Pygame - Zusammenfassung

Aufbau eines Programms:

```python
import pygame
pygame.init()

FPS = 60
screen_width = 800
screen_height = 600
clock = pygame.time.Clock()

screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("Mein Pygame Spiel")

#----

while True:
    screen.fill((0, 0, 0))  # Fenster schwarz füllen

    #----

    clock.tick(FPS)
    pygame.display.flip()
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

Pygame importieren und initialisieren

Variablen für das Spiel festlegen (FPS, Punktestände…), Bilder, Sounds, Schriftarten und weiteres laden.

Fenster für das Spiel erstellen.

Funktionen für das Spiel definieren.

Game Loop - die Funktionen des Spiels ausführen:
- Fenster leeren (-> einfarbig füllen)
- Input vom Benutzer (z.B über Tastatur) abfragen
- Spiellogik: Funktionen, Grafiken zeichnen
- Prüfen, ob das Fenster geschlossen werden soll.

Pygame installieren

Das kann man normalerweise mit dem Paketmanager von Python, pep, erledigen.
Um ein Paket zu installieren, tippt man „pip install“ und dann den Paketname in ein Terminal ein.

Im Falle von pygame müssten wir also „pip install pygame“ in das Terminal eingeben.

Pygame verwenden

Um pygame zu verwenden, muss es zuerst importiert werden. Dies wird mit „Import pygame“ in der ersten Zeile einer Python Datei, in der pygame verwendet wird, erledigt.

Funktionen und wie sie funktionieren

Aufbau einer Funktion - Beispiel: blit(surface, Position)

„blit“ ist hier der Name der Funktion.

Innerhalb der Klammern stehen Argumente (=Daten), die man der Funktion geben muss, dass sie funktioniert.
Surface muss ein pygame.Surface - also ein Bild oder Fenster sein.
Position ist eine Liste aus zwei Zahlen: eine x- und eine y-Koordinate. z.B [400,200]

Es kann auch vorkommen, dass eine Funktion auf eine bestimmte Variable „angewendet“ wird.

Dies erkennt man daran, dass diese Funktion dann mit einem Punkt an eine Variable angeschlossen wird.

Beispiel: Surface.blit(surface, Position)

Die Funktion blit wird hier auf Surface angewendet - an Stelle von Surface könnte auch irgendeine andere Variable stehen, solange sie ein Pygame Surface (also ein Bild oder Fenster) ist.

Funktionen von Pygame

(Hinter den Argumenten steht jeweils in grün, was für Daten die Funktion hier erwartet, und in orange, was für Daten die Funktion zurückgibt. Diese Daten kann man zur weiteren Verwendung in einer Variable speichern.

• pygame.init() - initialisiert pygame

• pygame.display.set_mode(size - Liste aus zwei Zahlen) -> Bild (das Fenster des Programms)
-> Erstellt ein Fenster mit einer bestimmten Größe. In Size sind Höhe und Breite des Fensters gespeichert. Ist size bspw. [800, 600] ist das Fenster 800 x 600 Pixel groß. Dieses Fenster kann man dann als Variable speichern.
Beispiel: screen = pygame.display.set_mode((800,600)) - Das Fenster wird in der Variable „screen“ gespeichert.

• Surface.fill(RGB Farbe) Surface = ein Bild oder Fenster
-> Füllt das Bild/Fenster mit einer RGB-Farbe (Liste aus drei Zahlen zwischen 0 und 255)
Solche RGB-Farben kann man online mischen. Sucht einfach „RGB Farbmischer“ im Internet.

• Surface.blit(Bild, Position - Liste aus zwei Zahlen)
-> Zeigt ein Bild an einer Position.

• pygame.image.load(Dateipfad) -> Bild
-> Lädt Bild aus einem Dateipfad (= „Ordner/Ordner/Dateiname.Endung“, z.B „Bilder/test.png)
Dieses kann dann als Variable gespeichert werden. Beispiel: Player_image = pygame.image.load(„player.png“)

• pygame.transform.scale(Bild, Zielgröße - Liste aus zwei Zahlen) -> Bild
-> Skaliert das eingegebene Bild auf die eingegebene Zielgröße - Dieses Bild kann dann in einer Variable gespeichert werden - Beispiel: neues_Bild = pygame.transform.scale(altes_Bild, (200,200))

• pygame.transform.rotate(Bild, Winkel - in Grad als Zahl) -> Bild
-> Dreht das eingegebene Bild um den eingegebenen Winkel (in Grad).
Beispiel: gedrehtes_Bild = pygame.transform.rotate(altes_Bild, 45)

• pygame.display.flip() - updatet das Fenster

• pygame.quit() - beendet pygame

• pygame.key.get_pressed()[pygame-taste] -> True oder False
-> Gibt True (Ja) oder False (Nein) zurück, je nachdem ob die abgefragte Taste gedrückt ist. Pygame-tasten sind z.B pygame.K_A für die Taste A, pygame.K_Space für Leertaste, Pygame.K_UP für den Pfeil hoch, etc…
Beispiel: ist_A_gedrückt = pygame.key.get_pressed()[pygame.K_A]

• pygame.font.Font(Dateipfad, Größe - Zahl) -> pygame-Schriftart
-> Lädt eine Schriftart aus einem Dateipfad mit einer bestimmten Größe (z.B 24, 32…)

• Font.render(Text, True, RGB Farbe) Font = eine mit pygame.font.Font geladene Schriftart -> Bild (der Text)
-> Zeichnet einen Text in einer bestimmten Farbe und Schriftart. Dieser kann dann z.B mit .blit auf dem Bildschirm angezeigt werden.

Wo kriegt man weitere Infos / Hilfe?

Wenn man beim Programmieren bei einem Thema Hilfe benötigt oder Fehler aufkommen, empfiehlt es sich, auf die Dokumentation der Sache, bei der ihr Hilfe benötigt (z.B. Pygame, Python) zurückzugreifen.
Zum Lernen sind auch Tutorials sehr hilfreich, und vielleicht etwas spannender als das Lesen einer Dokumentation.

Die Dokumentation von pygame findet ihr unter: https://pyga.me/docs/

In was kann man noch programmieren?

Es gibt hunderte Programmiersprachen und Programmierumgebungen…
Für Spiele gibt es zusätzlich Game Engines, die einem viel Arbeit abnehmen können. (Unity, Godot,…)
Besonders bekannt sind Programmiersprachen wie C, C#, Java, Python, C++ und Javascript.