What are the Issues?
- Are we using MM to teach CS or teaching MM software development?
- RFP for MM-in-CSED: integrating MM assignments in CS education, “modern software like Squeak makes multimedia approachable and understandable”

What’s the Focus?
- Teaching programming and procedural thinking -- low-level languages
- Teaching OO analysis/design -- knowledge-domain-modeling
- Teaching IDE and CASE tool usage
- Teaching real-world job (“vocational”) skills
- Preparing students for a comprehensive CS curriculum
- Teaching non-CS-majors to build domain-specific SW tools

Teaching Software Design
- In the traditional CS curriculum, methodology, CASE, IDE tools, and A/D are taught after programming(!)
- In ST curricula, A/D is taught first.
- What does the CS freshman need?
- What does the MM artist/technician need?
- What do other knowledge workers (science/humanities/arts) need?

Teaching Real-World Tools
- ST as a niche language…
- UofW experience with ADA (1990s)
- The ubiquity of C, C++, Java (C*)
- The ubiquity of file/project-based IDEs
- The matter of principle (buck or go)
The Role of Smalltalk/Squeak

- Perfect OOA/D teaching platform
- Simple concepts and syntax
- Low-overhead IDE (?)
- Platform-independent
- Libraries/APIs for many application domains, including MM
- Not “mainstream,” however...

Smalltalk as a First Language

- My experience with teaching ST to many groups
- Non-programmers have an easier time than programmers
- Programmers get frustrated by the A/D focus and the IDE
- This is good news!

Multiple Planes and Paradigm Shifts

- The two technologies that have the most profound impact on musical practice in the last 100 years were both invented before 1920: recording and broadcasting
- “The impact of digital technology on music has been as shallow as it is broad” (F. R. Moore)

Generations of Computer Technology

- Mainframes
- Mini-computers
- Engineering workstations
- PCs
- The Internet
  - The Next Big Things:
    - Distributed Systems
    - Immersive user interfaces
    - Multimedia integration

Dimensions of Moore’s Law
(With thanks to L. Rabiner)

- RAM density (doubles every 18 months)
- LAN speed
- Disk throughput
- Disk access time

CS Education and “vocalitionalism”

- Preparing CS majors to build the next-generation systems
  - WA Distributed
  - Media rich, immersive UIs
  - Thin/wireless clients
  - Parallel or thread-based simulations
  - Non-sequential: ES, GA, NN, CSP, IFS, ...
  - DSCP (distributed, sensing, computation, and presentation)
  - Model-based “biology in the computer”
**Good News**

- We know what to do (see below)

  - E.g., GaTech CS2340
    - Course on design for novices
    - Build interactive maps
    - Team-focus

**CS Language Options**

- Scheme/CommonLISP
- ML/Haskell
- ES Shells
- Uniformly high-level Java
  - OOA/D focus
  - Use of reflection
  - Still quite painful
- Various Smalltalks

**CS @ UCSB**

- Java, Scheme, C on UNIX/Linux
- Several integrated labs
  - PSL, SCC, Clusters
- Wireless infrastructure
- Media classrooms
- VE centers
- Integration with application domains: MAT, CREATE, dStudio, eStudio, BioInformatics, CompChem, MaterialsIT, CNSI, CITS
- Dual appointments

**The UCSB MAT Program**

- Two-year interdisciplinary Master’s degree program in Media Arts and Technology (MAT)
- Partner depts: Music, Art, CS, EE
- Five core courses, electives, thesis project; optional internship, independent study, RA-ship
- Emphasis on group projects
- Three areas of specialization:
  - Multimedia engineering
  - Electronic music and sound design
  - Visual and spatial arts

**MAT Students**

- Generally have BA/BS in
  - Music composition or performance
  - Visual arts
  - Computer science
  - Electrical engineering
  - Recent exceptions include architecture, design, mechanical engr., film studies, etc.
- “Crossovers” possible (several already)

**MAT Courses**

- Five Core Courses
  - History/literature of electronic music
  - Digital media art and culture
  - Multimedia networking
  - Digital media signal processing
  - Survey of media technology and engineering
- Elective courses taken from the four partner departments and related fields
New MAT Electives

- Digital audio programming sequence (six quarters)
- Maya sequence (three quarters)
- Camera tracking and user input (project-oriented)
- Sound/music programming in SuperCollider
- New seminar series in each of the partner departments

What to teach in MAT?

- CS/EE students
  - Already know C*
  - Let them eat Squeak! (and SC, and others)
- Arts students
  - Some (though not many) know no programming language at all
  - Some know at least a scripting language (MEL, Lingo, etc.)
  - Some know C*
  - Teach them Java (+2D, 3D, Sound, etc.)

Example: MAT 240a-f

- Digital audio programming: the series
- Six quarters on various topics: using commercial APIs, frequency-domain processing, spatial audio and surround sound, synthesis techniques, multi-rate control and processing, etc.
- Use Linux, Windows, and Mac platforms
- Mainly in C* (with ObjC, SC, ST, etc.)

Example: MAT 220

- Sound/music system programming
- Focus on complex sound design, interactive performance, and compositional algorithms
- Students are mostly undergrad- or grad-level composers
- Use SuperCollider (a dialect of ST) on Macs

Example: MAT 41

- Beginning (minimal) programming skills for non-programmers in MAT
- (Not part of a comprehensive CS curriculum)
- Intro to Java

Example: MAT 596

- Independent R&D
- Several ST/SC projects (Siren, OSC, grammars, performance rules, CSP, NN, etc.)
- Lots of distributed projects (CORBA, OSC, CO, etc.)
So, why this way?

- The “mainstream” issue
- The learning curve issue
- ≥ Two languages needed
- Squeak vs. other ST platforms
- C* APIs: 2D, 3D graphics, spatial sound, CORBA

An Answer!

- Teach ST as a “first” language (assuming most frosh know C*)
- Integrate MM applications from day one (see below)
- Stress methods, CASE, A/D, tools
- Segue as needed to C*

Examples: MAT Student Projects

- Multimedia data/signal processing
- Robotics
- Multimedia databases
- Hardware input devices
- Virtual environments
- Advanced GUIs
- Work in Smalltalk, MATLAB, Java, C++, SuperCollider, Lingo, MEL

RKRobot (Gary Thomas)

Woon-grams

Matrix Controller (Dan Overholt)
Steganography (Gilroy Menezes)

Obsessive Remixer (C. Ramakrishnan)
- Sound example (Autumn Leaves)

Smart TextShredder (Brian O'Reilly)

```java
public class TextShredder {
    public static void main(String args[])
    throws IOException,
    FileNotFoundException {
        System.out.println("Starting Shredder!");
        // Instantiate shredder/cut up objects
        WordBank wb = new WordBank();
        Jumble j = new Jumble();
        StringMover mv = new StringMover();
        ... 
    }
}
```

I was also a hunter of birds. In this, the share of reaction to family members who want to live outside of a life of toil is great. I believe this has been touched upon in a novel as a "first rebellion" by an author from Turkey. As long as there was no provocation, animosity would never develop and an exemplary brotherly coexistence prevailed....

LLCH Models (Brent Yokota)

CREATE Auralizer (240C)
- Sources, geometry, processing, output
- Current: Stereo, 4 KLOC, selectable cues

Creatophone Installations in LLCH
The ATON VE (group)

DSCP Example: Sensing/Speaking Space

Sensing/Speaking Space at SFMoMA

Squeak Tools and Applications

Siren Composer’s DB & Tools
CS Curriculum
- Currently: ≥ two languages are still needed
- Teach ST first
  - OOA/D, CASE, methods, CM, tools focus
- Use C* as appropriate
  - R-T, OS, networking courses (i.e., low-level C)

(Current) MM Curriculum
- One language: Java or ST
- Teach Java first
  - APIs, portability, apps, level of documentation
  - Real-world-proximity
- Use ST for A/D, CASE, etc.
  - Beyond Q1
CS for Knowledge Majors

- Teach modeling, simulation, media integration, application composition
- Provide low-overhead development and distribution tools
- Duh! -- Squeak

The Squeak Pitch Thingy

- Language syntax (blocks as objects)
- 4th-generation class libraries
- “Rapid turn around” compilers
- Workspace-based system with multiple desktops/projects
- IDE tools, SCCS, CM, etc.
- Many compatible implementations
- SqueakOS
- Object/costume framework
- “Stuff is easy”
- “Parallel thinking”

Conclusions

- It’s just not a perfect world
- ST has strengths and weaknesses
- C* have strengths and weaknesses
- Teaching CS involves a lot of different topics at several levels
- We also need to teach some subset of these skills to non-CS majors
- We can design innovative and effective curricula for several target audiences