codecentric

Pong-Spiel in Python

Q4 2022

Das Spiel und die Quelldatei herunterladen

@codecentric

>> https://gitlab.codecentric.de/heinrich.braun/pong_schulpraktikum

>> https://github.com/h-braun/schulpraktikum

@public

Python als Programmiersprache

>> python.org/downloads/

1991 von Guido van Rossum entwickelt
> Die Standardbibliothek ist sehr umfangreich und wird ständig weiter ausgebaut

Dynamische Typisierung
> Variablen können unterschiedliche Typen innerhalb des selben Codeblocks zugewiesen werden

Interpretierte, höhere Programmiersprache
> Kann als Modul, oder sofort im Terminal (Jupyter Notebook) ausgeführt werden

Erweiterungen mit `pygame` und `numpy`


```
>> py -m pip install -U pygame --user
```
(unter **Windows**)

Einfache Anwendung durch modularen Aufbau

> Die wichtigsten Funktionalitäten können selbst angepasst werden (Game-Loop, Musik, Spielobjekte …)

Läuft auf unterschiedlichen Betriebssystemen

> Intern werden alle Funktionen an das auszuführende Betriebssystem angepasst

Optimierte mathematische Funktionen

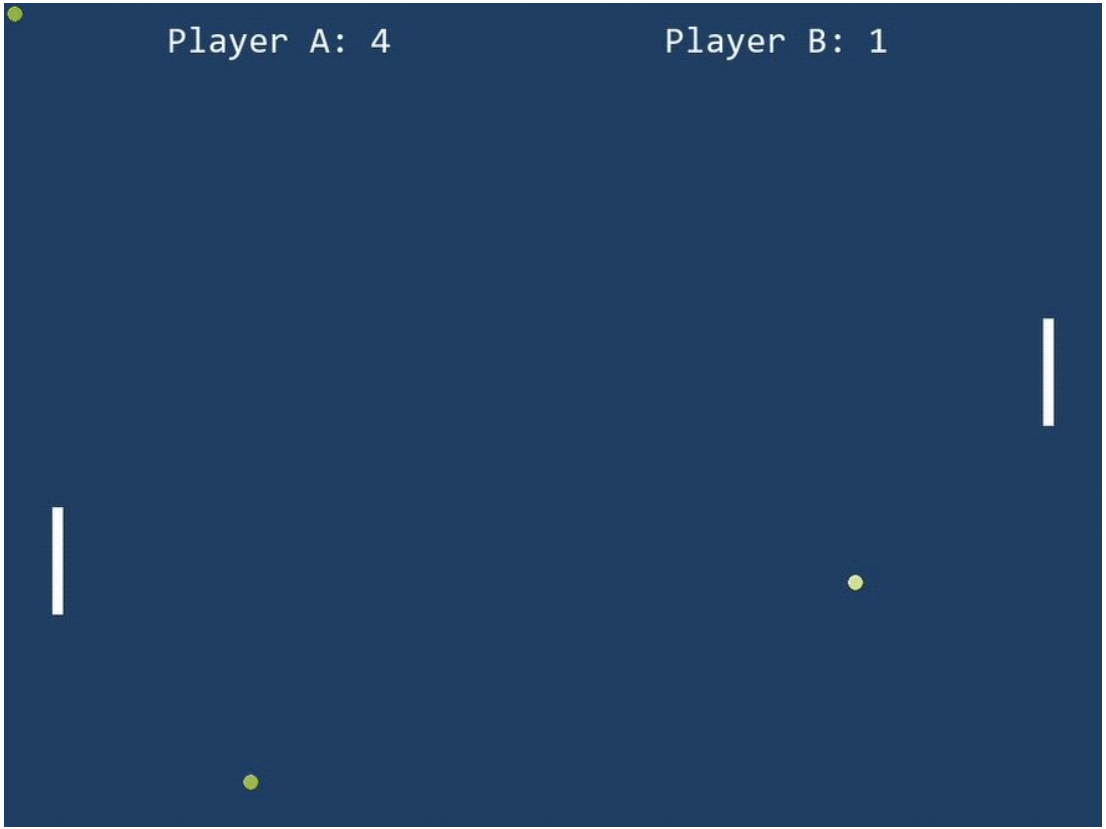
> Einfache Anwendung mathematischer Funktionen mit schneller Ausführung

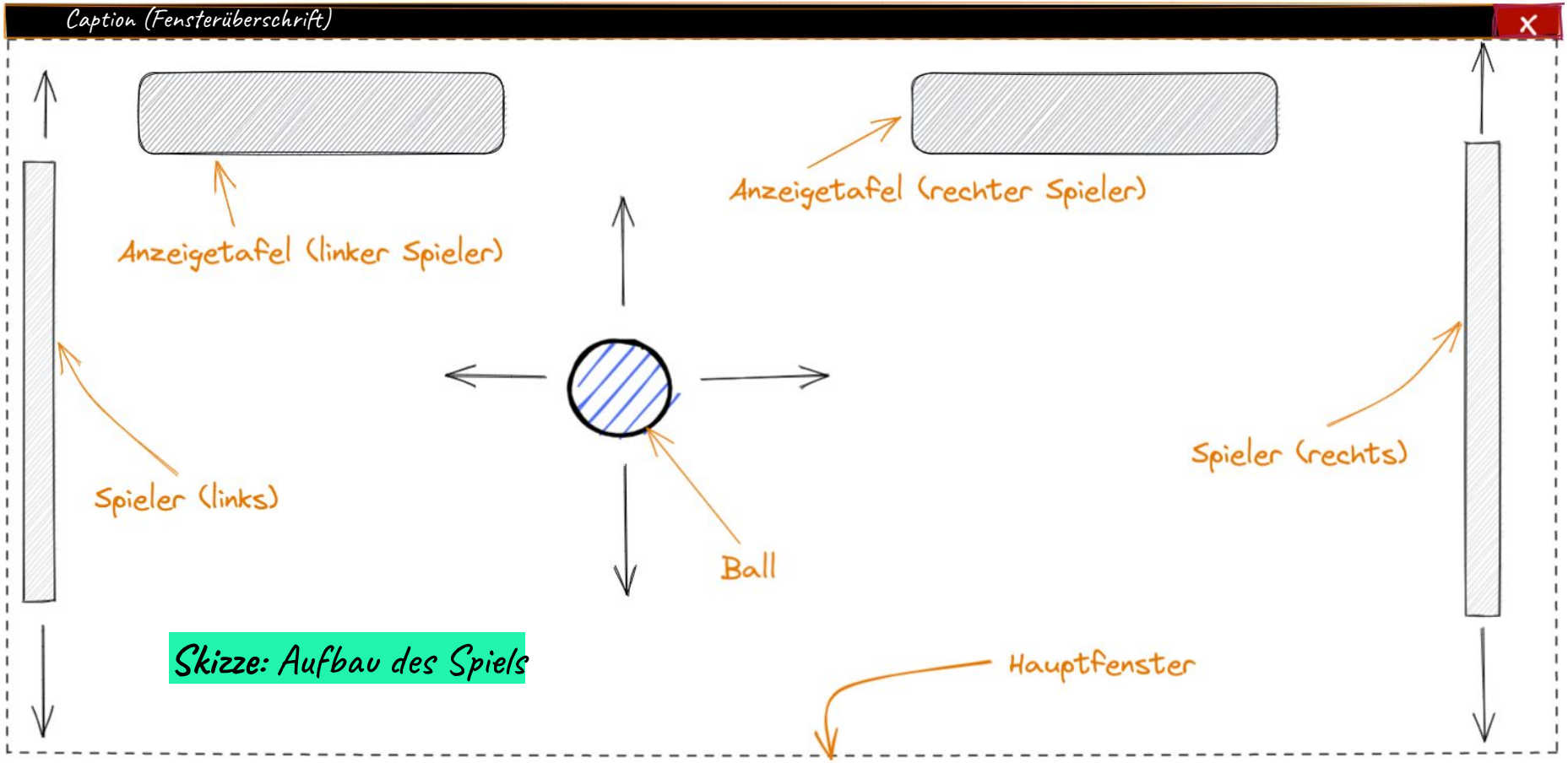
NumPy

```
>> py -m pip install -U numpy --user
```
(unter **Windows**)

Einfaches Beispiel

Pong by codecentric





Skizze: Aufbau des Spiels

Aktoren des Pong Spiels - Hauptfenster und Spieler

```
class Application
```

hauptfenster: `pygame.display.set_mode`
spieluhr: `pygame.time.Clock`
läuft das Spiel?: `True/False (boolean)`

Instanzvariablen

Vererbung

```
class Player(pygame.sprite.Sprite)
```

punktzahl: `int`
seite_des_spielers: `class PlayerSide(Enum)`
geschwindigkeit: `int`

```
# pygame Variablen
```

image: `pygame.Surface`
rect: `self.image.get_rect`

Initalisierung __init__

Parameter:
window_width, window_height, caption="Codecentric: Pong"
Fensterbreite x Fensterhöhe, sowie die Überschrift

Parameter:
side_of_field
Legt die Seite des Spielers im Spielfeld fest

Aktoren des Pong Spiels - Spielball

```
class Ball(pygame.sprite.Sprite)
```

ball_farbe: `tuple(rot, grün, blau)`
ball_radius: `int`
ball_form: `pygame.draw.circle`
geschwindigkeit: `np.array (numpy array)`
erster_aufschlag() `Callable (Funktionsaufruf)`

Besteht aus Geschwindigkeit [0] und Winkel [1]

Initalisierung __init__

Parameter:

background_color=(255, 255, 255, 255)

Gibt die Rot, Grün, Blau und Alpha werte der Hintergrundfarbe an

Aktoren des Pong Spiels - Spielfeld

```
class Matchfield
```

hauptfenster: Application.main window
_feld_erneuern() Callable (Funktionsaufruf)
hintergrundfarbe: tuple(Rot, Grün, Blau)
max_bälle_auf_feld: int
aktive_bälle_feld: int
ball_liste: List[Ball, …]
spieler_links: Player
spieler_rechts: Player
spiel_objekte: pygame.sprite.Group()
spiel_objekte.add()

Bälle, Spieler links und Spieler rechts hinzufügen

Initalisierung __init__

Parameter:

main_window, ball_count=1

'ball_count' gibt an, wie viele Bälle auf dem Spielfeld generiert werden sollen

Aktoren des Pong Spiels - Codeaufbau

```
class Application
    def __init__(self, window_width, window_height, caption)
    def root_path() # Statische Methode
class Player(pygame.sprite.Sprite)
    def position(self, move_px) # Bewegt den Schläger pro Pixel
class Ball(pygame.sprite.Sprite)
    def __init__(self, background_color)
    def _generate_ball_color(self)
    def first_serve(self) # Erster Aufschlag
    def move(self) # Bewegt den Ball pro 'tick'
    def remove_from_match(self) # Entfernt den Ball, falls
                                mehrere Bälle im Spiel sind
class Matchfield
    (...)
```

Bibliotheken

```
from enum import Enum,
unique, auto
import random
import os
import pathlib
import pygame
from pygame.locals import *
from dataclasses import
dataclass, field
import numpy as np
```