Pygame Zero auf Pydroid - Teil 2: Bilder und Sound! 🎨🔊

Super! Du hast Teil 1 geschafft und kannst jetzt einfache Spiele mit Formen machen. In diesem Teil lernst du, wie du deine Spiele mit **echten Bildern** und **Sounds** noch cooler machst!

Was du am Ende kannst:

- Bilder statt Formen in deinen Spielen benutzen
- Sounds abspielen wenn etwas passiert
- Animationen erstellen
- Deine Spiele professioneller aussehen lassen

Was du vorher können musst:

- Teil 1 durchgearbeitet haben
- Ein funktionierendes Spiel (z.B. Flappy Box) haben

Kapitel 1: Vorbereitung - Ordner erstellen

Ziel dieses Kapitels

Du erstellst die Ordnerstruktur die Pygame Zero braucht um Bilder und Sounds zu finden.

Warum brauchen wir spezielle Ordner?

Pygame Zero sucht Bilder und Sounds an bestimmten Stellen. Wenn du sie woanders hinlegst, findet das Programm sie nicht!

So muss dein Projekt aussehen

```
MeinSpiel/              ← Dein Hauptordner
├── meinspiel.py        ← Dein Python-Code
├── images/             ← Ordner für BILDER
│   ├── spieler.png
│   └── feind.png
└── sounds/             ← Ordner für SOUNDS
    ├── sprung.wav
    └── punkt.wav
```

Schritt-für-Schritt: Ordner erstellen

Schritt 1: Öffne den Datei-Manager auf deinem Handy

Schritt 2: Finde den Ordner wo dein Spiel gespeichert ist

- Meistens unter: Interner Speicher → pydroid3

Schritt 3: Erstelle einen neuen Ordner namens `images`

- WICHTIG: Kleingeschrieben!
- Keine Großbuchstaben!

Schritt 4: Erstelle einen neuen Ordner namens `sounds`

- Wieder: Kleingeschrieben!

Fertig! Jetzt kannst du Bilder und Sounds hinzufügen.

Wichtige Regeln für Dateinamen

✅ Richtig	❌ Falsch
`spieler.png`	`Spieler.png` (Großbuchstabe!)
`mein_vogel.png`	`mein vogel.png` (Leerzeichen!)
`feind1.png`	`feind 1.png` (Leerzeichen!)
`ball.png`	`ball.PNG` (Großbuchstaben!)

Merke dir:

- Nur **Kleinbuchstaben**
- Keine **Leerzeichen** (benutze _ stattdessen)
- Keine **Sonderzeichen** wie ä, ö, ü

Kapitel 2: Bilder finden oder erstellen

Ziel dieses Kapitels

Du findest oder erstellst Bilder für deine Spiele.

Woher bekomme ich Bilder?

Option 1: Kostenlose Bilder herunterladen

Diese Webseiten haben kostenlose Spielgrafiken:

1. **Kenney.nl** (am besten für Anfänger!)
    - Viele einfache, schöne Bilder
    - Alles kostenlos
    - Suche nach "Kenney game assets"
2. **OpenGameArt.org**
    - Sehr viele Bilder
    - Verschiedene Stile
    - Achte auf die Lizenz!

Option 2: Selbst malen

Es gibt Apps um Pixel-Bilder zu malen:

1. **Pixel Studio** (kostenlos im Play Store)
    - Einfach zu benutzen
    - Perfekt für kleine Spielfiguren
2. **Dotpict** (kostenlos)
    - Sehr einfach
    - Gut für Anfänger

Welche Größe sollten Bilder haben?

Was	Empfohlene Größe
Spielfigur	32x32 oder 64x64 Pixel
Feind	32x32 oder 64x64 Pixel
Münze/Item	16x16 oder 32x32 Pixel
Hintergrund	400x600 Pixel (ganzer Bildschirm)

Tipp: Kleinere Bilder sind besser! Große Bilder machen das Spiel langsam.

Schritt-für-Schritt: Ein Bild herunterladen

Ziel: Ein Vogel-Bild für Flappy Bird herunterladen.

Schritt 1: Öffne den Browser auf deinem Handy

Schritt 2: Suche nach "kenney bird sprite png"

Schritt 3: Finde ein Bild das dir gefällt

Schritt 4: Tippe lange auf das Bild → "Bild speichern"

Schritt 5: Benenne es um zu `vogel.png` (kleingeschrieben!)

Schritt 6: Verschiebe es in deinen `images/` Ordner

Kapitel 3: Actors - Spielfiguren mit Bildern

Ziel dieses Kapitels

Du lernst wie man Bilder als Spielfiguren benutzt.

Was ist ein Actor?

In Pygame Zero heißen Spielfiguren mit Bildern **"Actors"** (englisch für "Schauspieler"). Ein Actor ist wie ein Rechteck, aber mit einem Bild drauf!

Vergleich: Rechteck vs. Actor

Vorher (mit Rechteck):

```
spieler_x = 100
spieler_y = 300

def draw():
    screen.draw.filled_rect(Rect(spieler_x, spieler_y, 40, 40), "red")
```

Nachher (mit Actor):

```
spieler = Actor("spieler")  # Lädt images/spieler.png
spieler.x = 100
spieler.y = 300

def draw():
    spieler.draw()
```

Was ist besser am Actor?

- Zeigt ein echtes Bild
- Einfacher zu bewegen
- Hat eingebaute Kollisionserkennung

Schritt-für-Schritt: Deinen ersten Actor erstellen

Ziel: Ein Bild auf dem Bildschirm anzeigen.

Vorbereitung: Du brauchst ein Bild namens `spieler.png` im `images/` Ordner.

Schritt 1: Actor erstellen

```
# Der Name in Klammern ist der Dateiname OHNE .png
spieler = Actor("spieler")
```

Schritt 2: Position setzen

```
spieler.x = 200  # Horizontal (links/rechts)
spieler.y = 300  # Vertikal (oben/unten)
```

Schritt 3: Actor zeichnen

```
def draw():
    screen.fill("skyblue")
    spieler.draw()  # Das ist alles!
```

Komplettes Beispiel

```
# === MEIN ERSTER ACTOR ===

WIDTH = 400
HEIGHT = 600

# Actor erstellen
# WICHTIG: "spieler" bedeutet → images/spieler.png muss existieren!
spieler = Actor("spieler")
spieler.x = 200
spieler.y = 300

def draw():
    screen.fill("skyblue")
    spieler.draw()

def update():
    # Actor nach rechts bewegen
    spieler.x = spieler.x + 2

    # Wenn rechts raus, links wieder rein
    if spieler.x > WIDTH + 50:
        spieler.x = -50
```

```
def on_mouse_down(pos):
    # Actor springt zu Fingerposition
    spieler.x = pos[0]
    spieler.y = pos[1]
```

Was du mit Actors machen kannst

Was	Wie	Beispiel
Position ändern	`.x` und `.y`	`spieler.x = 100`
Bild wechseln	`.image`	`spieler.image = "spieler_springt"`
Drehen	`.angle`	`spieler.angle = 45`
Größe ändern	`.scale`	`spieler.scale = 2`
Breite abfragen	`.width`	`print(spieler.width)`
Höhe abfragen	`.height`	`print(spieler.height)`

Bild wechseln (für Animationen)

Ziel: Der Spieler sieht anders aus wenn er springt.

```
def on_mouse_down(pos):
    spieler.image = "spieler_springt"  # Bild wechseln

def update():
    # ... wenn auf dem Boden ...
    spieler.image = "spieler"  # Zurück zum normalen Bild
```

Du brauchst dafür:

- `images/spieler.png` (normales Bild)
- `images/spieler_springt.png` (Sprung-Bild)

Kapitel 4: Kollisionen mit Actors

Ziel dieses Kapitels

Du lernst wie man prüft ob sich zwei Actors berühren.

Warum ist das einfacher als vorher?

Vorher (Teil 1) - Kompliziert:

```
# Manuell prüfen ob sich Rechtecke überlappen
if spieler_x + spieler_breite > feind_x and spieler_x < feind_x + feind_breite:
    if spieler_y + spieler_hoehe > feind_y and spieler_y < feind_y + feind_hoehe:
        print("Getroffen!")
```

Mit Actors - Einfach:

```
# Ein Befehl!
if spieler.colliderect(feind):
    print("Getroffen!")
```

Die wichtigsten Kollisions-Befehle

1. colliderect() - Berührt einen anderen Actor?

```
if spieler.colliderect(feind):
    print("Spieler berührt Feind!")
```

2. collidepoint() - Wurde angeklickt/getippt?

```
def on_mouse_down(pos):
    if spieler.collidepoint(pos):
        print("Spieler wurde angetippt!")
```

Beispiel: Münzen sammeln

Ziel: Wenn der Spieler eine Münze berührt, verschwindet sie.

```
WIDTH = 400
HEIGHT = 600

spieler = Actor("spieler")
spieler.pos = (200, 500)

muenze = Actor("muenze")
muenze.pos = (200, 200)

punkte = 0
muenze_da = True

def draw():
    screen.fill("darkgreen")
    spieler.draw()
    if muenze_da:
        muenze.draw()
    screen.draw.text(f"Punkte: {punkte}", (10, 10), color="white", fontsize=30)

def update():
    global punkte, muenze_da

    # Prüfen ob Spieler Münze berührt
    if muenze_da and spieler.colliderect(muenze):
        punkte = punkte + 1
        muenze_da = False
        # Hier könnte man auch: sounds.muenze.play()

def on_mouse_down(pos):
```

```
    spieler.x = pos[0]
    spieler.y = pos[1]
```

Kapitel 5: Sounds hinzufügen

Ziel dieses Kapitels

Du lernst wie man Sounds in dein Spiel einbaut.

Sound-Dateien vorbereiten

Welche Formate funktionieren?

- `.wav` (am besten!)
- `.ogg` (auch gut)

MP3 funktioniert NICHT immer! Konvertiere MP3 zu WAV.

Woher bekomme ich Sounds?

1. **Freesound.org** - Viele kostenlose Sounds
2. **OpenGameArt.org** - Auch Sounds für Spiele
3. **Selber aufnehmen** - Mit dem Handy!

Schritt-für-Schritt: Einen Sound hinzufügen

Ziel: Einen "Sprung" Sound abspielen.

Schritt 1: Sound-Datei finden

- Suche nach "jump sound effect free wav"
- Lade eine .wav Datei herunter

Schritt 2: Datei umbenennen

- Nenne sie `sprung.wav` (kleingeschrieben!)

Schritt 3: In sounds/ Ordner verschieben

- Die Datei muss in `sounds/sprung.wav` sein

Schritt 4: Im Code abspielen

```
def on_mouse_down(pos):
    sounds.sprung.play()  # Das ist alles!
```

Wichtig: Wie Sounds benannt werden

Der Dateiname wird zum Befehl:

Dateiname	Befehl im Code
`sounds/sprung.wav`	`sounds.sprung.play()`
`sounds/punkt.wav`	`sounds.punkt.play()`
`sounds/game_over.wav`	`sounds.game_over.play()`

Dateiname	Befehl im Code
`sounds/muenze_sammel.wav`	`sounds.muenze_sammel.play()`

Achte auf:

- Kleinbuchstaben
- Unterstriche statt Leerzeichen
- Keine Bindestriche (-)

Beispiel: Spiel mit Sounds

```
WIDTH = 400
HEIGHT = 600

spieler = Actor("spieler")
spieler.pos = (100, 500)

punkte = 0
spiel_vorbei = False

def draw():
    screen.fill("skyblue")
    spieler.draw()
    screen.draw.text(f"Punkte: {punkte}", (10, 10), color="black", fontsize=30)

    if spiel_vorbei:
        screen.draw.text("GAME OVER", (100, 250), color="red", fontsize=50)

def on_mouse_down(pos):
    global punkte, spiel_vorbei

    if not spiel_vorbei:
        # Sprung-Sound abspielen
        sounds.sprung.play()
        spieler.x = pos[0]

        # Punkt sammeln (Beispiel)
        punkte = punkte + 1
        sounds.punkt.play()

    if punkte >= 10:
        spiel_vorbei = True
        sounds.game_over.play()
```

Wenn Sounds nicht funktionieren

Problem: `AttributeError: 'module' object has no attribute 'sprung'`

Lösung:

1. Prüfe ob die Datei im `sounds/` Ordner ist
2. Prüfe ob der Name kleingeschrieben ist
3. Prüfe ob es `.wav` oder `.ogg` ist (nicht `.mp3`)

Kapitel 6: Flappy Bird mit Bildern und Sound

Ziel dieses Kapitels

Du baust das Flappy Box Spiel zu einem echten Flappy Bird um!

Was du brauchst

Bilder (in `images/` Ordner):

- `vogel.png` - Dein Vogel (ca. 40x30 Pixel)
- `roehre_oben.png` - Röhre von oben
- `roehre_unten.png` - Röhre von unten

Sounds (in `sounds/` Ordner):

- `flap.wav` - Flügelschlag beim Tippen
- `punkt.wav` - Wenn du durch eine Röhre kommst
- `game_over.wav` - Wenn du stirbst

Schritt 1: Actors statt Rechtecke

Vorher (Box):

```
box_x = 100
box_y = 300

def draw():
    screen.draw.filled_rect(Rect(box_x, box_y, 30, 30), "yellow")
```

Nachher (Vogel Actor):

```
vogel = Actor("vogel")
vogel.x = 100
vogel.y = 300

def draw():
    vogel.draw()
```

Schritt 2: Röhren als Actors

```
roehre_oben = Actor("roehre_oben")
roehre_unten = Actor("roehre_unten")

def setze_roehren_position():
    # Obere Röhre: Unterkante bei luecke_y
    roehre_oben.midbottom = (roehre_x, luecke_y)
    # Untere Röhre: Oberkante bei luecke_y + luecke_hoehe
    roehre_unten.midtop = (roehre_x, luecke_y + luecke_hoehe)
```

Was ist `midbottom` und `midtop`?

- `midbottom` = Die Mitte der Unterkante
- `midtop` = Die Mitte der Oberkante

Das macht das Positionieren einfacher!

Schritt 3: Kollision mit Actors

Vorher (kompliziert):

```
if box_x + box_groesse > roehre_x and box_x < roehre_x + roehre_breite:
    if box_y < luecke_y or box_y + box_groesse > luecke_y + luecke_hoehe:
        spiel_laeuft = False
```

Nachher (einfach):

```
if vogel.colliderect(roehre_oben) or vogel.colliderect(roehre_unten):
    spiel_laeuft = False
    sounds.game_over.play()
```

Schritt 4: Sound beim Flügelschlag

```
def on_mouse_down(pos):
    global vogel_speed

    if spiel_laeuft:
        vogel_speed = sprung_kraft
        sounds.flap.play()  # Flügelschlag-Sound!
```

Schritt 5: Vogel dreht sich

Der Vogel soll nach oben zeigen wenn er steigt und nach unten wenn er fällt:

```
def update():
    # ... andere Code ...

    # Vogel dreht sich basierend auf Geschwindigkeit
    if vogel_speed < 0:
        vogel.angle = 15   # Nach oben schauen
    else:
        vogel.angle = -15  # Nach unten schauen
```

Das komplette Spiel

```
# === FLAPPY BIRD MIT BILDERN UND SOUND ===
# Benötigte Bilder: vogel.png, roehre_oben.png, roehre_unten.png
# Benötigte Sounds: flap.wav, punkt.wav, game_over.wav

import random

WIDTH = 400
HEIGHT = 600

# === DER VOGEL ===
```

```
vogel = Actor("vogel")
vogel.x = 100
vogel.y = 300

vogel_speed = 0
schwerkraft = 0.5
sprung_kraft = -10

# === DIE RÖHREN ===
roehre_oben = Actor("roehre_oben")
roehre_unten = Actor("roehre_unten")

roehre_x = 400
luecke_y = 250
luecke_hoehe = 180
roehre_speed = 4

# === SPIELSTAND ===
punkte = 0
spiel_laeuft = True

def setze_roehren_position():
    roehre_oben.midbottom = (roehre_x, luecke_y)
    roehre_unten.midtop = (roehre_x, luecke_y + luecke_hoehe)

# Am Anfang positionieren
setze_roehren_position()

def draw():
    # Himmel
    screen.fill("skyblue")

    # Wolken (Dekoration)
    screen.draw.filled_circle((100, 80), 30, "white")
    screen.draw.filled_circle((130, 80), 40, "white")
    screen.draw.filled_circle((160, 80), 30, "white")

    # Röhren
    roehre_oben.draw()
    roehre_unten.draw()

    # Vogel
    vogel.draw()

    # Punkte
    screen.draw.text(f"{punkte}", (WIDTH/2 - 20, 30), color="white", fontsize=60)

    # Game Over
    if not spiel_laeuft:
        screen.draw.text("GAME OVER", (80, 250), color="red", fontsize=50)
        screen.draw.text("Tippe zum Neustarten", (70, 320), color="white", fontsize=25)

def update():
    global vogel_speed, roehre_x, luecke_y, punkte, spiel_laeuft
```

```
    if not spiel_laeuft:
        return

    # Schwerkraft
    vogel_speed = vogel_speed + schwerkraft
    vogel.y = vogel.y + vogel_speed

    # Vogel dreht sich
    if vogel_speed < 0:
        vogel.angle = 15
    else:
        vogel.angle = -15

    # Röhren bewegen
    roehre_x = roehre_x - roehre_speed
    setze_roehren_position()

    # Neue Röhren
    if roehre_x < -50:
        roehre_x = WIDTH + 50
        luecke_y = random.randint(100, HEIGHT - luecke_hoehe - 100)
        punkte = punkte + 1
        sounds.punkt.play()

    # Kollision mit Röhren
    if vogel.colliderect(roehre_oben) or vogel.colliderect(roehre_unten):
        spiel_laeuft = False
        sounds.game_over.play()

    # Kollision mit Rand
    if vogel.y < 0 or vogel.y > HEIGHT:
        spiel_laeuft = False
        sounds.game_over.play()

def on_mouse_down(pos):
    global vogel_speed, spiel_laeuft, roehre_x, punkte

    if spiel_laeuft:
        vogel_speed = sprung_kraft
        sounds.flap.play()
    else:
        # Neustart
        spiel_laeuft = True
        vogel.y = 300
        vogel_speed = 0
        roehre_x = 400
        punkte = 0
```

Kapitel 7: Münzen sammeln Spiel

Ziel dieses Kapitels

Du baust ein Spiel wo du Münzen sammelst und Feinden ausweichst.

Was du brauchst

Bilder:

- `spieler.png` - Deine Spielfigur
- `muenze.png` - Eine Münze
- `feind.png` - Ein Feind/Monster

Sounds:

- `muenze.wav` - Wenn Münze gesammelt
- `autsch.wav` - Wenn Feind berührt

Das Spielkonzept

```
┌─────────────────────────────────────────┐
│  • Spieler folgt deinem Finger          │
│  • Sammle Münzen = Punkte               │
│  • Feinde bewegen sich hin und her      │
│  • Berührst du einen Feind = Leben weg  │
│  • 3 Leben, dann Game Over              │
└─────────────────────────────────────────┘
```

Schritt 1: Listen von Actors

Ziel: Mehrere Münzen und Feinde erstellen.

```python
import random

# Münzen-Liste erstellen
muenzen = []
for i in range(5):  # 5 Münzen
    muenze = Actor("muenze")
    muenze.x = random.randint(30, WIDTH - 30)
    muenze.y = random.randint(50, 400)
    muenzen.append(muenze)

# Feinde-Liste erstellen
feinde = []
for i in range(3):  # 3 Feinde
    feind = Actor("feind")
    feind.x = random.randint(30, WIDTH - 30)
    feind.y = random.randint(100, 300)
    feind.speed_x = random.choice([-3, 3])  # Links oder rechts
    feinde.append(feind)
```

Was macht `random.choice([-3, 3])`? Es wählt zufällig einen Wert aus der Liste. Also entweder -3 (nach links) oder 3 (nach rechts).

Schritt 2: Alle Actors zeichnen

```python
def draw():
    screen.fill("darkgreen")

    # Alle Münzen zeichnen
    for muenze in muenzen:
        muenze.draw()

    # Alle Feinde zeichnen
    for feind in feinde:
        feind.draw()

    # Spieler
    spieler.draw()
```

Schritt 3: Feinde bewegen

```python
def update():
    # Feinde hin und her bewegen
    for feind in feinde:
        feind.x = feind.x + feind.speed_x

        # Am Rand umdrehen
        if feind.x < 30 or feind.x > WIDTH - 30:
            feind.speed_x = -feind.speed_x  # Richtung umkehren
```

Schritt 4: Münzen sammeln

```python
def update():
    global punkte
    # ... Feinde-Code ...

    # Münzen sammeln
    for muenze in muenzen[:]:  # [:] macht eine Kopie!
        if spieler.colliderect(muenze):
            muenzen.remove(muenze)  # Münze entfernen
            punkte = punkte + 1
            sounds.muenze.play()

            # Neue Münze erstellen
            neue_muenze = Actor("muenze")
            neue_muenze.x = random.randint(30, WIDTH - 30)
            neue_muenze.y = random.randint(50, 400)
            muenzen.append(neue_muenze)
```

Schritt 5: Unverwundbarkeit nach Treffer

Problem: Wenn du einen Feind berührst, verlierst du sofort alle Leben!

Lösung: Nach einem Treffer bist du kurz unverwundbar.

```
unverwundbar = 0  # Timer

def update():
    global leben, unverwundbar

    # Timer runterzählen
    if unverwundbar > 0:
        unverwundbar = unverwundbar - 1

    # Feind-Kollision nur wenn nicht unverwundbar
    if unverwundbar <= 0:
        for feind in feinde:
            if spieler.colliderect(feind):
                leben = leben - 1
                unverwundbar = 120  # 2 Sekunden unverwundbar (60 FPS × 2)
                sounds.autsch.play()

def draw():
    # Spieler blinkt wenn unverwundbar
    if unverwundbar <= 0 or unverwundbar % 10 < 5:
        spieler.draw()
```

Das komplette Spiel

```
# === MÜNZEN SAMMELN ===
# Sammle Münzen und weiche Feinden aus!

import random

WIDTH = 400
HEIGHT = 600

# Spieler
spieler = Actor("spieler")
spieler.x = 200
spieler.y = 500

# Münzen erstellen
muenzen = []
for i in range(5):
    muenze = Actor("muenze")
    muenze.x = random.randint(30, WIDTH - 30)
    muenze.y = random.randint(50, 400)
    muenzen.append(muenze)

# Feinde erstellen
feinde = []
for i in range(3):
    feind = Actor("feind")
    feind.x = random.randint(30, WIDTH - 30)
    feind.y = random.randint(100, 300)
    feind.speed_x = random.choice([-3, 3])
```

```
    feinde.append(feind)

# Spielstand
punkte = 0
leben = 3
unverwundbar = 0

def draw():
    screen.fill("darkgreen")

    # Gras-Muster
    for x in range(0, WIDTH, 20):
        for y in range(0, HEIGHT, 20):
            if (x + y) % 40 == 0:
                screen.draw.filled_rect(Rect(x, y, 20, 20), "green")

    # Münzen
    for muenze in muenzen:
        muenze.draw()

    # Feinde
    for feind in feinde:
        feind.draw()

    # Spieler (blinkt wenn unverwundbar)
    if unverwundbar <= 0 or unverwundbar % 10 < 5:
        spieler.draw()

    # Info
    screen.draw.text(f"Münzen: {punkte}", (10, 10), color="yellow", fontsize=30)
    screen.draw.text(f"Leben: {leben}", (10, 50), color="red", fontsize=30)

    if leben <= 0:
        screen.draw.text("GAME OVER", (100, 280), color="red", fontsize=45)

def update():
    global punkte, leben, unverwundbar

    if leben <= 0:
        return

    # Unverwundbar-Timer
    if unverwundbar > 0:
        unverwundbar = unverwundbar - 1

    # Feinde bewegen
    for feind in feinde:
        feind.x = feind.x + feind.speed_x
        if feind.x < 30 or feind.x > WIDTH - 30:
            feind.speed_x = -feind.speed_x

    # Münzen sammeln
    for muenze in muenzen[:]:
        if spieler.colliderect(muenze):
```

```
            muenzen.remove(muenze)
            punkte = punkte + 1
            sounds.muenze.play()

            neue_muenze = Actor("muenze")
            neue_muenze.x = random.randint(30, WIDTH - 30)
            neue_muenze.y = random.randint(50, 400)
            muenzen.append(neue_muenze)

    # Feind-Kollision
    if unverwundbar <= 0:
        for feind in feinde:
            if spieler.colliderect(feind):
                leben = leben - 1
                unverwundbar = 120
                sounds.autsch.play()

def on_mouse_down(pos):
    if leben > 0:
        spieler.x = pos[0]
        spieler.y = pos[1]

        # Grenzen
        if spieler.x < 20: spieler.x = 20
        if spieler.x > WIDTH - 20: spieler.x = WIDTH - 20
        if spieler.y < 20: spieler.y = 20
        if spieler.y > HEIGHT - 20: spieler.y = HEIGHT - 20
```

Kapitel 8: Animationen

Ziel dieses Kapitels

Du lernst wie man Bilder wechselt um Animationen zu erstellen.

Was ist eine Animation?

Eine Animation ist, wenn Bilder schnell hintereinander gezeigt werden:

```
Bild 1:  🚶       (Bein vorne)
Bild 2:  🚶       (Bein hinten)
Bild 3:  🚶       (Bein vorne)
... und so weiter

→ Sieht aus wie Laufen!
```

Bilder für Animation vorbereiten

Du brauchst mehrere Bilder mit Nummern:

- `images/laufen1.png`
- `images/laufen2.png`
- `images/laufen3.png`
- `images/laufen4.png`

Schritt-für-Schritt: Einfache Animation

Schritt 1: Liste mit Bildnamen

```
animation_bilder = ["laufen1", "laufen2", "laufen3", "laufen4"]
animation_frame = 0  # Welches Bild gerade?
animation_timer = 0  # Zähler für Bildwechsel
```

Schritt 2: In update() Bilder wechseln

```
def update():
    global animation_timer, animation_frame

    animation_timer = animation_timer + 1

    # Alle 10 Frames Bild wechseln
    if animation_timer >= 10:
        animation_timer = 0
        animation_frame = animation_frame + 1

        # Zurück zum ersten Bild wenn am Ende
        if animation_frame >= len(animation_bilder):
            animation_frame = 0

        # Bild des Actors ändern
        spieler.image = animation_bilder[animation_frame]
```

Animation nur beim Bewegen

Ziel: Animation läuft nur wenn der Spieler sich bewegt.

```
ist_am_laufen = False

def update():
    global animation_timer, animation_frame

    if ist_am_laufen:
        # Animation abspielen
        animation_timer = animation_timer + 1
        if animation_timer >= 10:
            animation_timer = 0
            animation_frame = (animation_frame + 1) % len(animation_bilder)
            spieler.image = animation_bilder[animation_frame]
    else:
        # Stehendes Bild zeigen
        spieler.image = "spieler_stehen"
```

Komplettes Animationsbeispiel

```
# === ANIMATION BEISPIEL ===
```

```
WIDTH = 400
HEIGHT = 600

# Spieler
spieler = Actor("laufen1")
spieler.pos = (200, 300)

# Animation
animation_bilder = ["laufen1", "laufen2", "laufen3", "laufen4"]
animation_frame = 0
animation_timer = 0

# Bewegung
ziel_x = 200
ziel_y = 300
ist_am_laufen = False

def draw():
    screen.fill("lightblue")
    screen.draw.filled_rect(Rect(0, 450, WIDTH, 150), "green")
    spieler.draw()
    screen.draw.text("Tippe irgendwo!", (100, 30), color="black", fontsize=25)

def update():
    global animation_timer, animation_frame, ist_am_laufen

    if ist_am_laufen:
        # Zum Ziel bewegen
        speed = 4
        diff_x = ziel_x - spieler.x
        diff_y = ziel_y - spieler.y

        if abs(diff_x) > speed:
            spieler.x = spieler.x + (speed if diff_x > 0 else -speed)
        else:
            spieler.x = ziel_x

        if abs(diff_y) > speed:
            spieler.y = spieler.y + (speed if diff_y > 0 else -speed)
        else:
            spieler.y = ziel_y

        # Angekommen?
        if spieler.x == ziel_x and spieler.y == ziel_y:
            ist_am_laufen = False
            spieler.image = animation_bilder[0]

        # Animation
        animation_timer = animation_timer + 1
        if animation_timer >= 8:
            animation_timer = 0
            animation_frame = (animation_frame + 1) % len(animation_bilder)
            spieler.image = animation_bilder[animation_frame]
```

```
def on_mouse_down(pos):
    global ziel_x, ziel_y, ist_am_laufen
    ziel_x = pos[0]
    ziel_y = pos[1]
    ist_am_laufen = True
```

Kapitel 9: Tipps für bessere Spiele

Partikel-Effekte

Kleine Teilchen die bei Aktionen erscheinen machen Spiele lebendiger!

```
partikel = []

def erstelle_partikel(x, y, farbe):
    import random
    for i in range(10):
        p = {
            "x": x,
            "y": y,
            "speed_x": random.uniform(-5, 5),
            "speed_y": random.uniform(-5, 5),
            "size": random.randint(3, 8),
            "farbe": farbe,
            "leben": 30  # Wie lange sichtbar
        }
        partikel.append(p)

def update_partikel():
    for p in partikel[:]:
        p["x"] = p["x"] + p["speed_x"]
        p["y"] = p["y"] + p["speed_y"]
        p["size"] = p["size"] * 0.95  # Wird kleiner
        p["leben"] = p["leben"] - 1
        if p["leben"] <= 0:
            partikel.remove(p)

def draw_partikel():
    for p in partikel:
        screen.draw.filled_circle(
            (int(p["x"]), int(p["y"])),
            int(p["size"]),
            p["farbe"]
        )
```

Benutzen:

```
# Bei Münze sammeln:
erstelle_partikel(muenze.x, muenze.y, "yellow")
```

```
# Bei Treffer:
erstelle_partikel(spieler.x, spieler.y, "red")
```

Highscore speichern

```
def speichere_highscore(punkte):
    try:
        with open("highscore.txt", "w") as datei:
            datei.write(str(punkte))
    except:
        pass  # Ignoriere Fehler

def lade_highscore():
    try:
        with open("highscore.txt", "r") as datei:
            return int(datei.read())
    except:
        return 0  # Kein Highscore vorhanden
```

Benutzen:

```
highscore = lade_highscore()

def update():
    global punkte, highscore
    # ... Spiel-Code ...

    if punkte > highscore:
        highscore = punkte
        speichere_highscore(highscore)
```

Hintergrundbild

```
hintergrund = Actor("hintergrund")
hintergrund.pos = (WIDTH/2, HEIGHT/2)

def draw():
    hintergrund.draw()  # ZUERST Hintergrund!
    spieler.draw()      # DANN andere Dinge
```

Zusammenfassung

Was du gelernt hast

Thema	Was du jetzt kannst
Ordner	`images/` und `sounds/` Ordner erstellen
Actors	Bilder als Spielfiguren benutzen
Kollision	`colliderect()` für einfache Kollisionsprüfung

Thema	Was du jetzt kannst
Sounds	`sounds.name.play()` für Sound-Effekte
Animation	Bilder wechseln für Bewegungseffekte
Listen	Mehrere Actors verwalten

Checkliste für ein fertiges Spiel

- [ ] Spieler kann sich bewegen
- [ ] Es gibt ein Ziel (Punkte, überleben, etc.)
- [ ] Es gibt Hindernisse oder Feinde
- [ ] Punkte werden angezeigt
- [ ] Game Over wenn man verliert
- [ ] Möglichkeit zum Neustarten
- [ ] **Bilder für alle Elemente**
- [ ] **Sounds für Aktionen**
- [ ] **(Bonus) Animationen**
- [ ] **(Bonus) Partikel-Effekte**
- [ ] **(Bonus) Highscore speichern**

Problemlösungen

"Actor not found" Fehler

```
Actor 'spieler' not found
```

Lösung:

1. Existiert `images/spieler.png`?
2. Ist der Name kleingeschrieben?
3. Ist die Endung `.png` (nicht `.PNG`)?

Sound spielt nicht

```
AttributeError: 'module' object has no attribute 'sprung'
```

Lösung:

1. Existiert `sounds/sprung.wav`?
2. Ist es `.wav` oder `.ogg` (nicht `.mp3`)?
3. Ist der Name kleingeschrieben?

Bild ist zu groß/klein

Lösung:

```
spieler.scale = 0.5  # Halb so groß
spieler.scale = 2    # Doppelt so groß
```

Super gemacht! 🎉

Du kannst jetzt richtig professionelle Spiele machen mit Bildern, Sounds und Animationen. Zeig deinen Freunden was du programmiert hast!

Tipp: Du kannst deine Spiele teilen! Schicke die `.py` Datei und die `images/` und `sounds/` Ordner zu deinen Freunden. Sie können dann dein Spiel auf ihrem Handy spielen!