Python Programmieren

Dodger Game

Jugend Hackt

Was ist das Dodger Game?

Wir programmieren gemeinsam ein Arcade-Spiel, bei dem ihr:

- Einen Spieler nach **links und rechts** steuert
- **Hindernissen ausweicht**, die von oben fallen
- Versucht, so lange wie möglich zu überleben
- Euren **Highscore** verbessert!

Was ist programmieren?

- Dem Computer sagen, was er tun soll 💻
- **Programmieren** = Eine Sprache sprechen, die der Computer versteht
- **Code** = Schritt-für-Schritt Anweisungen für den Computer
- **Python** = Eine weit verbreitete Programmiersprache

Gemeinsames Setup

1. Repository herunterladen

```
git clone https://github.com/kadir5858/dodger-game.git
cd dodger-game
```

2. Abhängigkeiten installieren

```
pip install -r requirements.txt
```

3. Spiel starten

```
cd src
python main.py
```

Grundkonzept 1 - Variablen

- Variablen = Boxen mit Namen 📦
- Eine Variable ist wie eine beschriftete Box, in der wir Dinge aufbewahren.

Codebeispiel:

```
spieler_name = "Max"
punkte = 100
geschwindigkeit = 7
```

Aus dem Spiel:

```
BILDSCHIRM_BREITE = 800   # Wie breit ist das Fenster?
SPIELER_GESCHWINDIGKEIT = 7  # Wie schnell bewegt sich der Spieler?
```

Grundkonzept 2 - Datentypen

Die wichtigsten Datentypen:

Datentyp	Beschreibung	Beispiel
Integer (int)	Ganze Zahlen	`punkte = 100`
Float	Kommazahlen	`geschwindigkeit = 3.5`
String (str)	Text	`name = "Max"`
Boolean (bool)	Wahr/Falsch	`hat_schild = True`
Liste (list)	Sammlung von Dingen	`hindernisse = []`

Grundkonzept 3 - Dictionaries

- Dictionaries = Steckbriefe 📋
- Ein Dictionary speichert zusammengehörige Informationen mit Schlüssel-Wert-Paaren.

Aus dem Spiel :

```
spieler = {
    "x": 400,              # Position horizontal
    "y": 500,              # Position vertikal
    "breite": 48,          # Wie breit ist der Spieler?
    "hoehe": 48,           # Wie hoch ist der Spieler?
    "geschwindigkeit": 7,  # Wie schnell bewegt er sich?
    "hat_schild": False    # Hat er ein Schild?
}
```

So greifen wir auf Werte zu:

```
spieler["x"]              # Gibt 400 zurück
spieler["hat_schild"]     # Gibt False zurück
```

Grundkonzept 4 – If-Bedingung

- Wenn… Dann… 🔀
- Mit Bedingungen kann der Computer Entscheidungen treffen.

Struktur:

```
if bedingung:
    # Mache das, wenn die Bedingung wahr ist
else:
    # Mache das, wenn die Bedingung falsch ist
```

Aus dem Spiel:

```
if powerup_typ == "shield":
    schild_aktivieren(spieler)
elif powerup_typ == "slow":
    slow_motion_aktivieren(spiel_status)
```

Grundkonzept 5 - Schleifen

- Für Wiederholungen 🔁
- Zwei Arten von Schleifen:

FOR-Schleife - Wiederhole X mal:

```
for hindernis in hindernisse:
    if hindernis["y"] < BILDSCHIRM_HOEHE:
        sichtbare_hindernisse.append(hindernis)
```

WHILE-Schleife - Wiederhole solange Bedingung wahr:

```
while laeuft:
    # Spiel läuft weiter
    # Zeichne alles
    # Prüfe Eingaben
```

Grundkonzept 6 - Funktionen

- Wiederverwendbare Code-Blöcke 🧩
- Funktionen sind wie Rezepte - einmal schreiben, beliebig oft verwenden!

Struktur:

```
def funktions_name(parameter):
    # Tu etwas
    return ergebnis
```

Aus dem Spiel:

```
def spieler_erstellen():
    spieler = {
        "x": 400,
        "y": 500,
        "geschwindigkeit": 7
    }
    return spieler
```

Aufrufen der Funktion:

```
mein_spieler = spieler_erstellen()
```

Grundkonzept 7 – Listen

- Sammlungen von Dingen 📝
- Listen speichern mehrere Elemente in einer geordneten Reihenfolge.

Codebeispiel:

```
liste = [1, 2, "Hallo", True]

liste[0]    # 1
liste[2]    # "Hallo"
liste[-1]   # True (letztes Element)
```

Listen-Operationen:

```
liste.append(element)   # Element hinzufügen
liste.pop(index)        # Element entfernen
len(liste)              # Anzahl der Elemente
liste[0]                # Erstes Element holen
```

Aus dem Spiel:

```
hindernisse = []  # Leere Liste am Anfang
# Neues Hindernis hinzufügen
hindernisse.append(neues_hindernis)
# Durch alle Hindernisse gehen
for hindernis in hindernisse:
    hindernis["geschwindigkeit"] = 5  # Bewege nach unten
```

Grundkonzept 8 – Modulo Operator

- Timing im Spiel ⏱️
- Das Problem: Wie spawnen wir alle 45 Frames ein neues Hindernis?

Die Lösung - Der Modulo-Operator (%):

Was macht %?

- 45 % 45 = 0 ✓ (Hindernis spawnen!)
- 46 % 45 = 1 ✗
- 47 % 45 = 2 ✗
- ...
- 90 % 45 = 0 ✓ (Wieder spawnen!)

Grundkonzept 9 - Imports

- Code von anderen nutzen 🧩
- Erklärung: Mit import laden wir fertigen Code von anderen, damit wir nicht alles selbst schreiben müssen.

Wichtige Imports im Spiel:

```
import pygame       # Für Grafik und Spiele
import random       # Für Zufallszahlen
import time         # Für Zeit-Funktionen
import sys          # Für System-Funktionen
```

Eigene Programmierdateien importieren:

```
from config import BILDSCHIRM_BREITE
from player import spieler_erstellen
from obstacles import kollision_pruefen
```

Die Spielschleife

- Das Herz jedes Spiels ❤️

Struktur:

SPIELSCHLEIFE

1. 🎮 Events prüfen (Tastatur, etc.)
2. 🧠 Spiellogik berechnen
3. 🎨 Alles auf Bildschirm zeichnen
4. 🔄 Bildschirm aktualisieren
5. ⏱️ Auf nächstes Frame warten

└────── Wiederholen! ──────┘

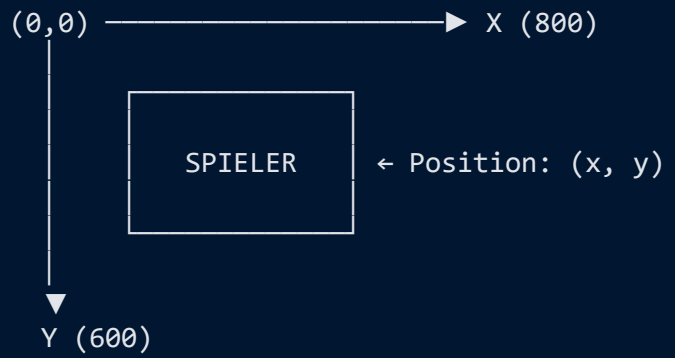
Aus dem Spiel:

```
# HAUPTSCHLEIFE - Läuft bis das Spiel endet
laeuft = True
while laeuft:
    # EVENTS VERARBEITEN
    for event in pygame.event.get():
        ...
```

Koordinatensystem in Pygame

- Wo ist was auf dem Bildschirm? 📍

Wichtig zu wissen:



Besonderheit:

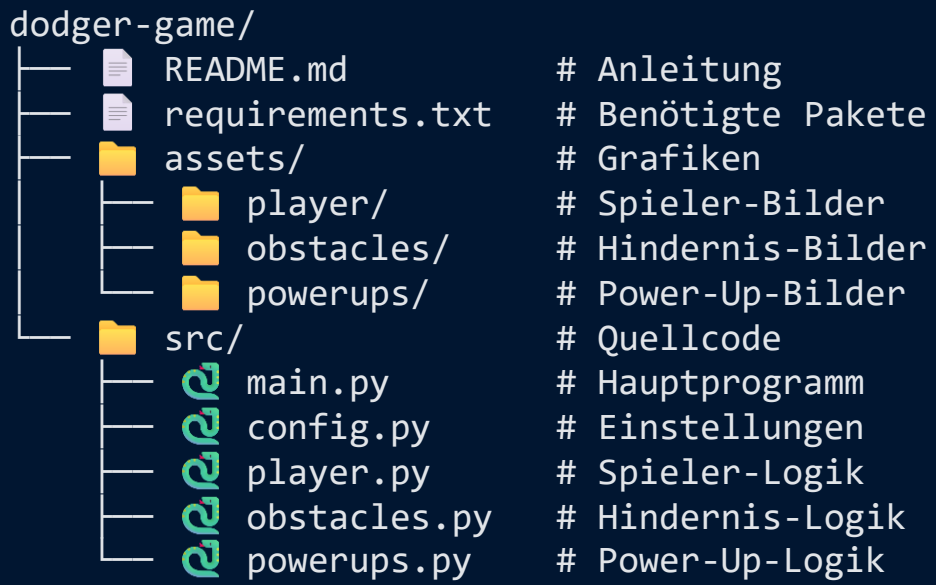
- (0, 0) ist OBEN LINKS (nicht unten links wie in Mathe!)
- X geht nach RECHTS (größer = weiter rechts)
- Y geht nach UNTEN (größer = weiter unten)

Aus dem Spiel:

```
BILDSCHIRM_BREITE = 800    # X geht von 0 bis 800
BILDSCHIRM_HOEHE = 600     # Y geht von 0 bis 600
spieler["x"] = 400         # Mitte horizontal
spieler["y"] = 500         # Unten (nah am Boden)
```

Projektstruktur

So ist unser Projekt aufgebaut:



Eure Aufgaben (TODOS)

TODO 1: Spielerbewegung 🏃

- Links und rechts bewegen mit Pfeiltasten
- Im Bildschirm bleiben

TODO 2: Hindernisse spawnen 🧱

- Neue Hindernisse erscheinen lassen
- Timing mit Modulo-Operator

TODO 3: Kollisionserkennung 💥

- Prüfen ob Spieler ein Hindernis berührt
- Game Over bei Kollision

TODO 4: Score-System 🏆

- Punkte pro Sekunde
- Zeit-basierte Erhöhung

BONUS: Power-Ups ⚡

- Schild aktivieren
- Slow-Motion aktivieren

Tipps

Wenn ihr nicht weiterkommt:

- 🔍 Lest die Kommentare im Code - da stehen Tipps!
- 🗣️ Fragt eure Nachbarn oder die Mentor*innen
- 🧪 Probiert Dinge aus - Fehler sind zum Lernen da!