Dokumentation

TEAM STOLPERSTEIN

Einreichung von Jonas Huhndorf und Sandra Bernich
NORDAKADEMIE - HOCHSCHULE DER WIRTSCHAFT

Inhaltsverzeichnis

Ansatz und Idee

Zu Beginn wurden unterschiedliche Lösungen skizziert. Darunter ein Programm, welches mehrere Schritte im Voraus berechnet, eine Artifical Intelligence (AI) die über ein Belohnungssystem angelernt wird und einige Abwandlungen der genannten Ideen. Neben der Kompliziertheit der Ideen fand in der Betrachtung auch eine positive Einschätzung der Kenntnisse von Programmiersprachen und dem Hintergrundwissen zu AI statt. Zusätzlich fand auch die benötigte Ressource Zeit und mögliche weitere Teammitglieder Einfluss in die Betrachtung. Es ging dann zu zweit los mit dem Motto „Das schaffen wir schon“.

Im folgendem wurde sich mit der Entwicklung einer AI beschäftigt und parallel dazu wurde die Verbindung zum Websocket aufgebaut, sowie eine Logik zum Senden einer Antwort gebaut. Auf dessen Grundgerüst wurde weiter programmiert werden.

Im weiteren Verlauf, entschieden wir uns die Aufgabe nicht mit einer AI zu lösen, sondern manuell zu rechnen. Das Dokument ist die Dokumentation, viel Spaß beim Lesen.

Umsetzung

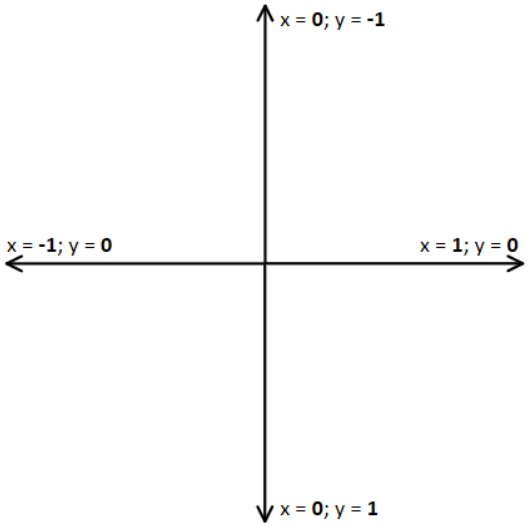
Das Programm kann eigenständig das Spiel spe_ed spielen. Geschrieben ist es in Python. Voraus berechnet wird jeweils der aktuelle Schritt des Bots. Es kommt nur zu Änderungen der Richtung oder der Geschwindigkeit, sofern Hindernisse oder möglicherweise Spieler in der Nähe sind. Insgesamt handelt es sich um, einen eher passiven Bot, welcher kaum Taktik beinhaltet, um andere Spieler zu töten. Durch das passive Verhalten ist es in der Testphase oft gelungen lange zu überleben und zu gewinnen.

Architektur

Das Projekt wurde thematisch geordnet in die Verbindung zum Websocket, die Berechnung des nächsten Schrittes, in benötigte Datenklassen und das Testen. Relevante Dateien wurden bewusst nicht in Ordner gepackt, da diese das zurechtfinden im Projekt erleichtern.

Es wurde sich an die allgemeingültigen Coding Conventions von Python gehalten. Der Umfang des Projektes ist überschaubar, benötigte Libaries sind in den Requirements, mit den entsprechenden Versionen, zu finden.

Strategie



Die Grafik auf der rechten Seite zeigt die Grundidee der Berechnung. Wir mappen die aktuelle Bewegungsrechnung auf das Modell und können somit für jede mögliche Bewegung, die wir ausführen können, alle Kombinationen durchtesten, die von den anderen Spielern ausgeführt werden können. Vorher werden die bereits ausgeschiedenen Spieler und diejenigen, die zu weit weg sind, aus der Berechnung entfernt. Anschließend wird evaluiert, welcher Schritt am besten ist. Dafür wird primär derjenige genommen, bei dem wir mit der geringsten Anzahl sterben. Sollten die Anzahl gleichwertig sein, wird überprüft, in welchem Zug alle Gegner zusammen öfters sterben. Wenn diese Werte immer noch identisch sind, dann wird der erste Wert aus der internen Schrittliste (siehe config.json) genommen.

Diese Berechnungsweise führt dazu, dass sich der Bot in der Regel am Rand des Spielfeldes aufhält. Wir haben auch getestet was passiert, wenn der Bot in die Mitte des Spielfeldes geht. Dort überleben wir jedoch bei weitem nicht so oft. Daher haben wir die Kalkulation wieder zurück auf das ursprüngliche Verhalten geändert. Ein bisschen Gelassenheit ist angesagt.

Test

Die einzelnen Funktionen des Programmes sind über Unit-Test getestet. Dazu gehören:

- Alle möglichen Züge überprüfen
- Das Mappen eines Spielers
- Sterbemöglichkeit für uns und die anderen Spieler bei einem festgelegten Zug berechnen
- Berechnung unseres Zuges

Die Verbindung, sowie die API des Websockets wurde manuell durch Beobachten getestet, da der aktuelle Aufbau des Programmes keine Unit-Test in dem Bereich zu lässt.