## Multi-system Integration



Fig. 4. Audio clip available in the original CD-Rom version. The SensOrg. Click picture to hear the SensOrg composition *Fingerprint no. 1* by Tamas Ungvary.

264

## Multi-system Integration

- iPhone-based game controller



265

## Applications



266

## Projects

- Tools
- Instruments
- Installations
- Objects
- Interfaces
- Situations



267

## MAT 594O Review

1. Space & gesture
2. Electronics introduction
3. Human-computer interfaces
4. Transducers & signals
5. Microcontrollers and interfaces
6. Application integration



268

## Where do we go from here?

- MAT 240X: Digital Audio Programming
- MAT 201B: Computing with Media Data
- MAT 200C: Multimedia Engineering
- MAT 275: Systems in Supercollider
- AlloSphere!



269

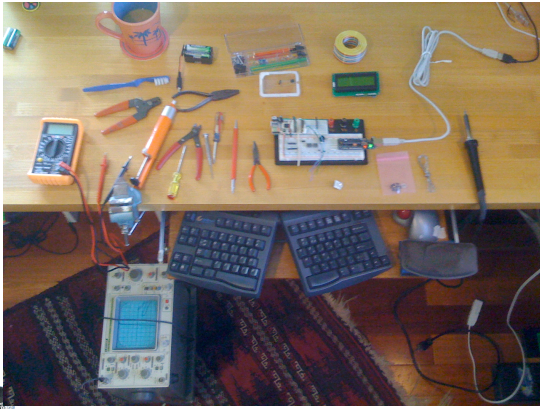
## Conclusions

- **Assertions**
  - Sensors and distributed controllers are being ever more tightly integrated into a variety of appliances, tools, and instruments.
  - Reactive microsystems are easy to build and program!



270

Thank You



271

**MEDIA ARTS & TECHNOLOGY PROGRAM**

## MAT 594O: Sensors and Interfaces for Media Art

Formerly "Media Interface Technology"

UCSB, Fall, 2008

Instructors

Stephen Pope, Matt Wright, Amichi Amar

<http://www.mat.ucsb.edu/594O>



272