Aufgabenblatt 03

Heute entwickelt ihr mit PyGame Zero ein eigenes Weltraumspiel. Ihr steuert ein Raumschiff am linken Fensterrand mit den Pfeiltasten nach oben und unten. Von rechts kommen Asteroiden auf euch zu. Weicht ihnen aus oder schießt sie mit einem Laser ab. Mit der Leertaste feuert ihr; jeder abgeschossene Asteroid gibt einen Punkt. Euer Schiff hat drei Leben und verliert bei jeder Kollision eines. Bei null Leben wird der Punktestand ausgegeben und das Spiel beendet.

Verwendet dabei die Konzepte aus der Vorlesung. Arbeitet gerne zu zweit, probiert Zwischenschritte regelmäßig aus und besprecht Fehlermeldungen miteinander.

1. Vorbereitung

1. Erstellt ein Verzeichnis `weltraumspiel` für euer Projekt.
2. Wechselt im Terminal in das Verzeichnis und erstellt eine leere `game.py`.
3. Startet sie mit `pgzrun game.py`. Es sollte ein leeres Fenster erscheinen.
4. Öffnet `game.py` in einem Editor.
5. Erstellt ein kurzes Entwicklungslogbuch. Notiert nach jedem größeren Schritt, was funktioniert, was noch fehlt und welchen Fehler ihr zuletzt behoben habt.

Bevor ihr programmiert, beschreibt den Game Loop eures Spiels mit eigenen Worten: Welche Eingaben gibt es? Welcher Zustand verändert sich? Was wird gezeichnet? Wann ist das Spiel vorbei?

2. Das Raumschiff

Erstellt zunächst ein Rechteck als Platzhalter für das Raumschiff. Es soll nahe am linken Fensterrand bleiben und mit den Pfeiltasten nach oben und unten bewegt werden können.

1. Welche Werte beschreiben Position und Größe des Schiffes?
2. In welcher Funktion fragt ihr eine dauerhaft gehaltene Taste ab?
3. Warum sollte die Bewegung mit `dt` multipliziert werden?
4. Wie verhindert ihr, dass das Schiff das Fenster verlässt?

Testet auch sehr große und sehr kleine Geschwindigkeiten. Ersetzt das Rechteck erst später durch einen `Actor("raumschiff")` mit einer eigenen Grafik.

3. Asteroiden

Asteroiden sollen rechts auf einer zufälligen Höhe erscheinen und nach links fliegen. Beginnt mit einem Asteroiden und erweitert den Zustand anschließend zu einer Liste.

1. Erstellt eine Funktion, die einen neuen Asteroiden erzeugt.
2. Verwendet `randint`, um seine y-Koordinate auszuwählen.
3. Ruft die Funktion mit `clock.schedule_interval` regelmäßig auf. Lest dazu den Abschnitt Clock in der PyGame-Zero-Dokumentation.
4. Bewegt in `update(dt)` alle Asteroiden nach links.
5. Entfernt Asteroiden, sobald sie vollständig links aus dem Fenster geflogen sind.

Verändert eine Liste nicht unüberlegt, während ihr über genau diese Liste iteriert. Ihr könnt beispielsweise über `asteroids.copy()` laufen.

4. Kollisionen und Leben

Das Schiff startet mit drei Leben. Berührt ein Asteroid das Schiff, wird der Asteroid entfernt und ein Leben abgezogen. Bei null Leben beendet sich das Spiel.

1. Findet heraus, wie `Rect` beziehungsweise `Actor` mit `colliderect` auf eine Überschneidung geprüft werden.
2. Zeigt die verbleibenden Leben mit `screen.draw.text` an.
3. Beendet das Spiel bei null Leben mit `raise SystemExit`.

5. Laser

Mit der Leertaste feuert das Schiff. Ihr müsst dafür keine Projektile über mehrere Frames verwalten: Beim Tastendruck entsteht ein schmales, unsichtbares Rechteck vom Bug des Schiffs bis zum rechten Fensterrand.

1. Findet alle Asteroiden, die dieses Rechteck schneiden.
2. Falls mehrere getroffen werden, zerstört nur den Asteroiden mit der kleinsten x-Koordinate – also den nächsten vor dem Schiff.
3. Erhöht für einen Treffer den Punktestand.
4. Zeichnet den Laserstrahl für ungefähr `0.1` Sekunden und blendet ihn danach mit `clock.schedule_unique` wieder aus.
5. Spielt beim Feuern und beim Treffer unterschiedliche Töne.

Prüft besonders diese Fälle: kein Asteroid im Strahl, mehrere Asteroiden im Strahl und ein Asteroid knapp ober- oder unterhalb des Strahls.

6. Bilder und Feinschliff

Erstellt einfache Bilder für das Raumschiff und die Asteroiden, oder lade welche aus dem Internet herunter. Legt neben `game.py` ein Verzeichnis `images/` an und speichert die Bilder dort als `raumschiff.png` und `asteroid.png`. Ersetzt die Platzhalter durch Actors. PyGame Zero verwendet beim Laden nur den Namen ohne Dateiendung.

1. Ergänzt Punkte- und Lebensanzeige.
2. Gebt dem Spiel einen Hintergrund und gut unterscheidbare Farben.

3. Gebt vor dem Beenden den Punktestand mit `print()` aus.
4. Prüft einen vollständigen Ablauf vom Start bis zum Verlust aller drei Leben.
5. Lasst eine andere Gruppe spielen, ohne den Code zu erklären. Beobachtet, wo Regeln oder Rückmeldungen unklar sind.

7. Bonusaufgaben

1. Erhöht Geschwindigkeit oder Spawnrate mit wachsendem Punktestand.
2. Gebt Asteroiden zufällige Größen oder Geschwindigkeiten.
3. Baut ein Schutzschild mit begrenzter Dauer ein.
4. Führt eine Combo ein, die nach einer Kollision zurückgesetzt wird.
5. Speichert einen Highscore in einer Datei.
6. Erstellt einen cool-down zum Abfeuern des Lasers.
7. Ersetzt den geraden Laser durch sichtbare Projektile. Welche zusätzlichen Zustände und Kollisionsprüfungen benötigt ihr jetzt?
8. Entwerft eigene Bilder oder ein zweites Level.
9. Lasst das Schiff sich auch nach links und rechts bewegen.

8. Nützliche PyGame-Zero-Bausteine

```
def update(dt): ...
def draw(): ...
def on_key_down(key): ...

keyboard.up
keys.SPACE
Actor("raumschiff", pos=(x, y))
rechteck.colliderect(anderes_rechteck)
clock.schedule_interval(funktion, sekunden)
clock.schedule_unique(funktion, sekunden)
tone.play("A5", 0.1)
```