Outline

1. Development Platforms for Multimedia Programming
   1.1. Classification of Development Platforms
   1.2. A Quick Tour of Various Development Platforms
2. Multimedia Programming with Python and Pygame
   2.1. Introduction to Python
   2.2. Pygame: A Multimedia/Game Framework for Python
3. Multimedia Programming with Java FX
4. Multimedia Programming with JavaScript and CreateJS
5. History of Multimedia Programming
6. Programming with Images
7. Programming with Vector Graphics and Animations
8. Programming with Sound
9. Programming with Video
10. Software Engineering Techniques for Multimedia Programs
   10.1. Design Patterns
   10.2. Multimedia Modeling Languages
11. Development Process for Multimedia Projects
4 Multimedia Programming with JavaScript and CreateJS

4.1 HTML5, JavaScript and Multimedia
4.2 CreateJS Libraries
4.3 Slideshow Example with CreateJS
4.4 Generation of JavaScript Code

Literature:
www.createjs.com
www.gskinner.com
HTML5

• WHATWG:
  “Web Hypertext Application Technology Working Group”
  – Founded in 2004 by individuals from Apple, Mozilla Foundation, and Opera
  – Editor of WHATWG specs, Ian Hickson, now at Google
  – Response to slow development of W3C standards for HTML
  – Opposition against XML dominance in W3C HTML standards

• HTML5
  – Collection of various technologies, being unbundled now
  – Realized in leading browsers, in parallel to standardization process
  – Since 2007, W3C HTML working group takes WHATWG HTML5 as starting point for refining HTML

• Multimedia features in HTML5:
  – Audio & video elements
  – “Canvas” 2D graphics element
  – Support for OpenGL 3D graphics ("WebGL")
Immediate and Retained Mode Graphics

- **Immediate mode:**
  - Application directly draws on graphics surface
  - Application has full responsibility, must re-issue adequate drawing commands in case of changes

- **Retained mode:**
  - Graphics library contains model of objects to be rendered
  - Graphics library takes over part of the responsibility for updating display in case of changes

- **Types of retained mode data structures (increasing sophistication):**
  - Display list
  - Hierarchical display list
  - Scene graph
HTML5 Canvas Element

• Origin:
  – Apple MacOS X WebKit component 2004
  – Later adopted in Gecko engine and Opera browser
  – Standardized by WHATWG

• Idea: 2D graphics surface
  – Rendering of graphics through JavaScript drawing API

• Attention: Immediate mode graphics
  – Danger of low performance
  – Additional programming techniques needed (e.g. pre-rendering of canvas)
  – Libraries/frameworks to approach retained mode graphics
4 Multimedia Programming with JavaScript and CreateJS

4.1 HTML5, JavaScript and Multimedia
4.2 CreateJS Libraries
4.3 Slideshow Example with CreateJS
4.4 Generation of JavaScript Code

Literature:
www.createjs.com
CreateJS - Idea and History

• Grant Skinner, March 2012: “Announcing CreateJS”
  – Private project with high ambitions
  – Has quickly won support from major industry players:
    Adobe, Microsoft, AOL, Mozilla foundation

• Basic idea: HTML5 canvas for Flash-educated developers
  – Creates abstraction layer on top of (pixel-based) HTML5 canvas
  – EaselJS: Hierarchical display list, mouse interaction mode

• Uses Metaphors very close to Flash terms
  – Stage, Timeline, Tween, Movieclip, ….
Libraries in CreateJS

- **EaselJS**
  - Assistance in working with HTML5 canvas
  - Hierarchical display list, interaction model, helper classes
- **TweenJS**
  - “Tweening" library for JavaScript (= interpolation of object properties)
- **SoundJS**
  - Cross-Browser audio support for HTML5
- **PreloadJS**
  - Assistance in pre-loading assets, like graphics, sound, code
  - Uses queues and supports multiple connections
- **Movieclip**
  - Analogy to Flash “swf” movieclips
  - Not meant for use in manually written programs
  - Currently *not* part of standard EaselJS
CreateJS Embedding in HTML

- Make library accessible, e.g. by loading from network:
  <script "http://code.createjs.com/
  createjs-2013.12.12.min.js"></script>

- Provide “init” function to be called after loading of HTML is complete
- Provide HTML5 canvas element and associate with “init” function

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <script src="http://code.createjs.com/..."></script>
  <script>
    function init() { ... CreateJS API usage ... }
  </script>
</head>
<body onload="init();">
  <canvas id="myCanvas" width=... height=...></canvas>
</body>
</html>
```
CreateJS Stage and HTML5 Canvas

- **Stage (Bühne):**
  - Frequently used metaphor for the space for animated program behavior

- **createjs.Stage class:**
  - To be instantiated in script code
  - HTML id of stage element as parameter for constructor
    (or selection through classical means, e.g. DOM)

- For naming the window use HTML window title

- Stage contents are not organized as full scene graph here!

- Stage contents have to be added and made visible explicitly
  (hierarchical display list):
  - `stage.addChild()`
  - `stage.update()` (or “tick” method calling `update()` regularly)
Example: Pre-Loading Assets (1)

```javascript
var manifest = [
    "pics/tiger.jpg",
    "pics/elephant.jpg",
    "pics/jbeans.jpg",
    "pics/peppers.jpg",
    "pics/butterfly.jpg"
];

var loader = new createjs.LoadQueue(false);
// do not use XMLHttpRequest
loader.addEventListener("fileload", handleFileLoad);
loader.addEventListener("complete", handleComplete);
loader.loadManifest(manifest);
```
Example: Pre-Loading Assets (2)

```javascript
var images = new Array();
var imagecount = 0;

function handleFileLoad(event) {
    if (event.item.type == "image") {
        images[imagecount] = event.result;
        imagecount++;
    }
}

function handleComplete() {
    proceed after all assets have been loaded
}
```
4 Multimedia Programming with JavaScript and CreateJS

4.1 HTML5, JavaScript and Multimedia
4.2 CreateJS Libraries
4.3 Slideshow Example with CreateJS
4.4 Generation of JavaScript Code

Literature:
www.createjs.com
CreateJS JavaScript Kernel for Slideshow

```javascript
var timeline = new createjs.Timeline({}, null,{loop: true});
for (var i=0; i<imagecount; i++) {
    var image = new createjs.Bitmap(images[i])
        .set({x: 50, y: 50, visible: false});
    stage.addChild(image);
    timeline.addTween(createjs.Tween
        .get(image)
        .wait(4000*i)
        .wait(4000)
        .to({visible: true})
        .wait(4000)
        .to({visible: false}));
}
createjs.Ticker.setFPS(30);
createjs.Ticker.addEventListener("tick", stage);
timeline.gotoAndPlay();
```
Method Chaining in CreateJS

- Many methods return the object on which they were invoked as result
- Chains of method calls: simple to write and easy to read
- Example:
  ```javascript
  new createjs.Bitmap(images[i])
    .set({x: 50, y: 50, visible: false});
  ```
- Technique often used in modern JavaScript frameworks
Tweens in CreateJS

• Tween = Interpolation
• Modifies properties of graphical object over time
• Methods used here:

  tween.get(object)        Obtain target object to modify
  tween.to(prop-values)   Modification of properties,
                          specified as object containing property/value pairs;
                          may specify additional time duration for linear interpolation
  tween.wait(msecs)       Pause for a specified time

• Automatic updates work only if an event listener for the “tick” event of the Ticker class is registered.

• Stage as parameter:
  Citing “createjs.com” on Stage:

  “Each time its Stage/tick method is called, it will render its display list to its target canvas.”
Timeline in CreateJS

- Timeline object:
  - Provides temporal structure to execute several tweens
- Constructor parameters:
  - Array of tweens to add (alternatively add with `addTween()`)
  - Object defining labels to be addressed in “goto” method calls
  - Initial timeline properties as object, for instance `{loop: true}`
- Looping timeline causes embedded tweens to be executed repeatedly
- Executing timeline at a defined point in time (time in msecs or label):
  - `gotoAndPlay()`
  - `gotoAndStop()`
Timeline for Slideshow Example

Time (secs)
0  4  8  12

- **image[0]**. visible
- wait() **image[1]**. visible
- wait() wait() **image[2]**. visible
- wait() wait() wait() **image[3]**. visible
4 Multimedia Programming with JavaScript and CreateJS

4.1 HTML5, JavaScript and Multimedia
4.2 CreateJS Libraries
4.3 Slideshow Example with CreateJS
4.4 Generation of JavaScript Code

Literature:
www.createjs.com
Flash Toolkit for CreateJS

- Adobe-supported plug-in for “Flash Pro” authoring system
- Provides additional exporting function
  - Standard export: As “swf” (shockwave flash) file for Flash player
  - Added export: JavaScript/CreateJS code, relatively well readable
Publishing CreateJS Code from Flash
Created JavaScript Code (Excerpt)

(function (lib, img, cjs) {

var p; // shortcut to reference prototypes

// stage content:
(lib.slideshow0 = function(mode,startPosition,loop) {
    this.initialize(mode,startPosition,loop,

    // Layer 1
    this.instance = new lib.tiger();
    this.instance.setTransform(50,50);

    ...

    this.timeline.addTween(cjs.Tween.get({})
        .to({state:[{t:this.shape},{t:this.instance}]})
        .to({state:[{t:this.instance_1}],96)
        .to({state:[{t:this.instance_2}],96)
        ...
        .wait(96)); ...

...