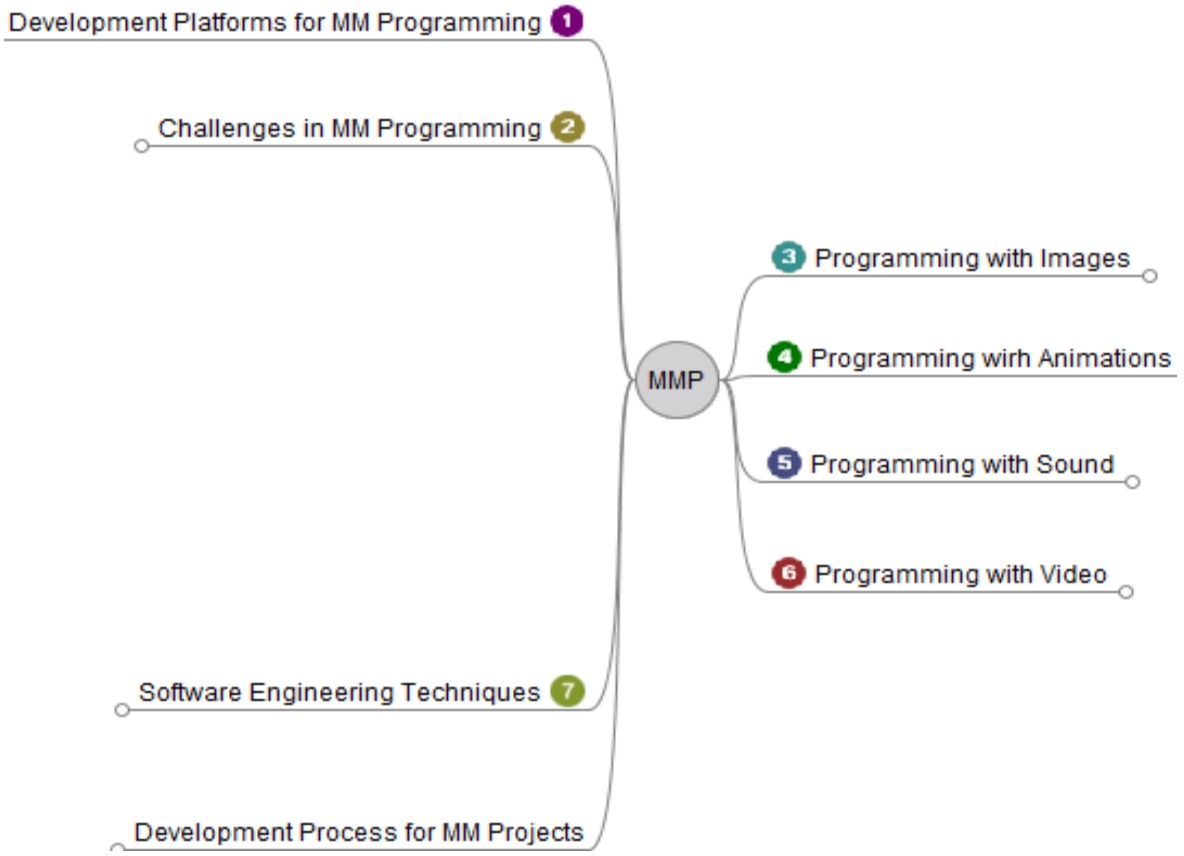
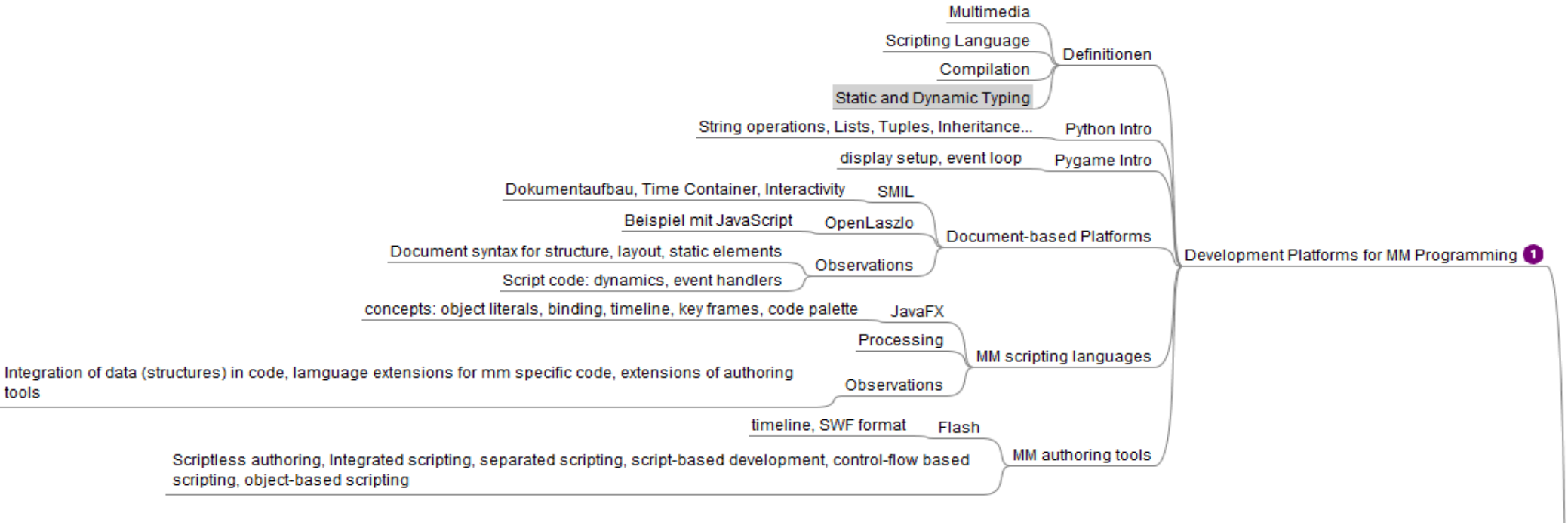
Multimedia-Programmierung
Fragestunde zur Klausur

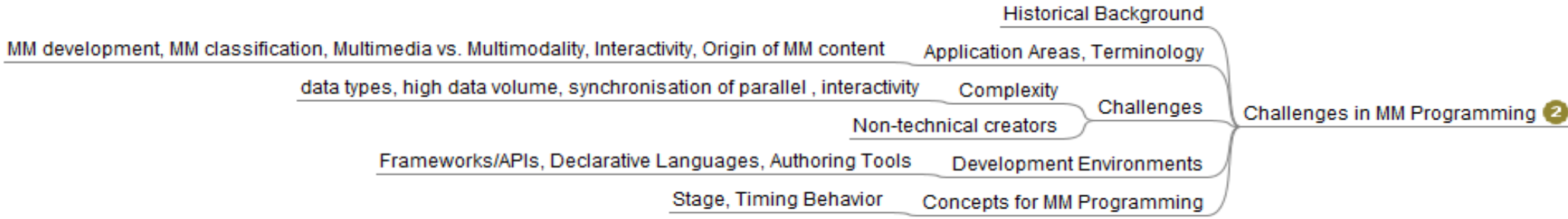
Ludwig-Maximilians-Universität München

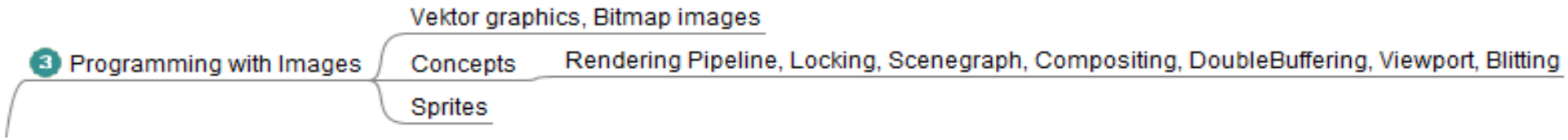
Sommersemester 2011

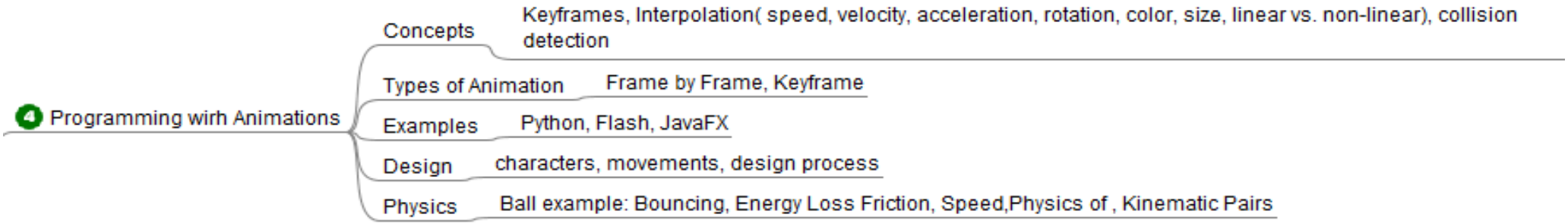
Zusammenfassung Vorlesung

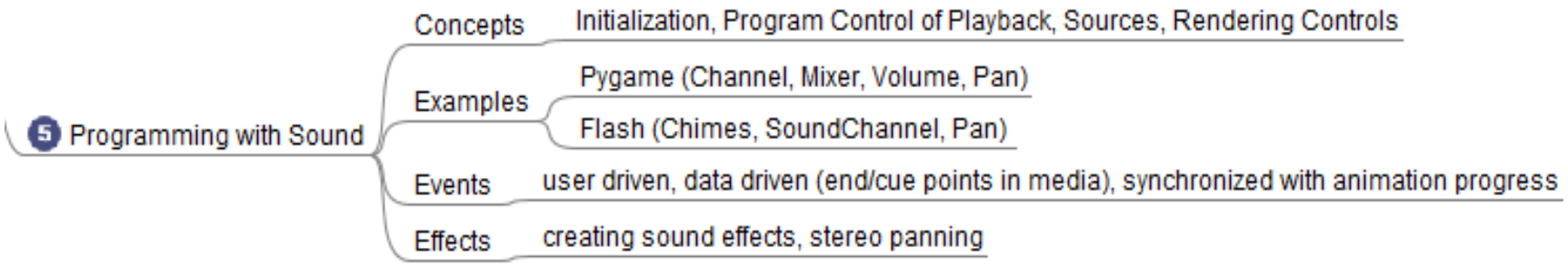


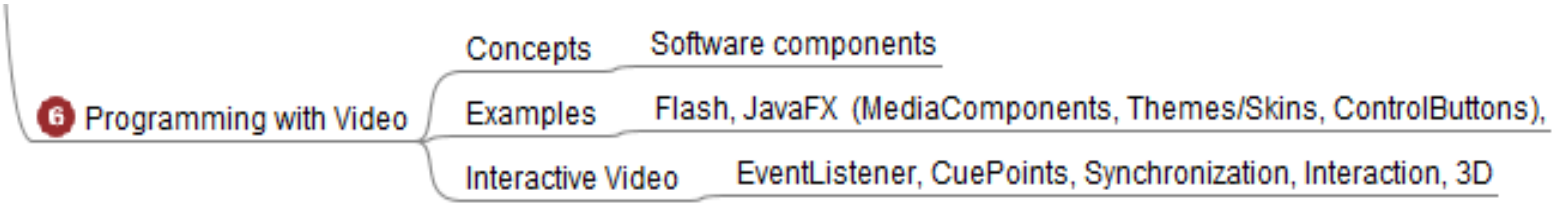

Development Platforms for MM Programming
Definitionen
Multimedia
Scripting Language
Compilation
Static and Dynamic Typing
Python Intro
String operations, Lists, Tuples, Inheritance...
Pygame Intro
display setup, event loop
Document-based Platforms
SMIL
Dokumentaufbau, Time Container, Interactivity
OpenLaszlo
Beispiel mit JavaScript
Observations
Document syntax for structure, layout, static elements
Script code: dynamics, event handlers
MM scripting languages
JavaFX
concepts: object literals, binding, timeline, key frames, code palette
Processing
Observations
Integration of data (structures) in code, lamguage extensions for mm specific code, extensions of authoring tools
MM authoring tools
Flash
timeline, SWF format
Scriptless authoring, Integrated scripting, separated scripting, script-based development, control-flow based scripting, object-based scripting

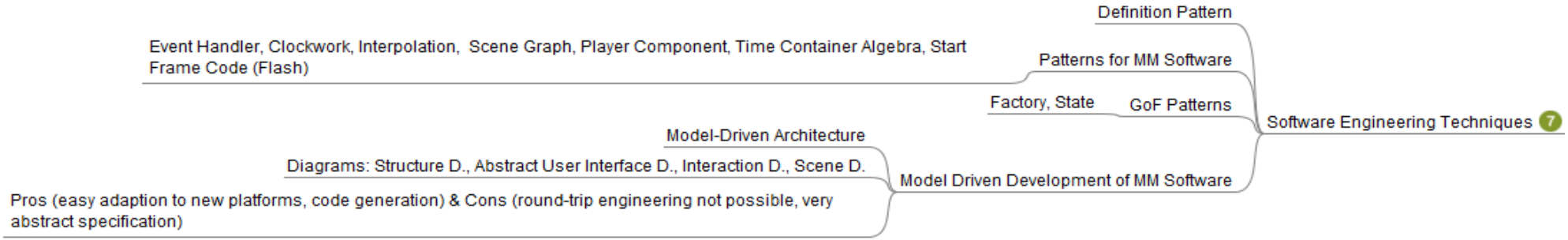

Challenges in MM Programming
2
Historical Background
Application Areas, Terminology
Challenges
Complexity
Non-technical creators
Development Environments
Concepts for MM Programming
MM development, MM classification, Multimedia vs. Multimodality, Interactivity, Origin of MM content
data types, high data volume, synchronisation of parallel , interactivity
Frameworks/APIs, Declarative Languages, Authoring Tools
Stage, Timing Behavior

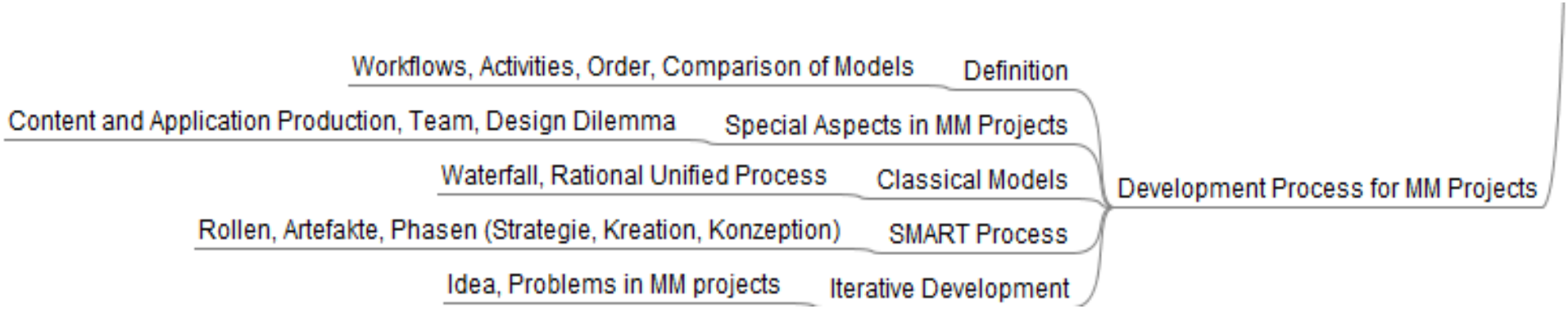

3 Programming with Images
Vektor graphics, Bitmap images
Concepts
Rendering Pipeline, Locking, Scenegraph, Compositing, DoubleBuffering, Viewport, Blitting
Sprites


4 Programming wirh Animations
Concepts
Keyframes, Interpolation(speed, velocity, acceleration, rotation, color, size, linear vs. non-linear), collision detection
Types of Animation
Frame by Frame, Keyframe
Examples
Python, Flash, JavaFX
Design
characters, movements, design process
Physics
Ball example: Bouncing, Energy Loss Friction, Speed,Physics of , Kinematic Pairs


5 Programming with Sound
Concepts
Initialization, Program Control of Playback, Sources, Rendering Controls
Examples
Pygame (Channel, Mixer, Volume, Pan)
Flash (Chimes, SoundChannel, Pan)
Events
user driven, data driven (end/cue points in media), synchronized with animation progress
Effects
creating sound effects, stereo panning


6 Programming with Video
Concepts
Software components
Examples
Flash, JavaFX (MediaComponents, Themes/Skins, ControlButtons),
Interactive Video
EventListener, CuePoints, Synchronization, Interaction, 3D


Software Engineering Techniques 7
Definition Pattern
Patterns for MM Software
Event Handler, Clockwork, Interpolation, Scene Graph, Player Component, Time Container Algebra, Start Frame Code (Flash)
GoF Patterns
Factory, State
Model Driven Development of MM Software
Model-Driven Architecture
Diagrams: Structure D., Abstract User Interface D., Interaction D., Scene D.
Pros (easy adaption to new platforms, code generation) & Cons (round-trip engineering not possible, very abstract specification)

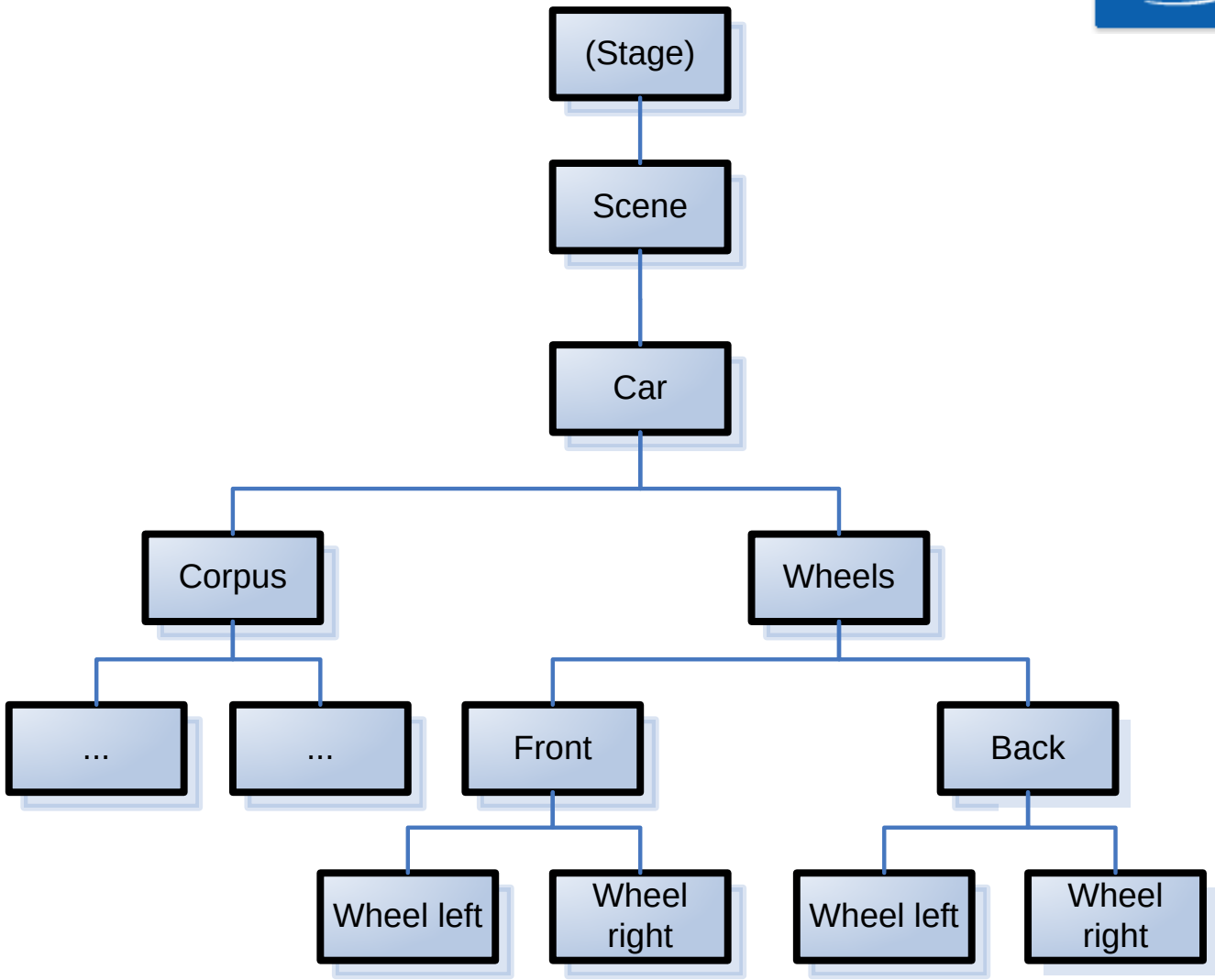

Development Process for MM Projects
Definition
Workflows, Activities, Order, Comparison of Models
Special Aspects in MM Projects
Content and Application Production, Team, Design Dilemma
Classical Models
Waterfall, Rational Unified Process
SMART Process
Rollen, Artefakte, Phasen (Strategie, Kreation, Konzeption)
Iterative Development
Idea, Problems in MM projects

Beispielaufgaben

Aufgabe 1: Grafik und Animation in MM Anwendungen

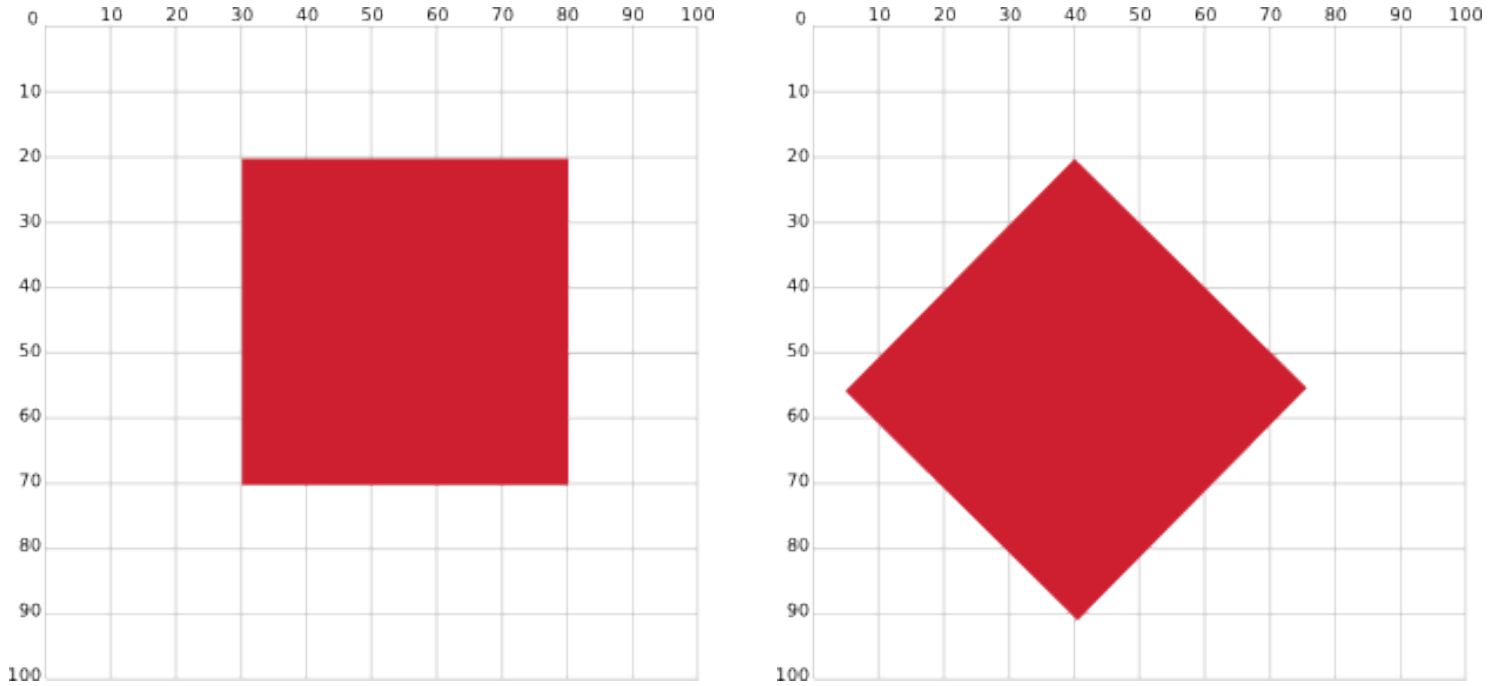
- Zur Realisierung von Objekten, die in mehrere Animationen unterteilt sind, bietet es sich an, Szenengraphen zur logischen Gliederung zu verwenden. Skizzieren Sie beispielhaft einen Szenengraphen für ein einfaches Auto (Sicht von oben). Dieser muss so ausgelegt sein, dass folgende Animationen leicht realisiert werden können:
  - Das Auto kann als ganzes animiert werden.
  - Jedes der vier Räder ist animiert.
  - Die beiden Vorderräder lassen sich auf einfache Art gemeinsam animieren.

Aufgabe 1a,


(Stage)
Scene
Car
Corpus
Wheels
...
...
Front
Back
Wheel left
Wheel right
Wheel left
Wheel right

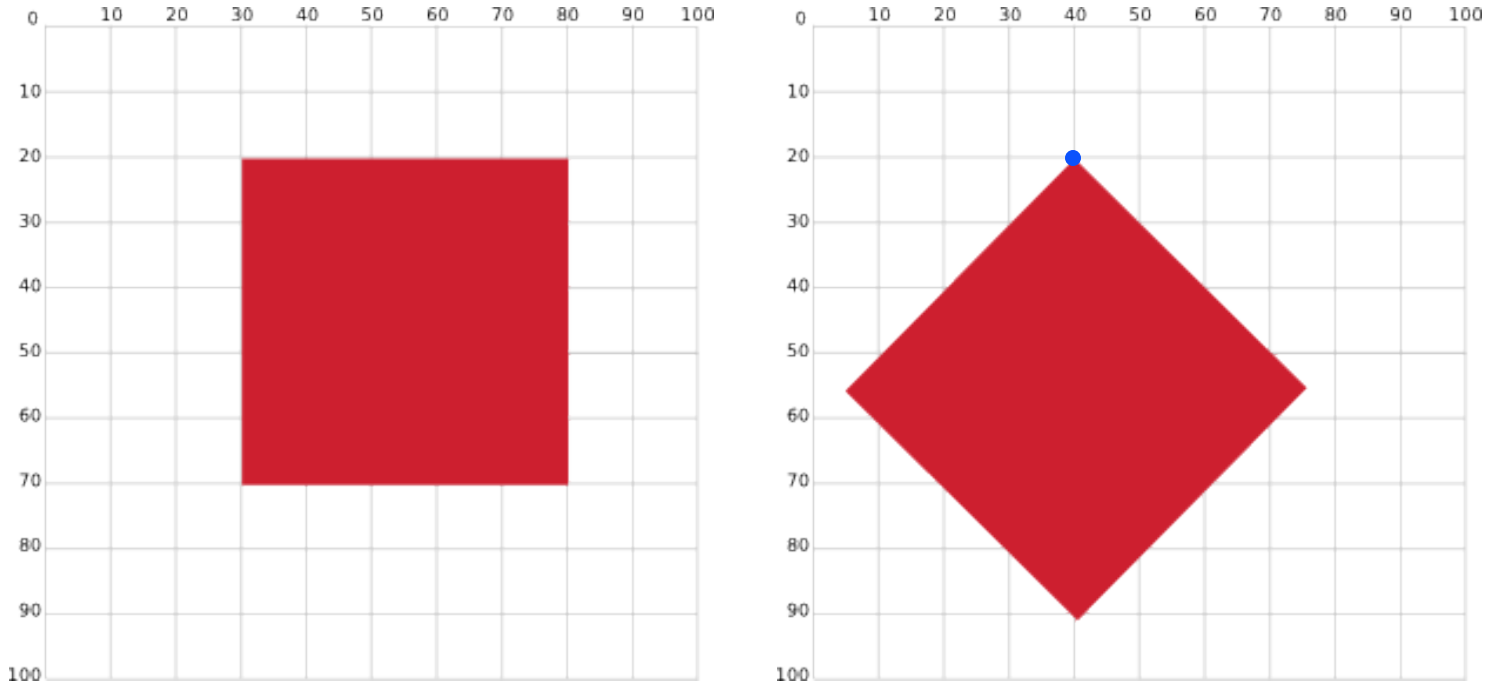
Aufgabe 1b,

a) Die folgenden Abbildungen skizzieren ein Rechteck in JavaFX vor (links) und nach einer Transformation (rechts). Bearbeiten Sie die Transformation in folgendem Code so, dass der Zustand in Abbildung 2 (rechts) erreicht wird. Sie müssen dazu exakt zwei Transformationen verwenden. Auch darf kein Transformationstyp zweimal vorkommen.



Aufgabe 1b,

a) Die folgenden Abbildungen skizzieren ein Rechteck in JavaFX vor (links) und nach einer Transformation (rechts). Bearbeiten Sie die Transformation in folgendem Code so, dass der Zustand in Abbildung 2 (rechts) erreicht wird. Sie müssen dazu exakt zwei Transformationen verwenden. Auch darf kein Transformationstyp zweimal vorkommen.



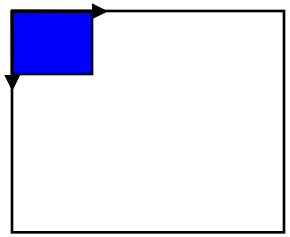
Aufgabe 1b,


```
 1 Stage {
2     title: "Transformationen"
3     width: 400
4     height: 400
5     scene: Scene {
6         content: [
7             Rectangle {
8                 x: 30, y: 20
9                 width: 50, height: 50
10                  fill: Color.RED
11                  transforms: [




14                  ]
15              }
16          ]
17      }
18 }
19
```

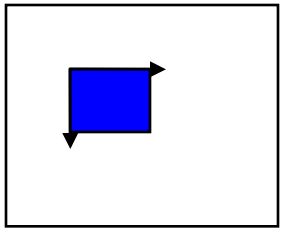
Transformations

the transform variable

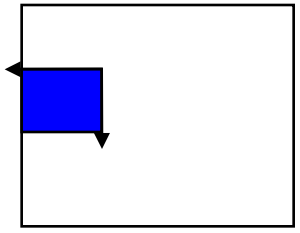
- **Transformations are applied in order of their appearance within the transform sequence**



1. translate(100,100)



2. rotate(90,0,0)



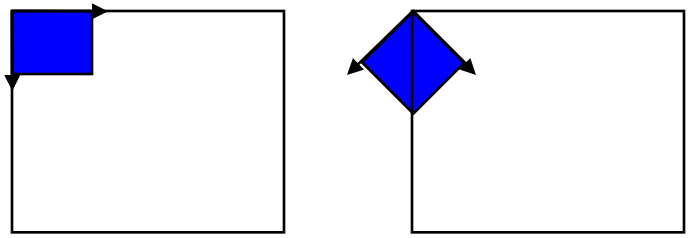
```
Stage {
    title : "Transformations"
    scene: Scene {
        width: 400
        height: 400
        content: [
            Rectangle {
                x: 0, y: 0
                width: 100, height: 100
                fill: Color.BLUE
                stroke: Color.BLACK
                transforms: [
                    Transform.translate(100,100),
                    Transform.rotate(90, 0, 0)
                ]
            }
        ]
    }
}
```

sequence of transformations

Transformations

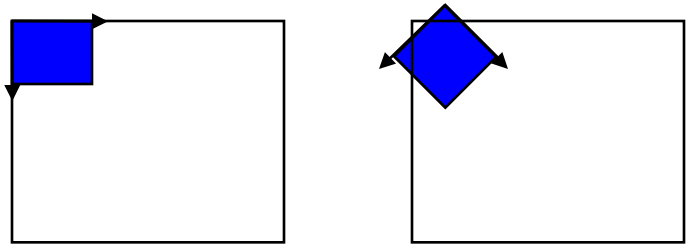
some examples 1

- **Transform.rotate(angle,x,y) rotates clockwise around a pivot point**



```
...
transforms: [
    Transform.rotate(45, 0, 0)
]
...
```

rotate 45° clockwise around 0,0



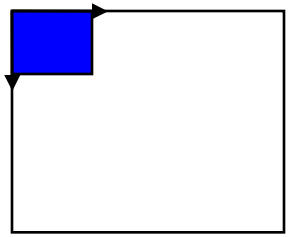
```
...
transforms: [
    Transform.rotate(45, 50, 50)
]
...
```

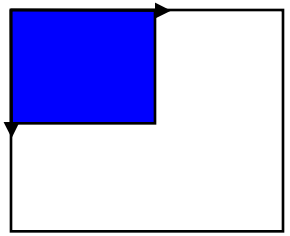
around the center (if rectangle is 100x100px)

Transformations

some examples 2

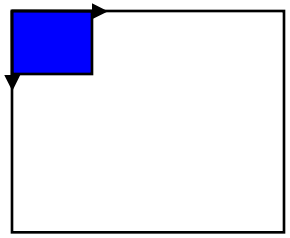
- **Transform.scale(xfactor,yfactor) scales the node's axes**

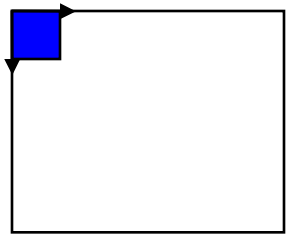




```
...
transforms: [
    Transform.scale(2.0, 2.0)
]
...
```

scale 200%





```
...
transforms: [
    Transform.scale(0.50, 0.50)
]
...
```

scale 50%

Aufgabe 1b,


```
 1 Stage {
2     title: "Transformationen"
3     width: 400
4     height: 400
5     scene: Scene {
6         content: [
7             Rectangle {
8                 x: 30, y: 20
9                 width: 50, height: 50
10                  fill: Color.RED
11                  transforms: [

12                      Transform.translate(10,0),
13                      Transform.rotate(45,40,20)



14                  ]
15              }
16          ]
17      }
18 }
19
```

Aufgabe 2: Programming with Animations

- Beschreiben Sie **kurz** das Prinzip der Interpolation im Zusammenhang mit der Animation von Objekten.

Animation by Interpolation

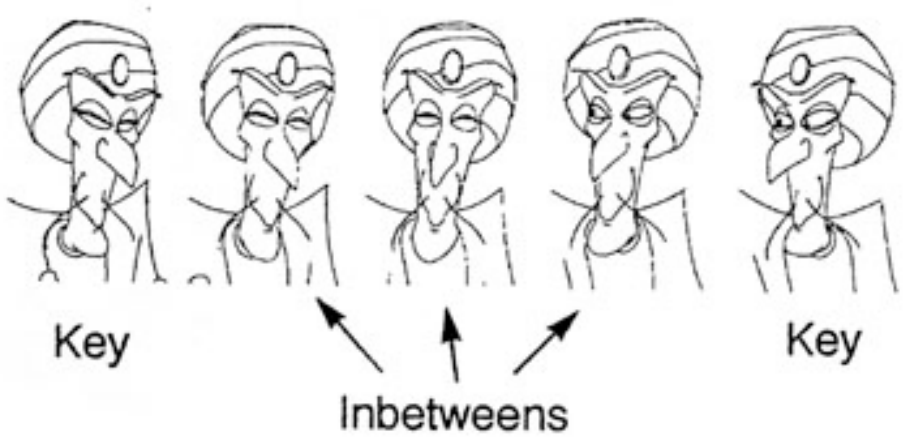
- Key frame:
  - Contains manually defined objects & object attributes
- In-between frame:
  - Object attributes computed automatically
- Computation of attribute values:
  - Discrete interpolation:
    - » Start and end value given
    - » Intermediate position given by frame number
  - E.g. (linear interpolation):

    $delta = (end - start) / steps$

    $value(i) = start + delta * i$

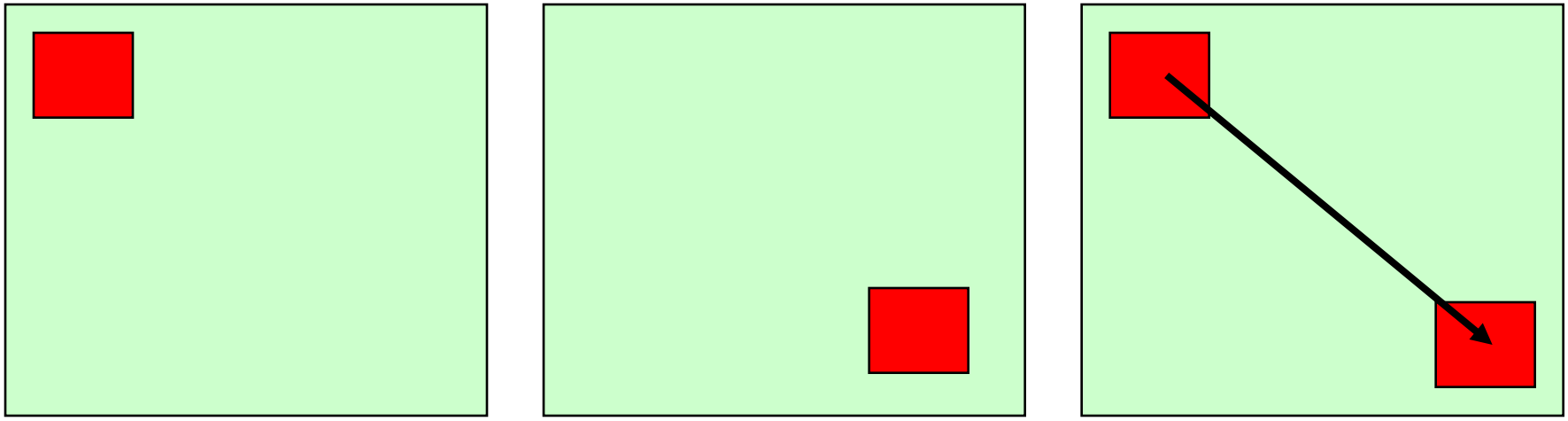
Keyframe Animations

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

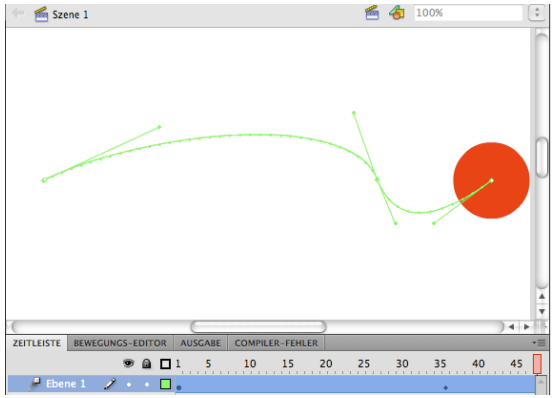


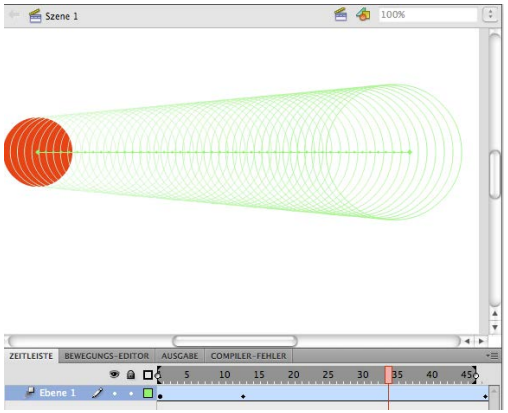
Keyframe Animations

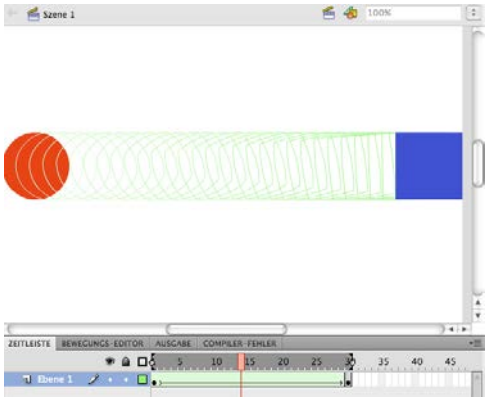
- Keyframes are defined
- Intermediate steps are interpolated
- Basic interpolators/tweens/... built into many programming environments (e.g. Flash, JavaFX)
- Examples: motion, color, shape

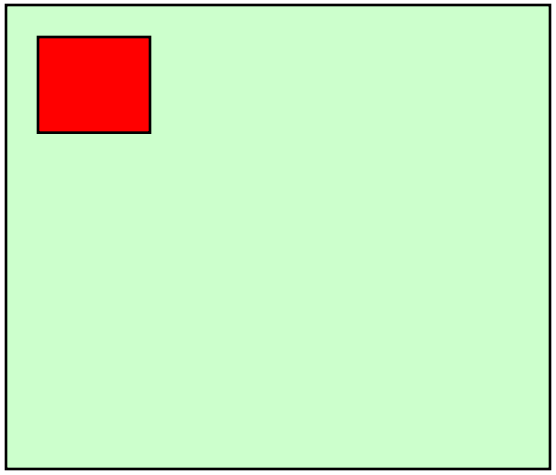


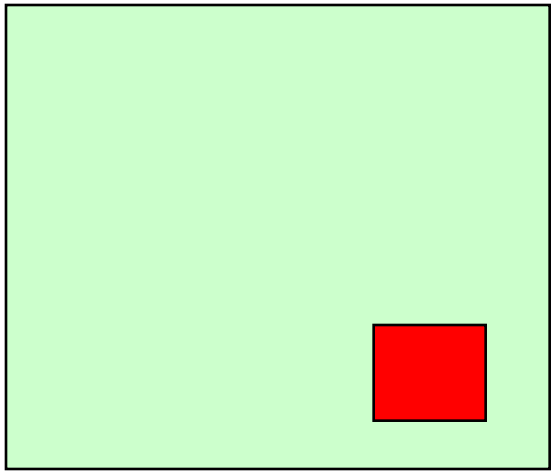
E.g. Tweens in Flash..

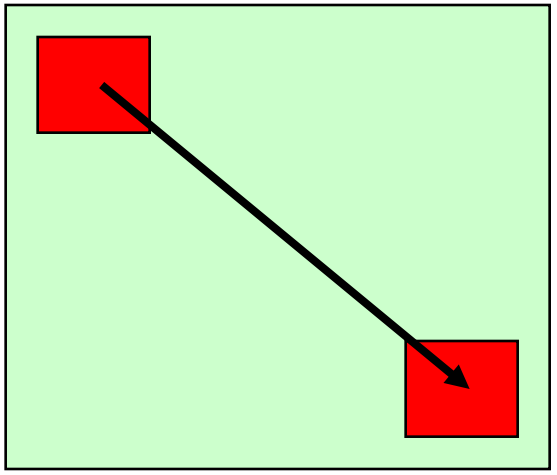








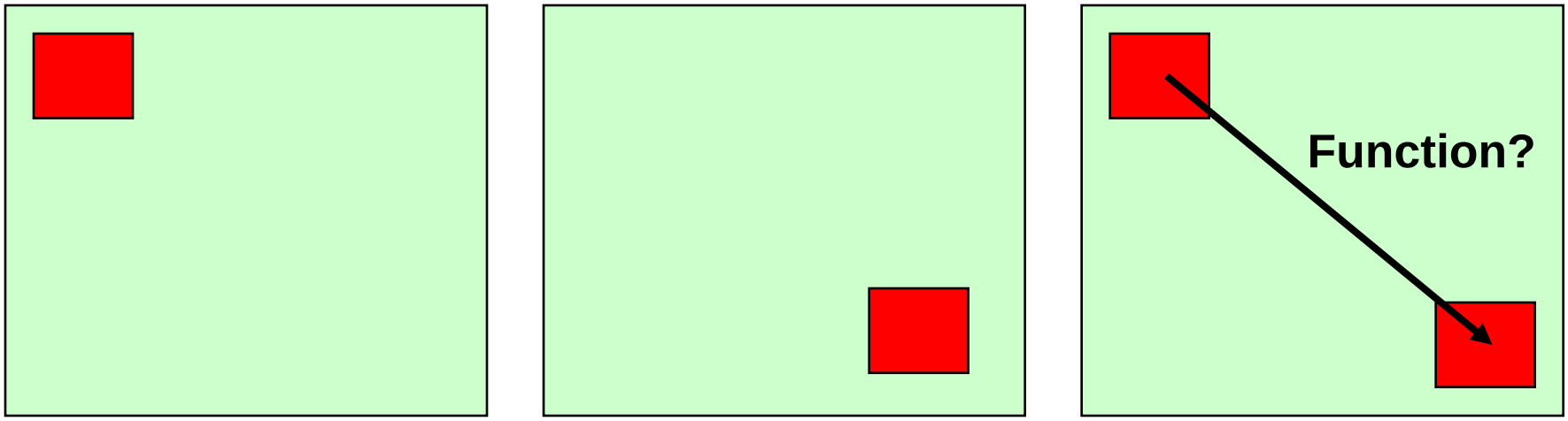




Keyframe Animations

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

```
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()
```

animated in steps of 10 pixels

Result:

Moving an object time-based

```
...

clock = pygame.time.Clock()

speed = 300.0

x = 0 - mouse_cursor.get_width()
y = 10
while True:
    time_passed = clock.tick() / 1000.0
    moved_distance = time_passed * speed
    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+=moved_distance
    pygame.display.update()
```

speed in pixels per second

time passed since last tick() in seconds

distance moved since last call

move the sprite the calculated distance

Result:

Aufgabe 3

- Event-Handler sind unter anderem ein gängiges Konzept zur Abfrage von Benutzereingaben. Ergänzen Sie den folgenden Pygame Code so, dass überprüft wird, ob der Nutzer eine Maustaste gedrückt hat. Sollte dieser Fall eintreten, werden die x- und y-Koordinate der Maus auf der Konsole ausgegeben.

Aufgabe 3

```
 1 import pygame
2 from pygame.locals import *
3
4
5 pygame.init()
6 screen = pygame.display.set_mode((640, 480), 0, 32)
7
8
9 while True:
10     for event in pygame.event.get():
11         if event.type == QUIT:
12             exit()
```

Aufgabe 3

```
 1 import pygame
2 from pygame.locals import *
3
4
5 pygame.init()
6 screen = pygame.display.set_mode((640, 480), 0, 32)
7
8
9 while True:
10     for event in pygame.event.get():
11         if event.type == QUIT:
12             exit()
13         if event.type == MOUSEBUTTONDOWN:
14             print str(event.pos[0])+" "+str(event.pos[1])
```

Aufgabe 4

- Das **Observer Pattern** ist ein Entwicklungsmuster, bei dem abhängige Objekte über Änderungen (welcher Art auch immer) benachrichtigt werden. In den meisten Programmiersprachen wird dieses Konzept über Listen implementiert, die alle Observer (Beobachter) beinhalten, die über Änderungen informiert werden müssen. Der Programmierer muss sich darum zumeist selber kümmern.
  - In JavaFX gibt es ein Konstrukt, welches dem Programmierer diese Arbeit erspart. Wie lautet dieses Schlüsselwort?

Data Binding

or: who needs the observer pattern

- Variables can be bound to expressions
- That is, whenever the expression changes, the variable will be updated accordingly
- Example:

Define

```
var a = 1;
var b = 2;

var c = bind a + b;
println(c);
a = 2;
println(c);
```

Output

```
3
4
```

Aufgabe 4 b,

- Gegeben ist folgender Code. Setzen sie die Variable y so, dass diese den String „Hallo“ enthält, wenn x den Wert 4 hat.

  Hat x einen anderen Wert als 4, dann wird y auf den Wert „Welt“ gesetzt. Das bedeutet, dass die Ausgabe des Skripts „Hallo“ „Welt“ ergeben soll.

```
1 var x = 4;
2
3 var y =
4
5 println(y);
6 x = 2;
7 println(y);
8
```

Aufgabe 4 b,

- Gegeben ist folgender Code. Setzen sie die Variable y so, dass diese den String „Hallo“ enthält, wenn x den Wert 4 hat.

  Hat x einen anderen Wert als 4, dann wird y auf den Wert „Welt“ gesetzt. Das bedeutet, dass die Ausgabe des Skripts „Hallo“ „Welt“ ergeben soll.

```
1 var x = 4;
2
3 var y = bind if(x==4) "hallo" else "welt";
4
5 println(y);
6 x = 2;
7 println(y);
8
```

Aufgabe 4: Sound

- Stereo Panning ist eine einfache Technik, die in Multimedia-Anwendungen wie z.B. Computerspielen oft verwendet wird um eine räumliche Illusion zu erzeugen.

  – Beschreiben Sie diese Technik **kurz** in eigenen Worten.

  – Kennen Sie eine Methode, um mit Ton eine räumliche Illusion zu schaffen, die über Stereo hinaus geht? Wenn ja, dann beschreiben Sie diese. Wenn nicht, denken Sie sich selber eine Methode aus.

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

Aufgabe 5: Collision Detection

- Kollisionserkennung ist für viele Multimedia-Anwendungen ein entscheidender Mechanismus, um Realismus zu erzeugen. Es gibt eine Vielzahl von möglichen Kollisionserkennungsalgorithmen. Je nach Anforderung der Anwendung können unterschiedliche Methoden geeigneter sein.
  - Bei der pixelbasierten Kollisionserkennung wird jeder Pixel eines Objekts mit der Kollisionsfläche verglichen, die Kollisionserkennung also pixelgenau durchgeführt. Wieso wird nicht einfach immer auf pixelbasierte Kollisionserkennung zurückgegriffen? Nennen Sie einen entscheidenden Grund dafür.

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

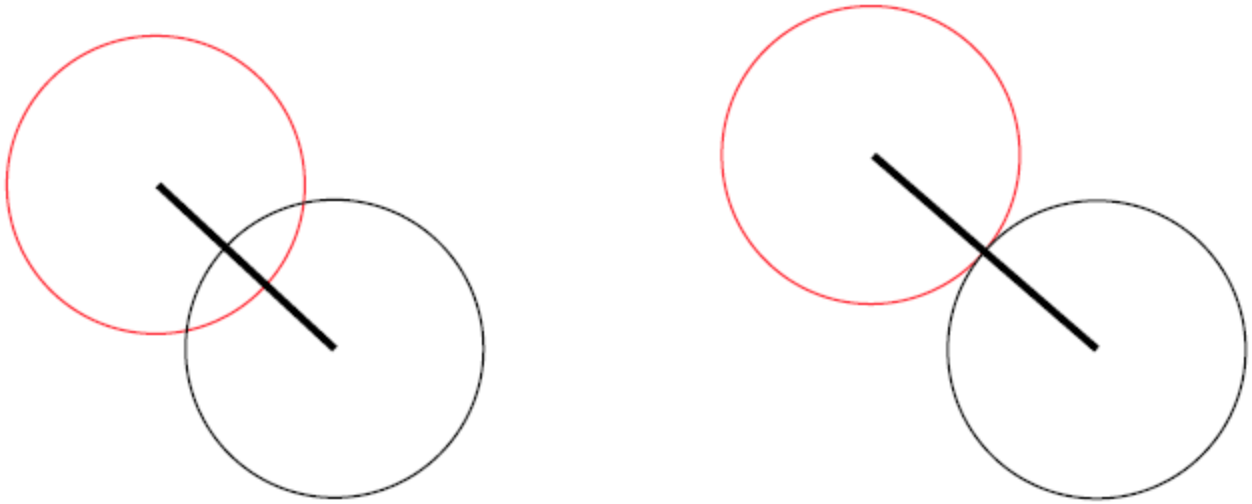
Aufgabe 5 b, Collision Detection

- Für einfache Formen wie Rechtecke und Kreise gibt es sehr effiziente Verfahren, um diese auf Kollision zu überprüfen.

  Gegeben seien nun zwei Kreise (mit Mittelpunkt und Radius). Skizzieren Sie (grafisch), wie man einfach überprüfen kann, ob diese zwei Kreise miteinander kollidiert sind.

  Beschreiben Sie außerdem, in eigenen Worten, eine Funktion, welche die beiden Kreise auf Kollision prüft (z.B. „wenn der Fall x eintritt dann … ansonsten …“).

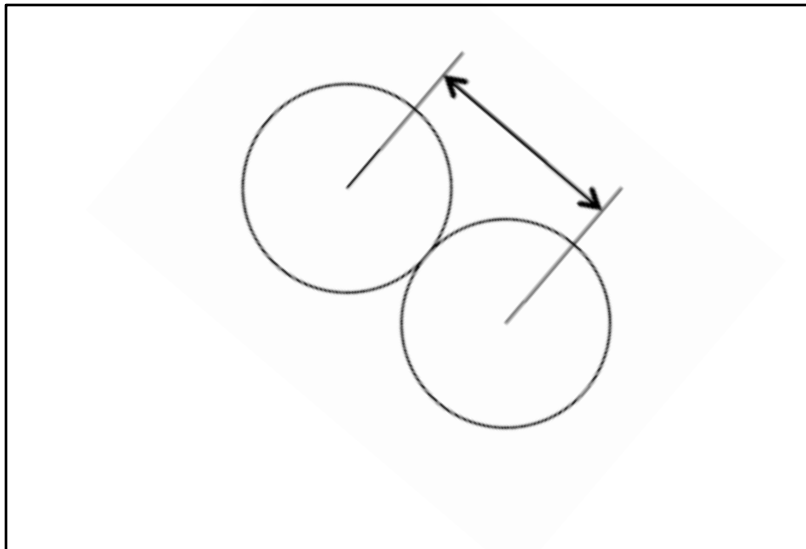
Placing Balls At Collision Time



This is a simplification compared to the actual physical laws.

Collision Detection Between Balls (2)

```
private function checkCollision(ball0:Ball,
        ball1:Ball):void
{
      var dx:Number = ball1.x - ball0.x;
      var dy:Number = ball1.y - ball0.y;
      var dist:Number = Math.sqrt(dx*dx + dy*dy);
      if(dist < ball0.radius + ball1.radius)
      {
            Collision detected ...
      }
}
```



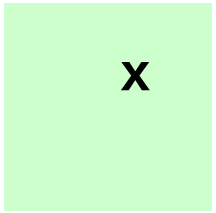
Collision Detection with Rect

```
....

#check bomb - ball collision
    for bomb in bombs:
        for ball in balls:
            if bomb.rect.colliderect(ball.rect):
                #explosion = AnimatedSprite(explosion_images)
                #explosion.rect = ball.rect
                #explosions.add(explosion)
                ball.kill()
```

Collision Detection

Rect

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

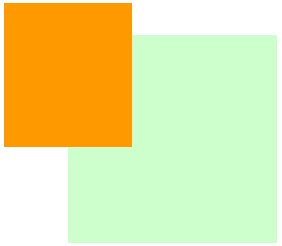


True

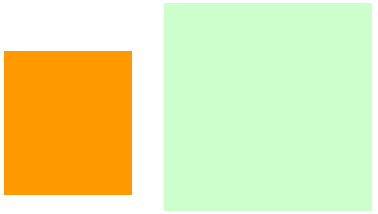
False

x

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



True



False

2 main options for Collision Detection

- Rect.collidepoint(point):return bool can be used to see whether a coordinate is within the area of a Rect object
- Rect.colliderect(Rect):return bool can be used to see whether any portion of either rectangle overlap (except the top+bottom or left+right edges).
- pygame.sprite has advanced methods to check for collisions
  - E.g. pygame.sprite.collide_rect(a,b) checks whether two sprites intersect

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 method 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

- The module sprite provides several method to test collision http://www.pygame.org/docs/ref/sprite.html
- sprite.spritecollide(sprite, group,...): return Sprite_list returns a list of sprites within a group that intersect with a sprite
- sprite.collide_rect(sprite_a,sprite_b): return bool 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 as attribute in the sprite.

False

True

Sprite-Group Collision with spritecollide()

```
import pygame
import math
from pygame.locals import *

class Box(pygame.sprite.Sprite):
    def __init__(self, color, initial_position):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.Surface((100,100),pygame.SRCALPHA,32)
        pygame.draw.rect(self.image,color,(0,0,100,100))
        self.rect = self.image.get_rect()
        self.rect.topleft = initial_position
        self.color = color
        self.collide = False

    def update(self):
        if self.collide:
            print "collide"
            pygame.draw.rect(self.image,(255,0,255),(0,0,100,100))
        else:
            pygame.draw.rect(self.image,self.color,(0,0,100,100))
```

Sprite-Group Collision with spritecollide()

```
pygame.init()
screen = pygame.display.set_mode((640, 480), 0, 32)
boxes = ([(255,0,0),(0,200)],[(0,255,0),(150,200)])
box1 = Box((0,100,255),(250,250))
sprites = pygame.sprite.Group()
for box in boxes:
    sprites.add(Box(box[0],box[1]))

clock = pygame.time.Clock()

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
        if event.type == MOUSEMOTION:
            box1.rect.center = pygame.mouse.get_pos()

    for sprite in pygame.sprite.spritecollide(box1,sprites, False): # return list of sprites
        sprite.collide = True

    screen.fill((0, 0, 0))
    sprites.update()
    sprites.draw(screen)
    screen.blit(box1.image,box1.rect)
    pygame.display.update()

    #set back to original color
    for sprite in sprites:
        sprite.collide = False
```

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 Sprite collide on a bitmap level (non-transparent pixels overlap)

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

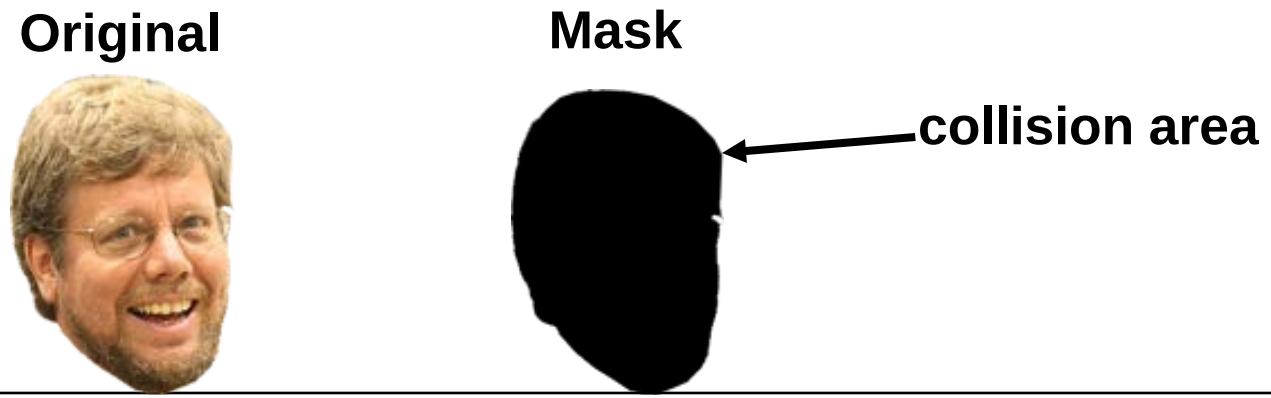
False

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
  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



Sprite Collision with Masks

```
import pygame
from pygame.locals import *

class Box(pygame.sprite.Sprite):
    def __init__(self, initial_position):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load("head.png").convert_alpha()
        self.rect = self.image.get_rect()
        self.rect.topleft = initial_position
        # sprite should have mask attribute (else is computed every time)
        # makes the transparent parts of the surface not set,
        # and the opaque parts set
        self.mask = pygame.mask.from_surface(self.image)

    def update(self, pos):
        self.rect.center = pos
```

Sprite Collision with Masks

```
pygame.init()

screen = pygame.display.set_mode((640, 480), 0, 32)
box1 = Box((200,200))
box2 = Box((100,10))
clock = pygame.time.Clock()


while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
        if event.type == MOUSEMOTION:
            box2.update(pygame.mouse.get_pos())
            if pygame.sprite.collide_mask(box1,box2):
                print "collide"
            else:
                print "no"

    screen.fill((100, 200, 0))
    screen.blit(box1.image,box1.rect)
    screen.blit(box2.image,box2.rect)
    pygame.display.update()
```

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

Bonusblatt