Ausarbeitung

Ein Snake Spiel mit Darstellung über einen Raspberry Pi

Zoe Luca Günther

10. August 2022

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In diesem Projekt habe ich ein Snake Spiel entwickelt. Hierbei ist das besondere, dass man dieses Snake Spiel über die Lagesensoren des Handys steuern kann. Man verbindet sich mit einer Website, geht in die Warteschlange rein und sobald man an der Reihe ist, wird man zur Spielsteuerung weitergeleitet [1][2]. Am Anfang jedes Spiels wird ein kurzer Hinweis am Handy eingeblendet, welcher einem sagt, dass man das Handy vertikal und flach halten soll und die Bildschirmrotation deaktiviert werden sollte [3][4]. Dies ist wichtig, damit die Steuerung richtig funktioniert, da man um das Spiel zu steuern, lediglich sein Handy neigen muss. Um Das Spielerlebnis so gut wie möglich zu halten, muss das Handy flach und vertikal gehalten werden, da so die Steuerung am besten funktioniert, da diese sehr sensibel sein kann. Um das Spielgeschehen zu sehen, wird ein Raspberry Pi mit einer 64x64 LED Matrix verwendet, worauf der Score und das Spiel dargestellt werden. Um das Spiel zu pausieren, klickt man lediglich auf seinem Handy auf den Pause Button [5]. Dies geht jedoch nur, wenn man gerade an der Reihe ist. Um dies zu überprüfen, wird beim ersten betreten der Spiel Website ein Cookie gesetzt, worin die User ID steht. Jede ID ist unterschiedlich, um eine genaue Identifizierung zu ermöglichen. Sollte sich ein Spieler für mehr als 20 Sekunden im Pause Status befinden, oder mehr als 20 Sekunden keine Bewegung mehr machen, wird das Spiel beendet und der nächste Spieler aus der Warteschlange kann spielen. Nachdem man sich selber als Schlange gefressen hat, oder man zu lange im Spiel war, wird man auf eine Game Over Seite weitergeleitet, wo der eigene Score eingeblendet wird [6].

2 Vorgehensweise

Das Projekt wurde in verschiedene Unterpunkte eingeteilt, um die bestmögliche Implementierung sicherzustellen.

2.1 Grundgerüst

Das Grundgerüst besteht aus verschiedenen Klassen, welche für den Spielablauf essentiell sind. Folgende Klassen beinhaltet das Grundgerüst:

Klassenname	Beschreibung
Player.py	Zuständig für die Bewegung des Spielers, des Überprüfens auf Kollision, das Score Handling und das Fressen von Futter
Logger.py	Ausgeben von Nachrichten in der Konsole zu Debug Zwecken. Es kann ein Prefix der Klasse Prefix.py angegeben werden
Playground.py	Handling des Spielfeld, setzen von Blöcken[1] auf dem Spielfeld, setzen einer Random Futter Position, Prüfen ob das Spielfeld voll ist
Queue.py	Generelles Queue Handling (Spieler hinzufügen, entfernen aus Queue), Nächsten Spieler nehmen, UserId verifizieren[2]
SnakeGame.py	Game Status setzen, Game starten / stoppen / pausieren, Game loop, game resetten, game over Handling

Tabelle 1: Grundgerüst Klassen

Klassenname	Beschreibung
Direction.py	Richtungsangaben
GameStatus.py	Gamestatus mit Gamestatus Text
Message.py	Game over Nachrichten
PlaygroundTile.py	Spielfeld Blöcke (zum Beispiel Wand, Snake, Futter)
Prefix.py	Prefix zu Debug Zwecken

Tabelle 2: Enum Klassen

Zum testen der Funktionalitäten, wurde das Spiel vorerst mithilfe der Bibliothek *pygame* implementiert. Somit konnte das Spiel dargestellt werden, die Bewegungsabläufe getestet und der Spielablauf angepasst werden. Für den Gameloop wurde jedoch zu diesem Zeitpunkt die pygame eigenen Funktionen verwendet. Die Steuerung wurde vorerst über die Tastatur implementiert, da die spätere Steuerung über die Website mithilfe der API Schnittstelle funktioniert.

2.2 Website

2.3 API Schnittstelle

2.4 Darstellung

3 Schwierigkeiten

Die größte Schwierigkeit war, die API nicht zu unterbrechen, während das Spiel läuft. Es wurde *multithreading* verwendet, um verschiedene Thread parallel laufen zu lassen. Somit laufen im fertigen Projekt der Thread für die Darstellung des Spiels auf dem

[1] Das Spielfeld besteht aus Blöcken, welche in einem 2d Array gespeichert sind
[2] Überprüfen ob die UserId die nötige länge hat

Raspberry Pi, der Gameloop, die Überprüfung ob ein Spieler keine Inputs mehr macht und somit nicht mehr anwesend ist (AFK[3]) und die API Schnittstelle parallel. Es wurde versucht, einen neuen Prozess für die Darstellung auf dem Display zu verwenden, jedoch können live updates in einem separaten Prozess nicht mehr nachverfolgt werden, weshalb die Thread Variante verwendet worden ist. Die Thread wurden als *daemon* Threads gestartet, damit man sie zu jedem Zeitpunkt durch ein Interrupt unterbrechen kann. Die Implementierung sieht wie folgt aus, sobald die start Methode von der API Schnittstelle aufgerufen wird:

```
# Threads initialisieren
displayThread = threading.Thread(name="Display", target=display.
    process)
displayThread.daemon = True

gameLoopThread = threading.Thread(name="Gameloop", target=self.loop)
gameLoopThread.daemon = True

gameLoopAfkThread = threading.Thread(name="GameloopAfk", target=self.
    loopAfkCheck)
gameLoopAfkThread.daemon = True

# Threads starten
gameLoopThread.start()
displayThread.start()
gameLoopAfkThread.start()

# Threads beenden
gameLoopThread.join()
self.loopAfkCheckRunning = False
gameLoopAfkThread.join()
display.terminate()
displayThread.join()
```

Entscheidend ist hierbei die Reihenfolge, in der die Threads gestartet und gestoppt werden. Zu beachten ist außerdem, dass der Display Thread eine Funktion benötigt, um terminiert zu werden, ansonsten kann er nicht gestoppt werden. die Loops greifen jeweils auf den Gamestatus zu, um den in jedem Thread integrierten *infinity loop* zu unterbrechen, sollte sich der Gamestatus ändern und der Thread nicht mehr benötigt werden. Somit erklärt sich auch die Reihenfolge, in welcher die Thread beendet werden. Sobald das Spiel vorbei ist und der Gameloop beendet wird, kann die afk Überprüfung ebenfalls beendet werden. Da das Spiel vorbei ist wird außerdem zum Schluss der Displaythread beendet, da es kein Spiel mehr zum darstellen gibt. Somit gehen alle Aktionen der restlichen Threads vom eigentlichen Gameloop aus.

4 Ergebnisse

[3]away from keyboard, abwesend

5 Abbildungen

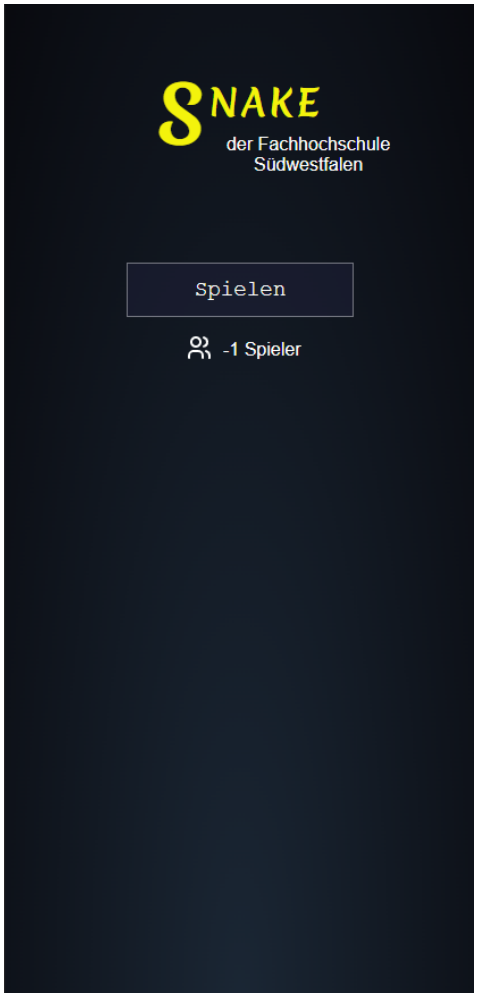

Abbildung 1: Snake Website Hauptmenü

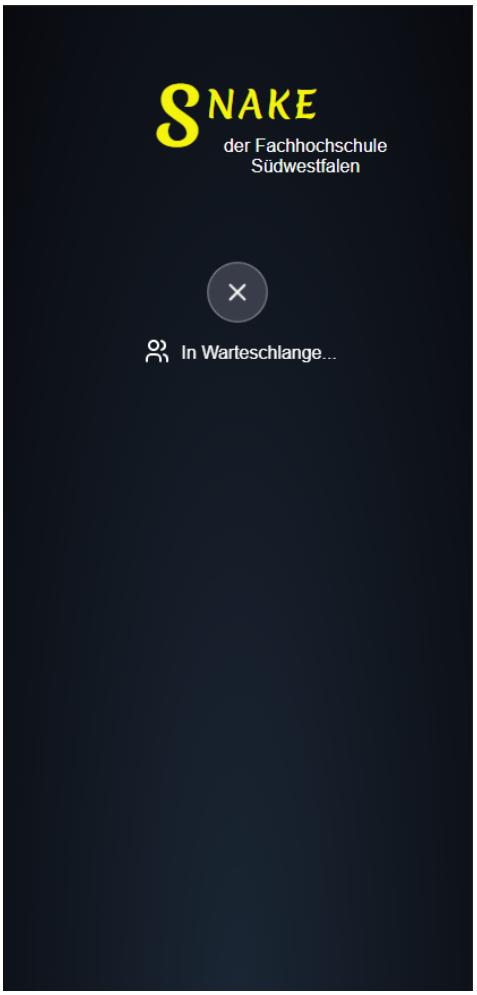

Abbildung 2: Snake Website Hauptmenü in Warteschlange

Abbildung 3: Snake Website Controller Hinweis 1

Abbildung 4: Snake Website Controller Hinweis 2

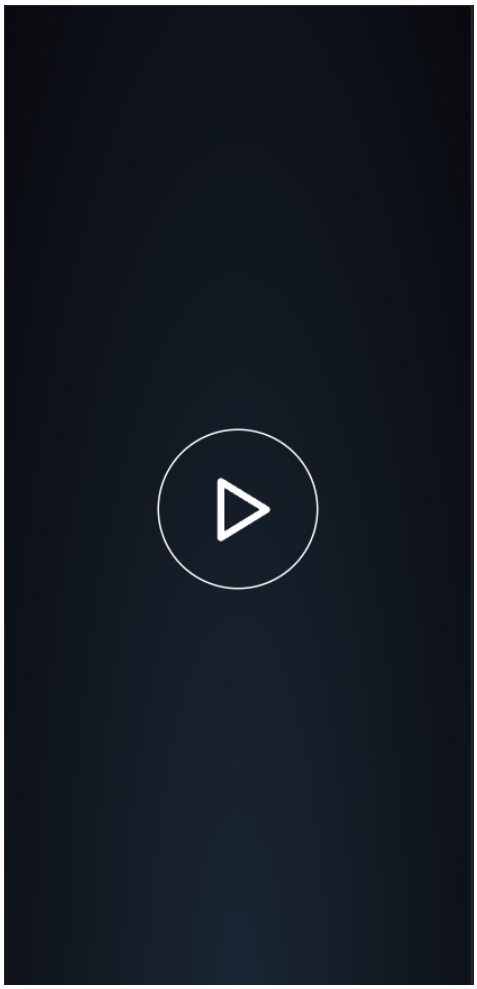

Abbildung 5: Snake Website Controller

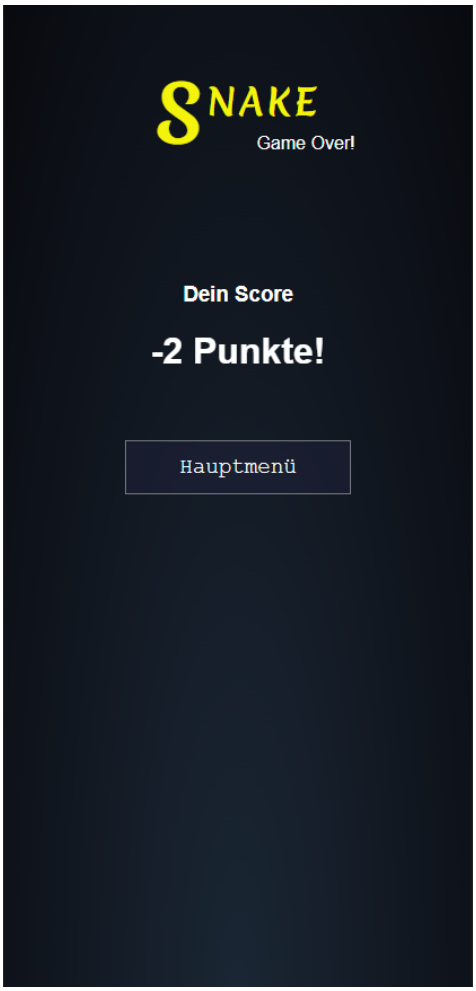

Abbildung 6: Snake Website Game Over Score

6 Quellenverzeichnis

- test