AI Drone Race

Author: Kaito Soga
Fach: IN29
Sprachen: JS + Python

12 Januar 2026

Übersicht

In dem geplanten Game tritt ein User in einem Rennen gegen eine KI gesteuerte 2D-Drone an. Das Ziel ist es, so schnell wie möglich von einem Checkpoint zum nächsten zu fliegen, mit einer endlichen Anzahl and Checkpoints.

Die Steuerung für den User besteht aus zwei Inputs, welche die zwei Antriebe der Drone kontrollieren. Die KI stuert ihre Drone mit einem simplen feedforward network (FFN).

Zusätzlich kann ein User einen manuellen algorithmus erstellen, durch das Bauen einer Funktion die den Zustandsvektor als input nimmt, und als output die Antriebswerte der Drone gibt. Diese kann ebenfalls gegen die KI oder den User selbst antreten.

Die Drone selbst unterliegt realistischen Physikalischen Eigenschaften (Schwerkraft, Momentum, Momentum, Beschleunigung, Rotation (-sgeschindigkeit)), und die Umgebung wird zufällig mit vorbestimmten Bildern generiert, wobei die Drone im Mittelpunkt des Canvas bleibt, sodass sich der Hintergrund bewegt.

*Sollte die Zeit noch reichen, werde ich im Backend auf einem Raspberry-Pi ein simples Login erstellen, um Usern einen Top-Score, Index auf einem öffentlichen Leaderboard, und Top-Score-abhängige Outfits/Styles für die Drone zuzuweisen. Andernfalls ist es mein Ziel, Teil 0 zu implementieren.

Details + Schritte der Implementierung

1

1.1 Realistische Physik für die Drone

Aufgrund der KI, die ich aus praktischen Gründen in Python trainiere (mir bekannte, optimierte Bibliothek PyTorch, die es in JS nicht gibt), werde ich auch die Physik-Logik vorerst in Python implementieren. Diese soll beeinhalten:

- Schwerkraft - Momentum - Momentum - Beschleunigung - Rotation - Rotationsgeschindigkeit

Diese werde ich ebenfalls in JS implementieren, und auf einem Canvas Element visualisieren.

0.2 KI Training in Python (verschiedene Schwierigkeitslevel): Die KI wird eine parametrische Funktion sein, dessen Parameter das im Training optimiert werden. Ziel ist es, dass sie den Zustandsvektor als Input nimmt, und als Output die beiden (stetigen) Werte für die Thrusters gibt. Dafür werde ich ein feedforward network (FFN) mit PID outputs trainieren. Das bedeutet, dass ein manueller algorithmus beispiele von optimalen Output Sequenzen (der Physik-Logik folgend) generiert, und diese als "Vergleich" für die KI gebraucht werden, um dessen Verslust zu bestimmen. Dieser Verlust wird mit MSE berechnet, und mit Adam Optimisation schrittweise optimiert. Damit lernt die KI, was für Werte die Dronenantriebe bekommen sollten, basierend auf deren Zustand.

Das Resultat wird ein Set von Parametern sein, das ich für die Inferenz der KI in JS laden kann, und somit nur die Inferenz (die deutlich simpler als das Training ist) in JS ausgeführt wird.

Da die KI auf Zufällig generierten Beispielsequenzen trainiert wird, sollte die Anzahl solcher Beispiele die Performance der KI beeinflusse. Dadurch kann ich verschieden gute KIs speichern, indem ich die Qualität und Quantität der Trainingsdaten variiere. Diese weredn im Game die verschiedenen Schwierigkeitslevels bilden.

0.3 Inferenz in JS: Wie angedeutet, wird die Inferenz in JS implementiert (in Python würde sie ebenfalls für Debugging implementiert). Das bedeutet, dass die Architektur (als Funktion mit noch unbestimmten Parametern zu verstehen) zuerst definiert, und dann die von Python gespeicherten Parameter laden wird. Diese Parameter müssen nicht privat sein, und können auf der Client-Side geladen werden (async). Somit wird eine Instanz eines trainierten KI Models in JS geladen. Diese kann dann angewendet werde, um Zustand->Antriebwerte transformationen durchzuführen, um die Drone zu steuern.

0.4 Visualisierung in Canvas: Das Verhalten der Dronen soll auf einem Canvas Element visualisiert werden, mit verschiedenen Farben / Designs für die Dronen. Die Antriebe und desser nummerischen Werte sollen ebenfalls visualisiert werden.

Der Hintergrund wird in Sektion 0.6 genauer beschrieben.

0.5 User Inputs für Steuerung (Handy + Laptop): Damit ein User gegen die KI-gesteuerte Drone antreten kann, müssen Input möglichkeiten für User bestehen, sowohl auf dem Handy als auch auf grösseren Screens mit externer Tastatur. Für die beiden Antriebe (j, k) werden zwei Tasten, UP/DOWN oder A/D, die Inputs geben. Anstatt stetige Werte zu generieren, werden dabei binäre Werte (0, 1) generiert, wobei User die Tasten beliebig clicken oder nicht clicken können, um die Drone zu kontrollieren.

0.6 Zufällig generierte Umgebung (bspw. Bilder): Um die Umgbung (in Canvas) spannend zu machen, soll der Hintergrund (die Map) des Spiels Zufällig generiert werden, d.h., es soll ein vorbestimmtes Set an Features (visuelle Objekte als PNGs) Anzeigen, bspw. sollen Checkpoints ein bestimmtes Aussehen haben, und der Rest wird als Weltall dargestellt (bspw. mit Weltraumobjekten, Dronen sind Raumschiffe, Planete könne Teil der Map sein).

Um dies effektiv zu machen, soll die von Usern kontrollierte Drone im Mittelpunkt des Screens

bleiben, und der Hintergrund soll sich bewegen, um das Vorkommen einer unendlich grossen Map zu erzeugen.

0.7 Drag & Drop Interface für eigene Funktion: Sollte das eigene interagieren mit dem Spiel anstrengend, oder an den eigenen Fähigkeiten gezweifelt werden, soll es auch die Möglichkeit geben einen eigenen Algorihmus zu bauen, um, im Sinne des Programmierens, selber weniger machen zu müssen. Somit können KIs oder Users selbst gegen eigenen Algorithmen antreten.

Dazu werden die Elemente des Zustandsvektors (Input der KI), Operatoren, und Float Werte als Ziehbare HTML Elemente dargestellt, wobei die User diese selbst selbst Elemente hinzufügen / entfernen (Drag & Drop), Operatoren und Faktoren definieren, und eine (deterministische) Funktion zur Berechnung der Antriebe erstellen können. Im gegensatz zur Probabilisitischen KI wird diese Funktion (bspw. $a * geschwindigkeit_x - b * winkel$ ...) im Spiel getestet, sodass User die Parameter der Funktion anpassen (quasi debuggen) können.

0.8 Game-Logik (Schwierigkeitslevels, Scores, Sieger, Start/Pause/Quit, Menu): Die oben genannten Features sollen möglichst übersichtlich implementiert und verbunden werden. Dazu soll es verschiedene Tabs und ein Menu geben.

Tabs: Game, Scores, My PID

Menus: "Game": Menu zur Auswahl von Input Art (Keyboard / PID), der Schwierigkeit, Länge des Spiels, "Play", "Pause", "Quit" Button, Input möglichkeiten für User "Scores": Vergangene Punkte, Wins/Defeats, Art von Spieler (Mensch/PID) "My PID": Das interface, um den eigenen Algorithmus zusammenzustellen

Die Navigation für die Tabs soll dabei immer sichtbar sein.

2

2.1 Backend / Login

Sollte die Zeit noch reichen, werde ich auf einem Raspberry Pi Server als Backend ein simples Login für User erstellen, mit Email, Passwort, und Username. Mit einem Userobjekt können dann folgende Werte assoziiert werden:

- Erreichte Scores (User, PID, Schwierigkeitslevel) - Index auf dem Leaderboard - Skins/Outfits der Drone

Ebenfalls werden die Algorithmen eines Users als Option gelistet, bei der Wahl vom Gegner (KIs, PIDs)

1.2 Skins/Outfits für Drone (automatisch angewendet, abhängig von Scores, Siegen, Schwierigkeitslevels) Das Bild, das das Aussehen der Drone bestimmt soll Stufenweise abhängig vom erreichten Top Score eins Users sein, wobei dieses Aussehen ("Skin" oder "Outfit") mit höheren Top Scores auch "besser aussehen" soll, obwohl das eine subjektive Sache bleibt.

1.3 Leaderboard: Das Leaderboard soll die Usernamen und deren Top Scores anzeigen. Diese sollen geordnet augelistet werden, sodass der die besten Spieler und Algorithmen sichtbar sind. Es wird also kategorisiert, ob die Scores von Usern oder eigenen Algorithmen erreicht wurden.

1.4 Benchmarks für Algorithmen Die eigen erstellten Algorithmen sollen als Option nebst den KIs hochgeladen werden, damit andere User ihren eigene Algorithmus gegen den von anderen Usern testen können (und das Rennen visuell anzeigen lassen).

Bemerkungen

Für die Implementierung ohne Backend sollten Scores, Algorithmen, geladene KI modelle in der Browser-Cache bleiben, um diese beim Neuladen nicht zu verlieren.