4 Overview on Approaches to Multimedia Programming

4.1 Historical Roots of Multimedia Programming
4.2 Squeak and Smalltalk: An Alternative Vision
4.3 Frameworks for Multimedia Programming
4.4 Further Approaches & Systematic Overview

Literature:
Video lecture available at http://www.archive.org/details/AlanKeyD1987
Mark Guzdial: History of Squeak
Lecture notes at http://coweb.cc.gatech.edu/cs2340/3608
http://wiki.squeak.org/squeak/3139

Ivan Sutherland’s Sketchpad, 1963

First object-oriented drawing program
Master and instance drawings
Rubber bands
Simple animations
Douglas C. Engelbart 1962

- Born 1925, Ph.D. Berkeley 1955
- Influenced by Vannevar Bush’s article “As We May Think” (1945)
- 1962: Research Project at SRI (Stanford Research Institute):
  - “Augmenting Human Intellect: A Conceptual Framework”
  - Research support triggered by the “Sputnik shock” (1957)
- Basic ideas:
  - Computer supported learning
  - Computer supported collaboration
  - Seamless integration of computer interaction into workflows
- Development of the “NLS” (onLine System)
  - Demonstrated 1968 in Brooks Hall, San Francisco
- 1970: Patent application for “X-Y pointing device” (mouse)

http://www.bootstrap.org/augdocs/friedewald030402/augmentinghumanintellect/ahi62index.html

NLS Demo 1968
Alan C. Kay

- U. Utah PhD student in 1966
  - Read Sketchpad, Ported Simula
- Saw “objects” as the future of computer science
- His dissertation:
  - Flex, an object-oriented personal computer
  - A personal computer was a radical idea then
  - How radical?

"There is no reason anyone would want a computer in their home." (Ken Olsen, Digital Equipment Corp, 1977)

Further stations of Alan Kay’s life:
- Stanford Artificial Intelligence Laboratory
- Xerox PARC
- Atari
- Apple
- Disney Interactive
- Viewpoints Research Institute
- Hewlett-Packard

from M. Guzdial

The Dynabook Vision

- Small, handheld, wireless(!) device – a new medium
- Can be used creatively by everybody, in particular children, for learning
- Xerox PARC Learning Research Group, early 70s

Tablet PC, 2004
Xerox PARC Learning Research Group: Smalltalk-72

- Object-oriented programming system
  - Mouse
  - Windows
  - Icons
  - Pop-up menus
- Uses simple object-oriented language “Smalltalk”
- Idea of user interface: Make computers easy to use for everybody
- Idea of language: make programming both more simple and more powerful (e.g. include multimedia: sound)

The Alto

- The machine the prototype of which impressed Steve Jobs so much that he decided to produce the Lisa/Macintosh kind of computers for the mass market (1979)
  - Graphical user interface
  - Networked via Ethernet
  - Programming language Smalltalk
4 Overview on Approaches to Multimedia Programming

4.1 Historical Roots of Multimedia Programming

4.2 Squeak and Smalltalk: An Alternative Vision

EToys: Visual Programming in Squeak
Introduction to Smalltalk
Multimedia in Squeak

4.3 Frameworks for Multimedia Programming

4.4 Further Approaches & Systematic Overview

Literature:
http://www.squeakland.org
Back to the Future: Squeak

- Smalltalk:
  - Developed 1972
  - Commercial versions from 1980 on
- 1995: Alan Kay, Dan Ingalls, Ted Kaehler at Apple
  - Build on Open Source Software strengths
    - Use the distributed power of Internet-based programmers
  - Available Smalltalk versions had lost many media capabilities
- Later on, the Squeak team moves to Disney
  - “It’s all about media”
- Multimedia in Squeak:
  - 16 voice music synthesis
  - 3-D graphics, MIDI, Flash, sound recording
  - Network: Web, POP/SMTP, zip compression/decompress

Basics of Squeak Interaction (1)

- Squeak assumes a three-button mouse
- Menus are invoked by clicking on objects
  - clicking on surface opens “world” menus
- “Red”
  - Windows: left-button click
  - MacOS: simple click
- “Yellow”
  - Windows: middle-button click
  - MacOS: option + click
- “Blue”
  - Windows: right-button click
  - MacOS:  + click

(A different colour mapping... )
Basics of Squeak Interaction (2)

• Flaps:
  – Areas which can be opened or closed in a drawer-style
  – Often used as repositories ("parts-bins")

• Collapsing windows:
  – A window can be collapsed or expanded

• Tiles:
  – Objects can be represented by "tiles"

Etoys: Example “Car Race” (1)

• Step 0:
  Create a new empty project
  – world menu -> open...
  -> morphic project
  – enter new project by double-click

• Step 1: Draw the things with which we want to play
  – Very simplistic bitmap-oriented painting tool

• Step 2: "Keep" the drawing
  – We get a Squeak object
    » Free form, not square
    – Can be moved around

Note: Slides refer to Squeak 3.6, slight changes in version 3.8!
“Halo” of a Squeak Object

- The “halo” is a circular graphic menu which can be invoked on any object by a mouse click
  - “blue” click
  - special “playfield configuration” (preferences): invoked just by mouse over
- The “halo” consists of the following buttons:
  - menu
  - pick
  - move
  - delete
  - collapse
  - viewer
  - make tile
  - rotate
  - duplicate
  - debug
  - repaint
  - change color
  - resize

Squeak Viewers

- Step 3: Create a viewer (e.g. via the object’s halo)
  - Special flap for quickly showing and hiding the viewer
  - Rename sketch in viewer e.g. to “Car”
- Shows categories of properties and commands for objects
  - Categories: Object is derived from a subclass in a complex class hierarchy
  - Viewer can show many different categories in parallel
- Commands can be immediately executed (exclamation mark button)
  - Car can be moved, turned (Note: Orientation to be set in “rotate” mode to define direction of movement)
Squeak Scripts

- **Script:**
  - simple sequence of commands
  - executed under user control or automatically through a timer ("ticking").
- Represented by windows
  - created by drag-and-drop
  - "Tiles" represent objects and actions
- Step 4: Create a script
  - "add new script" in viewer
  - drag "empty script" onto surface
- Step 5: Add forward command
  - drag it from the Car viewer
  - adjust the parameter(s)

Running a Script

- Step 6: To control all scripts, use a new script control object.
  - To be found under the "Widgets" flap, like many other helpful tools
- All scripts of the project are simultaneously started and stopped through one button
  - Again just one drag operation to instantiate the object
- Example: Now car can be "driven" forward (till the border of the screen)
Object Interaction in Scripts

- Parameters of script commands can be computed from other objects’ properties (by dragging the property onto the parameter location)
- Local adjustments can be added at the end (factor, offset etc.)

User Control through Graphical Objects

- Graphical manipulations can be used to control other objects
- Example:
  - Steering wheel graphics
    - Drawn by hand
    - Viewer attached
  - Rotated by user (e.g. through halo operations)
  - Heading of wheel is transferred to car
  - A “servo steering” i.e. a less sensitive transfer is recommendable
Watcher

• The values of object properties can be easily shown on the screen
  – Updated regularly and automatically
• Technically, this is an “Observer” mechanism
  – Hidden behind simple drag&drop interface
• Watcher:
  – Simple watcher (value), Detailed watcher (value plus label)
  – Can be obtained from menu left of property (in viewer)
  – Can be placed anywhere on screen

Sensors for Environment

• Squeak objects can easily observe where they are currently located
  – Through coordinates
  – Simpler: through colours
• Sensors:
  – Realizable as special parts of the graphics with a unique colour
  – “color x sees color y” test: Which colour is below the sensor?
• Example:
  – Grey road, car with two sensors
  – Alert lamp shall go red when one of the sensors is not on road
Example: Alert Lamp

Example: Auto-Steering

- Interaction among objects can be designed in control loops
- Example:
  - Car automatically moves forward
  - Sensor detects border of road
  - Car automatically steers to stay on the road
- Enables complex interactive learning experiences (setting up feedback loops)
4 Overview on Approaches to Multimedia Programming

4.1 History of Multimedia Programming

4.2 Squeak and Smalltalk: An Alternative Vision
- EToys: Visual Programming in Squeak
- Introduction to Smalltalk
  - Multimedia in Squeak

4.3 Frameworks for Multimedia Programming

4.4 Further Approaches & Systematic Overview

Literature:
http://www.squeak.org (tutorials)

Smalltalk Programming is Open & Interactive

- Smalltalk programs are always ready for execution, even small parts of the code can be evaluated instantly.
- The interpreter state is saved/loaded in an “image” file.
- The full code of the runtime system can be inspected at any time.

“do it” (ctrl-d)
“print it” (ctrl-p)
Basic Rules of Smalltalk

• Every variable is an object.
  – There are no basic types which are not objects!
  – Even classes are objects!
• Code is always triggered by sending a message to an object.
• All methods return a value.
• There are three types of messages
  – Unary, e.g. 3 \textit{negated}.
  – Binary, e.g. \texttt{a + b}.
  – Keyword, e.g. \texttt{Transcript show: a}.
    » \texttt{show} message with parameter \texttt{a} is sent to object \texttt{Transcript}
• All code is evaluated from left to right.
  – Unary messages first, then binary, then keyword messages
  – There are no operator precedence rules.
• Assignment evaluates right hand side and assigns the result to left hand side.

Smalltalk Blocks

• \texttt{a := [2 + 3].}
  \texttt{a value.} \hspace{1cm} \textit{Result: 5}

• \texttt{c := [:a :b | a + b].}
  \texttt{c value: 5 value: 7.} \hspace{1cm} \textit{Result: 12}
  \textit{(a multiple-part message)}

• \texttt{x := 3.}
  \texttt{y := 5.}
  \texttt{(x = y)}
  \texttt{ifTrue: [Transcript show: 'equal']}
  \texttt{ifFalse: [Transcript show: 'not equal'].}
  \textit{Control flow realized by message passing mechanism}
Interval Objects and Loops

- An Interval object:
  \( a := 10 \text{ to } 20. \)
  \( a \) inspect.
- Looping through the interval:
  \( a \) do: [:i | Transcript show: i; cr].

Advanced Language Constructs in Squeak

- Infinite number precision
  - 1000 factorial / 999 factorial. \( 1000 \)
  - \((1/3) + (2/3). \) \( 1 \)
  - Float infinity + 1. \( \text{Infinity} \)
  - Float infinity / Float infinity. \( \text{NaN} \)
- Lazy evaluation
- High level iterators
  - \( a := \#(1 \ 2 \ 3). \)
  - \( a \) collect: [:x | x*2]. \( \#(2 \ 4 \ 6) \)
  - \( a \) reject: [:x | x odd]. \( \#(2) \)
Browser Window

BankAccount Example

- Constructed interactively
  - Create new class template
  - Fill in instance variable (balance)
  - Fill in methods
    » initialize
    » deposit
    » withdraw
- At any point in time, creation of objects and inspection is possible
- (Credits for the example: John Maloney)
### Defining Classes: BankAccount

Object subclass: #BankAccount
  instanceVariableNames: 'balance'

balance
  ^ balance.
initialize
  balance := 0.
deposit: amount
  balance := balance + amount.
withdraw: amount
  (amount > balance)
    ifTrue: [^ self inform: 'No more money!'].
    balance := balance - amount.

### BankAccount with History

- Extend class with history variable
  - Initialize with empty ordered collection
    history := OrderedCollection new.
- Update history
  balance: newBalance
    balance := newBalance.
    history addLast: newBalance.
deposit: amount
  self balance: (balance + amount).
withdraw: amount
  (amount > balance)
    ifTrue: [^self inform: 'No more money!'].
    self balance: (balance - amount).
Graphical Object (Morph) for BankAccount

```smalltalk
defHistoryMorph
    "displays account history as barchart"
    | bars m |
    bars := history collect:
        [:v | Morph new extent: 30@v].
    m := AlignmentMorph newRow
        hResizing: #shrinkWrap;
        vResizing: #shrinkWrap;
        cellPositioning: #bottomRight.
    m addAllMorphs: bars.
^m.

Make visible by:
acc historyMorph openInWorld.
```

---

Event Handling in Morphs

```smalltalk
Morph subclass: #TestMorph
category: 'My Stuff'
handlesMouseDown: evt
^ true
mouseDown: evt
    self position: self position + (10 @ 0).

TestMorph new openInWorld.
```
EToys and Smalltalk

- Squeak contains a full Smalltalk development system
- EToy scripts can be switched between iconic or textual representation
- EToy scripts are found in the browser hierarchy
- EToy scripts are just shortcuts in writing Smalltalk

4 Overview on Approaches to Multimedia Programming

4.1 History of Multimedia Programming

4.2 Squeak and Smalltalk: An Alternative Vision

Squeak
EToys: Visual Programming in Squeak
Introduction to Smalltalk

Multimedia in Squeak

4.3 Frameworks for Multimedia Programming

4.4 Further Approaches & Systematic Overview

Literature:
http://www.squeak.org
Wonderland: 3D Worlds in Squeak

- 3D objects can be moved around in intuitively simple manner
  - Prefabricated models
  - Simple self-drawn sketches ("Pooh drawings")
- 3D objects are EToys.
- 3D objects can be manipulated with Smalltalk programs.

Squeak as a Multimedia Experimentation Platform

- Example: Sound in Squeak
Example: Playing Musical Notes in Smalltalk

    instr := AbstractSound soundNamed: 'oboe1'.
    note1 := instr soundForPitch: #c4 dur: 0.5 loudness: 0.4.
    note2 := instr soundForPitch: #ef4 dur: 0.5 loudness: 0.4.
    note3 := instr soundForPitch: #g4 dur: 0.5 loudness: 0.4.
    (note1, note2, note3) play.
    (note1 + note2 + note3) play.

    song := AbstractSound noteSequenceOn: instr from: #(
      (c4 0.35 400)
      (c4 0.15 400)
      (d4 0.5 400)
      (c4 0.5 400)
      (f4 0.5 400)
      (e4 1.0 400)).
    song play.