Multimedia-Programmierung
Übung 8

Ludwig-Maximilians-Universität München
Sommersemester 2015
Today

• Sprite animations in pygame
• Advanced collision detection
• Sound
Keyframe Animations

• Keyframes are defined
• Intermediate steps are interpolated
• Basic interpolators/tweens/... built into many programming environments (e.g. CreateJS, JavaFX)
• Examples: motion, color, shape
Keyframe Animations

Keyframe Animations in Pygame

• Pygame has no built-in interpolators
• Logic has to be added by the programmer
• Question: How can we calculate the intermediate points?

Function?
```python
import pygame
from pygame.locals import *
from sys import exit

player_image = 'head.jpg'
pygame.init()

screen = pygame.display.set_mode((640, 280), 0, 32)
pygame.display.set_caption("Animate X!")
mouse_cursor = pygame.image.load(player_image).convert_alpha()

x = 0 - mouse_cursor.get_width()
y = 10

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
        screen.fill((255,255,255))
        if x > screen.get_width():
            x = 0 - mouse_cursor.get_width()
        screen.blit(mouse_cursor, (x, y))
x+=10
    pygame.display.update()
```

Horizontal Animation (old slides)
Sprite Animations

• Animations consist of frames that are displayed one after another

Frame 1  ...  Frame 7

• Either imported as single graphics or with sprite sheets

Contains small tutorial on creating animations with sprite sheets in Pygame
Sprite Sheets & Spriting

• Sprite sheets contain all possible movements for a character
• Each Sprite should have the same size for easy slicing in software

http://www.themysticalforestzone.com/Sprite_section.htm
Creating Sprite Sheets

• Editing with Photoshop, Gimp, Pixen etc.
• Pay attention to positioning of character and background color (should not appear in character)
In-class exercise

Be creative and create your own sprite sheet!

Pixen (Mac only)
Slicing Sprite Sheets

```python
def load_sliced_sprites(self, w, h, filename):
    images = []
    master_image = pygame.image.load(os.path.join('ressources', filename)).convert_alpha()
    master_image.set_colorkey((255,0,255))
    master_width, master_height = master_image.get_size()
    for i in xrange(int(master_width/w)):
        images.append(master_image.subsurface((i*w,0,w,h)))
    return images
```

set transparent color, background color of sprite sheet

create subsurfaces

More specialized slicing function may be needed due to incompatible sprite sheet (e.g. with borders)
import os, pygame
from pygame.locals import *

def load_sliced_sprites(self, w, h, filename):
    ....
class BombWithAnimation(pygame.sprite.Sprite):
    def __init__(self, color, initial_position, fps):
        pygame.sprite.Sprite.__init__(self)
        self.act_frame = 0
        # create the images for the animation
        self.frames = load_sliced_sprites(20,20, „explosed-sprite.png“)
        self.image = self.frames[0]
        self.rect = self.image.get_rect()
        self.rect.topleft = initial_position
        self.fps = fps
        self.change_time = 1.0/self.fps
        self.time = 0

def update(self, time_passed):
    self.time += time_passed
    if self.time >= self.change_time:
        self.act_frame = (self.act_frame + 1) % len(self.frames)
        self.image = self.frames[self.act_frame]
        self.time = 0

Remember current frame

Based on the frames per second (fps) calculate the time needed for animation changes

Frame changed? Change frame
First Sprite Animation 2

...  
pygame.init()

screen = pygame.display.set_mode((640, 480), 0, 32)
bomb1 = BombWithAnimation((0,0),4)
clock = pygame.time.Clock()

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
    screen.fill((100, 200, 0))
    time_passed = clock.tick() / 1000.0
    bomb1.update(time_passed)
    screen.blit(bomb1.image,bomb1.rect)
    pygame.display.update()
pygame.init()

screen = pygame.display.set_mode((640, 480), 0, 32)
bomb1 = BombWithAnimation((0,0),4)
bomb2 = BombWithAnimation((40,40),2)
clock = pygame.time.Clock()

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
    screen.fill((100, 200, 0))
    time_passed = clock.tick() / 1000.0
    bomb1.update(time_passed)
    screen.blit(bomb1.image,bomb1.rect)
    bomb2.update(time_passed)
    screen.blit(bomb2.image,bomb2.rect)
    pygame.display.update()
In-class exercise

Now animate your own sprite sheet. Try out different framerates!
Sprite Animation

```javascript
var data = {
    images: ["explosed-sprite.png"],
    frames: {width:20, height:20},
    animations: {explode:[0,6]}
};
var spriteSheet;
var animation;

function init() {
    stage = new createjs.Stage("canvas");
spriteSheet = new createjs.SpriteSheet(data);
animation = new createjs.Sprite(spriteSheet,"explode");

    stage.addChild(animation);
    stage.update();

    createjs.Ticker.setFPS(5);
    createjs.Ticker.addEventListener("tick", stage);
}
```
Collision Detection

Rect

• Rect provides several methods to test collisions

• `Rect.collidepoint(point)` tests whether a point is within the Rect’s area

• `Rect.colliderect(rect)` tests whether two Rects intersect
Collision Detection

Rect II

- `Rect.collide(list)` tests whether the Rect collides with at least one Rect in the given list
- `Rect.collideall(list)` tests whether the Rect collides with all Rects in the list
- `Rect.collide(dict)` tests whether the Rect collides with at least one Rect in the given dictionary
- `Rect.collideall(dict)` tests whether the Rect collides with all Rects in the dictionary
Collision Detection
Sprites

• The module sprite provides several methods to test collision

• `sprite.spritecollide(…)` returns a list of sprites within a group that intersect with a given sprite

• `sprite.collide_rect(a,b)` checks whether two sprites intersect (must have rects)

• `sprite.collide_circle(a,b)` checks whether the radius of two sprites intersect. Radius attribute should be defined in the sprite.

![False](image1)
False

![True](image2)
True
Collision Detection

Sprites 2

- `sprite.groupcollide(a,b)` returns a list of sprites of two groups that intersect
- `sprite.collide_mask(a,b)` checks whether two Sprites collide on a bitmap level (non-transparent pixels overlap)

```python
if pygame.sprite.collide_mask(head1, head2):
    print "collide"
```

False  
True
Collision Detection

Masks

- Masks are 1bit per pixel representations of areas that can collide
- Module mask contains functions and classes to create and use masks
- `mask.from_surface(surface, threshold=127)` creates a mask of a surface. Threshold defines the alpha value that counts as collideable
- Class Mask contains methods to work with classes

Original Mask

![Original Mask]

Collision area

Mask

![Mask]
Collision Detection

Conclusion

• Pygame offers various ways to check for collisions
• **Choose your collision detection algorithm wisely depending on the task**
• Pixel based collision detection is precise but slow
• Rect or radius based collision detection is fast but imprecise
In-class exercise

Implement a collision detection for the sprite animations from the previous examples. Two of these sprites should move freely through the scene. If they collide, they should move in opposite directions.
Sound

- Sound is an essential part of multimedia applications
- Provides immediate feedback about an action
- Supports realism (e.g. games)
- Provides accessibility (e.g. for blind people)
- ...
Sound in Pygame

Mixer

• Sounds are controlled using the `pygame.mixer` interface
• Mixer must be initialized
  `pygame.mixer.init(frequency,size,channels,buffer)`
• Automatically initialized with `pygame.init()` using the default values
• Default values can be changed using `pygame.mixer.pre_init()`
• The mixer mixes the sounds in background threads
  – Sounds are not blocking the rest of the application logic
Sound in Pygame

Sound Object

• `pygame.mixer.Sound` provides a class to load and control sound files (OGG and uncompressed WAV)

• `Sound.play(loops=0, maxtime=0, fade_ms=0)` plays the sound file

• Other methods: `stop()`, `fadeout(time)`, `set_volume(value)` etc.

```
click_sound = pygame.mixer.Sound("click.wav")
click_sound.play()
```

```
click_sound = pygame.mixer.Sound("click.wav")
click_sound.play(3)
```
Sound in Pygame

Channels

- A channel represents one of the channels that are mixed by the soundcard
- `Sound.play()` returns a Channel object (or None if all channels are blocked)
- Provides methods to manipulate the sound and create useful effects (e.g. `Channel.set_volume(left, right)`)
Sound in Pygame

Stereo Panning

• Create the illusion that sound is coming from a specific point at the screen
• Manipulate the volume of the different speakers
• Can be used to make a sound “move” over the screen

Stereo Panning Function

```python
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)
```

From: W. McGugan, Beginning Game Development with Python and Pygame, Apress 2007
Music in Pygame

• Don`t use pygame.mixer but **pygame.mixer.music**
• It enables **streaming** music which means that the file will be read in small chunks
• Supports MP3 and OGG files (OGG better supported across platforms)
• Other Methods include `stop()`, `pause()`, `rewind()` etc.
• **Attention**: only one song can be streamed at the same time

```
playing a song using pygame

pygame.mixer.music.load("music.ogg")
pygame.mixer.music.play()
```
Creating your own Sound

- Record real sounds and edit them
- Free sound editor Audacity
  (http://audacity.sourceforge.net/?lang=de)