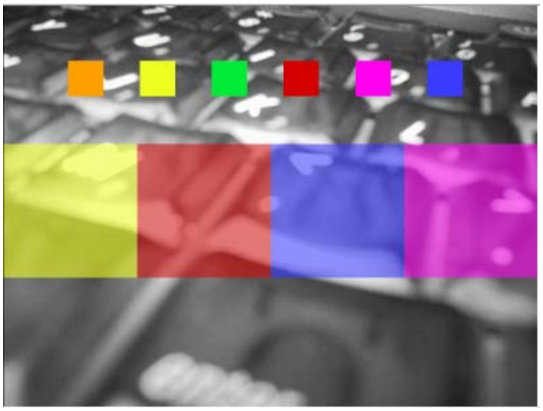
2 Development of multimedia applications

Literature: Brendan Dawes, Flash ActionScript für Designer: DRAGSLIDEFADE, Markt&Technik 2002

Drag and Drop

- Example application "Coloring Book" taken from Brendan Dawes, Flash ActionScript for Designers: Drag-Slide-Fade



Please note:

The example given here is functionally identical with the example of the book by B. Dawes. Also the design is identical.

The program logic has been adapted to ActionScript 2 and linked classes.

Built-in Dragging Support

- Dragging a symbol:
  - Symbol is moved according to mouse movement
  - Can be easily programmed within ActionScript
- Dragging in Flash:
  - Built-in dragging behaviour
  - MovieClip follows mouse between calls of **`startDrag()`** and **`stopDrag()`**
- Parameters to **`startDrag()`**
  - Locking to center (true) or to relative position of mouse within clip
  - Boundary rectangle for limiting possible movement

Hit Test, Path Syntax

- A *hit test* determines whether some position (usually the mouse) is within the bounds of a particular symbol at the time when a user interaction takes place (e.g. click, mouse button down).
  - Built-in function for **`MovieClip`** class in Flash
    - » Parameters: x- and y- position, shape/rectangle
- An object of the scene is always identified by a *path*.
  - Starting at level root scene, proceeding through nested symbols
- Path syntax in Flash:
  - Option 1 (older, ActionScript 1): Slash syntax
    - » Example: **`/block1/...`**
  - Option 2 (current, ActionScript 2): Dot syntax
    - » Example: **`_level0.block1`**
  - **`eval()`** function: Convert from slash to dot syntax
- **`_droptarget:`**
  - Built-in attribute giving the (uppermost) symbol on which drop took place

Dragging a Symbol

```
class Swatch extends MovieClip {

  private var theDrop;
  private var myColor:Color;
  private var startx:Number;
  private var starty:Number;

  public function onLoad() {
     myColor = new Color(this);
     startx = this._x;
     starty = this._y;
  }

  public function onMouseDown() {
     if (this.hitTest(_root._xmouse, _root._ymouse, false)) {
       this.startDrag(true, 0, 0, Stage.width, Stage.height);
           ...
     }
  } ...
```

Dropping a Symbol

```
...
  public function onMouseUp() {
     if (this.hitTest(_root._xmouse, _root._ymouse,
      false)) {
          this.stopDrag();
          theDrop = eval(this._droptarget);
          if (theDrop) // theDrop is not empty
               theDrop.changeColor(myColor.getRGB());
          this._x = startx;
          this._y = starty;
     }
  }
}
```

Why a hit test on mouse *down*: Isn’t it the target object obvious?

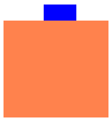
Mouse events are global to the stage, so without a hit test they affect *all objects on stage* which react to the event!

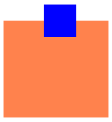
Dynamic Update to Representation of Symbol

```
class Block extends MovieClip {

  private var myColor:Color;

  public function onLoad() {
     myColor = new Color(this);
  }

  public function changeColor(rgb:Number) {
     myColor.setRGB(rgb);
  }

}
```

Stacking Order (z-Order)

- Objects on the two-dimensional screen need to be stacked on top of each other
- Z-Order:
  - Determines which object is “uppermost”
  - Higher numeric values are “upper”
- Flash:
  - Manually placed symbols get *negative* depth value (increased automatically)
  - Symbols placed via support explicit depth specification
- **`MovieClip.getNextHighestDepth():`**
  - Determines depth value to ensure “top level” (in example: 0)
- **`MovieClip.swapDepths`*`(depth):`***
  - Exchanges depth value of target with movie clip at specified depth (if any)


Coloring book with standard z-order

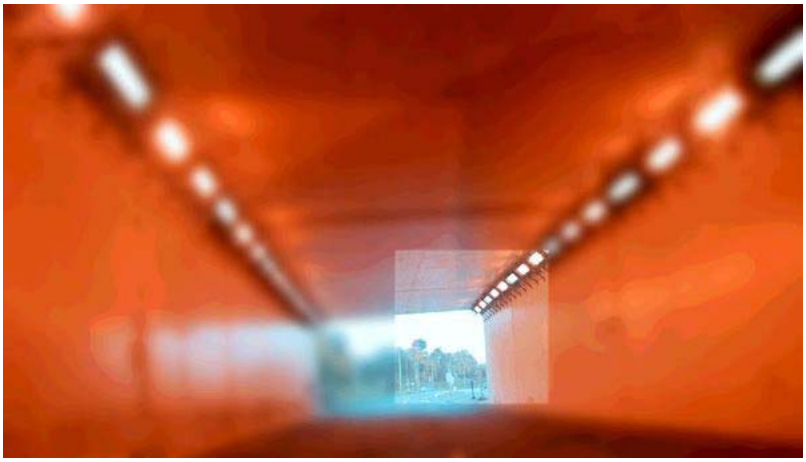

Coloring book with active symbol put “uppermost”

Putting Active Symbol on Top

```
class Swatch extends MovieClip {

  private var theDrop;
  private var myColor:Color;
  private var startx:Number;
  private var starty:Number;

  public function onLoad() {
     myColor = new Color(this);
     startx = this._x;
     starty = this._y;
  }

  public function onMouseDown() {
     if (this.hitTest(_root._xmouse, _root._ymouse, false)) {
       this.startDrag(true, 0, 0, Stage.width, Stage.height);
       this.swapDepths(getNextHighestDepth());
     }
  } ...
```

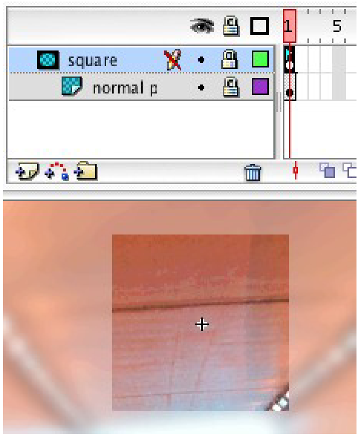
Example: Draggable Mask

- Make different version of a picture visible through a draggable mask
  - Example from Brendan Dawes; completely rewritten in ActionScript 2



Basic Architecture of “DragMask” Example

- Main timeline:
  - Contains blurred version of original picture as background
  - Contains an instance of symbol **`mask_square`** which acts as mask
- Symbol **`mask_square`**:
  - Composed of two elements in separate layers:
    - » Background is original picture (not blurred)
    - » Foreground is a square form
  - Square form (layer) is declared as a *mask*
    - » Achievable through context menu (of layer)
    - » Effects: Background becomes sub-layer, at runtime only intersection of background and mask is visible



Making the Mask Symbol Draggable

- Standard technique, associated class: **`Mask`**
- Problem: Mask uncovers originally picture only as placed statically, does not dynamically move over original picture

```
class Mask extends MovieClip {
  public var drag:Boolean; ...
  public function onMouseDown() {
      if (this.hitTest(_root._xmouse,_root._ymouse,true)) {
            drag = true;
            startDrag (this, false, 0, 0,
                  Stage.width, Stage.height);
      }
  }

  public function onMouseUp() {
      if (this.hitTest(_root._xmouse,_root._ymouse,true)) {
            drag = false;
            stopDrag();
      }
  } ...
```

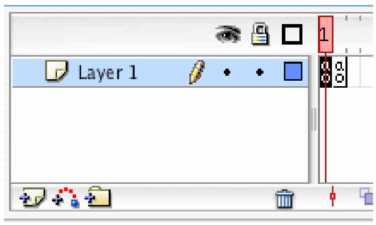
Aligning Pictures During Drag

- Idea for aligning pictures:
  - During drag, shift original picture in the mask_square symbol according to the relative movement of mouse
  - Technically: Event handler for EnterFrame events in class **Mask**

```
public function onLoad() {
    startx = this._x;
    starty = this._y;
}
public function onEnterFrame() {
    var pic_mc = eval(_target+"/picture_mc");
    var xdiff = startx - _x;
    var ydiff = starty - _y;
    if (drag) {
          pic_mc.move(xdiff,ydiff);
    }
}
```

Varieties of Programming Solutions

- Excerpts from the original solution by B. Dawes
- A special “script” MovieClip placed within the “mask drag” symbol



Actions in frame 2:
`gotoAndPlay(1);`

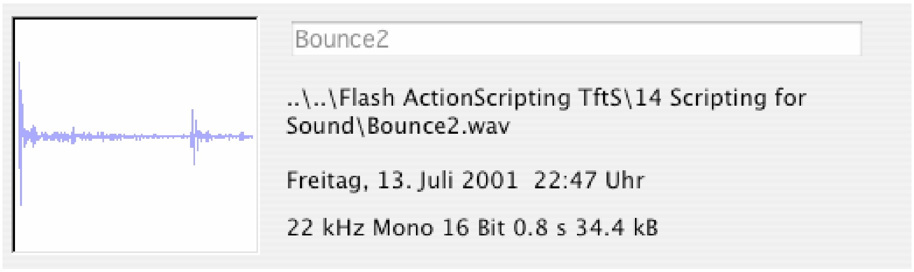
Actions in frame 1:

```
if (_parent.drag == true) {
_parent.picture._x = (_parent.startx -_parent._x);
_parent.picture._y = (_parent.starty -_parent._y);

};
```

2 Development of multimedia applications

Literature: Derek Franklin, Jobe Makar: Flash MX 2004 actionscript, Macromedia Press 2004 (Chapters 17 and 18)

Sounds in the Library



Bounce2

..\..\Flash ActionScripting TftS\14 Scripting for Sound\Bounce2.wav

Freitag, 13. Juli 2001 22:47 Uhr

22 kHz Mono 16 Bit 0.8 s 34.4 kB

- 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 Objects in Time-based Animations

- Sound object:
  - Encapsulates a (pre-produced) sound clip
  - Control of sound characteristics (in Flash)
    - » Length
    - » Volume
    - » Panning (panorama position in stereo sound)
- A sound 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 movieclip instance timelines
- 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:

  ```
  var soundObjectName:Sound = new Sound(TargetClip);
  ```

  Example:

  ```
  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:

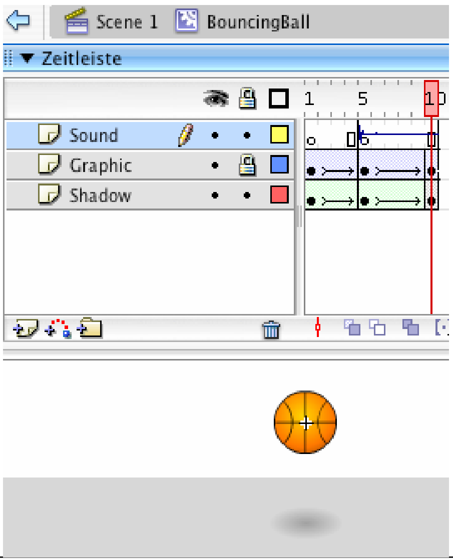
  ```
  mySound.setVolume(50);
  ```

- Attaching a library sound:

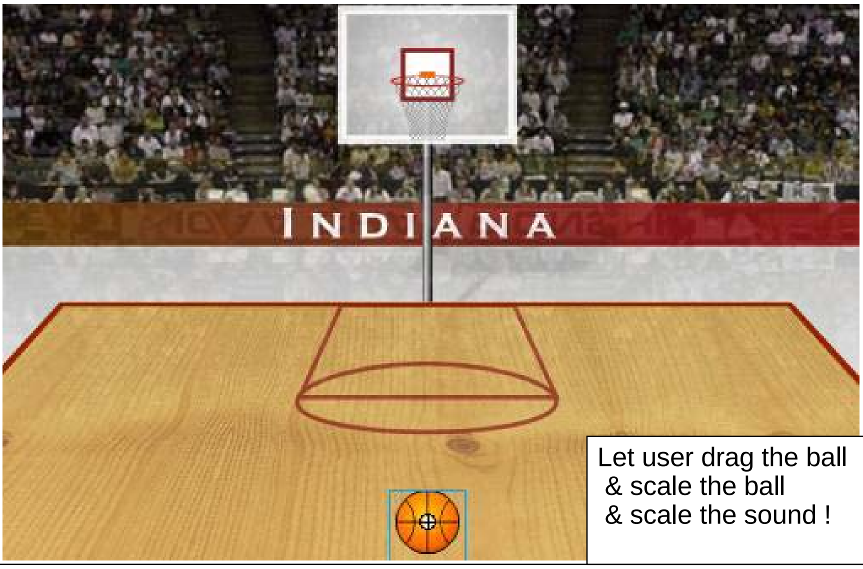
  ```
  mySound.attachSound("rockMusic");
  ```

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

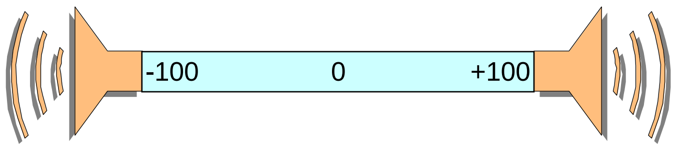


Let user drag the ball
& scale the ball
& scale the sound !

Dynamic Adjustment of Volume (and Scale)

```
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

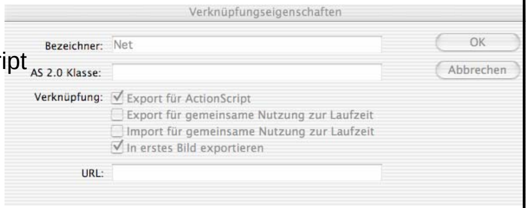
Example: Stereo Effect for Basketball

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

```
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



Verknüpfungseigenschaften

Bezeichner: Net

AS 2.0 Klasse:

Verknüpfung: ☑ Export für ActionScript
☐ Export für gemeinsame Nutzung zur Laufzeit
☐ Import für gemeinsame Nutzung zur Laufzeit
☑ In erstes Bild exportieren

URL:

- 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

```
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)

```
this.onKeyDown = function() {
   dynaSounds.stop();
}
Key.addListener(this);
```