Jesse Strathmann, Finn Geffken
DQI 17

Frau Felsmann, PP

TicTacToe

Programmier-Praktikum

von: Jesse Strathmann und Finn Geffken
Abgabe: 22.12.2019

Jesse Strathmann, Finn Geffken
DQI 17

Frau Felsmann, PP

Inhaltsverzeichnis

Jesse Strathmann, Finn Geffken
DQI 17
Frau Felsmann, PP

1. Projektbeschreibung

Das Projekt ist ein JavaFX Programm in dem man das bekannte Spiel Tic-Tac-Toe spielen kann. Bei der Entwicklung galt es sich nach dem Entwurfsmuster „Model-View-Controller“ oder kurz MVC zu richten, um eine modulare, saubere Softwarearchitektur zu erreichen und das Anwenden von Entwurfsmustern zu üben.

Die Mindestanforderungen umfassten eine Version des Spiels in der ein 3x3 Spielfeld mit zwei Spielern gespielt werden kann. Dabei sollen drei gleiche Horizontale, Vertikale oder Diagonale zu einem Sieg führen. Außerdem soll das Spiel über eine grafische Oberfläche gespielt werden, welche mit JavaFX entwickelt werden soll. Zum Einstellen von Optionen soll es hier ein Menü geben.

Als Erweiterungen sollte das Spielfeld in beliebiger, quadratischer Größe generiert werden können und es sollte eine Bestenliste geben.

Bonuspunkte gab es, wenn man gegen einen Computergegner spielen kann und eigene Verbesserungen einfließen lässt.

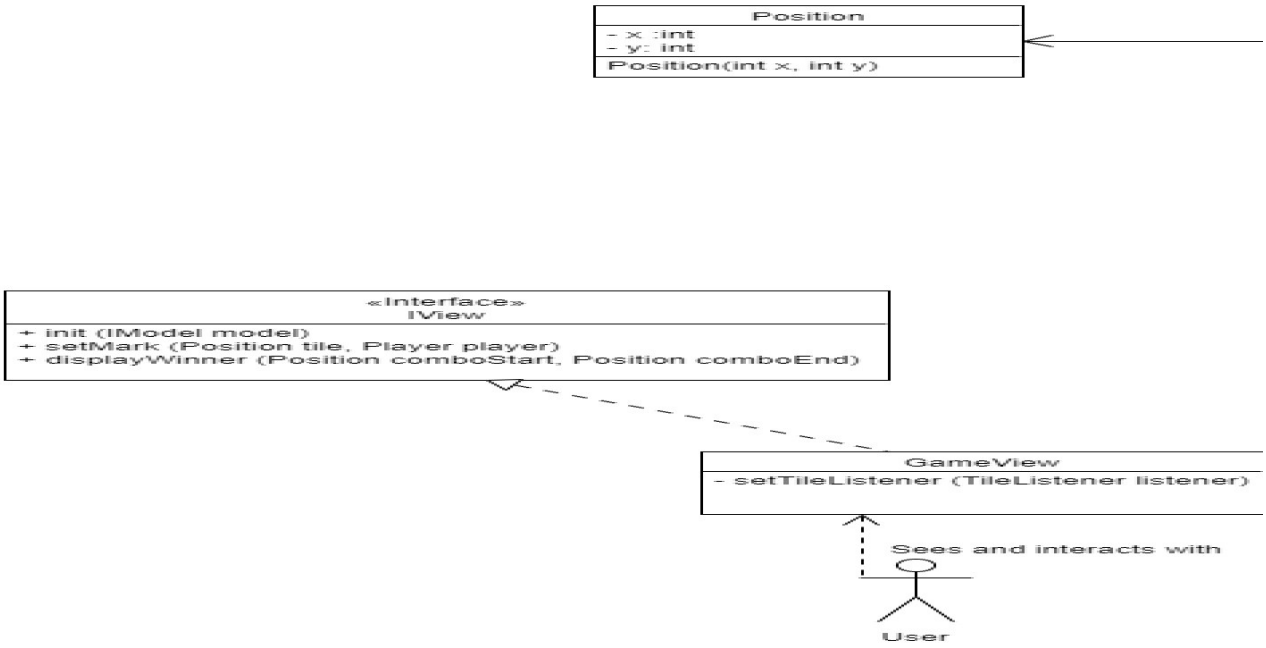
2. Theoretische Überlegungen

Wir haben aufgrund des vergleichsweise geringen Umfangs des Projektes gegen die Scrum-Methode entschieden und uns stattdessen getroffen und das Projekt im Pair-Programming umgesetzt.

Im Pair-Programming haben wir zunächst einen kleinen Prototypen erstellt, welchen wir später als Grundlage nutzten. Basierend auf dem Prototypen haben wir ein UML-Diagramm erstellt.

Wir haben uns außerdem gegen die Entwicklung in Java entschieden und Kotlin bevorzugt, da es zum einen keinen Nachteil bringt und zum anderen wir so eine neue Sprache näher kennenlernen konnten und auch beim Entwickeln Zeit sparten, aufgrund des optimierten Syntax und Umsetzung von Kotlin. Da Kotlin direkt in Java kompiliert wird und Java-Code mit Kotlin verwendet werden kann, war es kein Problem eine JavaFX GUI in Kotlin zu entwickeln.

3. Softwareentwurf



(das UML-Diagramm ist auch im Repository zu finden)

Das obige UML-Diagramm diente als Grundlage. Das fertige Programm weicht davon ab, die Architektur jedoch ist grundlegend gleich.

Es gibt drei Kernelemente in diesem Diagramm: „Game“, „GameController“ und „GameView“. Diese Klassen spiegeln das MVC-Pattern wieder. Game ist das Model, der GameController ist der Controller und GameView ist der View. Bei unseren Recherchen nach einem typischen Aufbau einer MVC-Architektur stellten wir fest, dass es verschiedene Arten gibt.

Wir haben uns dafür entschieden, dass das Model ein Observable ist und der Controller ein Observer. So kann das Model dem Controller Bescheid geben, wenn sich etwas auf dem Spielbrett ändert und der Controller gibt die notwendigen Informationen an den View weiter.

Der Controller hat außerdem einen „TileListener“, welcher als ClickHandler fungiert und im View jedem Feld zugewiesen wird. Bei jedem Klick auf ein Feld wird dem Controller so Bescheid gegeben, welches Feld geklickt wurde und kann so mit dem Model interagieren.

Im Controller ist außerdem auch der Spielablauf geregelt.

Der Computergegner basiert auf dem MiniMax-Algorithmus. Dieser Algorithmus berechnet den besten Zug indem er alle möglichen Züge durchgeht und basierend auf dem Spielzustand nach dem jeweiligen Zug entscheidet, wie gut dieser Zug war. Hat der Computer gewonnen, hat der Zug einen Wert von 10, hat der Gegner gewonnen, einen Wert von -10. Hat niemand gewonnen, hat der Zug einen Wert von 0.

Der Name „MiniMax“ kommt daher, dass der Computer immer den Zug wählt, welcher seinen Punktestand maximiert und den des Gegners minimiert.

4. Projektplanung

4.1 persönliche Zielsetzungen

Jesse

Mein Ziel bei diesem Projekt war es mit der Programmiersprache Kotlin vertrauter zu werden und den MiniMax-Algorithmus erfolgreich zu implementieren. Darüber hinaus wollte ich vertrauter mit dem MVC-Pattern werden, abseits von Android oder Django.

Finn

4.2 Arbeitsaufteilung

Da wir das Projekt im Pair-Programming umgesetzt haben, gab es keine klare Aufteilung der Arbeit in jeder Hinsicht. Die Model-Klasse haben wir gemeinsam erarbeitet, die View-Klasse haben wir ebenfalls gemeinsam konzipiert. Das Menü und die Bestenliste hat Finn gemacht. Jesse hat in der Zeit den MiniMax-Algorithmus versuch zu implementieren und hat schließlich den Controller gemacht.

5 Projektverlauf

5.1 Überblick

Wir haben das Projekt in einer Sitzung so gut wie beendet.

Zu Beginn entwickelten wir einen Prototypen um ein Gefühl für den Umfang des Projektes und eine Idee für eine Architektur bekommen konnten. Daraus erstellen wir dann das oben aufgeführte UML-Diagramm.

Mit dem UML-Diagramm haben dann angefangen eine neue Version zu erstellen und haben zunächst das Model und einen rudimentären Controller erstellt. Danach haben wir unsere GUI aus dem Prototypen genommen, da diese uns bereits sehr gut gefallen hat. So hatten wir bereits die Grundlage des Projektes geschaffen und optimierten die GUI noch ein wenig und implementieren den Computergegner.

Es gab am Ende der ersten Sitzung noch Probleme mit dem MiniMax-Algorithmus, welche Jesse noch bearbeitete und den Controller anpasste.

5.2 Testprotokolle

Das meiste Testen geschah bei der Entwicklung des Models und der Implementierung des Computergegners.

Wir haben den Controller erst entwickelt, nachdem wir das Model auf sämtliche Testfälle wie z.B. die Gewinnabfrage, verschiedene Brettgrößen, die Belegung von Feldern und das Klonen für MiniMax gesichert haben. Hier gab es insbesondere beim Klonen zunächst kleine Schwierigkeiten, da Java Pass-By-Reference orientiert ist, bei der Zuweisung von Objekten. So mussten wir das Spielbrett, welches wir als zweidimensionale ArrayList gespeichert haben, separat klonen.

Als das Spiel zunächst spielbar war, haben wir die GUI getestet und das Spiel ein wenig gespielt und versucht so viele verschiedene Kombinationen wie möglich zu testen. Dabei gab es anfangs Probleme bei der Darstellung des Gewinners, wenn er diagonal gewann. Die Linie wurde dabei immer von der oberen linken Ecke in die untere Rechte gezogen.

Bei dem MiniMax-Algorithmus gab es das Problem, dass die Abfrage, ob das Spiel zu Ende ist bzw. ob ein Spieler gewonnen, fehlerhaft war. Das Problem hierbei war, dass das Feld, wo der Algorithmus sich gerade befindet, immer als gültiges Feld einer Gewinn-Kombo galt, was dazu geführt hat, dass die Evaluation basierend auf dem Spielzustand nicht korrekt erfolgen konnte.

6 Projektergebnisse

6.1 Ergebnisse

Wir haben ein funktionstüchtiges Programm, mit welchem man TicTacToe spielen kann, sich in eine Bestenliste eintragen kann, verschiedene Feldgrößen generieren kann und gegen einen Computergegner spielen kann erreicht. Außerdem haben wir eigene Verbesserungen eingebracht, wie den Strich der den Gewinner markiert, die Einstellungsmöglichkeit des Computergegners und eine CSS-basierte GUI.

6.2 Mängel

Aus unserer Sicht ist das Endprodukt sehr gut geworden. Wir haben alles soweit getestet, dass das Programm jedenfalls funktionstüchtig ist und auch mit verschiedenen Einstellungen zurecht kommt, was uns sehr zuversichtlich macht.

Als Mangel identifizieren wir jedoch den Computergegner, wenn es zu größeren Spielfeldern als 3x3 kommt. Er funktioniert, jedoch wenn er perfekt Spielen soll, würde es bei Feldern die größer als 4x4 sind viel zu lange dauern, bis der Algorithmus zu Ergebnissen kommt. Auf einem 3x3 Feld jedoch, funktioniert es einwandfrei. Als Optimierung käme eventuell die Alpha-Beta-Suche in Frage, welche die Dauer des MiniMax-Algorithmus drastisch reduzieren würde, jedoch gäbe es auch dort Grenzen der Effizienz. Abseits vom MiniMax würde eventuell ein neuronales Netz oder Q-Learning in

Frage kommen um die Berechnungsdauer zu senken, aber das würde den Rahmen und das eigentliche Ziel dieses Projektes verfehlen.

6.3 Fazit

Jesse

Ich habe meine persönliche Zielsetzung definitiv erfüllt und bin sehr zufrieden mit dem Endprodukt. Außerdem hab ich vorher noch nie wirklich im Pair-Programming-Stil gearbeitet, was sich aber als durchaus positiv und angenehm herausgestellt hat. Vier Augen sehen mehr als zwei und man läuft nicht in Gefahr sich auf Gedanken festzufahren, welche die Umsetzung insgesamt nur schwerer machen würden.

Die Implementation des Computergegners fand ich sehr spannend und konnte dabei mehr über den MiniMax-Algorithmus lernen.

Finn