CAMP MINECRAFT AI

– Einführung –

Kapitel 1 – Willkommen im Camp!

Habt ihr euch jemals gefragt, wie sich die Figuren und die Spielwelt um den Spieler herum verhalten und nach welchen Mustern und Regeln diese auf den Spieler und seine Handlungen reagieren und sich verändern? Warum bewegt sich ein Schaf in Minecraft so wie es sich bewegt? Warum legt ein Huhn Eier, und vor allem, warum rennen die Tiere, wenn sie geschlagen von dem Spieler weg, aber die Monster auf den Spieler zu – und warum nur in einem bestimmten Bereich und nur für eine bestimmte Dauer?

Ein ganz herzliches Willkommen von mir – Matthias – zu eurem Camp bei den Forscher Freunden. In diesem Camp zum Thema Künstliche Intelligenz (kurz: „KI“; auf Englisch: Artificial Intelligence – kurz: „AI“), werden wir uns gemeinsam die Funktionsweise und die Einbindung von Künstlicher Intelligenz – zumindest sehr stark vereinfacht – am Beispiel von den Entitäten in Minecraft anschauen und erarbeiten. Dazu zählen, wie am Anfang bereits kurz erwähnt, viele Aspekte des Verhaltens der Tiere und Monster im Spiel. Dabei werden wir uns auch kurz mit dem Unterschied von Algorithmen und Pseudo-Algorithmen auseinandersetzen, und uns anschauen, was genau die Intelligenz im Namen Künstliche Intelligenz bedeutet.

Dabei werde ich euch durch dieses Camp begleiten und euch alle möglichen spannenden Infos und Tipps mit an die Hand geben – also lest diese Dokumentation sorgfältig, damit ihr möglichst einfach alles versteht, und damit später weniger Fehler im Code entstehen.

Kapitel 1.1 – eine Einführung in Algorithmen

Wie bereits erwähnt, wollen wir uns einmal mit dem Begriff Algorithmus, und was sich dahinter versteckt auseinandersetzen, denn dieser spielt in der Informatik, und vor allem bei künstlicher Intelligenz, eine ganz wichtige Rolle.

Ein Algorithmus ist im Grunde genommen eine Reihe von Anweisungen, die Schritt für Schritt befolgt werden, um ein bestimmtes Problem zu lösen oder eine Aufgabe zu erledigen. Stell dir vor, du möchtest einen Kuchen backen. Das Rezept, das dir sagt, welche Zutaten du brauchst und welche Schritte du unternehmen musst, um den Kuchen zu backen, ist wie ein Algorithmus. Jede Zutat und jeder Schritt ist wichtig, damit der Kuchen am Ende richtig gelingt.

In der Informatik funktioniert ein Algorithmus ähnlich. Er ist eine Reihe von klaren und präzisen Anweisungen, die ein Computer befolgt, um eine bestimmte Aufgabe zu erledigen, wie zum Beispiel ein Problem zu lösen oder Daten zu verarbeiten. Algorithmen sind die Bausteine der Programmierung und helfen Computern dabei, Aufgaben effizient und genau auszuführen.

Bei der Künstlichen Intelligenz (KI) spielen Algorithmen eine besonders wichtige Rolle. KI nutzt spezielle Algorithmen, um "intelligent" zu handeln und Entscheidungen zu treffen. Diese Algorithmen erlauben es der KI, aus Erfahrungen zu lernen, Muster zu erkennen und Vorhersagen zu treffen. Zum Beispiel in Minecraft: Die Art und Weise, wie ein Schaf sich bewegt oder ein Monster auf dich reagiert, basiert auf Algorithmen, die bestimmen, wie diese Figuren im Spiel agieren. Ohne diese Algorithmen wüsste der Computer nicht, wie sich die Tiere und Monster verhalten sollen.

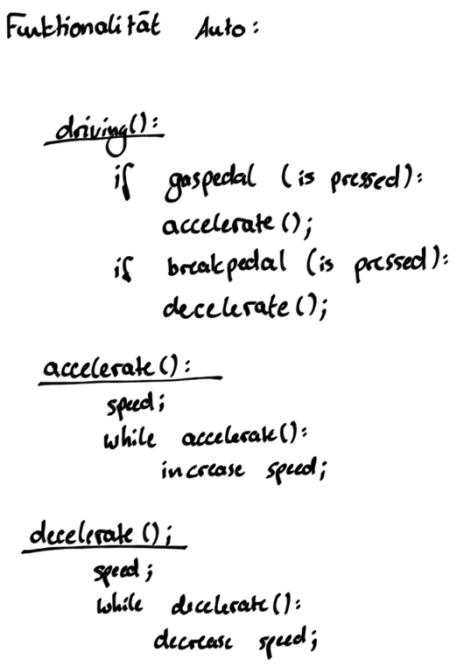
Zusammengefasst: Ein Algorithmus ist wie ein Kochrezept mit klaren Schritten, die ein Computer befolgen muss, und bei KI hilft er der Technik, "intelligent" zu agieren und Entscheidungen zu treffen.

Kapitel 1.2 – Pseudocode und die erste Aufgabe

In der Informatik verwenden wir gerne so genannten Pseudocode. Das bedeutet, dass wir einen Algorithmus erst mal ohne die passenden Fachbegriffe und Bezeichnungen einer spezifischen Programmiersprache schreiben, sondern in sehr einfachem Englisch, jeden Schritt, ohne auf Grammatik zu achten, sinngemäß in die gewollte Reihenfolge bringen.

Nehmen wir als Beispiel an, dass wir einen simplen Algorithmus für ein Auto schreiben wollen – sehr stark vereinfacht, ohne dass wir genau wissen müssen, wie wir das tatsächlich in das Programm am Ende integrieren würden.

Angewandt, sieht das Ganze dann in etwa wie folgt aus:



Dabei sehen wir, dass wir eine einfache Funktion fürs Fahren definiert haben, namentlich „driving()“. Diese Funktion schaut, ob das Gaspedal oder das Bremspedal betätigt wird, und führt dann entweder die Funktion „accelerate()“ oder „decelerate()“ aus, also beschleunigen oder abbremsen. Was genau diese Funktionen in einem Programm dann machen, beziehungsweise welche Attribute dann noch gebraucht werden, und wie das dann wirklich implementiert aussieht, ist in diesem Schritt völlig egal. Es geht nur darum, zu überlegen, welche Funktionalität man ganz grob umgesetzt haben will.

Diese Art an Code schreiben, wird uns in diesem Camp noch ein paar Mal über den Weg laufen – deshalb würde ich euch bitten, das einfach einmal zu üben an einem einfachen Beispiel.

Aufgabe 1

Wir hatten nun das Beispiel mit einem Auto – aber im Straßenverkehr gibt es auch noch andere Teilnehmer. Und diese ganzen Teilnehmer müssen sich an Regeln halten, damit niemand zu schaden kommt. Schreibe einen Pseudocode für das Verhalten von Fußgängern an einer Ampel an einer Kreuzung. Eine grobe Anleitung, welche Aspekte du beachten musst, findest du hier:

The Pedestrian is only allowed to walk across the crossing, if the pedestrian light is green and the traffic has come to a halt. Before crossing, the pedestrian must check if the traffic has actually stopped, by looking to the right, the left and the right again, and only then walk across. The pedestrian light will only turn green, after it was activated by a pedestrian.

Auf der letzten Seite der Dokumentation findest du eine mögliche Lösung, mit der du deinen Pseudocode vergleichen kannst, sobald du damit fertig bist.

Kapitel 1.3 – Die Aussichten und der Wochenplan:

Nun da wir die wichtigsten Punkte vorab geklärt hätten, schauen wir uns doch mal den Fahrplan für die weitere Reise diese Woche an. Heute wollen wir direkt einmal die Basis für die restliche Woche legen, also gehen wir im Folgenden noch ein paar weitere Basics durch, schauen uns die Funktionen von den Programmen an, mit denen wir arbeiten werden, aber vor allem entscheiden wir uns schon einmal für die weitere Richtung, die wir gehen wollen – dabei meine ich natürlich, ob ihr euch mit den Monstern in Minecraft auf die Dunkle Seite der Macht stellen wollt, oder lieber ein friedliches süßes Tier kreieren wollt. Dabei solltet ihr ein paar Aspekte im Hinterkopf behalten:

- Monster haben eine deutlich komplexere KI als die Tiere, dafür bieten Tiere mehr kreative Vielfalt in der Welt von Minecraft
- Ihr könnt euch sowohl bei den Tieren als auch bei den Monstern, an bestehenden Kreaturen entlanghangeln, und Code übernehmen, jedoch sollte euch bewusst sein, dass je komplexer euere Idee ist, desto mehr müsst ihr von dem Code und seiner Funktionsweise auch verstanden haben damit das dann auch funktioniert – Eure Teamer sind nicht dafür da, eure Ideen umzusetzen, sondern helfen euch nur dabei, euch zu betreuen und grundlegend anzuleiten!
- Grundsätzlich können wir die Monster und Tiere in 3 Kategorien unterteilen: Einfache Kreaturen, welche sich am Boden bewegen und einfachen Algorithmen folgen, also eine simple KI verwenden; Moderate Kreaturen, welche komplexere Verhaltensweisen zeigen und auf ganz spezielle Events und Trigger reagieren – also zum Beispiel nur ganz bestimmte Tiere angreifen, oder bestimmtes Angriffsverhalten aufweisen; und den Komplexen Entitäten, welche eine komplexe Form der Bewegung aufweisen, also sich zum Beispiel im Wasser bewegen, oder fliegen können, und dadurch eine komplexe KI und komplexe Algorithmen brauchen.
- Wenn ihr eine komplexe Kreatur euch vornehmt, muss euch klar sein, dass ihr dann eventuell keine funktionierende Entität mit nach Hause nehmen werdet, wenn ihr mit dem komplexen Code und seiner Implementierung nicht klarkommt!

Da wir das nun geklärt hätten, und euch hoffentlich die Konsequenzen eurer Wahl bewusst sind, schauen wir uns mal die Programme an mit denen wir arbeiten werden!