1 Example Technology: Macromedia Flash & ActionScript

1.1 Multimedia authoring tools - Example Macromedia Flash

1.2 Elementary concepts of ActionScript
   Scripting in General + „History“ of ActionScript
   Objects and Types in ActionScript
   Animation with ActionScript

1.3 Interaction in ActionScript
   Handling of Mouse Events
   Classical Model-View-Controller Programming

1.4 Media classes in ActionScript

Literature:
Colin Moock: Essential ActionScript 2.0, O'Reilly 2004
Creating a “Graphically Enhanced” User Interface

• Traditional programming
  – Example: Account with credit and debit function

• Additional “multimedia” features:
  – Auto-highlighting buttons
  – Visualization of money transfer direction
  – Visualization of “low” warning
The Account Class

class Account {

    var saldo:Number = 0;
    var num:Number;

    function Account(accnum:Number) {
        num = accnum;
    }

    function debit(n:Number) {
        saldo -=n;
    }

    function credit(n:Number) {
        saldo +=n;
    }

    function getNumber():Number {
        return (num);
    }

    function getSaldo():Number {
        return (saldo);
    }
}

Model-View-Controller (MVC) Paradigm

- **Model:**
  - Business model, mostly independent of user interface
  - Observable by arbitrary objects (application of Observer pattern)
- **View:**
  - Representation on user interface
  - Observes the model
  - Asks required data from the model
- **Controller:**
  - Modifies values in the model
  - Is driven by user interactions, therefore bound to elements of interface
  - Handles events mainly by calling methods of the model
Observer Design Pattern

• Classical design pattern, made publicly conscious by Gamma/Helm/Johnson/Vlissides
• Integrated into many frameworks, e.g. Java standard library (SDK)
• Idea:
  – *Observable* is a class from which any class is derived which shall notify other objects of changes
  – *Observer* is an interface through which objects can be notified of changes
    » Providing a callback method *update*
  – Observable provides a method *notifyObservers* to actually inform observing objects
  – Observers have to register (* addObserver*)
How to Realize an Observer Mechanism?

• Approach 1: Look for an existing standard mechanism
  – ActionScript: Contains standard class library
  – Possible solution: Use `mx.events.EventDispatcher`
  – However: Not officially documented, not fully identical

• Approach 2: Re-implement the pattern
  – Not difficult for ActionScript, just port the Java library source code
  – Own class library can be defined as a local package or at a central location

• Approach 3: Look for somebody who has already re-implemented the pattern
  – For ActionScript 2 e.g. Colin Moock, author of "Essential ActionScript 2.0"
Excerpt from util.Observable

```java
import util.*;

/**
 * A Java-style Observable class used to represent the "subject"
 * of the Observer design pattern. Observers must implement the Observer
 * interface, and register to observe the subject via addObserver().
 */
class util.Observable {
    private var changed:Boolean = false;
    private var observers:Array;

    /**
     * Constructor function.
     */
    public function Observable () {
        observers = new Array();
    }

    /**
     * Adds an observer to the list of observers.
     * @param o The observer to be added.
     */
    public function addObserver(o:Observer):Boolean {
        ...
    }
}
```
import util.*;

/**
 * The interface that must be implemented by all observers of an
 * Observable object.
 */
interface util.Observer {
  /**
   * Invoked automatically by an observed object when it changes.
   *
   * @param o The observed object (an instance of Observable).
   * @param infoObj An arbitrary data object sent by
   * the observed object.
   */
  public function update(o:Observable, infoObj:Object):Void;
}
Model: Observable Account Class

import util.*;

class Account extends Observable {

    var saldo:Number = 0;
    var accNum:Number;

    function Account(an:Number) {
        accNum = an;
    }

    function debit(n:Number) {
        if (n < 0) return;
        saldo -=n;
        if (n <> 0){
            setChanged();
            notifyObservers(false);
        }
    }

    function credit(n:Number) {
        if (n < 0) return;
        saldo +=n;
        if (n <> 0){
            setChanged();
            notifyObservers(true);
        }
    }

    ...
}

View: User Interface Design

- Main output form is a (dynamic) text field
- However:
  - Text fields cannot carry ActionScript code
  - Text field cannot be easily associated with AS class
- How can we stay object-oriented?
- Idea: Add a specific view object which just refers to the visible text field object
Class AccountView

import util.*;

class AccountView implements Observer {
    private var saldo_txt:TextField;
    private var lowWarning_mc:MovieClip;
    private var myAccount:Account;
    private var saldo:Number;

    private function animSaldo() {
        saldo = myAccount.getSaldo();
        saldo_txt.text = String(saldo);
        if (saldo < 0)
            lowWarning_mc.gotoAndPlay("startAnim");
        else
            lowWarning_mc.gotoAndStop("stopAnim");
    }

    public function AccountView(t:TextField, l: MovieClip, a: Account) {
        saldo_txt = t;
        lowWarning_mc = l;
        myAccount = a;
        myAccount.addObserver(this);
        animSaldo();
    }
} ...
More Animation...

- Extending AccountView to cover an animation for money transfers:
  - Add `credit_mc`, `debit_mc` as constructor parameters and local variables
- Call animation when update is issued by model
  - Depending on direction of money flow as given by info object

```javascript
public function update(o:Observable, infoObj:Object):Void {
  var credit:Boolean = Boolean(infoObj);
  if (credit)
    credit_mc.gotoAndPlay("startAnim");
  else
    debit_mc.gotoAndPlay("startAnim");
  animSaldo();
}
```
Controller: User Event Handling

• Using Flash’s built-in Button class makes highlighting easy.
• Event handling code (example “credit”, “debit” is similar):

```actionscript
on (release) {
    var amount:Number = Number(amount_txt.text);
    if (isNaN(amount) or (amount < 0)) {
        amount_txt.text += "?";
    } else {
        myAccount.credit(amount);
    }
}
```
Constructing the Objects in Main Timeline
Alternative: Extending a TextField Object

- This is from an alternative solution!
- saldo_txt is a TextField object generated in the authoring tool
- Text fields cannot be linked to ActionScript classes
- A method can be added as follows (in main timeline):

```javascript
saldo_txt.update = function(){
    var saldo: Number = myAccount.getSaldo();
    saldo_txt.text = saldo;
    if (saldo < 0)
        lowWarning_mc.gotoAndPlay("startAnim");
    else
        lowWarning_mc.gotoAndStop("stopAnim");
}
```
Further Literature (German)

• Ein schönes deutschsprachiges Buch mit ästhetisch ansprechenden Beispielen:

  Brendan Dawes, Flash ActionScript für Designer: DRAGSLIDEFADE, Markt&Technik 2002
1 Example Technology: Macromedia Flash & ActionScript

1.1 Multimedia authoring tools - Example Macromedia Flash
1.2 Elementary concepts of ActionScript
   Scripting in General + „History“ of ActionScript
   Objects and Types in ActionScript
   Animation with ActionScript
1.3 Interaction in ActionScript
1.4 Media classes in ActionScript

Literature:
  Derek Franklin, Jobe Makar: Flash MX 2004 actionscript, Macromedia Press 2004
Sounds in the Library

- Sounds are imported from a file (in Flash essentially WAV, MP3, AU)
  - Flash command: File -> Import -> Import into Library
- Sounds in the library are the raw material to be used in further design
Sound Processing in Authoring Tool

- Some simple effects can be created graphically
Sound Objects in Time-based Animations

• Sound object:
  – Encapsulates a (pre-produced) sound clip

• A sound object is associated with a specific timeline
  – Sound is played as the time in the timeline progresses
  – There may be many sounds in one presentation
    » Main timeline
    » Individual movie clip instance timelines
  – Sounds are mixed together

• Association of sound instance (from library) to timeline
  – Either graphically (e.g. dragging sound onto frame)
  – or using ActionScript method `attachSound()`
ActionScript Syntax for Sound Objects

• Creating a sound object:
  
  ```actionscript
  var soundObjectName:Sound = new Sound(TargetClip);
  ```
  
  Example:
  
  ```actionscript
  var mySound:Sound = new Sound(myMovieClip_mc);
  ```
  
  Omitting the TargetClip: Definition of global sound

• A Sound object is a handle like the Color object
• Controlling the sound’s volume:
  
  ```actionscript
  mySound.setVolume(50);
  ```
  
• Attaching a library sound:
  
  ```actionscript
  mySound.attachSound("rockMusic");
  ```
Example: A Bouncing Basketball

- Library contains the sound of the bouncing ball
- Movement of ball and coordinated change of shadow realised by tweening
- At the frame where ball touches ground (frame 5), sound is activated (e.g. through the object inspector)
- Sound is played from frame 5 till end of clip
  - Works only well with short sounds
Dragging the Ball over the Court

Let user drag the ball & scale the ball & scale the sound!
Dynamic Adjustment of Volume (and Scale)

```javascript
var bounce:Sound = new Sound(basketball_mc);
var leftBoundary:Number = 60;
var rightBoundary:Number = 490;
var topBoundary:Number = 220;
var bottomBoundary:Number = 360;
var boundaryHeight:Number = bottomBoundary - topBoundary;

this.onMouseMove = function() {
    if (_xmouse > leftBoundary && _ymouse > topBoundary &&
        _xmouse < rightBoundary && _ymouse < bottomBoundary) {
        basketball_mc.startDrag(true);
        var topToBottomPercent = (((_ymouse - topBoundary) /
                                     boundaryHeight) * 100) / 2) + 50;
        bounce.setVolume(topToBottomPercent);
        basketball_mc._xscale = topToBottomPercent;
        basketball_mc._yscale = topToBottomPercent;
    } else {
        stopDrag();
    }
}
```
Stereo Effect: “Panning”

- Panorama position or “balance”:
  - Relative volume of left and right stereo channel
  - Controls the perceived location of a monaural audio signal

- ActionScript (Class `Sound`):
  Method `setPan(relativeValue)`
  - Only left channel: –100
  - Only right channel: +100
  - Centered: 0
Example: Stereo Effect for Basketball

- Sound of bouncing ball draggable with mouse to left and right
  - According adjustment of sound balance

```javascript
var leftBoundary, rightBoundary,
    topBoundary, bottomBoundary...
var boundaryHeight:Number = bottomBoundary - topBoundary;
var boundaryWidth:Number = rightBoundary - leftBoundary;
var quadrantSize:Number = boundaryWidth / 2;
var centerPoint:Number = rightBoundary - quadrantSize;

this.onMouseMove = function() {
    if (_xmouse > leftBoundary && _ymouse > topBoundary &&
       _xmouse < rightBoundary && _ymouse < bottomBoundary) {
        ...
        var panAmount =
            ((_xmouse - centerPoint) / quadrantSize) * 100;
        bounce.setPan(panAmount);
    }
    ...
```
Dynamically Selected Sounds

• Sounds can be attached at runtime dynamically
  – as global sound and to movie clips

• Prerequisite in Flash:
  – Export library sound for ActionScript

• Attaching a sound from library:
  Class `Sound: attachSound("library name");`

• Playing the sound:
  Class `Sound: start(starttime, repetitions);` //time in secs
  Class `Sound: stop();`
Example: Random Basketball Sounds

• On mouse click: Random number between 0 and 2
  – 0: score for “North Carolina”  --> sound “boo”  (Sound0)
  – 1: score for “Indiana”       --> sound “cheer”   (Sound1)
  – 2: no score                --> sound “referee whistle” (Sound2)
    – Sound names chosen such that names can be computed from number
      (variable dynaSounds)

• In case of score:
  – Play “net sound”
  – Show basketball score animation (score_mc)
  – Update score fields of respective team (team_txt)
Code for Random Basketball Sounds

```javascript
var dynaSounds:Sound = new Sound();
var netSound:Sound = new Sound();
...
this.onMouseDown = function() {
    var randomSound = random(3);
    dynaSounds.attachSound("Sound" + randomSound);
    dynaSounds.start(0, 1);
    if(randomSound == 0) {
        northCarolina_txt.text = Number(northCarolina_txt.text) + 2;
        netSound.attachSound("Net");
        netSound.start(0, 1);
        score_mc.gotoAndPlay("Score");
    } else if(randomSound == 1) {
        indiana_txt.text = Number(indiana_txt.text) + 2;
        netSound.attachSound("Net");
        netSound.start(0, 1);
        score_mc.gotoAndPlay("Score");
    }
}
```
Code for Silencing the Dynamic Sounds

• Sound to be switched off when any key is pressed:
  – *Listener* concept used
    (appropriate for events broadcasted to many recipients)

    ```javascript
    this.onKeyDown = function() {
        dynaSounds.stop();
    }
    Key.addListener(this);
    ```
Playing Video from Animations

• Embedding video information into animation
  – Leads to very large files (SWF files in the case of Flash)
• External video clips:
  – Editable separately with specialized software
  – Progressive download: play during loading
  – Video played at its own frame rate, not at the rate of the animation
• Support for external video in Flash (MX 2004):
  – FLV (Flash Video) format
  – Converters from most well-known video formats to FLV exist
  – Special Media Components for easy integration of video
    » MediaDisplay
    » MediaController
    » MediaPlayback (= MediaDisplay + MediaController)
  – Media component can also play back MP3 audio
Flash Components

• **Software component**: „A **software component** is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.“
  
  ECOOP 1996, Workshop on Component-oriented Programming

• **Flash component**: A reusable unit of Flash design and ActionScript programming with clearly specified parameters and methods. A Flash component encapsulates a ready-made solution that can be incorporated into third-party Flash applications.

• Components delivered with Flash (MX 2004, examples):
  
  – User Interface components:
    » Button, CheckBox, ComboBox, DataGrid, DateChooser, Label, ProgressBar, ScrollPane, TextArea, TextInput, Window, ...

  – Data components:
    » DataHolder, DataSet, WebServiceConnector, ...

  – Manager:
    » PopUpManager, Depth Manager, ...

  – Media Components ...
Example Flash Component: Date Chooser

• Layout and basic behaviour pre-defined
• Component inspector allows customization, e.g.
  – Definition of string representation for days, months
  – Disabled days (not chosable)
  – Start day of week
• API allows dynamic ActionScript-based adaptation
  – E.g. setting selected date
• Components generate events
Events Generated by Media Components

• Various events are reported by Media Components to the surrounding application for flexible reaction:
  – Adjustments like change of volume
  – Media events like reaching end of media
  – User-defined events when reaching specific positions (cue events)

• Reaction to media events requires Listener objects, e.g.

```javascript
var myListener:Object = new Object();
myListener.volume = function() {
    // actions to react on volume change
}
myMediaComponent.addEventListener(“volume”, myListener);
```
Example: Video with Event-Triggered Animation
Cue Points

- A cue point marks a specific point in time during media playback.
  - Cue points can be defined independently of the movie (in ActionScript)
  - When reaching a cue point, an event is fired which can be handled by ActionScript.

```actionscript
display.addCuePoint("0", 1);
display.addCuePoint("1", 8);
display.addCuePoint("2", 14);
display.addCuePoint("3", 31);
display.addCuePoint("4", 35);
display.addCuePoint("5", 53);
display.addCuePoint("6", 56);
display.addEventListener("cuePoint", displayListener);
displayListener.cuePoint = function(eventObj:Object){
  var index = Number(eventObj.target.name);
  loadMovie("cue" + index + ".jpg", "cueBox_mc");
  cue_txt.text = cueTextArray[index];
}
```
Example for Cue Points

- Names of cue points chosen in a way such that conversion to number gives an index
- Two arrays of information to be displayed in the two extra windows
  - Still pictures
  - Text information

“Fluffy is crammed into dial-up pipe”
cue2.jpg
cueTextArray[2]
Flash Pattern: Names and Numbers

- **Problem:** Indexing and computing an index requires numbers to identify information instances. Storage in files and symbol identifiers require strings to identify information instances.

- **Solution:**
  - When a string is required to be used as an index: Choose a string representing a number and convert to number when required with function `Number()`
  - When a number is required to be used as a string: Compute an appropriate String by concatenating a base string with the number. Choose file names and identifiers appropriately.

- **Known Uses:**
  - String-to-Number: Cue point names in above example
  - Number-to-String: File names for CueX pictures in above example; Sound names in Basketball example