### 4 Fundamental Issues in Multimedia Programming

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Multimedia Programming in Context</td>
</tr>
<tr>
<td>4.2 History of Multimedia Programming</td>
</tr>
<tr>
<td>4.3 A Radically Alternative Approach: Squeak</td>
</tr>
<tr>
<td>Etoys</td>
</tr>
<tr>
<td>Video: Squeak in a School Project (Gravity)</td>
</tr>
<tr>
<td>Smalltalk: A Brief Introduction</td>
</tr>
<tr>
<td>4.4 The Programmers’ Way: Multimedia Frameworks for Java</td>
</tr>
<tr>
<td>4.5 Trends and Visions</td>
</tr>
</tbody>
</table>

**Literature:**
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>

Squeak as an Experimentation Platform

- Example: Sound in Squeak
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.

```
\text{Transcript}
\begin{verbatim}
5 * 5 \text{n} 
\text{Spared:} \text{n}
\end{verbatim}
```

“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!
- Squeak 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`.
    - `show` message with parameter `a` is sent to object `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**

- `a := [2 + 3].`
  `a value.`  
  Result: 5

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

- `x := 3.`  
  `y := 5.`  
  `(x = y)`  
  `ifTrue: [Transcript show: 'equal']`
  `ifFalse: [Transcript show: 'not equal']`.  
  Control flow realized by message passing mechanism
Example: Playing Musical Notes in Smalltalk

```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.
```

4 Fundamental Issues in Multimedia Programming

4.1 Multimedia Programming in Context
4.2 History of Multimedia Programming
4.3 A Radically Alternative Approach: Squeak

4.4 The Programmers’ Way: Multimedia Frameworks for Java
   Java 2D + Advanced Imaging
   Java Sound
   Java Media Framework

4.5 Trends and Visions

Literature:
   J. Knudsen, Java 2D Graphics, O'Reilly 1999
   http://java.sun.com/products/java-media/2D/
Types of Multimedia Features for Programs

- Still 2D images (includes vector graphics, sampled)
- Moving 2D images (includes vector graphics, sampled)
- Sound (includes MIDI, sampled)
- 3D Scenes

Playback
Create
Process

Java Media APIs

- Java was from its beginnings intended as a multimedia programming language:
  - “Oak”, Java’s predecessor: designed to control Set Top Boxes for Interactive TV
- Java Media APIs
  - Loose collection of APIs defined and maintained by Sun
  - Main APIs: Advanced Imaging (JAI), Java Media Framework (JMF), Java 3D
  - APIs which have become part of standard distribution: Java 2D, Java Sound
- Style rather heterogeneous
- Not all multimedia programming tasks covered
  - E.g. animation
  - “Unofficial” APIs and implementations try to fill the gap
Vector Graphics Framework

Media types
- Still 2D images
  - vector graphics
  - sampled
- Moving 2D images
  - vector graphics
  - sampled
- Sound
  - MIDI
  - sampled
- 3D Scenes

Playback
Create
Process

Example: Java 2D

Rendering-Pipeline, Example Java 2D

Form
- fill()
- draw()
- stroke

Text
- drawString()
- font

Image
- drawImage()
- rendering hints

Raster representation
- clipping shape
- paint

(output)

Compositing rule
(for images)
Example: Drawing a Path

```java
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;

public class Path extends Frame {
    public static void main(String[] args) {
        new Path();
    }
    public Path() {
        setSize(500, 400); setLocation(200, 200); setVisible(true); ...
    }
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D)g;
        GeneralPath p = new GeneralPath();
        p.moveTo(50, 50);
        p.lineTo(70, 44);
        p.curveTo(100, 10, 140, 80, 160, 80);
        p.lineTo(190, 40);
        p.lineTo(200, 56);
        p.quadTo(100, 150, 70, 60);
        p.closePath();
        g2.draw(p);
    }
}
```
Vector Animation Framework

- Media types
  - Still 2D images includes vector graphics
  - Moving 2D images includes vector graphics
  - Sound includes MIDI
  - 3D Scenes includes sampled

Examples (Non-official, not widespread!):
JGoodies (www.jgoodies.com)
SceneBeans (http://www-dse.doc.ic.ac.uk/Software/SceneBeans)

How to Design a Vector Animation Framework?

- Key concepts needed:
  - Clock for time-dependent change
  - Hooks on 2D graphical objects to change parameters
    » Location, orientation, size, colour etc.
- Disadvantage of Java 2D against Smalltalk, Flash:
  - No built-in objects with graphical properties (e.g. color, line thickness)
  - Instead drawing tools to be modified (more procedural approach):
    ```java
    g2.setPaint(Color.red);
    g2.fill(p);
    ```
- Some design ideas:
  - Interfaces for animation (e.g. in JGoodies)
    » Using event listener mechanism
  - Scene graphs (e.g. in SceneBeans)
JGoodies Example (1)

```java
private Animation createAnimation() {
    Animation welcome =
        BasicTextAnimation.defaultFade(label1, 2500, "Welcome To", Color.darkGray);

    Animation theJGoodiesAnimation =
        BasicTextAnimation.defaultFade(label1, 3000, "The JGoodies Animation", Color.darkGray);

    Animation description =
        BasicTextAnimation.defaultFade(label1, label2, 2000, -100, "An open source framework" +
        "for time-based real-time animations in Java.", Color.darkGray);...
}
```

JGoodies Example (2)

```java
Animation all =
    Animations.sequential(new Animation[] {
        Animations.pause(1000), welcome,
        Animations.pause(1000), theJGoodiesAnimation,
        Animations.pause(1000), description,
        Animations.pause(1000), features,
        Animations.pause(1000), featureList,
        Animations.pause(1500),
    });
```
Concepts for Time-Based Animation: SMIL

- E.g. the JGoodies framework clearly relates itself to the ideas of SMIL
- SMIL = Synchronized Multimedia Integration Language
  - XML application
  - Standardized by W3C
  - Not widespread but re-used in many other standards (e.g. MPEG-4)
- Idea:

  **Layout:**

  ![Layout Diagram]

  **Body:**

  Contents (here text) appearing over time

  Expression with concurrency operators (sequential, parallel)

SMIL Example

```xml
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <root-layout width="356" height="356"/>
      <region id="img_region" width="256" height="256" left="50" top="50"/>
    </layout>
    <transition id="img_wipe" type="barWipe" dur="3s"/>
  </head>
  <body>
    <par>
      <seq>
        <img region="img_region" src="....jpg" ... transIn="img_wipe" fill="transition"/>
        <img region="img_region" src="....jpg" ... transIn="img_wipe" fill="transition"/>
        ...
      </seq>
      <audio src="....mp3" end="32s"/>
    </par>
  </body>
</smil>
```
Sub-TYPES of Vector Graphics Animation

• Layout-bound animation:
  – Similar to the SMIL concepts
  – Basic layout of display regions determined statically
  – Used e.g. in SMIL, JGoodies, ...

• Layout-free animation:
  – Freely moving animated objects

• (As always) the border is not fully clear:
  – Objects in layout-bound animation may move away from their start positions

Scene Beans

• SceneBeans defines a graphical display using a "scene graph".
  – A Java Bean is a simple software component in Java following naming conventions to enable manipulation in authoring systems.

Material on SceneBeans adapted from Nat Bryce
Example: Spinning Square

Rectangle

Example: Spinning Square

RGBAColor

Rectangle
Example: Spinning Square

- Translate
- RGBAColor
- Rectangle

Example: Spinning Square

- Rotate
- Translate
- RGBAColor
- Rectangle
Example: Spinning Square

Loop

T

Rotate

T

Translate

S

RGBAColor

P

Rectangle

Example: Spinning Square

Loop

T

Rotate

T

Translate

S

RGBAColor

P

Rectangle
Composable Animations

An Animation object encapsulates a scene graph and the behaviours that animate it.

Can itself be embedded in a scene graph and run as an activity.

Animation objects are the unit of animation design and reuse.

XML File Format

- It’s not practical to create animations by programming Java
  - Non-programmers are not able to create animations
  - Frustrating edit/compile/debug cycle while fine-tuning animation parameters

- Therefore SceneBeans defines a file format for loading animations
  - based on XML

- XML document is used as a “wiring language”
  - Defines configuration of scene graph and behaviour beans
  - Beans dynamically loaded on demand
  - Animations not limited to fixed set of beans

- XML Processing instructions are used to introduce new packages of beans
  - Can load beans across the network - useful if animation is in an applet