4 Gewinnt Dokumentation

Hannes Höttinger - 1510585004

FH Technikum Wien, Game Engineering und Simulation, Wien, AUT

Abstract — Dieses Dokument dient als Dokumentation für die implementierte Version von 4 Gewinnt mittels MiniMax Algorithmus und als eine kurze Beschreibung der Programmausführung.

1 Umsetzung

Es sind drei verschiedene Versionen des Algorithmus implementiert:

1. Klassischer MiniMax
2. NegaMax
3. NegaMax mit Alpha Beta Pruning

Die Algorithmen basieren auf der gleichen Basis und liefern die gleichen Ergebnisse. NegaMax ist eine vereinfachte Form des MiniMax Algorithmus (hier wird immer maximiert). Mittels Alpha Beta Pruning kann die Rechenzeit deutlich verkürzt werden. Alternierend führt der Spieler bzw. der Computer einen Zug aus. Die Algorithmen wurden rekursiv aufgebaut, und führen für jeden möglichen Zug (Zug = Münze in Spalte: 7 Züge möglich) eine Tiefensuche durch. In der Tiefensuche wird für die möglichen Züge ein Spielfeldwert berechnet. Dieser gibt Aufschluss über die Wertigkeit. Ist das Spielfeld positiv → besser für Computer, ist es negativ → besser für Spieler. Die Berechnung erfolgt mittels einer Scorefunktion die in ein Array der Größe 9 das vorliegende Feld bewertet. Es wird horizontal, vertikal und in den beiden Diagonalen ausgewertet. Je nach Anordnung der Münzen wird ein Feld des Arrays upgedatet (+1). Liegen beispielsweise drei Münzen des Computers nebeneinander ergibt dies einen Score von +3. *counters[score + 4]++;* greift somit auf das Feld counters[7] zu und erhöht diesen Wert. Ergibt Das Array ist wie folgt aufgebaut:

1. counters[0] → Spieler gewinnt.
2. counters[1] → Spieler hat 3 Münzen nebeneinander.
3. counters[1] → Spieler hat 2 Münzen nebeneinander.
4. counters[1] → Spieler hat 1 Münze.
5. counters[4] → Leeres Feld.
6. counters[5] → Computer hat 1 Münze.
7. counters[6] → Computer hat 2 Münzen nebeneinander.
8. counters[7] → Computer hat 3 Münzen nebeneinander.
9. counters[8] → Computer gewinnt.

Nachdem alle Felder überprüft wurden, kann mit diesem Ergebnis sehr einfache eine Heuristik aufgestellt werden. Liefert counters[0] oder counters[8] einen Wert, bedeutet das, dass ein Siegspielzug möglich ist. Alle anderen Zugmöglichkeiten bekommen eine Wertigkeit. Drei Münzen sollten eine sehr hohe Wertigkeit bekommen. Absteigend alle anderen Möglichkeiten. Dies wurde wie folgt umgesetzt:

```
if (counters[0] != 0)
        return blueWins;
    else if (counters[8] != 0)
        return redWins;
    else // heuristic
        return
        counters[5] + 2 * counters[6] + 5 * counters[7] -
        counters[3] - 2 * counters[2] - 5 * counters[1];
```

Im Menü können folgende Einstellungen getroffen werden:

1. Welcher Spieler beginnen darf: YOU oder CPU
2. Welcher Algorithmus verwendet werden soll
3. Schwierigkeitsgrad: Tiefe der Suche

Die Tiefensuche kann durch den Schwierigkeitsgrad beschränkt werden. Easy bedeutet eine Tiefe von 1. Das bedeutet der Algorithmus wirft jeweils in jede Spalte eine Münze ein und bewertet das Spielfeld. Das heißt es kommt zu keiner Vorrechnung von möglichen Spielfeldern und der Computer kann leicht besiegt werden. Medium bedeutet eine Tiefe von 5. Das heißt es wird hier schon ein sehr groß verzweigter Baum aufgebaut und der Computer berechnet sich mögliche Varianten und wählt den besten Zug aus den möglichen Spielfeldern → bester Scorewert des Feldes für Computer. Ab der Tiefe 7 ist der Computer kaum schlagbar. Hard wurde auf die Tiefe 9 gestellt. Hier kann man bei MiniMax und NegaMax bereits eine sehr lange Rechenzeit beobachten. Wechselt man jedoch auf Alpha Beta Pruning wird hier deutlich die Rechenzeit verringert, mit dem exakt gleichen Ergebnis.

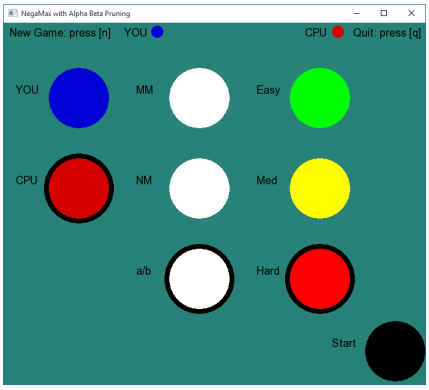

Abbildung 1: Game Menü mit Auswahlmöglichkeiten für das Spiel

1.1 Steuerung

Eine Münze kann mit der linken Maustaste gesetzt werden und wird dann automatisch auf das richtige Feld "gedropped". Anschließend berechnet der Computer seinen Zug mit dem gewählten Algorithmus. Im Spiel kann jedoch noch zwischen den Algorithmen gewechselt werden. Dies wird mit den Zahltasten 1 - 4 der Tastatur realisiert. Auswahlmöglichkeit 4 beschreibt den Algorithmus NegaMax ohne Tiefenbeschränkung. Dieser Wert führt jedoch zu keinem spielbaren Ergebnis aufgrund der Rechenzeit und wurde nur für Demonstrationszwecke implementiert. Mit [q] = quit bzw. [n] = new game wurden weitere In-Game Einstellungen implementiert.

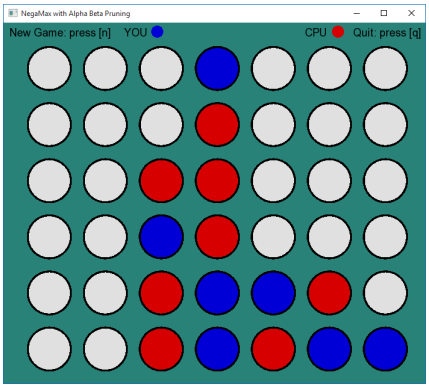

Abbildung 2: Spielfeld von 4 Gewinnt umgesetzt mittels SFML

1.2 Grafische Ausgabe

Für die grafische Ausgabe wurde *SFML (Simple and Fast Multimedia Library* (`http://www.sfml-dev.org/`)verwendet. Die verwendete Version befindet sich im Source Folder.

2 Programmausführung

In dem Abgabeordner im Verzeichnis *bin* befindet sich ein .EXE Files (x64), **MiniMax4.exe** das Hauptprogramm, welches die gesamte Funktionalität beinhaltet.

Weiters wurde eine Static Library für die Algorithmen angelegt. Diese ist als **MiniMax_Lib** im Verzeichnis *bin* zu finden.

Die notwendigen DLLs für SFML sind inkludiert. Die Solutions (MiniMax4 Main + Library) sind mit relativen Pfaden angegeben, somit sollte das Programm ohne weitere Einstellung ausgeführt bzw. neu kompiliert werden können.

3 NICE-TO-HAVE

Folgende Punkte wurden implementiert:

1. Implementierung eines GUI mit im UI wählbarem Schwierigkeitsgrade zu Spielbeginn.