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

4.3 Director and Lingo: Advanced Multimedia Authoring

4.4 Frameworks for Multimedia Programming

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

Target Persons for Multimedia Programming Technology

<table>
<thead>
<tr>
<th>Intended activities</th>
<th>Non-technical people</th>
<th>Experts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passive consumption</td>
<td>Flash</td>
<td>—</td>
</tr>
<tr>
<td>Design of complex systems</td>
<td>Squeak</td>
<td>Flash</td>
</tr>
<tr>
<td>Experimentation, learning</td>
<td>EToys</td>
<td>Squeak</td>
</tr>
</tbody>
</table>
Smalltalk and Learning

- Original intention of Smalltalk:
  - Make the programming system so intuitive that children can use it
  - In fact a first multimedia authoring system
  - Children projects at Xerox PARC
- What happened to Smalltalk?
  - Becomes commercially targeted programming language
  - ParcPlace
  - Smalltalk-80
- Smalltalk is still the environment in which most programming inventions appear:
  - Design patterns
  - Extreme programming

Smalltalk Interpreter in Squeak

- Smalltalk:
  - The language of the first systems with a graphical user interface
- Smalltalk-80:
  - Standardized syntax for Smalltalk
- Smalltalk in Squeak:
  - Squeak system contains a full interpreter for Smalltalk-80 syntax
  - Squeak system is written in Squeak mostly (and cross-compiled to C)
    - 95% of the system is in Squeak
  - Smalltalk is the serious programming language in Squeak
    - Squeak scripting is just for kids... ??
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.

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 negated.
  - Binary, e.g. a + b.
  - Keyword, e.g. Transcript show: a.
- 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

• \( a := [2 + 3]. \)
  \( a \) value.  
  Result: 5  
  Assignment either by 
typing ":=" or 
by typing ":."  

• \( c := [:a :b | a + b]. \)
  \( c \) value: 5 value: 7.  
  Result: 12  
  (a multiple-part message)

• \( x := 3. \)
  \( y := 5. \)
  \((x = y)\)
  
  ifTrue: \([\text{Transcript show: 'equal']}\]
  ifFalse: \([\text{Transcript show: 'not equal']}\].

  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: \([:i | \text{Transcript show: } i; \text{cr}]. \)
Infinite Number Precision

- 100 factorial.
  
  933262154439441526816992338856266700490715968264381621
  4655929852175999322991560894146397615651828625369
  79208272036758251185210916864000000000000000000000000

- 1000 factorial / 999 factorial. 1000

- \( \frac{1}{3} + \frac{2}{3} \). 1

- 1 class. SmallInteger
- 1 class maxVal. 1073741823
- 1 class maxVal + 1. 1073741824

- Float infinity + 1. Infinity
- Float infinity / Float infinity. NaN
While Loops, Lazy Evaluation, Local Variables

$f n$

\[
f := 1. \\
n := 4. \\
[n > 1] \text{ whileTrue: } \{ f := f * n. n := n - 1 \}. \\
f.
\]

Smaller Steps:
\[
s := [n > 1]. \\
s \text{ inspect.} \\
s \text{ whileTrue: } \{ \text{Transcript show: } n. n := n - 1 \}.
\]

High-Level Iterators

\[
a := \#(100 200 300). \\
a \text{ do: } \{ x | \text{Transcript show: } x; \text{cr} \}. \\
a \text{ collect: } \{ x | x^2 \}. \#(200 400 600)
\]

\[
a := \#(1 2 3). \\
15 \text{ odd.} \\
a \text{ reject: } \{ x | x \text{ odd} \}. \#(2)
\]
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
    \[\text{history} := \text{OrderedCollection new.}\]
- Update history
  \[
  \begin{align*}
  \text{balance} & : \text{newBalance} \\
  \text{balance} & := \text{newBalance}. \\
  \text{history addLast: newBalance}. \\
  \text{deposit: amount} \\
  \text{self balance: (balance + amount)}. \\
  \text{withdraw: amount} \\
  (\text{amount > balance}) \\
  \text{ifTrue: [^self inform: 'No more money!']}. \\
  \text{self balance: (balance - amount)}. \\
  \end{align*}
  \]

Graphical Object (Morph) for BankAccount

\[
\text{historyMorph} \quad \text{“displays account history as barchart”}
\]
\[
| \text{bars m |} \\
\text{bars := history collect:} \\
\quad [:v | \text{Morph new extent: 30@v}]. \\
\text{m := AlignmentMorph newRow} \\
\quad \text{hResizing: #shrinkWrap;} \\
\quad \text{vResizing: #shrinkWrap;} \\
\quad \text{cellPositioning: #bottomRight.} \\
\text{m addAllMorphs: bars.} \\
\quad ^\text{m.}
\]

Make visible by:
\[
\text{acc historyMorph openInWorld.}
\]
EToys and Smalltalk

- 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 Director and Lingo: Advanced Multimedia Authoring

4.4 Frameworks for Multimedia Programming

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

\[
\begin{align*}
\text{instr} & := \text{AbstractSound soundNamed: 'oboe1'}. \\
\text{note1} & := \text{instr soundForPitch: '#c4 dur: 0.5 loudness: 0.4'.} \\
\text{note2} & := \text{instr soundForPitch: '#ef4 dur: 0.5 loudness: 0.4'.} \\
\text{note3} & := \text{instr soundForPitch: '#g4 dur: 0.5 loudness: 0.4'.} \\
(\text{note1, note2, note3}) & \text{ play.} \\
(\text{note1 + note2 + note3}) & \text{ play.} \\
\text{song} & := \text{AbstractSound noteSequenceOn: instr from: #(} \\
& \quad \text{ (c4 0.35 400)} \\
& \quad \text{ (c4 0.15 400)} \\
& \quad \text{ (d4 0.5 400)} \\
& \quad \text{ (c4 0.5 400)} \\
& \quad \text{ (f4 0.5 400)} \\
& \quad \text{ (e4 1.0 400))}. \\
\text{song} & \text{ play.}
\end{align*}
\]