Music in Introductory Object Oriented Programming

Viera K. Proulx

College of Computer and Information Science
Northeastern University, Boston MA, USA
Outline

- TeachScheme/ReachJava curriculum
- Game design: motivation, challenge
- Sound and music: new opportunities
- Programmer’s view
- Design considerations
- Support for test-first design
- Student experiences
TeachScheme/ReachJava curriculum

• Design Recipe for functions:
  1. Analyze the problem, define data needed
  2. Contract and purpose for the function
  3. Examples of use with expected outcomes
  4. Inventory of available data, functions
  5. Design the function body
  6. Turn examples into tests and run tests
Game design: motivation, challenge

Special libraries that focus on model design

• World:
  • draw()
  • World onTick()
  • World onKeyDown(String)
  • World endOfWorld(String)
  • bigBang(width, height, speed)

• Canvas
  • new Canvas(width, height)
  • show()
  • drawDisk(center, radius color)
  • drawCircle(center, radius, color)
  • drawRect(nw, width, height, color)
  • drawString(pos, text, color)

Design the scene, program the next scene
Game design: motivation, challenge

Special libraries that focus on model design

• World:
  • draw()
  • void onTick()
  • void onKeyEvent(String)
  • void endOfWorld(String)
  • bigBang(width, height, speed)

• Canvas
  • new Canvas(width, height)
  • show()
  • drawDisk(center, radius color)
  • drawCircle(center, radius, color)
  • drawRect(nw, width, height, color)
  • drawString(pos, text, color)

Design the scene, program the next scene
Game design: motivation, challenge

Special libraries that focus on model design

Kyle Maguire, James Kandebo
Game design: motivation, challenge

Enforce Test-First Design

• Check the state of the world after a tick
• Check the state of the world in response to a key
• Check the end of world design
• Visually test the display of scenes

It promotes better understanding, cleaner design
(It is hard to test a function that does 20 things)
Game design: motivation, challenge - no more ...

Second semester: Java instead of Scheme; mutation

• Same games in a new language - little challenge
• New paradigm: object-oriented language
• New programming style: state change (mutation)

We needed new motivation
We needed more challenging environment
We needed a framework for imperative testing
Play music - Carry a Tune (in a bucket?)

Use Java MIDI support to play tunes:
Synthesizer, Program, Instruments, Notes

**Tune** = Instrument (MIDI) + Note (pitch)

- **tickTunes**
  - add tunes to play on each tick
  - we stop previous tunes
  - start new ones
  - no rythm

- **keyTunes**
  - add tunes to play for each key
  - we stop previous tunes
  - start new ones
  - no rythm

Original design: very simple, yet effective
Game design: motivation, challenge

Constructionism 2010, Paris, France, 16 August 2010

Special libraries that focus on model design
Pedagogical goals

• Creativity and constructionism
• Learn to work with sequences of notes
• Compose sequences into melodies
• See data that represents musical ideas
• See how such programs can generate music from data
• See how Test-First Design works in this setting

First semester results: surpassed expectations
Student experiences

• Define tempo by playing no tunes on some ticks

• Experiment with instruments, melodies

• Asked for a richer environment: note duration, program change, mouse actions, key press-release

A sample musical sequence:

```
public int synthhigh[] = {
    noteAp, 0, 0, noteAp, 0, 0,
    noteA, 0, 0, noteA, 0, 0,
    noteG, 0, 0, 0, 0, noteG, 0,
    0, 0, noteG, 0, 0, noteG, 0,

    noteAp, 0, 0, noteAp, 0, 0,
    noteA, 0, 0, noteA, 0, 0,
    noteG, 0, 0, 0, 0, noteG, 0,
    0, 0, noteG, 0, 0, noteG, 0,

    noteAp, 0, 0, noteAp, 0, 0,
    noteA, 0, 0, noteA, 0, 0,
    noteG, 0, 0, 0, 0, noteG, 0,
    0, 0, noteG, 0, 0, noteG, 0,

    noteAp, 0, 0, noteAp, 0, 0,
    noteA, 0, 0, noteA, 0, 0,
    noteG, 0, 0, 0, 0, noteG, 0,
    0, 0, noteG, 0, 0, noteG, 0,
};
```
Current SoundLib library:

- **Note**: e.g. C4s2 = C# in 4th octave 2 beats
  - multiple ways for defining a note (pitch only, pitch + duration, String)
- **Chord**: a collection of notes
- **Tune**: an Instrument and a **Chord**
- **SoundConstants**: names for notes, instruments, channels, programs
- **TuneBucket**: 16 **Tunes** (one for each MIDI channel)
- **MusicBox**: Synthesizer + SoundBank with play methods, support for tests
- **Melody**: a sequence of **Chords** with iterator
Qwortet: A sequencer for music composition

Visually compose music, allow replay

Robby Grodin
Qwortet: A sequencer for music composition

Constructionism 2010, Paris, France, 16 August 2010

Visually compose music, allow replay

Robby Grodin
BeatWorld: music recording and replay

Start recording - z
Stop recording - x
Octave down - c
Octave up - v

Recordings choice:

Kyle Montag
Design considerations

• **Note** class uses multiple representations of data
  ★ verifies validity
  ★ automatically converts between representations

• **Tests:** check the contents of the TuneBucket
  ★ after each tick
  ★ after each key event
  ★ check what notes are playing after several ticks

• **Composition:** work with sequences of data
  ★ compose a canon, reverse the sequence, repeat sequences
  ★ create a loop; build new way of displaying the data
Conclusion

• Simple environment that motivates and engages
• Number of pedagogical opportunities
• Careful design to support pedagogical goals

Acknowledgements

• Erich Neuwirth (Music and Spreadsheets)
• Jenny Sendova (Music sequences), Uri Wilensky (NetLogo)
• Matthias Felleisen (World and Universe libraries; TeachScheme/ReachJava)
• NSF DUE 0920182:

“Integrating Test Design into Computing Curriculum from the Beginning”
Find out more …

• http://www.ccs.neu.edu/home/vkp

• http://www.ccs.neu.edu/javalib

• http://www.ccs.neu.edu/javalib/SoundLib

• http://teachscheme.org

• http://www.bootstrapworld.org