

8 Programming with Sound

8.1 Playing Sound from File

8.2 Controlling Sound Objects

8.3 Sound Effects and Events



Literature:

W. McGugan, Beginning Game Development with Python and Pygame,
Apress 2007

<http://docs.oracle.com/javafx/2/media/overview.htm>
createjs.com/docs

Example: Background Music

- How to play back music while the program runs?
 - How to access the sound subsystem?
 - How to load a sound file?
 - » Supported file formats?
 - How to control playback?
- Sound playback always takes place *in parallel* to rest of program
 - Separate *thread* in program
 - Time container in *parallel* composition



Example: Background Music (1)

```
import pygame
from pygame.locals import *
from sys import exit

background = pygame.Color(255, 228, 95, 0)
sc_w = 356
sc_h = 356
music_file = "nancygroff_turntome.ogg"

pygame.init()
pygame.mixer.init(44100, -16, 2, 1024*4)

# Create program display area
screen = pygame.display.set_mode((sc_w, sc_h), 0, 32)
pygame.display.set_caption("Simple Slide Show")

# Set background color
screen.fill(background)

...contd.
```

Sound Initialization

- Sound subsystem:
 - Gateway between program and operating system
 - » Abstraction layer (e.g. Pygame)
 - » Possiböy plus system-specific plug-ins (e.g. CreateJS)
 - *Mixer* (name derived from audio mixer hardware)
- Audio Format:
 - Sample rate / playback rate: samples/second
 - Sample size: bits
 - Stereo channels (mono=1, stereo=2)
 - Buffer size: number of samples buffered for playback
 - » Relatively low-level interface in Pygame
- Pygame mixer initialization defines playback properties:
`pygame.mixer.init(44100, -16, 2, 1024*4)`
44100 samples/s, 16 bit samples (signed), stereo, 4k buffer

Example: Background Music (2)

... (cont.)

```
# Load and play background music
pygame.mixer.music.load(music_file)
pygame.mixer.music.play()
```

```
# Load slide and show it on the screen
slide = pygame.image.load('pics/tiger.jpg').convert()
screen.blit(slide, (50, 50))
pygame.display.update()
pygame.time.wait(4000)
```

```
...
# Load slide and show it on the screen
slide = pygame.image.load('pics/
    butterfly.jpg').convert()
```

```
...
pygame.time.wait(4000)
pygame.mixer.music.fadeout(3000)
```

mixer.music:

Special interface for long-running background sound

Program Control of Sound Playback

- Main functions (of `mixer` object):
 - `load()` : Loads a new sound from given filename
 - `play()` : Starts playback
 - `pause()` : Stops playback, ready for continuation
 - `unpause()` : Continues paused playback
 - `stop()` : Stops playback, no continuation possible
- Other functions realizable by combination
 - E.g. "restart" by combination of "stop" and "play"
- Example: Jukebox
 - Scans directory for sound files
 - Builds list of sound files
 - Interactive interface for skipping through files and playback

Example: Jukebox (1)

```

def get_music(path):
    raw_filenames = os.listdir(path)
    music_files = []
    for filename in raw_filenames:
        music_files.append(
            os.path.join(MUSIC_PATH, filename))
    return sorted(music_files)

...
music_filenames = get_music(MUSIC_PATH)
current_track = 0
max_tracks = len(music_filenames)
pygame.mixer.music.load(
    music_filenames[current_track] )
clock = pygame.time.Clock()
playing = False
paused = False

```

Example: Jukebox (2)

```

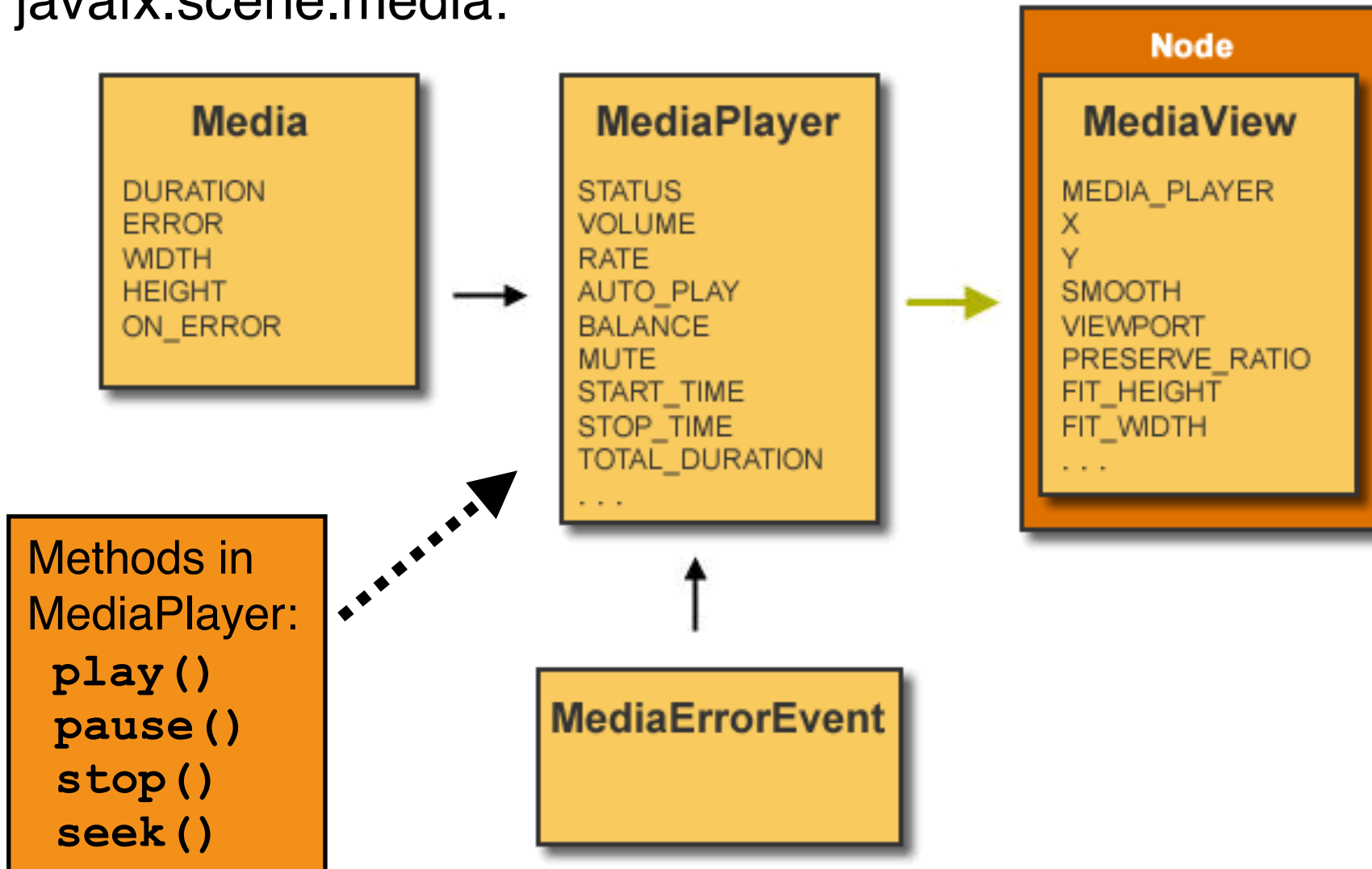
...
if button_pressed == "next":
    current_track =
        (current_track + 1) % max_tracks
    pygame.mixer.music.load(
        music_filenames[current_track] )
    if playing:
        pygame.mixer.music.play()
elif button_pressed == "prev":
    if pygame.mixer.music.get_pos() > 3000:
        pygame.mixer.music.stop()
        pygame.mixer.music.play()
    else:
        current_track = (current_track - 1) % max_tracks
        pygame.mixer.music.load(
            music_filenames[current_track] )
        if playing:
            pygame.mixer.music.play()
elif button_pressed == "pause":
    if paused:
        pygame.mixer.music.unpause()
        paused = False
    else:
        pygame.mixer.music.pause()

```



JavaFX Media Classes

javafx.scene.media:



Source: docs.oracle.com

Simple Audio Player in JavaFX

```
public class JavaFXAudioPlayer1 extends Application {  
  
    @Override  
    public void start(Stage primaryStage) {  
        Media media = new Media(  
            getClass().getResource("sounds/XXX.mp3").toString());  
        MediaPlayer mediaPlayer = new MediaPlayer(media);  
        mediaPlayer.play();  
  
        primaryStage.setTitle("Audio Player 1");  
        primaryStage.setWidth(200);  
        primaryStage.setHeight(200);  
        primaryStage.show();  
    }  
  
    ...  
}
```

JavaFXAudioPlayer1

Error Handling in Media Playback

- Synchronous errors:
 - Typically exceptions thrown when creating media player or accessing media
- Asynchronous errors:
 - Exception object stored in `Media.error` and `MediaPlayer.error` properties
 - Error callback methods can be registered through `Media.onError` and `MediaPlayer.onError`
- Example:

```
mediaPlayer.setOnError(new Runnable() {
    public void run() {
        // Handle asynchronous error, e.g. using:
        final String errorMessage =
            media.getError().getMessage();
        // Further error handling...
    }
});
```

Code partially from whant-when-how.com



Simple Audio Player in CreateJS

JavaScript:

```
function load() {
    document.getElementById("display").innerText = "Loading...";
    var loader = new createjs.LoadQueue();
    loader.installPlugin(createjs.Sound);
    loader.addEventListener("complete", handleComplete);
    loader.loadFile({id:"testSound", src:"sounds/XXX.mp3"});
}

function handleComplete() {
    document.getElementById("display").innerHTML = "Load complete";
    createjs.Sound.play("testSound");
}
```

HTML:

```
<body>
  <input id="loadBtn" type="button" value="Load Sound"
    onclick="load()" />
  <label id="display">To start click "Load Sound".</label>
</body>
```

Across-Platform Concepts

- Background sound vs. short audio clips
 - Background sound not loaded into working memory completely (e.g. `sound.music` in Pygame, `Media` in JavaFX)
- Loading sound from file
 - Pre-loading process
 - Format and sub-system dependencies
- Determining runtime characteristics for sound
 - E.g. default volume
- Sound rendering
- Runtime control for sound
 - Playback control (play, pause etc.) through *handler* object (mixer in Pygame, `MediaPlayer` in JavaFX, `Sound` instance in CreateJS)
 - Dynamic rendering control (e.g. volume)
 - » only for long-running sounds

8 Programming with Sound

8.1 Playing Sound from File

8.2 Controlling Sound with Objects



8.3 Sound Effects and Events

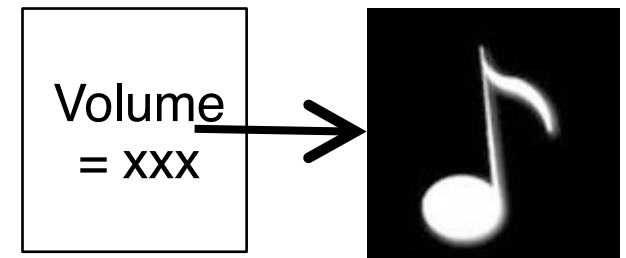
Literature:

W. McGugan, Beginning Game Development with Python and Pygame,
Apress 2007

<http://docs.oracle.com/javafx/2/media/overview.htm>

Rendering Control Objects

- Rendering process can be modified by parameter settings:
 - For images: e.g. compositing rules, clipping
 - For sound: e.g. volume, placement of mono source in stereo panorama
- Specific object representing rendering parameters: *rendering control*
 - Refers to media object (is a *handle* on the object)
 - Locally stores rendering parameters
 - Relative volume (to master volume) e.g. as float value between 0.0 and 1.0
- Examples:
 - *Channel* objects in Pygame
 - *SoundChannel* objects in ActionScript 3.0



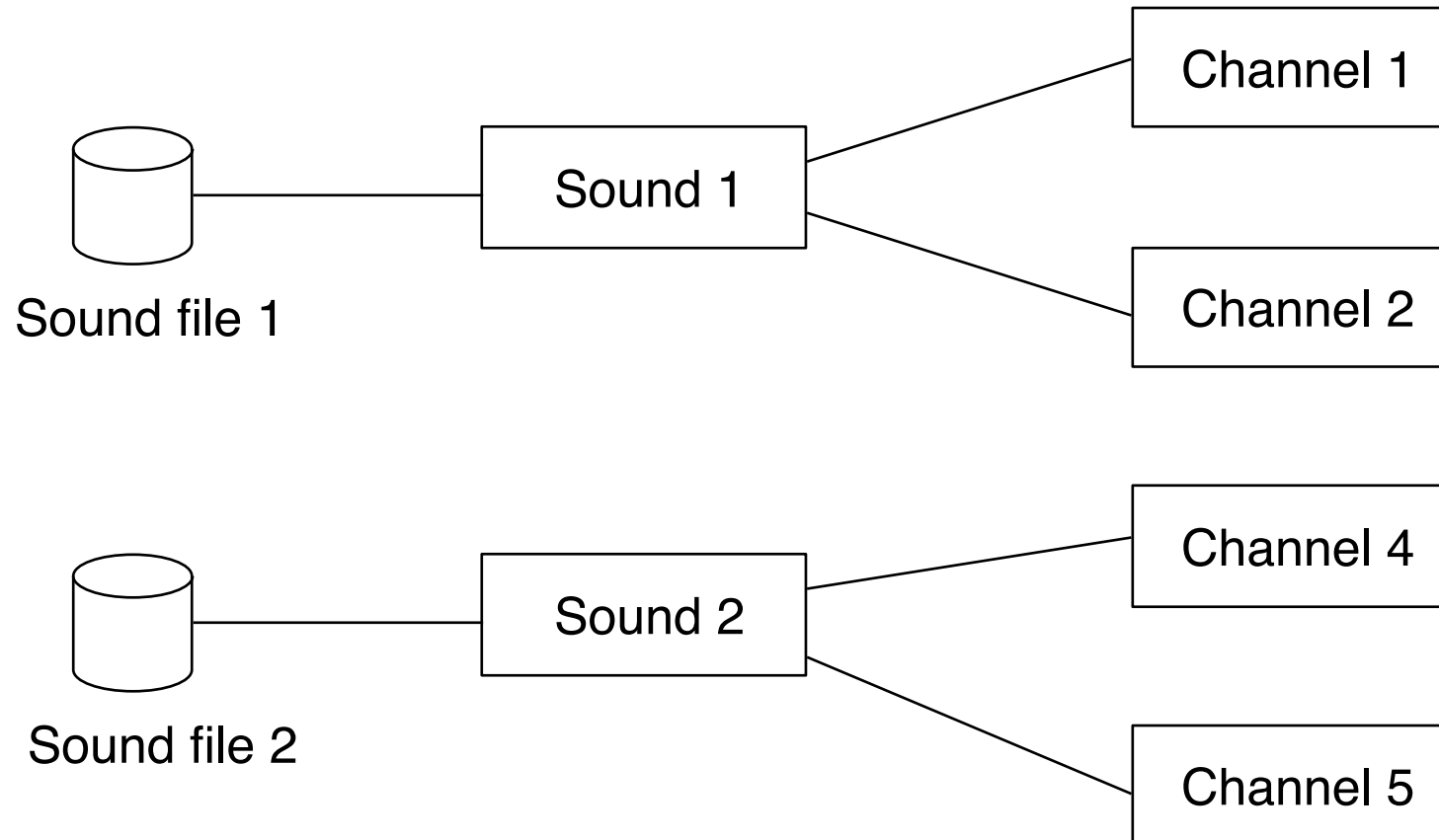
Panorama and Balance

- Balance:
 - Adjustment of relative level of stereo channels (attenuation)
 - Mainly for adapting to suboptimal speaker position
- Panorama (Pan):
 - Placing a sound source in the stereo panorama
 - Distribution of the signal over left/right channel
 - Mainly applied for mixing a monaural signal into a stereo signal
 - Can also be applied to stereo signals
- Both names often used interchangeably
- Typical parameter coding:
 - Either float value between 0.0 and 1.0
 - Or float value between -1.0 and $+1.0$ (where 0.0 means center)

Channels in Pygame

- Channel:
 - One out of several sources that are mixed together by the sound card
 - `play()` method returns a Channel object (or `None` if all channels are busy)
- Limited number of channels
 - Number of channels can be set (`pygame.mixer.set_num_channels`)
 - Channels are assigned to playing tasks automatically until maximum number is reached (all channels busy)
 - Channels for important audio information can be reserved (`pygame.mixer.set_reserved`)
- Typical methods for Channel objects:
 - Individual playback control (pause, play)
 - Volume control, for left and right speakers
 - Event handling for end of playing time
 - » Fire event at end of playing time
 - » Play queued sound object

Multiple Sounds and Channels



Asynchronous Playback

- Quiz question:
What do we hear when this code is executed?

```
sound1 = pygame.mixer.Sound(soundfile)
channel1 = sound1.play()
channel2 = sound1.play()
channel3 = sound1.play()
```

- The play() method triggers the start of playback only...

Setting Volume/Pan with Mouse (1)

```
SCREEN_SIZE = (480, 480)
cursor_image_file = "arrowcursor.png"
loop_file = "GuitarLoop.wav"

import pygame
from pygame.locals import *

pygame.mixer.init(44100, -16, 2, 1024*4)
pygame.init

screen = pygame.display.set_mode(SCREEN_SIZE, 0)
pygame.display.set_caption("Sound Control")
mouse_cursor =
    pygame.image.load(cursor_image_file).convert_alpha()

loop_sound = pygame.mixer.Sound(loop_file)
channel = loop_sound.play(-1)

...contd.
```

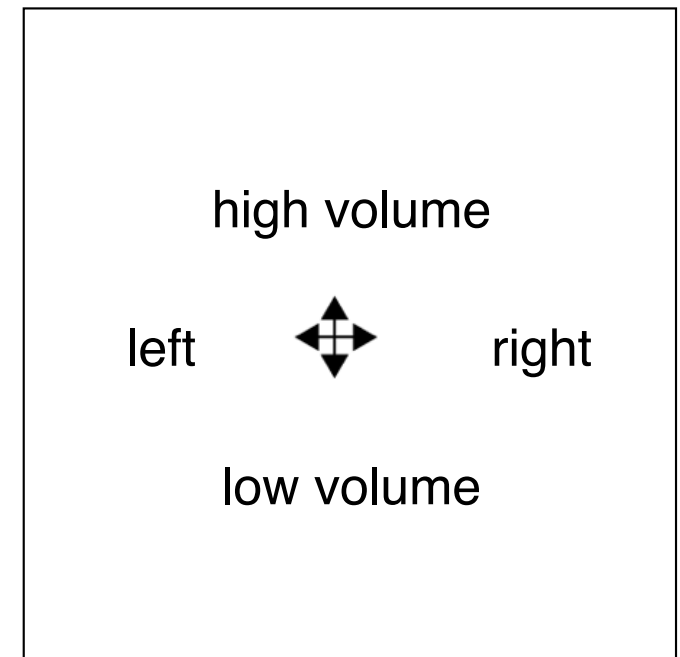
Setting Volume/Pan with Mouse (2)

```

...
xPos = max(0, float(mouseX)/SCREEN_SIZE[0])
yPos = max(0, float(mouseY)/SCREEN_SIZE[1])
vol = 1.0-yPos
pan = xPos
channel.set_volume(vol*(1.0-pan), vol*pan)

```

...contd.



Setting Volume/Pan with Mouse (3)

...

```

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
        screen.fill((255, 255, 255))
        mouseX, mouseY = pygame.mouse.get_pos()
        cursorX = mouseX - mouse_cursor.get_width()/2
        cursorY = mouseY - mouse_cursor.get_height()/2
        screen.blit(mouse_cursor, (cursorX, cursorY))
        pygame.display.update()

        if channel is not None:
            compute xpos, ypos and adjust volume/pan

```

Setting Volume/Pan with Mouse

```
Group root = new Group();
Scene scene = new Scene(root, SCRWIDTH, SCRHEIGHT);
Image cursorImg = new Image(
    getClass().getResource("arrowcursor.png").toString());
scene.setCursor(new ImageCursor(
    cursorImg, cursorImg()/2, image.getHeight()/2));

scene.setOnMouseMoved(new EventHandler<MouseEvent>() {
    public void handle(MouseEvent me) {
        double xPos = Math.max(0, me.getX()/SCRWIDTH);
        double yPos = Math.max(0, me.getY()/SCRHEIGHT);
        mediaPlayer.setVolume(1.0-yPos);
        mediaPlayer.setBalance(-1.0+2*xPos);
    }
});
```

Conceptual schema:

- Handle mousemove event
- Set rendering properties through handler object



Setting Volume/Pan with Mouse

```
function handleComplete() {  
    var soundInstance =  
        createjs.Sound.play("loopSound", {loop: -1});  
    var st_w = stage.canvas.width;  
    var st_h = stage.canvas.height;  
    stage.addEventListener("stagemousemove", function(evt) {  
        xPos = Math.max(0, evt.stageX/st_w);  
        yPos = Math.max(0, evt.stageY/st_h);  
        soundInstance.setVolume(1.0-yPos);  
        soundInstance.setPan(-1.0+2*xPos);  
    });  
}
```

Conceptual schema:

- Handle mousemove event
- Set rendering properties through handler object

8 Programming with Sound

8.1 Playing Sound from File

8.2 Controlling Sound with Objects

8.3 Sound Effects and Events



Literature:

W. McGugan, Beginning Game Development with Python and Pygame,
Apress 2007

<http://carlfx.wordpress.com/2012/08/26/javafx-2-gametutorial-part-5/>

Event-Driven Sound

- In interactive programs and animations:
 - Sound as part of presentation
 - Needs to be synchronized with user interactions and animation progress
 - Several sounds may play synchronously
- Examples:
 - Sound triggered by collision detection in animation (bounce, crash)
 - Sound triggered by user input (keyboard beep)
 - Sound synchronized with animation (pitch or volume analog to movement)
- Sound triggering events may be explicit program events or just implicit (position in program code)

Events Created by Sound System

- Specific conditions of the sound system may be made available as events to the programmer
 - Example: “End event” for playback in Pygame
`Channel.set_endevent(id)`
requests an event to be triggered when sound has finished playing.
Appropriate identifier for event is given as parameter
- Examples for other events possibly created by sound system (*not* Pygame-specific):
 - External change of volume or other parameters
 - Playback reaching a certain intermediate position (*cue point*)
 - Exceptional situations (e.g. too few channels)

Example: Bouncing Balls (1)

From Pygame book (excerpt):

```
class Ball(object):

    def __init__(self, position, speed, image, bounce_sound):

        self.position = Vector2(position)
        self.speed = Vector2(speed)
        self.image = image
        self.bounce_sound = bounce_sound
        self.age = 0.0

    def update(self, time_passed):

        w, h = self.image.get_size()

        screen_width, screen_height = SCREEN_SIZE

        x, y = self.position
        x -= w/2
        y -= h/2
        ...
```

Example: Bouncing Balls (2)

(update contd.)

```
# Has the ball bounced?
bounce = False
```

```
# Has the ball hit the bottom of the screen?
if y + h >= screen_height:
    self.speed.y = -self.speed.y * BOUNCINESS
    self.position.y = screen_height - h / 2.0 - 1.0
    bounce = True
```

```
# Has the ball hit the left of the screen?
if x <= 0:
    self.speed.x = -self.speed.x * BOUNCINESS
    self.position.x = w / 2.0 + 1
    bounce = True
```

```
# Has the ball hit the right of the screen
elif x + w >= screen_width:
    self.speed.x = -self.speed.x * BOUNCINESS
    self.position.x = screen_width - w / 2.0 - 1
    bounce = True
```

...

Example: Bouncing Balls (3)

(update contd.)

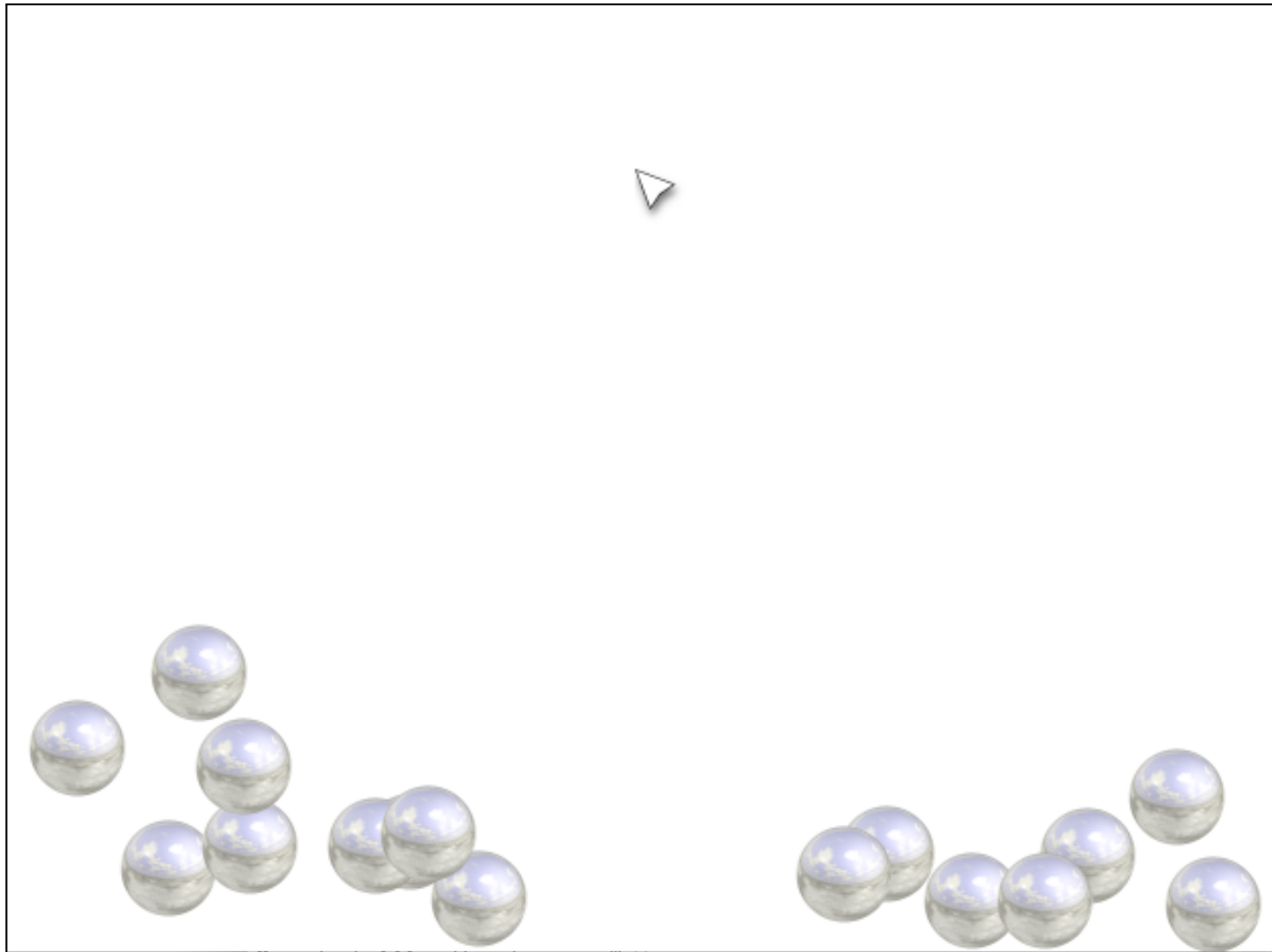
```
# Do time based movement
self.position += self.speed * time_passed
# Add gravity
self.speed.y += time_passed * GRAVITY
```

```
if bounce:
    self.play_bounce_sound()
    self.age += time_passed
```

```
def play_bounce_sound(self):
    channel = self.bounce_sound.play()
    if channel is not None:
        left, right =
            stereo_pan(self.position.x, SCREEN_SIZE[0])
        channel.set_volume(left, right)
```

```
def stereo_pan(x_coord, screen_width):
    right_volume = float(x_coord)/screen_width
    left_volume = 1.0 - right_volume
    return (left_volume, right_volume)
```

Multiple Bouncing Balls



Ball Collision and Bounce with Sound

```

AudioClip bounce = new AudioClip(
    getClass().getResource("sounds/ballbounce.wav")
        .toString());
AudioClip collision = new AudioClip(
    getClass().getResource("sounds/billiardballs.wav")
        .toString());
SpriteManager sprites = new SpriteManager(
    scW, scH, bounce, collision);
    
```

Sprite.update():

```

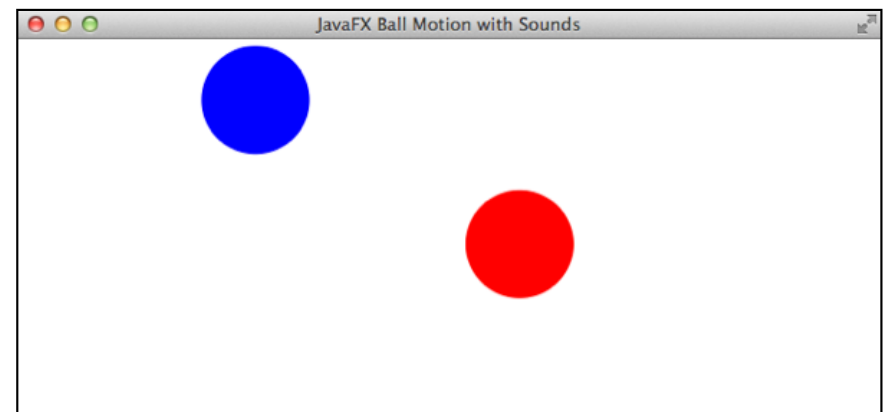
if (bounces) {
    bouncesound.play();
}
    
```

BallSprite.handleCollision():

```

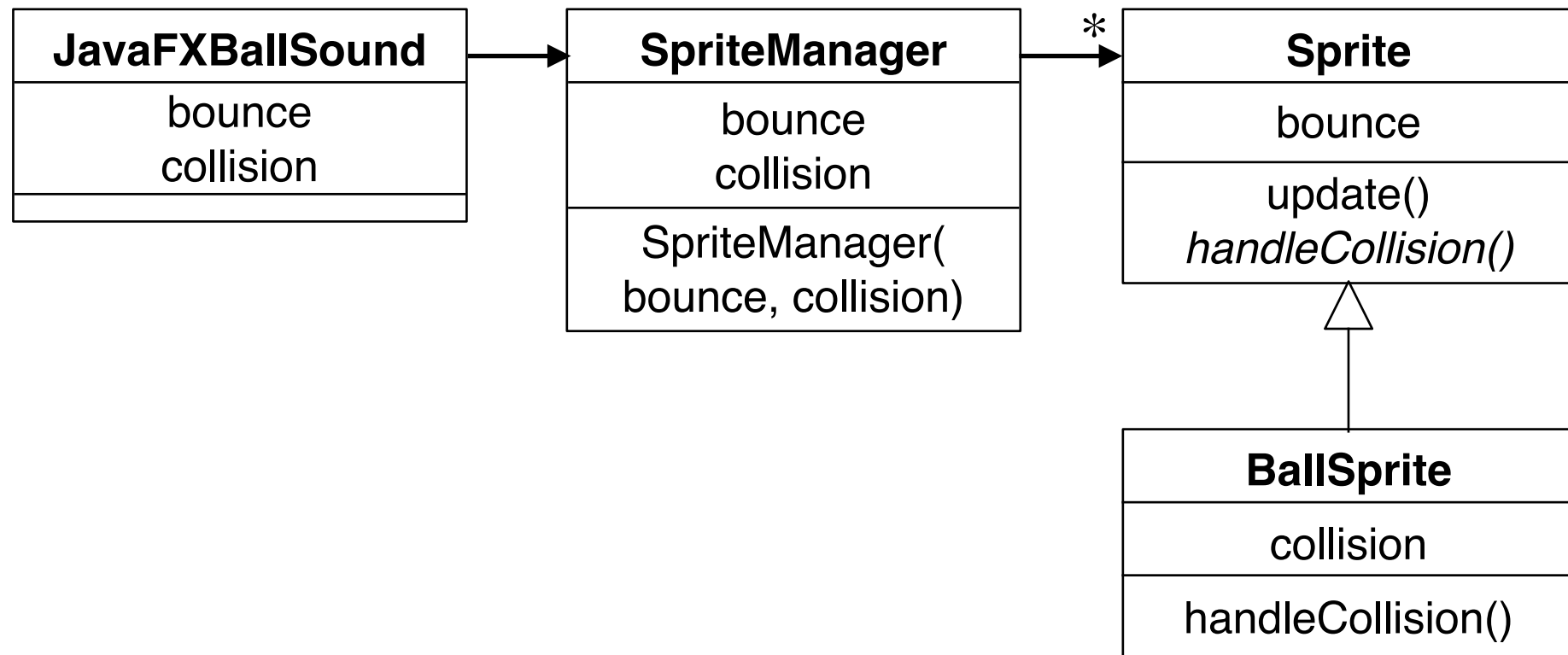
if (dist < r1+r2) {
    ...
    collsound.play();
}
    
```

Sounds from freesound.org



Media Management Software Architecture

- All media files to be accessed from a central class
- Re-use of created objects
 - Possibly re-use of threads (thread pool)



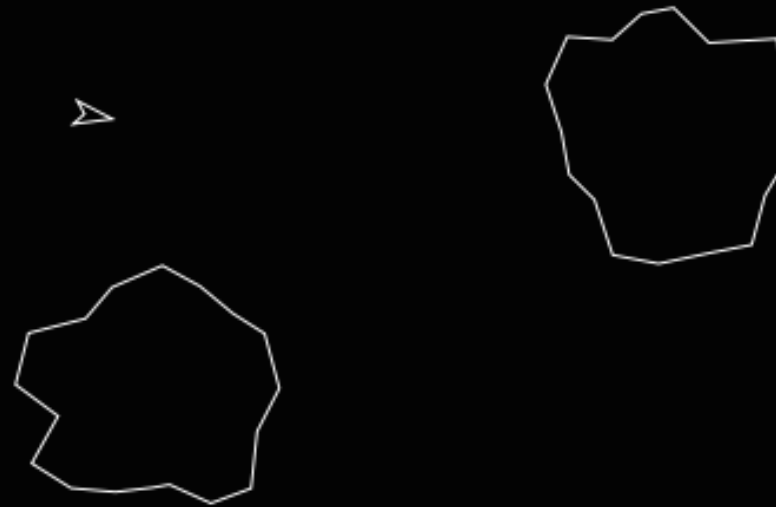
Dynamic Adaptation of Sound Effects

- Dependent of actual event:
 - Choosing between alternative sounds
 - Adaptation of sound properties
 - » Volume
 - » Panorama/Balance
- Audio clips:
 - Often property change during playback not supported
 - Adaptation performed at creation time or at playback start time



Example: Space Rocks Game

0



<http://www.cs.vu.nl/~eliens/media/mobile-graphic-easel-examples-Game.htm>



Code Snippets from Space Rocks

```
createjs.Sound.play("music",  
    createjs.Sound.INTERRUPT_NONE, 0, 0, -1, 0.4);
```

How to interrupt playing instances
if maximum number of instances
is already playing.

↑
↑
↑
↑
delay offset loop volume

```
createjs.Sound.play("break",  
    createjs.Sound.INTERRUPT_LATE, 0, 0.8);
```

```
createjs.Sound.play("death",  
    createjs.Sound.INTERRUPT_ANY);
```

Preloading of sounds from manifest
defining string ids for sounds