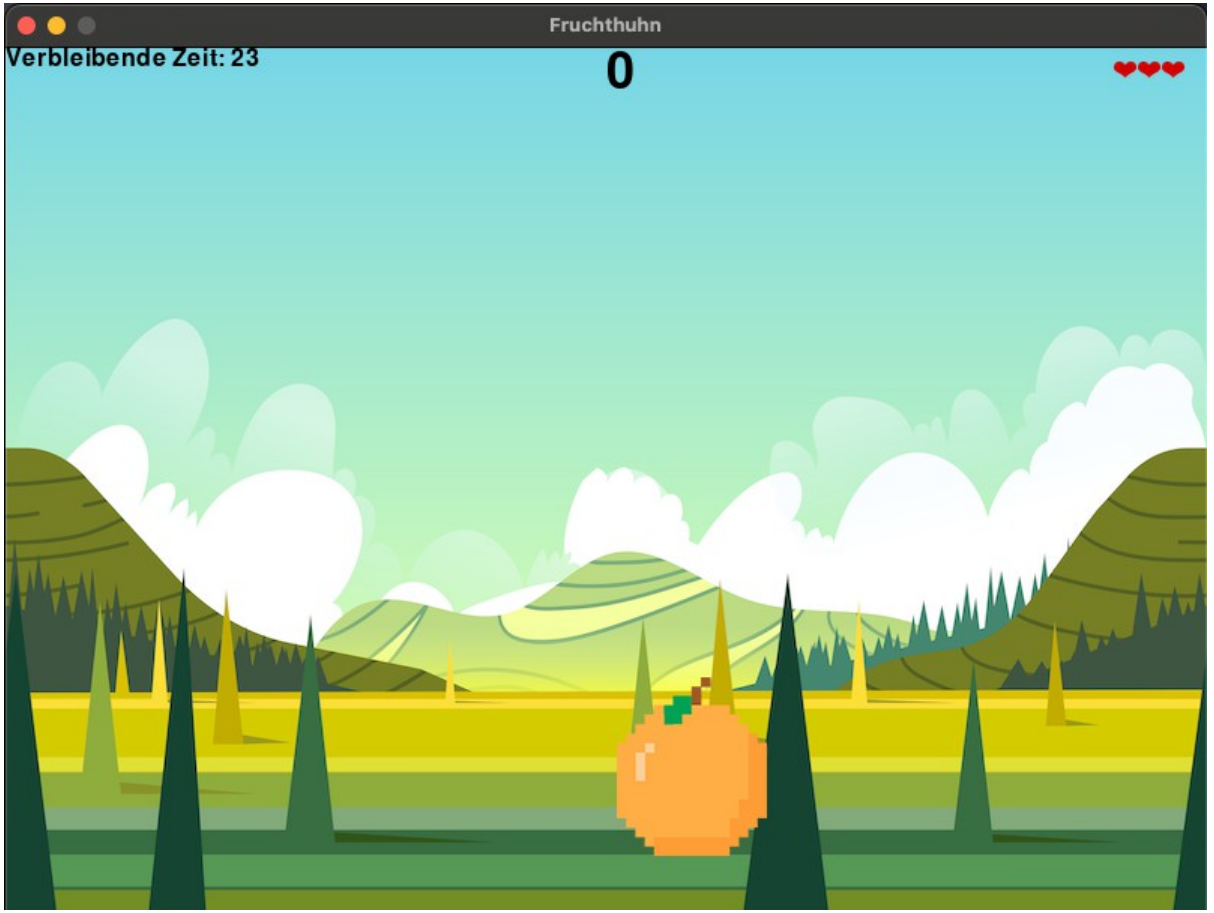
Python - Fruchthuhn Workshop

Python

Python

- Programmiersprache

Python - Fruchthuhn



Python - Fruchthuhn

- Spielscreen
- Früchte darstellen
- Mausklicks
- Score
- Zeit
- Spielerleben

Python - Fruchthuhn

- Für ein Spiel verwenden wir in Python:
  - pgzero

Falls es nicht da ist, muss es erst installiert werden.

Python - Fruchthuhn

- Python 3 installieren - https://www.python.org/downloads/

- Installation pgzero (https://pygame-zero.readthedocs.io/de/latest/):
  - Windows und Mac:
    - Kommandozeile:
      - pip install pgzero
  - Linux:
    - Kommandozeile:
      - sudo pip install pgzero

Python - Fruchthuhn

- Erstelle eine Text Datei
  - Windows: Notepad++, Atom, ...
  - Mac: TextEditor, Atom, ...
  - Linux: Nano Editor, Atom, …
  - Datei muss .py Endung haben! (wie eine PDF .pdf als Endung hat)
- Ausführen:
  - Schreibe in die Text Datei: import pgzrun
  - Ausführen/Starten:
    - Kommandozeile: navigiert zum Ordner wo die Datei gespeichert wurde
    - Kommandozeile: python textDateiName
    - Evtl. muss bei Windows noch der Pfad zum Python angegeben werden, wenn Windows den Befehl nicht kennt

Python & Pgzero

Funktion	Beschreibung
init()	Initialisiert alle importierten Pygame-Module (gibt ein Tupel zurück, das den Erfolg und Misserfolg von Initialisierungen anzeigt).
draw()	Zeichen Funktion → um Sachen auf den Bildschirm zu zeichnen Wird immer wieder automatisch während das spiel läuft aufgerufen.
update(dt)	Wird 60 mal in der Sekunde aufgerufen. dt = delta time = Zeitdifferenz zwischen jetzt und dem letzten Aufruf der Funktion update
on_mouse_down(pos)	Wird aktiviert sobald mit der Maus auf dem Monitor geklickt wird. Pos entspricht der Position der Maus.
on_key_down(key)	Wird aktiviert sobald eine Taste der Tastatur geklickt wurde. Key → entspricht der Taste die gedrückt wurde.
Actor('bildname')	Legt eine neue Spielfigur an, mit der viele Sachen möglich sind. Bildname → muss identisch zu dem Namen des Bildes, welches für die Figur ausgesucht wurde und im Projekt liegt, sein.

Python – Hintergrund

Code	Beschreibung
import pgzrun	
WIDTH = 300	Höhe des Bildschirms
HEIGHT = 300	Breite des Bildschirms
def draw(): screen.fill((128,0,0))	Füllt den Hintergrund mit einer Farbe die aus den drei (rot, grün, blau) gemischt werden.
pgzrun.go()	Wird benötigt um das spiel zu starten steht in der letzten Zeile der Datei

Welche Farbe hat der Hintergrund?

Versuch die Farbe zu wechseln, indem du für die drei Farben Werte von 0 bis 255 angibst.

Achtung: In Python muss auf die Einrückung der Zeilen geachtet werden --> je Einrückung werden 4 Leerzeichen verwendet

Python – Hintergrund

Für den Hintergrund wählen wir nun ein Bild aus dem Images Ordner.

→ Im „images“ Ordner liegt bereits ein bild „hintergrund“,
welches wir als Hintergrund für das Speil verwenden können.

→ screen.blit(“HINTERGRUND_NAME“, Startposition)

Als Startposition wählen wir (0,0)

Was passiert wenn ihr andere Werte für die Startposition wählt?

Tipp: Ersetze in der draw funktion das screen.fill() mit scree.blit()

Python – Hintergrund

Code	Beschreibung

def draw():
  screen.clear()
  screen.blit("hintergrund", (0, 0))
	→ löscht alles auf dem Screen
	→ zeichnet das ausgewählte Bild „hintergrund“ auf den screen beginnend bei x = 0 und y = 0

Python – Sprite (Figuren)

Wir wollen Figuren zeichnen

Benötigt wird dafür:

- Bild welches gezeichnet werden soll
- die Actor('bild_name') Klasse von pgzero

Python – Sprite (Figuren)

Code	Beschreibung
frucht = Actor('apple')	Actor ist unsere Spielfigur – in der klammer wird ein Bild eingebunden, was „apple“ heißt.

def draw():
  screen.clear()
  screen.blit("hintergrund", (0, 0))
  frucht.draw()
	Figur „frucht“ auf dem Bildschirm zeichnen

Python – Mausklicks

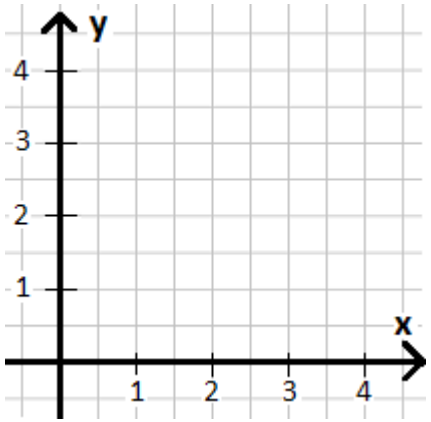
Lass die Frucht durch einen Mausklick verschwinden!

- nutze die on_mouse_down(pos) Funktion
    - pos = Position

- frucht.collidepoint(pos) – prüft ob die Frucht (Sprite) an der selben stelle is wie pos

Tipp: Frucht verschwinden lassen, indem die Position der Frucht geändert wird

Python – Mausklicks



frucht.pos(x,y)

Oder

frucht.x = 160
frucht.y = 300

Python – Zufällige Position

from random import randint → sollte ganz oben verwendet werden, wird verwendet um randint benutzen zu können.

randint(minWert, maxWert) → gibt einen zufallswert zwischen minWert und maxWert zurück

Python – Mausklicks

```
def on_mouse_down(pos):
    if frucht.collidepoint(pos):
        print("Treffer!")
        # 10 als Startwert damit etwas abstand zum
Oberen/links Rand existiert und die WIDTH/HEIGHT als
Endwerte für untere/rechte Begrenzung.
        frucht.x = randint(10, WIDTH)
        frucht.y = randint(10, HEIGHT)
    else:
        print("Daneben!")
```

Python – Score

Es soll angezeigt werden, wie viele Früchte der Spieler getroffen hat.

Sobald der Spieler eine Frucht angeklickt hat soll der Score um 1 erhöht werden.

Tipp: zum anzeigen benötigt ihr

```
screen.draw.text('text zum anzeigen', (x, y), fontsize=50, color="black")
```

Python – Score

Was passiert, wenn ihr die `fontsize` kleiner oder größer als 50 wählt?

Was passiert, wenn ihr für `color` eine andere Farbe eingebt?

Python – Score

```
frucht = Actor("apple")
# add score variable
score = 0
WIDTH = 800
HEIGHT = 600

def draw():
    screen.clear()
    screen.blit("background", (0, 0))
    # zeige den Score oben links an
    screen.draw.text(str(score), (400, 0), fontsize=50, color="black")
    frucht.draw()

def place_fruit():
    frucht.x = randint(10, WIDTH)
    frucht.y = randint(10, HEIGHT)

def on_mouse_down(pos):
    if frucht.collidepoint(pos):
        print("Treffer!")
        place_fruit()
        # update score
        score += 1
    else:
        print("Daneben!")

# setze alle start werte auf den entsprechenden Anfangswert
def init():
    global score
    score = 0
    place_fruit()
```

Python – Verschiedene Früchte

Es sollen verschiedene Früchte angezeigt werden, nicht nur der Apfel.

Dafür benötigen wir verschiedene Frucht Bilder, die im selben Ordner wie das Apfelbild gespeichert sind.

Tipp: Ein Liste mit Fruchtbildnamen und dann dem Actor zufällig eins der Bilderübergeben
Die Frucht muss auch zufällig neu gesetzt werden, wenn die Frucht angeklickt wurde

Python – Verschiedene Früchte

```
from random import randint, choice

# liste mit den frucht bildern
fruechte = ['apple', 'orange', 'pineapple']
# waehle ein bild zufaellig aus 'fruechte' aus
frucht = Actor(choice(fruechte))




def place_fruit():
    frucht.x = randint(10, WIDTH)
    frucht.y = randint(10, HEIGHT)

    # aendere das bild der aktuellen frucht durch ein neues zufaelliges bild aus
'fruechte'
    frucht.image = choice(fruechte)
```

Python – Timer

Es soll ein Timer mit der verbleibenden Zeit angezeigt werden.

Der Timer soll bei 30 Sekunden starten und langsam runterzählen.

Tipp: Verwendet hierfür

```
def update(dt):
```

Python – Timer

```
# neue variable timer mit 30
timer = 30

# standard methode updated den screen etc.
def update(dt):
    global timer
    timer -= dt

def draw():
    screen.clear()
    screen.blit("background", (0, 0))
    screen.draw.text(str(score), (400, 0), fontsize=50, color="black")
    frucht.draw()

    # zeigt die verbleibende Zeit in der mitte des bildschirms an
    screen.draw.text("Verbleibende Zeit: %s" % round(timer), (0, 0), color="black")
```

Python – Timer

Prüfe, ob der Timer abgelaufen ist.

Zeige einen Gameover screen an, wenn der Timer abgelaufen ist.

Timer nur runter zählen, wenn er nicht 0 ist.

Reagiere nur auf Mausklicks, solange der Timer nicht 0 ist.

Python – Timer

```
timer = 30

def update(dt):
    global timer
    # zähle den timer nur runter, wenn er nicht 0 ist
    if timer >= 0:
        timer -= dt

def draw():
    screen.clear()
    screen.blit("background", (0, 0))
    screen.draw.text(str(score), (400, 0), fontsize=50, color="black")
    frucht.draw()
    screen.draw.text("Verbleibende Zeit: %s" % round(timer), (0, 0), color="black")

    # zeigt ein Game Over wenn der Timer abgelaufen ist
    if timer < 0:
        screen.draw.textbox("Game Over!\n %s Punkte erreicht" % score, Rect(100, 100, 600, 400),
background="black")

def on_mouse_down(pos):
    global score, timer
    # reagiere nur auf maus klicks wenn der timer noch nicht abgelaufen ist
    if timer > 0:
        if frucht.collidepoint(pos):
            print("Treffer!")
            place_fruit()
            score += 1
        else:
            print("Daneben!")
```

Python – Restart

Nachdem der Timer abgelaufen ist und der Spieler ein Gameover angezeigt bekommt, soll es nun auch möglich sein über einen Tasten druck das Spiel neu zu starten.

Dafür soll nicht immer das ganze Programm neugestartet werden.

Tipp: Verwendet hierfür

```
def on_key_down(key):
```

Python – Restart

```
# standard methode wird aufgerufen bei jedem Tasten druck
def on_key_down(key):

    # pruefe ob R gedrückt wurde um jederzeit das spiel neu zu starten
    if key == keys.R:
        init()
```

Python – Spielerleben

Der Spieler soll 3 Leben bekommen.

Wenn der Spieler die Frucht nicht trifft soll ihm ein Leben abgezogen werden.

- gib dem Spieler 3 Leben

- zeige die Anzahl der Leben an

- ziehe dem Spieler ein Leben ab, wenn die Frucht nicht getroffen wurde

Python – Spielerleben

```
# neue variable für die Anzahl der Leben
player_lives = 3

WIDTH = 800
HEIGHT = 600

def update(dt):
    global timer, player_lives

    # update den timer nur wenn Leben da sind und timer nicht 0
    if timer >= 0 and player_lives > 0:
        timer -= dt


def draw():
    screen.clear()
    screen.blit("background", (0, 0))
    screen.draw.text(str(score), (400, 0), fontsize=50, color="black")
    frucht.draw()
    screen.draw.text("Verbleibende Zeit: %s" % round(timer), (0, 0), color="black")

    draw_lives()
    if timer < 0:
        screen.draw.textbox("Game Over!\n %s Punkte erreicht" % score, Rect(100, 100, 600, 400),
background="black")


# funktion welche die Leben anzeigt auf der rechten seite
def draw_lives():
    for life in range(player_lives):
        screen.blit('heart', (WIDTH-30-(life*16), 10))
```

Python – Spielerleben

```
def place_fruit():
    frucht.x = randint(10, WIDTH)
    frucht.y = randint(10, HEIGHT)
    frucht.image = fruechte[randint(0, len(fruechte) - 1)]

def on_mouse_down(pos):
    global score, timer, player_lives
    if timer > 0:
        if frucht.collidepoint(pos):
            print("Treffer!")
            place_fruit()
            score += 1
        else:
            # ziehe ein Leben ab.
            player_lives -= 1
            print("Daneben!")

def on_key_down(key):
    if key == keys.R:
        init()

def init():
    global score, timer, player_lives
    score = 0
    place_fruit()
    timer = 30
    # setze das leben wieder auf 3
    player_lives = 3
```

Python – Spielerleben

- prüfe ob der Spieler noch ein Leben hat
- zeige einen anderen Gameover Ansicht an, wenn der Spieler keine Leben mehr hat
- Stoppe die Zeit, wenn der spiele keine Leben mehr hat

Python – Spielerleben

```
player_lives = 3

WIDTH = 800
HEIGHT = 600

def update(dt):
    global timer, player_lives

    # update den timer nur wenn Leben da sind und timer nicht 0
    if timer >= 0 and player_lives > 0:
        timer -= dt


def draw():
    screen.clear()
    screen.blit("background", (0, 0))
    screen.draw.text(str(score), (400, 0), fontsize=50, color="black")
    frucht.draw()
    screen.draw.text("Verbleibende Zeit: %s" % round(timer), (0, 0), color="black")

    draw_lives()
    if player_lives == 0 or timer < 0:
        draw_game_over()


def draw_lives():
    for life in range(player_lives):
        screen.blit('heart', (WIDTH-30-(life*16), 10))
```

Python – Spielerleben

```
# Funktion, die prüft was für ein Game over angezeigt werden soll
def draw_game_over():
    global timer, score
    if timer > 0:
        screen.draw.textbox("Game Over!\n %s Punkte erreicht\n mit %s Sekunden Restzeit!" % (score, round(timer)), Rect(100, 100, 600,
400), background="black")
    else:
        screen.draw.textbox("Game Over!\n %s Punkte erreicht" % score, Rect(100, 100, 600, 400), background="black")

def place_fruit():
    frucht.x = randint(10, WIDTH)
    frucht.y = randint(10, HEIGHT)
    frucht.image = fruechte[randint(0, len(fruechte) - 1)]

def on_mouse_down(pos):
    global score, timer, player_lives
    if timer > 0:
        if frucht.collidepoint(pos):
            print("Treffer!")
            place_fruit()
            score += 1
        else:
            player_lives -= 1
            print("Daneben!")

def on_key_down(key):
    if key == keys.R:
        init()

def init():
    global score, timer, player_lives
    score = 0
    place_fruit()
    timer = 30
    player_lives = 3
```

Python – Früchte bewegen sich

Damit der Spieler es nicht so einfach hat, die Früchte zu erwischen,
Sollen die Früchte sich bewegen.

- Lass die Früchte sich nach rechts oder links bewegen

- Lass die Früchte sich nach oben oder unten bewegen

Tipp: Ändere hierfür in der draw() Funktion die x und y werte der Frucht

Python – Früchte bewegen sich

Was passiert mit der Frucht wenn ihr x und y gleichzeitig ändert werden?

Was passiert mit der Frucht wenn ihr x und y gleichzeitig geändert werden und den gleichen Wert haben?

Was passiert mit der Frucht wenn ihr x und y gleichzeitig geändert werden und die Wert sich unterscheiden?

Python – Früchte bewegen sich

```
def draw():
    screen.clear()
    screen.blit("background", (0, 0))
    screen.draw.text(str(score), (400, 0), fontsize=50, color="black")
    frucht.draw()
    screen.draw.text("Verbleibende Zeit: %s" % round(timer), (0, 0), color="black")
    move_frucht()
    draw_lives()
    if player_lives == 0 or timer < 0:
        draw_game_over()


def move_frucht():
    # addiere 1 zur Position der Frucht hinzu
    frucht.x += 1
    frucht.y += 1
```

Python – Früchte bewegen sich

Achte darauf, dass die Früchte sich innerhalb des vorgegebenen Spielebildschirms Bewegen.

- Prüfe dass die Frucht nichts rechts/links außerhalb des Bildschirm angezeigt wird

- Prüfe dass die Frucht nichts oben/unten außerhalb des Bildschirm angezeigt wird

Python – Früchte bewegen sich

```
def move_frucht():
    # addiere zufälligen wert zwischen -20 und 20 zur Position
der Frucht
    frucht.x += randint(-20, 20)
    frucht.y += randint(-20, 20)

    if frucht.x <= 10:
        frucht.x = 30

    if WIDTH - 20 <= frucht.x:
        frucht.x = WIDTH - 30

    if frucht.x <= 10:
        frucht.y = 30

    if HEIGHT - 20 <= frucht.y:
        frucht.y = HEIGHT - 20
```

Python – tanzende Früchte

Um den Schwierigkeitsgrad für den Spieler zu erhöhen, lassen wir die Früchte tanzen.

- setze den x Wert der Frucht auf einen zufälligen wert zwischen -10 und 10
  → benutze hierfür die randint() Funktion

→ was kannst du beobachten?

Python – tanzende Früchte

- lass x und y zufällig einen Wert zwischen -20 und 20 aus wählen

→ was passiert mit der Frucht?

→ wähle unterschiedliche werte aus auch für x und y – was passiert dann mit der Frucht?

Python – Früchte bewegen sich

```
def move_frucht():
    # addiere zufälligen wert zwischen -20 und 20 zur Position der Frucht
    frucht.x += randint(-20, 20)
    frucht.y += randint(-20, 20)

    if WIDTH - 20 <= frucht.x <= 0:
        frucht.x = 10
    if HEIGHT - 20 <= frucht.y <= 0:
        frucht.y = 10
```