


Multimedia-Programmierung

Übung 8

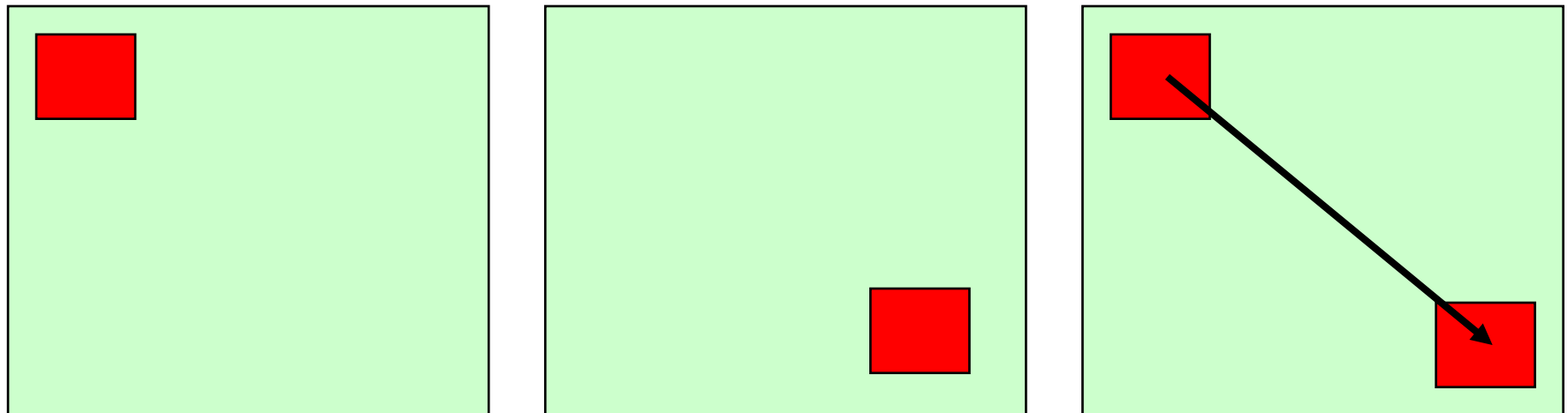
Ludwig-Maximilians-Universität München
Sommersemester 2011

Today

- Sprite animations in  Pygame
- Advanced collision detection
- Sound

Keyframe Animations

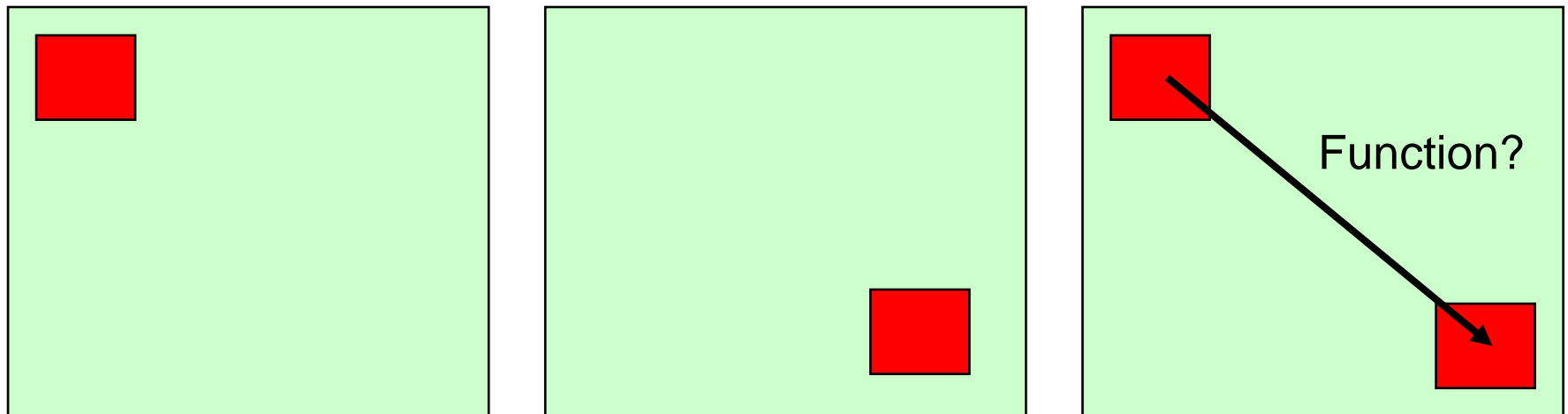
- Keyframes are defined
- Intermediate steps are interpolated
- Basic interpolators/tweens/... built into many programming environments (e.g. Flash, JavaFX)
- Examples: motion, color, shape



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?



Horizontal Animation (old slides)



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

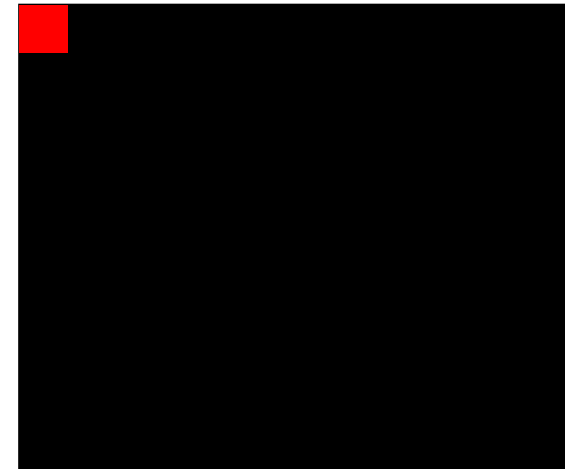
player_image = 'head.jpg'
pygame.init()

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

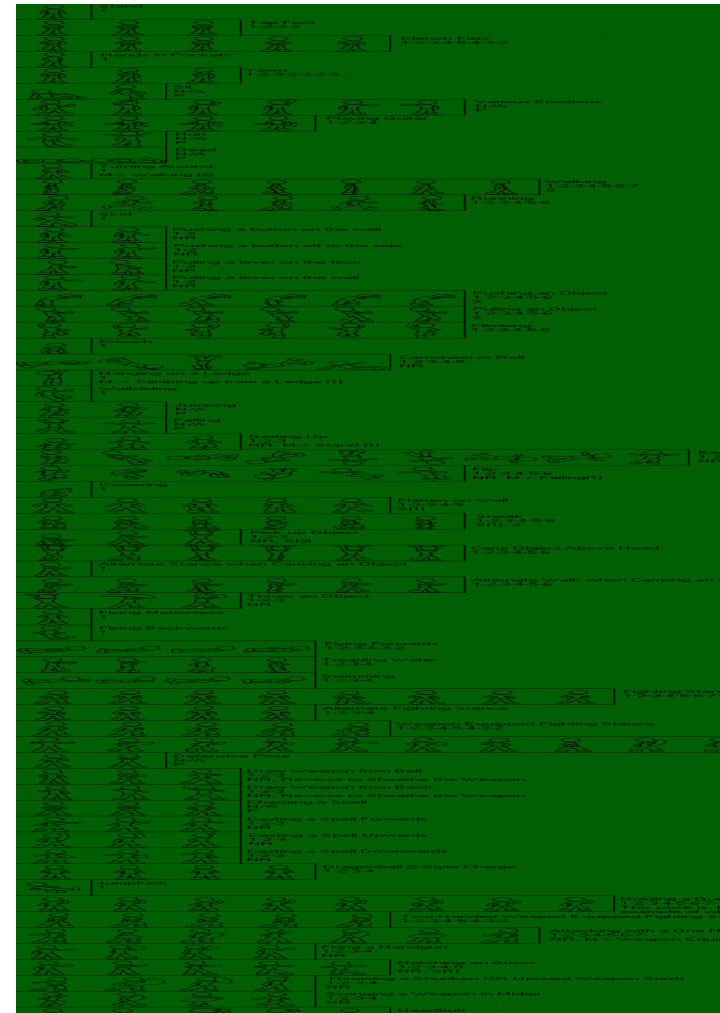
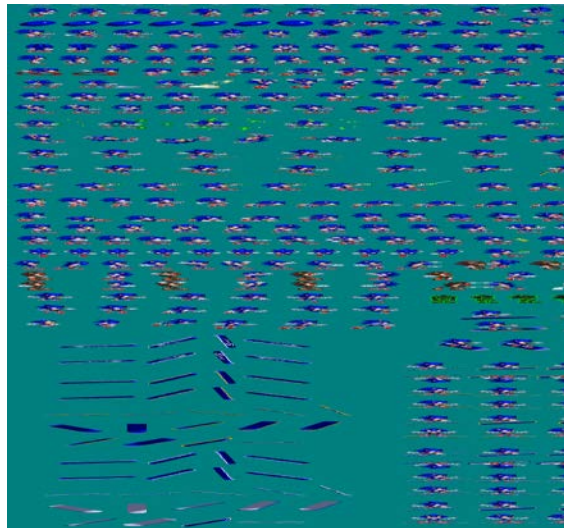
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
        screen.fill((255,255,255))
        if x > screen.get_width():
            x = 0 - mouse_cursor.get_width()
            screen.blit(mouse_cursor, (x, y))
            x+=10
        pygame.display.update()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

Result:



Sprite Sheets & Spriting

- Sprite sheets contain all possible movements for a character
- Each Sprite should have the same size for easy slicing in software
- Spriting means to adapt existing sprites or sprite sheets or create new ones (e.g. with empty outlines)



http://www.yoyogames.com/extras/resource/file/san1/90/90/male_character_outline_final.png

http://www.themysticalforestzone.com/Sprite_section.htm

Slicing Sprite Sheets



```
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)

First Sprite Animation 1



```
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, „exploded-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
```

remember the current frame
create the frames (defined later)
Based on the frames per second (fps) calculate the time needed for animation changes

```
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
```

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()
```

Multiple Parallel Animations



```
...
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()
```

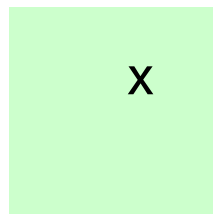
two bombs in two
different framerates

```
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()
```

Collision Detection

Rect

- Rect provides several methods to test collisions
<http://www.pygame.org/docs/ref/rect.html>
- **Rect.collidepoint(point)** tests whether a point is within the Rect's area



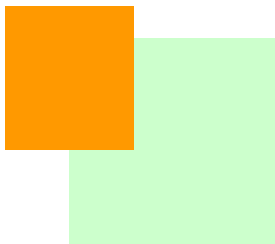
True



False

x

- **Rect.colliderect(rect)** tests whether two Rects intersect



True



False

Collision Detection

Rect II

- `Rect.collidelist(list)` tests whether the Rect collides with **at least one** Rect in the given list
- `Rect.collidelistall(list)` tests whether the Rect collides with **all** Rects in the list
- `Rect.collidedict(dict)` tests whether the Rect collides with **at least one** Rect in the given dictionary
- `Rect.collidedictall(dict)` tests whether the Rect collides with **all** Rects in the dictionary

Collision Detection

Sprites

- The module `sprite` provides several methods to test collision
<http://www.pygame.org/docs/ref/sprite.html>
- `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



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)

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



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
<http://www.pygame.org/docs/ref/mask.html>
- `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



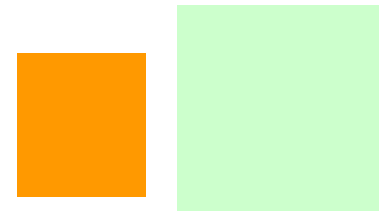
collision area



Collision Detection

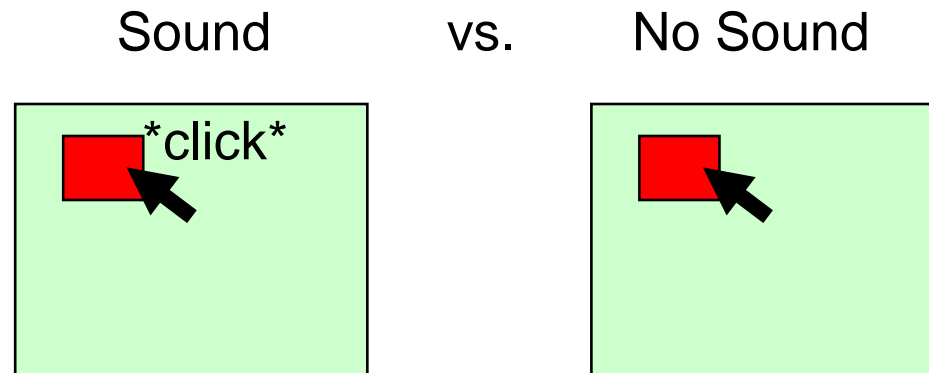
Conclusion

- 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



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.

playing a sound file

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

playing a sound file in a loop 4(!) times

```
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)`)

playing a sound file from the right speaker only

```
channel = click_sound.play()  
channel.set_volume(0.0, 1.0)
```

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

```
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

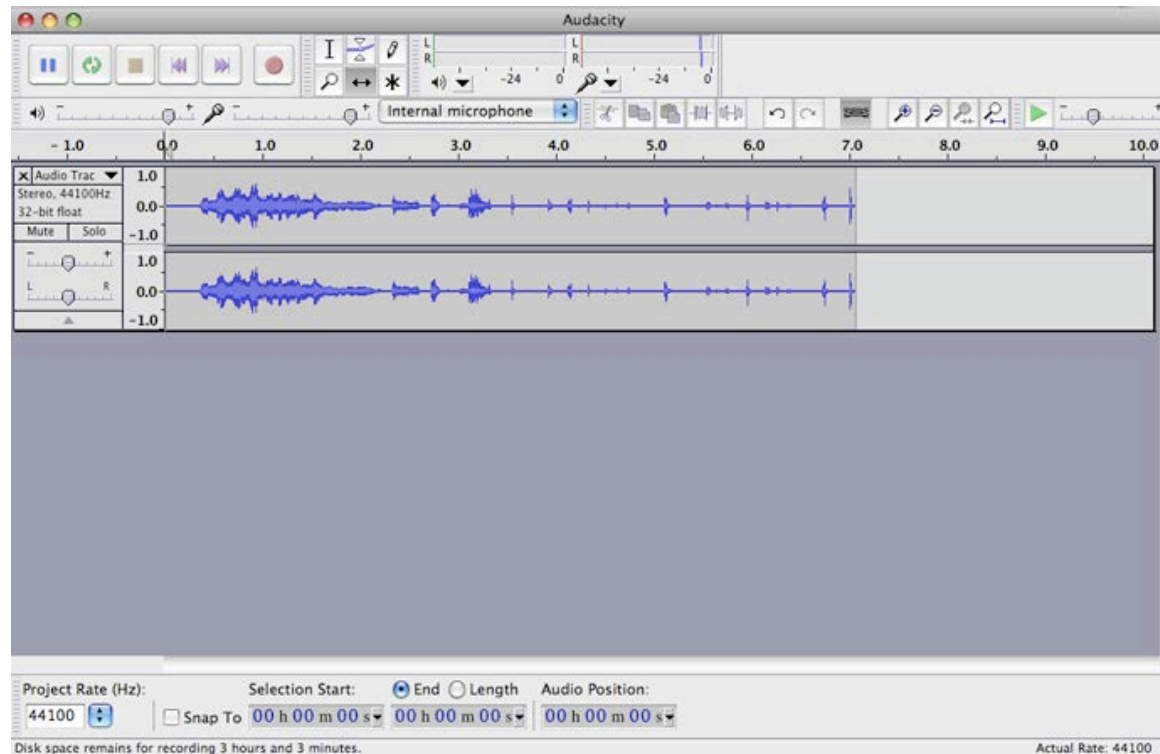
- Don't use `pygame.mixer` but [pygame.mixer.music](#)
- It enables **streaming** music which means that the file will be read in small chunks
- Supports MP3 and OGG files (OGG better supported across platforms)
- Other Methods include [stop\(\)](#), [pause\(\)](#), [rewind\(\)](#) etc.
- **Attention**: 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>)



Useful Links

- Pygame Sprite Tutorial
<http://kai.vm.bytemark.co.uk/~piman/writing/sprite-tutorial.shtml>
- Pygame API !!!!
<http://www.pygame.org/docs/>