Dokumentation zur Projektaufgabe für das Modul Künstliche Intelligenz für Spiele im Wintersemester 2022/23

Oliver Jakobs

13. Februar 2023

Thema und Motivation

Als Thema für meine Modulaufgabe habe ich mich für **Decision Trees** entschieden.

Einlesen eines Decision Trees aus einer XML-Datei und Frage-Antwort Spiel oder textbasiertes Adventure.

Das Konzept lässt sich neben künstlicher Intelligenz auch auf viele andere Bereiche anwenden. Ein Beispiel ist eine Alternative zu Zustandsautomaten um festzulegen welche Animation gespilet werden soll.

Allgemeine Designentscheidungen

Mein Programm lässt sich in zwei Abschnitte aufteilen. Der erste Teil ist die Implementiation des Decision Tress. Dieser wird aus einer .xml-Datei eingelesen und in eine passenden Datenstruktur geladen. Den zweiten Teil habe ich TreeWalker gennant. Dieser Teil beschäftigt sich mit Kommunikation zwischen dem Nutzer und dem Baum.

Das Programm habe ich bewusst im C-Style programmiert. Das heißt ich habe mich bei der Struktur des Programmes an C gehalten und habe lediglich einzelene Features (z.B. `std::vector` oder `std::variant`) aus C++ verwendet. Der Grund für diese Entscheidung war, dass ich privat hauptsächlich in C programmiere und so am komfortabelsten mit diesem Stil bin. Außerdem habe ich so weniger Aufwand, wenn ich die Implementation anpassen muss, wenn ich Decision Trees in meinen C-Programmen verwenden will.

Entscheidungen zur DecisionTree Implementation

Der DecisionTree wird aus mehreren TreeNodes aufgebaut. So eine TreeNode hat einen Type, einen Namen, einen Wert und eine Liste an Nachfolgern. Der Type gibt an ob es sich um einen Knoten oder um ein Blatt des Baumes handelt, und um was für eine Art Entschiedung es sich handelt. Dabei werden zwei Arten unterscheiden:

Decision
Eine `decision` bildet den Grundbaustein meiner Implementation. An sich lässt sich alle Funktionalität von DecisionTrees nur mit TreeNodes dieser Art umsetzen. Der Wert einer `decision` ist eine Boolean Expression, in die zum Zeitpunkt der Entscheidung eine Variable eingesetzt werden kann.

Option
Eine `option` ist ein Spezialfall der Decision

Entscheidungen zur XML-Datei

Ich bin eigentlich kein Fan vom XML Dateiformat, aber um Baumstrukturen darzustellen ist es die einfachste und verbreitetsten Form.

Der eigentliche DecisionTree besteht aus drei Tags (`decision`, `option`, `final`). Diese Tags entsprechen dem Type des aus dem Element generierten TreeNode. Der `final`-Tag dient lediglich der anschaulicheren Darstellung und im generierten Baum hat jede TreeNode den Type `final`, wenn dieser keine Nachfolger hat. Jedes Element mit einem dieser Tags kann als Attribut einen Namen und einen Wert haben. Der Name gibt den Namen der Entscheidung an, die dieses Element repräsentiert.

Tag: `decision`
Der `decision`-Tag ist der grundlegende Baustein des DecisionTrees.

Für den TreeWalker kann man noch einen weiteren Tag (`prompt`) angeben, der dem TreeWalker zusätzliche Informationen gibt. Mit deisem Tag lässt sich angeben welche Frage der TreeWalker für eine Entscheidung stellen soll.

Bedienung

Um das Programm zu starten muss die Datei `MeshSimplifier.exe` ausgeführt werden. Dafür wird neben der `.exe` auch der gesammte Inhalt des Orderns `./res/` benötigt.

Die Ausführung des Programms findet in einem `read-eval-print loop (REPL)` statt. Dem Nutzer wird eine Frage gestellt, auf die er dann antworten muss. Ist die Antwort ungültig, das heißt die Antwort entspricht nicht der erwarteten Form oder ist keine der möglichen Antwortmöglichkeiten, so wird der Nutzer erneut aufgefordert zu antworten.

Bibliotheken

Da dieses Projekt textbasiert ist benötige ich nur eine Möglichkeit XML-Dateien einzulesen. Für diese Funktionalität habe ich mich für die Bibliothek **TinyXML-2** (`https://github.com/leethomason/tinyxml2`) entschieden.

Build

Zur Project Generation benutze ich Premake (`https://premake.github.io/`). Ich habe die entsprechenden Scripts meiner Abgabe beigefügt. Falls also Probleme mit der VisualStudio Solution autreten sollten, kann mit dem folgenden Befehl das Projekt neu generiert werden und so die Probleme hoffentlich gelöst werden:

```
.\premake\premake5.exe [action]
```

Für mein Projekt habe ich `vs2019` als `action` verwendet. Andere Möglichkeiten sind hier `https://premake.github.io/docs/Using-Premake` aufgelistet.