Sabir Yusuf, G4A

# Spider Invasion

Bei diesem Projekt ist man in der Rolle eines Jägers, welcher ausschliesslich vertikal und horizontal bewegt werden kann. Dieser hat eine limitierte Magazingrösse, welche nur durch das Stehenbleiben nachgeladen werden kann. Die Aufgabe besteht darin, Spinnen zu erschiessen, welche aus allen Himmelsrichtungen den Spieler verfolgen. Diese Spinnen sind jedoch in der Lage, sich auch diagonal zu bewegen, was ihnen einen Vorteil beim Fangen des Spielers bietet. Wird der Jäger von einer Spinne berührt, so erhält der Jäger Schaden und seine Lebensanzeige sinkt. Der Sinn des Spiels ist es, möglichst lange zu überleben und dabei viele Spinnen zu töten. Die Lebensanzeige kann nicht regeneriert werden.

## Notwendigkeiten

Um dieses Projekt zu starten sind keine speziellen Anforderungen nötig. Alles wurde bereits im Code importiert.

## Anforderungen

**Der Spieler kann in einer 2D-Welt horizontal und vertikal bewegt werden.**

2D-Welt

```
#Bildschirm
screen = pygame.display.set_mode((SCREEN_WIDTH,SCREEN_HEIGHT))
pygame.display.set_caption("Projektarbeit")

#Hintergrund
hintergrund_bild = pygame.image.load("Game/assets/hintergrund.jpg")
hintergrund = pygame.transform.scale(hintergrund_bild, (SCREEN_WIDTH, SCREEN_HEIGHT))
```

Horizontal und vertikale Bewegung

```
if eingabe[pygame.K_a] and self.rect.x > 0:
    self.velx = -self.tempo
    self.vely = 0
    self.face_left = True
    self.face_right = False
    self.face_up = False
    self.face_down = False
if eingabe[pygame.K_d] and self.rect.x < SCREEN_WIDTH - 50:
    self.velx = self.tempo
    self.vely = 0
    self.face_left = False
    self.face_right = True
    self.face_up = False
    self.face_down = False
if eingabe[pygame.K_w] and self.rect.y != 0:
    self.velx = 0
    self.vely = -self.tempo
    self.face_left = False
    self.face_right = False
    self.face_up = True
    self.face_down = False
if eingabe[pygame.K_s] and self.rect.y < SCREEN_HEIGHT - 50:
    self.velx = 0
    self.vely = self.tempo
    self.face_left = False
    self.face_right = False
    self.face_up = False
    self.face_down = True

self.rect.x += self.velx
self.rect.y += self.vely
```

**Gegner verfolgen den Spieler horizontal, vertikal und diagonal.**

Spielerverfolgung

```python
#Spinne Angriff
def move(self, player):
    self.dx = player.rect.x - self.rect.left
    self.dy = player.rect.y - self.rect.top
    dist = math.hypot(self.dx, self.dy)
    if dist > 0:
      self.dx, self.dy = self.dx / dist, self.dy / dist
    else:
        self.dx = 0
        self.dy = 0
```

**Der Spieler soll mit einer Waffe die Gegner besiegen, jedoch hat die Waffe ein limitiertes Magazin. Um diese nachzuladen, muss man stehenbleiben, was gefährlich ist, da die Gegner dennoch den Spieler verfolgen.**

Limitierte Schüsse und Nachladung beim Stehenbleiben

```python
#Spieler Schuss
def shoot(self):
    self.cooldown()
    self.move_player(eingabe)
    if eingabe[pygame.K_SPACE] and self.cooldown_dauer == 0 and self.schuss > 0:
        bullet = Bullet(self.rect.x, self.rect.y, self.direction())
        self.bullets.append(bullet)
        bullets_group.add(bullet)
        self.cooldown_dauer = 1
        self.schuss -= 1

#Waffe nachladen
    if self.velx == 0 and self.velx == 0:
        if leben > 3000:
            self.schuss += 1/60
        else:
            self.schuss += 1/30

    if self.schuss > 20:
        self.schuss = 20
```

# Reflexion 1

Sprite Animation

Wir haben gelernt, ein einzelnes Bild auf den Bildschirm zu zeichnen, jedoch keine Animation. Aus diesem Grund habe ich durch eigene Recherche gelernt, wie das geht.

```python
    self.vorne = [pygame.image.load("Game/assets/spinne/spinne_vorne (1).jpg").convert(), pygame.image.load("Game/assets/spinne/spinne_vorne (2).jpg").convert(),
    pygame.image.load("Game/assets/spinne/spinne_vorne (3).jpg").convert(), pygame.image.load("Game/assets/spinne/spinne_vorne (4).jpg").convert(),
    pygame.image.load("Game/assets/spinne/spinne_vorne (5).jpg").convert(), pygame.image.load("Game/assets/spinne/spinne_vorne (6).jpg").convert(),
    pygame.image.load("Game/assets/spinne/spinne_vorne (7).jpg").convert(), pygame.image.load("Game/assets/spinne/spinne_vorne (8).jpg").convert(),
    pygame.image.load("Game/assets/spinne/spinne_vorne (9).jpg").convert(), pygame.image.load("Game/assets/spinne/spinne_vorne (10).jpg").convert(),
    pygame.image.load("Game/assets/spinne/spinne_vorne (11).jpg").convert(), pygame.image.load("Game/assets/spinne/spinne_vorne (12).jpg").convert()]

    self.vorne_scaled_list = []

    for element in self.vorne:
        element.set_colorkey((95, 73, 59), RLEACCEL)
        self.vorne_scaled = pygame.transform.scale(element, (50,50))
        self.rect = self.vorne_scaled.get_rect()
        self.vorne_scaled_list.append(self.vorne_scaled)
```

Man platziert jeden Frame in eine Liste. Dieser Prozess musste in meinem Fall für jede Himmelsrichtung separat erstellt werden. Anschliessend werden die Frames angepasst und in eine neue Liste eingesetzt, so wie sie im Spiel angezeigt werden sollen.

```python
#Spinne Animation
def step(self):
    if self.stepIndex >= 120:
        self.stepIndex = 0

#Spinne zeichnen
def draw(self, screen):
    self.step()
    self.direction()

    if self.face_up:
        screen.blit(self.vorne_scaled_list[self.stepIndex//10], self.rect)
    if self.face_down:
        screen.blit(self.runter_scaled_list[self.stepIndex//10], self.rect)
    if self.face_left:
        screen.blit(self.links_scaled_list[self.stepIndex//10], self.rect)
    if self.face_right:
        screen.blit(self.rechts_scaled_list[self.stepIndex//10], self.rect)

    self.stepIndex += 1
```

Um die Frames nacheinander anzuzeigen, wurde ein Schrittmacher eingebaut. Dieser steigt mit der Zeit und zeichnet immer den nächsten Frame. Die Geschwindigkeit, wie schnell die Frames gewechselt werden sollen, können dadurch angepasst werden, wie hoch der Schrittmacher gesetzt ist, bis dieser sich wieder auf den ersten Frame zurücksetzt.

## Schönes Programmieren mit OOP

Bei einem grösseren Projekt kann man schnell den Faden verlieren, arbeitet man unsauber, versteht man den eigenen Code nicht mehr. Kommentare und klare Funktionen schaffen Abhilfe. Müssen Anpassungen getätigt oder Fehler ausfindig gemacht werden, weiss man genau, wo zu suchen ist.

```python
#Spieler
class Player:
    def __init__(self, x, y): ···

    #Spieler Steuerung & Blickrichtunganpassung
    def move_player(self, eingabe): ···

    #Spieler auf den Bildschirm zeichnen
    def draw(self, screen): ···

    #Blickrichtung
    def direction(self): ···

    #Waffencooldown
    def cooldown(self): ···

    #Spieler Schuss
    def shoot(self): ···

#Bullet
class Bullet(pygame.sprite.Sprite):
    def __init__(self, x, y, direction): ···


    #Bullet Zeichnung
    def draw_bullet(self): ···

    #Bullet Movement
    def move(self): ···

    #Bullet Offscreen löschen
    def off_screen(self): ···
```

Gruppenzuweisung und dessen Notwendigkeit

```
#Kollision zwischen Spinne und Schuss
bullet_collision = pygame.sprite.groupcollide(mobs, bullets_group, True, True)

#Kollision zwischen Spieler und Spinne
player_collision = pygame.sprite.spritecollide(player, mobs, False)
```

Während des ganzen Projektes wurden die Klassen keiner Gruppe zugeordnet, dies musste dringend angepasst werden, um die Kollisionsfunktionen von PyGame nutzen zu können. Ausserdem wurde ausschliesslich mit Surfaces gearbeitet, welche nicht in der Lage sind, zu kollidieren. Aus diesem Grund mussten sehr viele Zeilen die .get_rect() Funktion erhalten.

```
    self.rect = self.aussehen_skaliert.get_rect()
```

# Reflexion 2

Ich kann Ein- und Ausgaben in einer Programmiersprache realisieren.

In diesem Abschnitt des Codes wurde die Bewegung des Spielers programmiert. Die Variable «eingabe» beinhaltet die Funktion pygame.key.get_pressed(), womit die gedruckten Tasten wahrgenommen werden. In meinem Code ist die Ausgabe der Bewegungsmuster des Charakters und dessen Blickrichtung, welche an einem anderen Ort programmiert wurden.

```
    def move_player(self, eingabe):

        self.velx = 0
        self.vely = 0

        if eingabe[pygame.K_a] and self.rect.x > 0:
            self.velx = -self.tempo
            self.vely = 0
            self.face_left = True
            self.face_right = False
            self.face_up = False
            self.face_down = False
        if eingabe[pygame.K_d] and self.rect.x < SCREEN_WIDTH - 50:
            self.velx = self.tempo
            self.vely = 0
            self.face_left = False
            self.face_right = True
            self.face_up = False
            self.face_down = False
        if eingabe[pygame.K_w] and self.rect.y != 0:
            self.velx = 0
            self.vely = -self.tempo
            self.face_left = False
            self.face_right = False
            self.face_up = True
            self.face_down = False
        if eingabe[pygame.K_s] and self.rect.y < SCREEN_HEIGHT - 50:
            self.velx = 0
            self.vely = self.tempo
            self.face_left = False
            self.face_right = False
            self.face_up = False
            self.face_down = True
```

# Quellen von Programmcode

Spielerverfolgung von den Spinnen

https://stackoverflow.com/questions/20044791/how-to-make-an-enemy-follow-the-player-in-pygame

Folgender Abschnitt wurde als Grundlage übernommen und im eigenen Code in angepasster Form eingebaut:

```
def move_towards_player(self, player):
    # Find direction vector (dx, dy) between enemy and player.
    dx, dy = player.rect.x - self.rect.x, player.rect.y - self.rect.y
    dist = math.hypot(dx, dy)
    dx, dy = dx / dist, dy / dist  # Normalize.
    # Move along this normalized vector towards the player at current speed.
    self.rect.x += dx * self.speed
    self.rect.y += dy * self.speed
```

```
#Spinne Angriff
def move(self, player):
    self.dx = player.rect.x - self.rect.left
    self.dy = player.rect.y - self.rect.top
    dist = math.hypot(self.dx, self.dy)
    if dist > 0:
      self.dx, self.dy = self.dx / dist, self.dy / dist
    else:
        self.dx = 0
        self.dy = 0
```

Animation
https://www.techwithtim.net/tutorials/game-development-with-python/pygame-tutorial/pygame-animation/

Dieses Prinzip für die Animierung inklusive stepIndex von der Spinne wurden übernommen.

```
walkRight = 
[pygame.image.load('R1.png'), pygame.image.load('R2.png'),
pygame.image.load('R3.png'), pygame.image.load('R4.png'),
pygame.image.load('R5.png'), pygame.image.load('R6.png'),
pygame.image.load('R7.png'), pygame.image.load('R8.png'),
pygame.image.load('R9.png')]
```

```
def __init__(self, x, y):
    pygame.sprite.Sprite.__init__(self)
    self.vorne = [pygame.image.load("Game/assets/spinne/spinne_vorne (1).jpg").convert(), pygame.image.load("Game/assets/spinne/spinne_vorne (2).jpg").convert(),
    pygame.image.load("Game/assets/spinne/spinne_vorne (3).jpg").convert(), pygame.image.load("Game/assets/spinne/spinne_vorne (4).jpg").convert(),
    pygame.image.load("Game/assets/spinne/spinne_vorne (5).jpg").convert(), pygame.image.load("Game/assets/spinne/spinne_vorne (6).jpg").convert(),
    pygame.image.load("Game/assets/spinne/spinne_vorne (7).jpg").convert(), pygame.image.load("Game/assets/spinne/spinne_vorne (8).jpg").convert(),
    pygame.image.load("Game/assets/spinne/spinne_vorne (9).jpg").convert(), pygame.image.load("Game/assets/spinne/spinne_vorne (10).jpg").convert(),
    pygame.image.load("Game/assets/spinne/spinne_vorne (11).jpg").convert(), pygame.image.load("Game/assets/spinne/spinne_vorne (12).jpg").convert()]
```