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

Selected other approaches
- Classification of multimedia applications
- Classification of concepts for multimedia programming
- Classification of development tools & languages

Adobe Director Desktop (German Version)
Motion Tweening in Director

- Very similar to Flash but easier
  - Each sprite has a default registration point for a motion path
  - Drawing motion paths is straightforward
  - Key frames used to reshape motion path

Director: The Lingo Paradigm

- Lingo is the programming language of the authoring tool Adobe Director.
- Lingo is very much inspired by "HyperTalk" (Apple)
- All programming is programming event handlers
- There is no main program
  - Effectively the event handler of “prepareMovie” is kind of a main program
- Program code is only meaningful together with project file of the authoring system
  - No stand-alone programs
- All code is scattered over the project
Object-Orientation in Director:
“Parent-Child Programming” (1)

- “Parent script” (class):

```plaintext
property pVorname, pNachname

on new me
    return me
end

on fill me, vorname, nachname
    pVorname = vorname
    pNachname = nachname
end
```

Object-Orientation in Director:
“Parent-Child Programming” (2)

- Global script (film script):

```plaintext
global lUsers

on prepareMovie
    lUsers = []
end

on fillOut
    temp = new(script "parent script")
    fill(temp, member("vorname").text, member("nachname").text)
    append(lUsers, temp)
    clearFields
end

on clearFields
    member("vorname").text = ""
    member("nachname").text = ""
end
```

vorname, nachname are text input fields
SMIL Example: Slide Show (1)

```xml
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <root-layout width="356" height="356"/>
      <region id="brush_region" z-index="1"/>
      <region id="img_region" width="256" height="256"
              left="50" top="50" z-index="2"/>
    </layout>
    <transition id="img_wipe" type="barWipe"
               dur="3s"/>
    <transition id="bkg_wipe" type="barWipe"
               direction="reverse" dur="3s"/>
  </head>
</smil>
```

SMIL Example: Slide Show (2)

```xml
...
<body>
  <par>
    <seq>
      <img_region="img_region" src="....jpg" ...>
      ... transIn="img_wipe" fill="transition"/>
      ...
    </seq>
    <seq>
      <img_region="brush_region"
      ...
      transIn="bkg_wipe" fill="transition"/>
    </seq>
    <audio src="....mp3" end="32s"/>
  </par>
</body>
```

Seite <Nr.>
QuickTime for Java

- The QuickTime media framework (Apple) is available as a programming framework as well for
  - C, C++
  - Java (Wrapper, QT for Java)
  - (for Windows and MacOS only)
- Programs can use the QuickTime for instance to
  - Play movies and audio files
  - Play SMIL presentations
  - Display images
  - Use an extensive file conversion library
  - Compose images by compositing overlays
  - Using built-in graphical transition effects
  - Display simple animations (QT Sprites)
    » Including movement paths, interactive controls, event handlers

QuickTime for Java Example

```java
public Zool(String s) {
    super(s);
    setResizable( false );
    setBounds( 0, 0, WIDTH, HEIGHT );
    QTCanvas myQTCanvas = new QTCanvas( 
        QTCanvas.kInitialSize, 0.5F, 0.5F );
    add( myQTCanvas );
    try {
        QTFiler imageFile = new QTFiler(
            QTFactory.findAbsolutePath("xyz.jpg "));
        GraphicsImporterDrawer mapDrawer =
            new GraphicsImporterDrawer( imageFile );
        myQTCanvas.setClient( mapDrawer, true );
    } catch ... 
}
```

# 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

- **Selected other approaches**
- **Classification of multimedia applications**
  - Classification of concepts for multimedia programming
  - Classification of development tools & languages

---

## Space, Time and Interactivity

- **Multimedia applications** have space and time extension
- **Space:**
  - Conveys essential information through graphical design
  - Example: Game
  - Advanced concepts (e.g. ubiquitous applications, AR): 3D space, real world
- **Time:**
  - Progress of time influences presentation
  - Example: Video, audio, animation
- **Third dimension:** **Interactivity**
  - Degree of user interaction
- **Multimedia objects**
  - occupy a certain range in space, time and interactivity
  - address a certain number of modalities (auditive, visual, tactile)
Classification Space

Time usage:
- Still picture
- Linear progress
- Interaction dependent progress

Interactivity:
- Fully automatic
- Confirmations & questions
- Interactive controls
- Dragging of objects

Space usage:
- Static layout
- Scenes
- Scenes & objects
- Fully dynamic

Example 1: Slide Show

Time usage:
- Still picture & Linear progress

Interactivity:
- Fully automatic

Space usage:
- Static layout

Degree of interactivity
Example 2: Animated Product Presentation

- **Time usage:**
  - Linear progress
  - Interaction dependent progress

- **Interactivity:**
  - Confirmations & questions
  - Interactive controls

- **Space usage:**
  - Scenes

Example 3: Game

- **Time usage:**
  - Interaction dependent progress

- **Interactivity:**
  - Interactive controls

- **Space usage:**
  - Scenes & objects
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

Selected other approaches
Classification of multimedia applications
Classification of concepts for multimedia programming
Classification of development tools & languages
Multimedia Development Pattern: Space Layout

- The location of objects in the presentation space is fixed by assigning coordinates to the objects.
- Space usage:
  - Static layout, scenes or scenes&objects
- Usually combined with highly automatic time usage and low interactivity
- Examples:
  - SMIL layout
  - Flash & Director stage
  - JGoodies Swing layout

Pattern Space Layout: Application Range

Each pattern has an application range. It is suitable for multimedia applications the properties of which are within its range.
Multimedia Development Pattern: Clockwork

- The current properties of presentation elements are derived from the current value of a “clock” ticking at regular time intervals
- Time usage: Linear progress
- Limited interactivity: Automatic or confirmations and questions
- Usually combined with static layout or scenes and objects
- Examples:
  - Timeline in Flash, Director
  - EnterFrame-Events in Flash ActionScript
  - Ticking scripts in Squeak
  - PAActivity in Piccolo

```java
PAActivity flash = new PAActivity(-1, 500, currentTime + 5000) {
    protected void activityStep(long elapsedTime) {
    ...
```

Multimedia Development Pattern: Interpolation

- A parameter (usually regarding a graphical property) is assumed to change its value continuously dependent of another parameter (e.g. time). The dependency can follow a linear or other rules of computation.
  - Fixed values for the dependent parameter are given for certain values of the base parameter.
  - Intermediate values of the dependent parameter are computed by interpolation.
- Space usage: scenes and objects mainly
- Time usage: Linear progress only
- Usually combined with low interactivity (on this level)
- Examples:
  - Tweening in Flash
  - Animation methods in Piccolo

```java
PAActivity al = aNode.animateToPositionScaleRotation(0, 0, 0.5, 0, 5000);
```
Multimedia Development Pattern: Scheduled Time

- An activity is assumed to start at a given point in time. The start time is specified
  - in absolute terms, or
  - relatively to another activity
- Time usage: Mainly automatic
- Low interactivity
- Examples:
  - SMIL time specifications (begin attribute)
  - Placement of code or object in certain frame in Flash
  - setStartTime() and startAfter() methods in Piccolo

Multimedia Development Pattern: Process Algebra

- Presentation is built from atomic parts (processes) each of which is executed sequentially.
- Presentation is constructed using operations similar to mathematical process algebra: sequential composition, parallel composition, repetition, mutual exclusion, synchronisation options
- Time usage: Linear progress
- Space usage: Scenes or scenes&objects
- Low interactivity
- Examples:
  - Animations class of JGoodies
  - SMIL body: seq, par, excl
  - Sequence of frames and parallelism of layers in Flash
Various Representations of a Single Concept

```
<layout>
  <region id="r1" ...>
  <body>
    <seq>
      ...frame1
      ...frame2
    </seq>
  </body>
  ...
  <animation id="frame1" ...>
  <animation id="frame2" ...>
</body>
```

Component r1 = ...;
Animation frame1 = ...;
Animation frame2 = ...;
Animation all = Animations_sequential {
  new Animation[] {
    frame1, frame2}
};

Multimedia Development Pattern: Scene Graph

- Graph structure comprises all represented objects together with the operations (transformations) applied to them
- Space usage: Scenes & objects or fully dynamic
- Time usage: Linear progress or interaction dependent
- Examples:
  - Scene graph of SceneBeans
  - Scene graph of Piccolo
  - Implicit: Film Explorer view in Flash
Multimedia Development Pattern: Player Component

- For standardized time-dependent media types, a pre-fabricated component is made available which provides
  - Playback of associated media files
  - Standard VCR-style controls (play, pause, stop, rewind)
- Space usage: any
- Time usage: Linear progress
- Interactivity: Interactive controls
- Examples:
  - Flash MediaPlayback component
  - JMF Player component
  - QuickTime player in QT4Java

```java
try {
    p = Manager.createPlayer(new MediaLocator("file:"+file);
    p.addControllerListener(new ContrEventHandler());
    p.realize();
}
```

Multimedia Development Pattern: Event Handler

- Program code is not executed sequentially but triggered by events
- Space usage: any
- Time usage: Interaction dependent
- Interactivity: any
- Examples:
  - ActionScript event handlers
  - Lingo event handlers
  - JMF event handlers
  - Squeak/Smalltalk event handlers
  - ...

```java
try {
    p = Manager.createPlayer(new MediaLocator("file:"+file);
    p.addControllerListener(new ContrEventHandler());
    p.realize();
}
```
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

Selected other approaches
Classification of multimedia applications
Classification of concepts for multimedia programming
Classification of development tools & languages

Properties of Development Tools & Languages

- Supported multimedia development patterns (see above)
- Supported kinds of application (see above classification)
- Visual programming vs. Textual programming
  - Graphical editor vs. Textual vector data
    - Example: Flash vs. Piccolo
  - Integration of scripting language
    (script-less, integrated, separated, script-based)
    - Example: Script types in Flash
- Degree of abstraction & platform-independence
  - Modeling language vs. Programming language
  - Code generation, compilation, reverse engineering, round-trip engineering
    - Example: MML
- Run time/ Design Time Trade-Off
  - Compilation process or seamless interaction with living world
    - Example: Flash vs. Squeak


Conclusion

- What do you expect from the next generation of multimedia programming technologies?
- What is your forecast about the role multimedia programming will play in the future of software development techniques?