AI Pilots

Autor: Kaito Soga | Fach: IN29 | Sprachen: JS + Python | Datum: 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 an Checkpoints.

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

Zusätzlich kann ein User durch das Bauen einer Funktion, die den Zustandsvektor als Input nimmt und als Output die Antriebswerte der Drone zurückgibt, einen eigenen Controller erstellen. Dieser kann ebenfalls gegen die KI oder den User selbst antreten.

Die Drone selbst unterliegt realistischen physikalischen Eigenschaften (Schwerkraft, Momentum, Beschleunigung, Rotation (Winkelgeschwindigkeit)), 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 das Innere der KI visualisieren, um Usern eine Einsicht in deren Funktion zu geben. Der Fokus liegt dabei nicht auf dem Game, sondern darauf, die KI durch intuitive Visualisierung zu verstehen und in Echtzeit zu testen.

Für mehr Interaktion werde ich im Backend auf einem Raspberry Pi ein simples Login erstellen, um Usern einen Top Score, einen Index auf einem öffentlichen Leaderboard sowie vom Top Score abhängige Outfits/Styles für die Drone zuzuweisen und eigene Controller zu teilen. Andernfalls ist es mein Ziel, Teil 1 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 Library PyTorch, die es in JS nicht gibt), werde ich auch die Physik-Logik vorerst in Python implementieren. Diese soll beinhalten:

- Schwerkraft
- Momentum
- Beschleunigung
- Rotation (Winkel)
- Rotationsgeschwindigkeit

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

1.2 KI Training in Python (verschiedene Schwierigkeitslevel)

Die KI wird eine parametrische Funktion sein, deren Parameter im Training optimiert werden. Ziel ist es, dass sie den Zustandsvektor als Input nimmt und als Output die beiden (stetigen) Werte für die Antriebe zurückgibt. Dafür werde ich ein Feedforward Network (FFN) mit PID-ähnlichen 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 deren Verlust zu bestimmen. Dieser Verlust wird mit MSE berechnet und mit Adam Optimisation schrittweise optimiert. Damit lernt die KI, welche Werte die Dronenantriebe erhalten sollten, basierend auf dem Zustand der Umgebung.

Das Resultat wird ein Set von Parametern sein, das ich für die Inferenz der KI in JS laden kann, sodass 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 beeinflussen. Dadurch kann ich verschieden starke KIs speichern, indem ich die Qualität und Quantität der Trainingsdaten variiere. Diese werden im Game die verschiedenen Schwierigkeitslevel bilden.

1.3 Inferenz in JS

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

1.4 Visualisierung in Canvas

Das Verhalten der Dronen soll auf einem Canvas Element visualisiert werden, mit verschiedenen Farben und Designs für die Dronen. Die Antriebe und deren numerische Werte sollen ebenfalls visualisiert werden.

Der Hintergrund des Canvas wird in Sektion 1.6 genauer beschrieben.

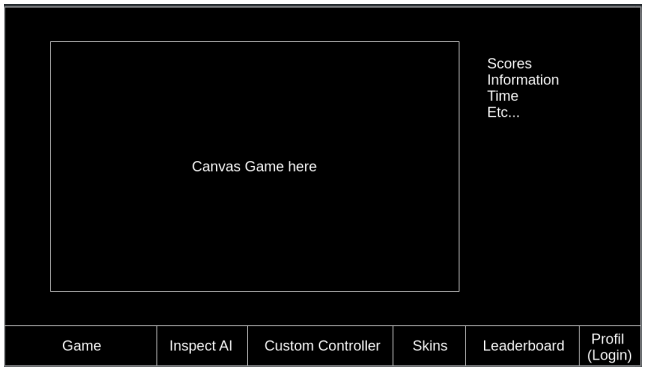
1.5 User Inputs für Steuerung (Handy + Laptop)

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, verwendet, die Inputs liefern. Anstatt stetige Werte zu generieren, werden dabei binäre Werte (0, 1) erzeugt, wobei User die Tasten beliebig klicken oder nicht klicken können, um die Drone zu kontrollieren.

1.6 Zufällig generierte Umgebung (bspw. Bilder)

Um die Umgebung (im 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. Beispielsweise sollen Checkpoints ein bestimmtes Aussehen haben, und der Rest wird als Weltall dargestellt (z. B. mit Weltraumobjekten; Dronen sind Raumschiffe, Planeten können Teil der Map sein).

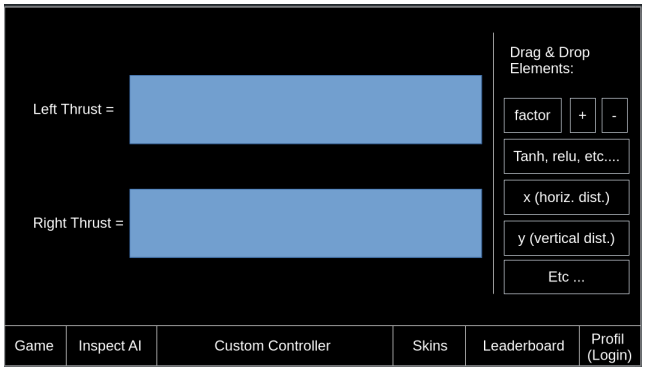
Um dies effektiv zu machen, soll die von Usern kontrollierte Drone im Mittelpunkt des Screens bleiben, und der Hintergrund soll sich bewegen, um das Vorkommen einer scheinbar unendlich grossen Map zu erzeugen.



1.7 Drag & Drop Interface für eigene Funktion

Sollte das eigene Interagieren mit dem Spiel anstrengend sein oder an den eigenen Fähigkeiten gezweifelt werden, soll es auch die Möglichkeit geben, einen eigenen Algorithmus zu bauen, um im Sinne des Programmierens selbst weniger machen zu müssen. Somit können KIs oder User selbst gegen eigene Algorithmen antreten.

Dazu werden die Elemente des Zustandsvektors (Input der KI), Operatoren und Float Werte als ziehbare HTML Elemente dargestellt, wobei die User diese Elemente hinzufügen oder entfernen (Drag & Drop), Operatoren und Faktoren definieren und eine (deterministische) Funktion zur Berechnung der Antriebe erstellen können. Im Gegensatz zur probabilistischen KI wird diese Funktion (bspw. $a \times Geschwindigkeit_x - b \times Winkel \dots$) im Spiel getestet, sodass User die Parameter der Funktion anpassen (quasi debuggen) können.



1.8 Game-Logik (Schwierigkeitslevels, Scores, Sieger, Pause/Quit, Menü)

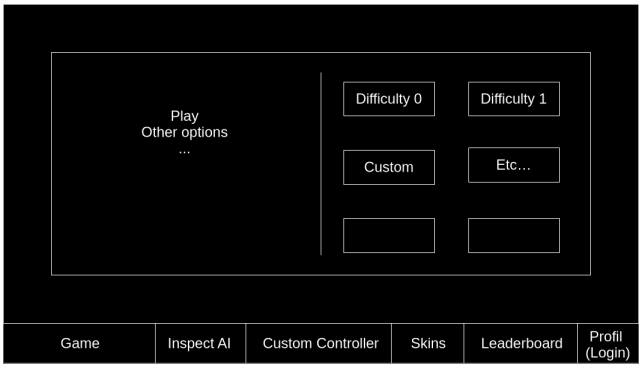
Die oben genannten Features sollen möglichst übersichtlich implementiert und verbunden werden. Dazu soll es verschiedene Tabs und ein Menü geben.

Tabs: “Game”, “Inspect AI”, “Custom Controller”, “Skins”, “Leaderboard”, “Profile”

Menüs:

- “Game”: Menü zur Auswahl der Input Art (Keyboard / Custom Algorithm), der Schwierigkeit, der Länge des Spiels sowie “Play”, “Pause” und “Quit” Buttons und Input-Möglichkeiten für User.
- “Scores”: Vergangene Punkte, Wins/Defeats, Art des Spielers (Mensch/Custom Algorithm).
- “Custom Controller”: Das Interface, um den eigenen Algorithmus zusammenzustellen.

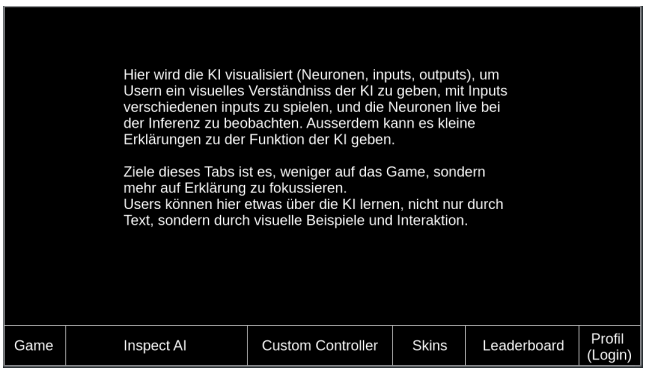
Die Navigation für die Tabs soll dabei immer sichtbar sein.



2

2.1 Erklärung / Inspector

Es soll zusätzlich die Möglichkeit geben, die KI zu "untersuchen", das heisst, die Neuronen der KI werden visualisiert, User können Inputs der KI modifizieren und die daraus resultierenden Outputs beobachten. Es soll ebenfalls angezeigt werden, welche Inputs wie viel Einfluss auf einen bestimmten Output hatten (Explainability Methoden). Ausserdem wird kurz der mathematische Aufbau der KI erklärt und soll Usern ein intuitives, visuelles Tool geben, um das Innere der KI in Echtzeit zu beobachten.



2.2 Backend / Login

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

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

Ebenfalls werden die Algorithmen eines Users als Option gelistet, bei der Wahl des Gegners (KIs, Custom Algorithm).

2.3 Skins/Outfits für Drone

Das Bild, das das Aussehen der Drone bestimmt, soll stufenweise abhängig vom erreichten Top Score eines Users sein, wobei dieses Aussehen ("Skin" oder "Outfit") mit höheren Top Scores auch "besser aussehen" soll, obwohl das eine subjektive Sache bleibt.

2.4 Leaderboard

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

2.5 Benchmarks für Algorithmen

Die selbst erstellten Algorithmen sollen als Option neben den KIs hochgeladen werden, damit andere User ihren eigenen Algorithmus gegen den von anderen Usern testen können (und das Rennen visuell anzeigen lassen).

Bemerkungen

Für die Implementierung ohne Backend sollten Scores, Algorithmen und geladene KI-Modelle im Browser Cache bleiben. Ausserhalb von Canvas soll, falls angemessen, Angular verwendet werden.