Projektdokumentation

Konzeption und Implementierung einer Computerversion

des Strategiespiels Reversi

Schüler: David Härer

Klasse: TGI J1

Schuljahr: 2012 / 2013

Schule: Grafenbergschule Schorndorf

Betreuer: Herr Keim

Inhaltsverzeichnis

1 Vorwort und Danksagung

Ich habe mich Reversi als Projekt entschieden, da mir das Spiel sehr gut gefällt, und ich gerne eine Künstliche Intelligenz schaffen wollte. Die Entwicklung des Programms, besonders der KI, hat mir viel Spaß gemacht.

Für die tatkräftige Unterstützung bei der Umsetzung dieses Projekts, insbesondere für das ausführliche Feedback zu jedem Meilenstein, bedanke ich mich bei meinem Informatiklehrer Herr Keim. Dank gilt außerdem meinen Mitschülern, die mir oft weitergeholfen haben.

2 Einleitung

Bei dem Projekt „Konzeption und Implementierung einer Computerversion des Strategiespiels Reversi“ geht es darum, dass Brettspiel Reversi als Computerspiel umzusetzen. Zentrales Element ist die Künstliche Intelligenz, gegen welche der Spieler spielt. Leitfragen sind, wie man das Konzept des realen Brettspiels in eine Computerversion überträgt, und wie man eine möglichst gute KI dafür entwickelt. Die Projektdokumentation beginnt mit einer Erläuterung des realen Brettspiels. Es folgt ein Informationsteil über Künstliche Intelligenz, und wie bei der Entwicklung einer solchen vorgegangen wird. Danach geht es um die Anforderungen an das Computerprogramm, sowie den Entwurf der GUI, Systemarchitektur und KI.

3 Das Spiel Reversi

3.1 Anwendungsbereich

Reversi ist ein Strategiespiel. Als Programm umgesetzt ist es ein strategisches Computerspiel. Das Spiel Reversi eignet sich für Personen ab acht Jahren.

3.2 Spielbeschreibung

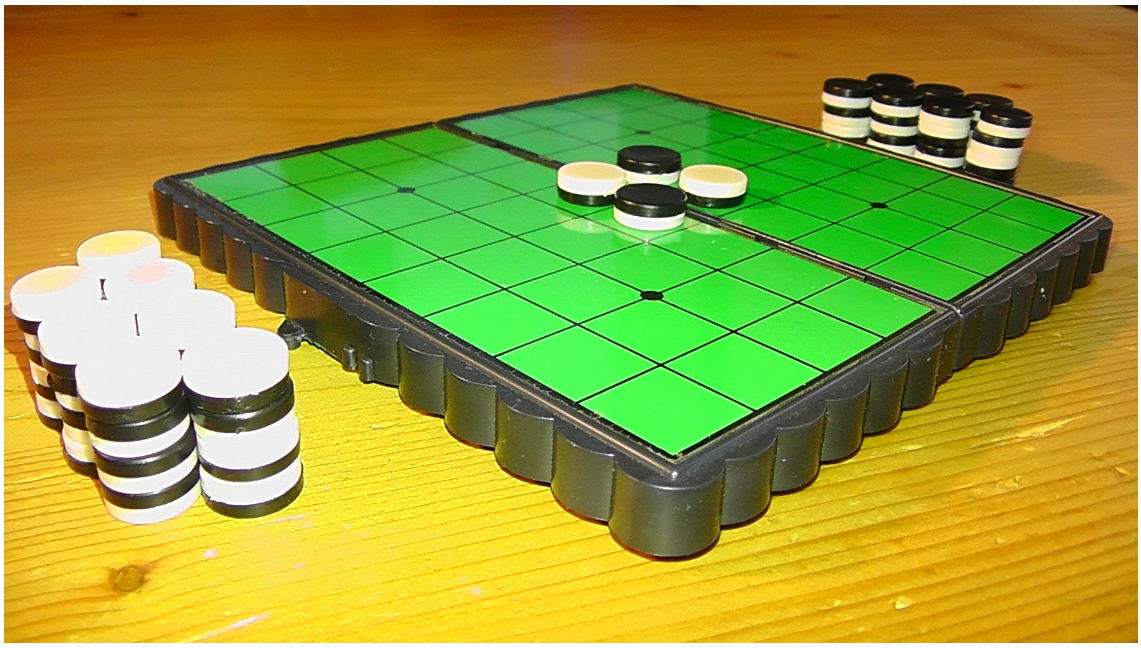

Abbildung 1: Foto von einem realen Reversispiel

Das Spiel Reversi hat ein 8*8 großes Spielfeld. Es ist für zwei Spieler ausgelegt, ein Spieler hat die Farbe Weiß, der andere hat Schwarz. Zu Beginn sind zwei Steine beider Farben in der Mitte des Spielfelds vorhanden. Es beginnt der Spieler der Farbe Weiß. Dieser setzt einen Stein neben einen schwarzen Stein. Man kann entweder waagerecht (rechts oder links), senkrecht (oben oder unten) oder diagonal (oben links, oben rechts oder unten links,

unten rechts) spielen. Entscheidend dabei ist, das er den, oder auch mehrere schwarze Steine mit seinem Zug einschließt. Die Reihe der schwarzen Steine muss also mit einem bereits gesetzten weißen Stein abgeschlossen werden. Wenn in mehrere Richtungen (z.B. nach oben und nach rechts) Steine des Gegners eingeschlossen werden zählen alle eingeschlossenen Steine. Wenn der Stein gesetzt ist, werden alle eingeschlossenen Steine des Gegners in eigene Steine umgewandelt, das heißt ihre Farbe wird auf die eigene Farbe geändert. Damit ist der erste Zug abgeschlossen, und der Gegner ist an der Reihe. Für ihn gelten die gleichen Regeln. Dieser Prozess geht so lange hin und her, bis kein Zug auf dem Spielfeld mehr möglich ist. Dann wird gezählt welcher der beiden Spieler mehr Steine seiner Farbe hat, und dieser hat dann gewonnen.

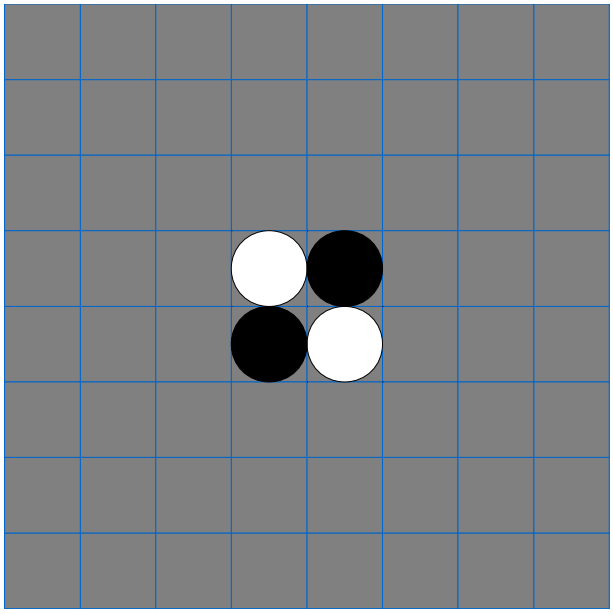

Abbildung 2: Skizze des Spielfelds zu beginn des Spiels

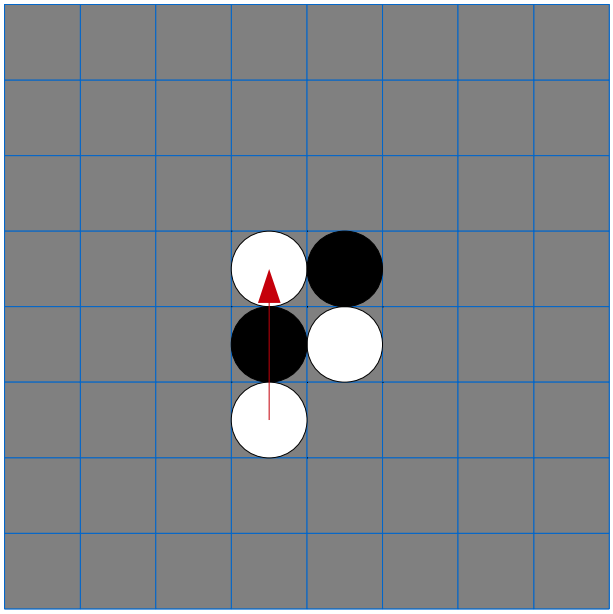

Abbildung 3: Weiß kann mit diesem Zug beginnen

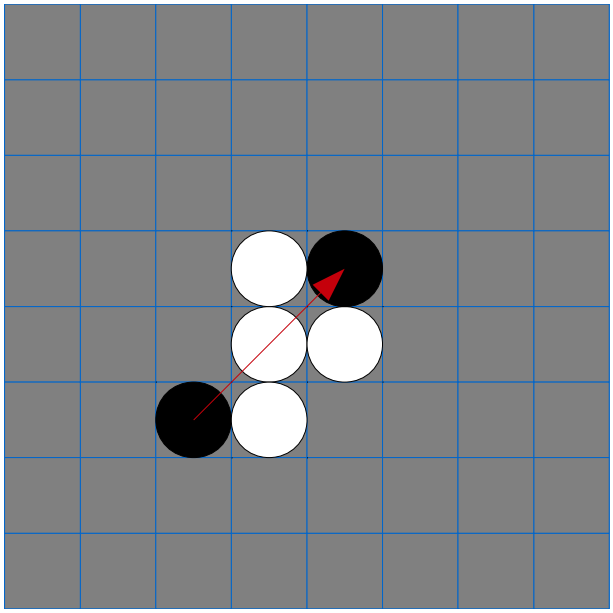

Abbildung 4: Schwarz kann diesen Gegenzug machen

4 Objektorientierte Analyse

4.1 Funktionale Anforderungen

- auf Eingaben des Spielers reagieren → künstliche Intelligenz
- Spiel verwalten (siehe 'Beschreibung Reversi')
- Benutzeroberfläche als Benutzerschnittstelle anbieten
- Möglichkeit ein neues Spiel zu starten
- Meldungen ausgeben (Spielstand, Gewinner)
- es gibt keine Benutzerkonten
- Spielstände werden nicht gespeichert

4.1.1 Analyse der Benutzeroberfläche

Die grafische Benutzeroberfläche des Programms muss folgende Dinge beinhalten:

- Fenster
- Spielfeld (welches aus einzelnen Zellen besteht)
- Steine (schwarz und weiß)
- Spielstandanzeige
- Button für neues Spiel

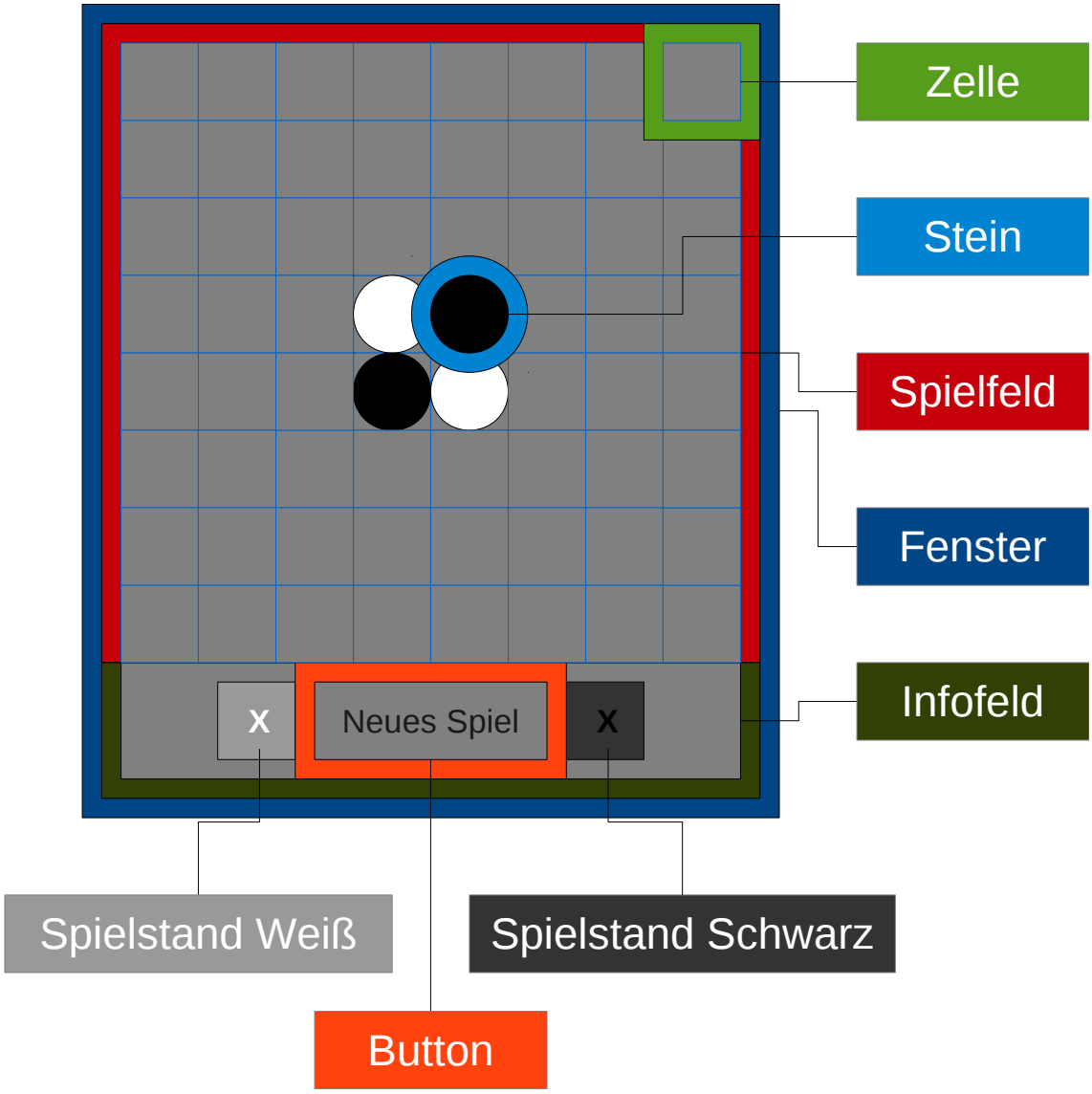

Abbildung 5: Grafische Analyse der GUI

4.1.2 Künstliche Intelligenz

Die Computerversion des Strategiespiels Reversi wird von einer Person gespielt. Daher muss der Gegner von dem Computer erstellt werden. Dies erfolgt über eine künstliche Intelligenz.

4.1.2.1 Allgemeine Beschreibung

Der Begriff künstliche Intelligenz [1] [2] (kurz KI) wird in der Informatik in zwei Bereiche unterteilt. Zum einen gibt es die starke KI, bei welcher versucht wird eine real existierende menschliche Intelligenz nachzubilden. Eine schwache KI ist darauf ausgelegt, Probleme zu lösen, für die es im allgemeinen Verständnis Intelligenz benötigt. Während der Versuch eine starke KI zu schaffen seit Jahrzehnten an grundlegenden philosophischen Fragen scheitert, ist die Entwicklung schwacher KI's schon sehr weit fortgeschritten. So gibt es verschiedene Teilgebiete, wie beispielsweise Expertensysteme, Mustererkennung oder Robotik, in denen KI's unerlässlich sind. Bei Expertensysteme ist das Wissen eines speziellen Fachgebiets formalisiert und für Computer aufbereitet. Diese können daraufhin durch logische Schlüsse Probleme dieses einen Fachgebiets lösen. Ein Beispiel ist die Diagnose von speziellen Krankheiten. Bei der Mustererkennung ist es möglich, Bilder, Formen oder Sprache zu erkennen und zu analysieren, wie zum Beispiel bei einer Gesichts- oder Spracherkennungssoftware. Bei der Robotik bekommen die Programme zusätzlich einen Körper, nämlich einen Roboter. Damit können sie Aufgaben, wie Schweißen, oder aber auch bei Katastrophen zu helfen, in der realen Welt lösen. Natürlich werden KI's auch in Computerspielen eingesetzt, um einen Gegenspieler des echten Spielers darzustellen.

4.1.2.2 Vorgehensweise bei der Entwicklung einer KI

Die Basis bei der Entwicklung einer künstlichen Intelligenz ist, dass man das Problem, welches später von der KI gelöst werden soll, verstanden hat. Darauf hin muss man sich entscheiden, was für eine Art von künstlicher Intelligenz sich am besten eignet um das Problem zu lösen.

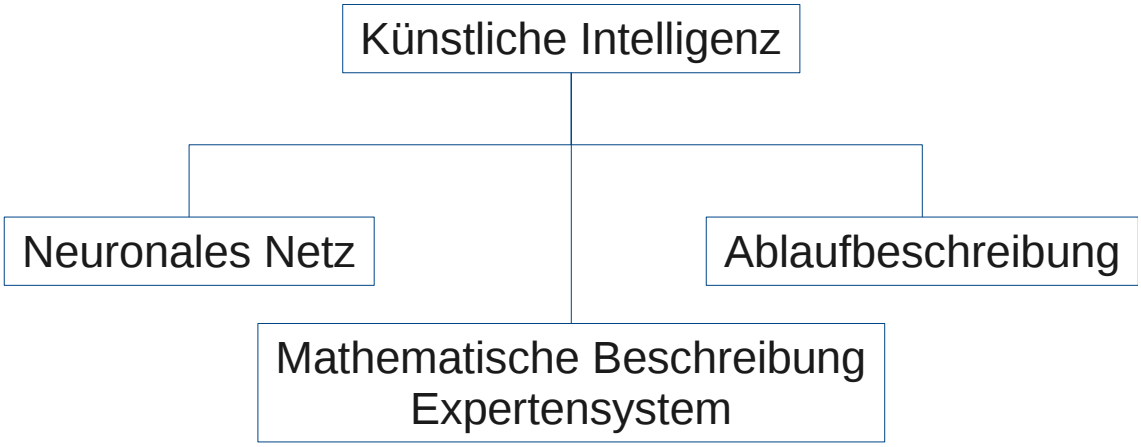

Abbildung 6: Unterarten einer künstlichen Intelligenz

4.1.2.2.1 Neuronales Netz

Ein Neuronales Netz [3] bildet die Struktur des Gehirns nach. Es reagiert auf eine Eingabe auf der einen Seite mit einer Ausgabe auf der anderen Seite. Dazwischen sind verschiedene Schichten aus Synapsen, welche miteinander verbunden sind. Die Ausgabe basiert daher auf der Eingabe und den verschieden gewichteten Verbindungen zwischen den Synapsen und den Synapsen selbst. Der Vorteil ist, dass man mittels Neuronaler Netze Aufgaben lösen kann, bei denen man selbst gar nicht weiß wie sie zu lösen sind. Der Nachteil ist aber, dass man nicht genau nachvollziehen kann, wie das Ergebnis entstanden ist, und es viele Testdaten benötigt, bis ein Neuronales Netz wirklich das macht, was man will. Eine typische Anwendung ist die Mustererkennung.

4.1.2.2.2 Mathematische Beschreibung

Bei einer Mathematischen Beschreibung des Problems, versucht man zum Beispiel eine Gleichung zu finden, welche für die einzelnen Aspekte des Problems eine Lösung liefert. Wenn man es schafft das Problem mathematisch zu lösen, ist es sehr einfach die KI zu verwirklichen, doch ist es meist nicht möglich, oder nur sehr schwer das Problem mit einer Funktion zu beschreiben. Ein Expertensystem [4] basiert im Grunde genommen genau darauf, denn für ein Expertensystem wird alles relevante Wissen formalisiert und für den Computer lesbar gemacht. Doch es ist häufig sehr aufwändig, oder bei vielen Problemen auch unmöglich, jedes relevante Wissen für den Computer verständlich zu machen. Eie Ergebnisse sind am Ende aber auch dementsprechend gut. Ein weiterer Nachteil ist, dass ein Expertensystem nur sehr eingeschränkt eingesetzt werden kann, und mit fachfremden Problemen hoffnungslos überfordert ist. Das klassische Einsatzgebiet ist die Medizin, bei der Expertensysteme Ärzte bei der Diagnose unterstützen.

4.1.2.2.3 Ablaufbeschreibung

Für eine Ablaufbeschreibung der Problemlösung müssen zuvor bestimmte Regeln festgelegt werden, das Problem lässt sich daraufhin anhand dieser Regeln lösen. Dies ist aber oft nur bei einfachen Problemen möglich, da mit zunehmender Komplexität des Problems, auch die Beschreibung des Lösungswegs immer komplexer wird, oder stark vereinfacht werden muss, was sich negativ auf das Ergebnis am Ende auswirkt. Eine Ablaufbeschreibung ist die klassische Lösung bei einfachen Computerspielen gegen eine KI.

4.2 Nichtfunktionale Anforderungen

- Java Runtime muss installiert sein
- PC muss mit Maus und Tastatur (oder entsprechenden Eingabemöglichkeiten) bedient werden können
- Bildschirm mit mindestens 1024*768 Pixeln zur Darstellung der GUI

5 Objektorientiertes Design

5.1 Use-Case-Diagram

In dem folgenden Use-Case-Diagram sieht man, dass der Mensch die GUI mit der Maus bedient. Er kann zum einen seinen Zug ausführen, zum anderen ein neues Spiel starten. Die KI (Maschine) kann ebenfalls ihren Zug ausführen. Dies wird über die Leertaste ausgelöst und durch Methoden umgesetzt.

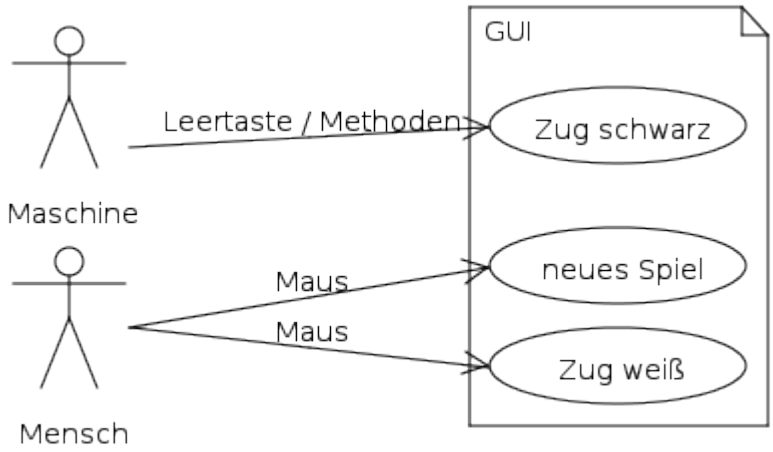

Abbildung 7: So benutzen Mensch und Maschine das Programm

5.2 Graphical User Interface (GUI)

Vom Aufbau her befinden sich alle Elemente der GUI in einem Fenster, dieses beinhaltet zum einen das Spielfeld, und ein Infofeld. Das Spielfeld wiederum besteht aus einzelnen Zellen. Auf diesen Zellen befinden sich die Steine. Das Infofeld beinhaltet drei Elemente. Den Button für ein neues Spiel, sowie die Spielstandanzeigen für Weiß und Schwarz. Über den Button für das neue Spiel wird ebenfalls am Ende des Spiels der Gewinner in Textform ausgegeben.

Je nach dem wie der PC, auf dem das Programm läuft, konfiguriert ist, dauert es unterschiedlich lange, bis alle Grafiken der GUI geladen sind. Daher gibt es während dem Ladevorgang des Programms einen Startbildschirm (Splash Screen).

Abbildung 8: Screenshot des Splash Screens

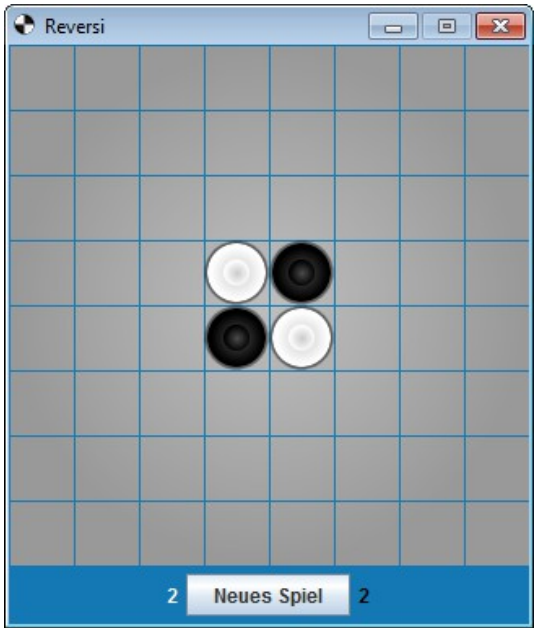

Abbildung 9: Screenshot der Grafischen Benutzeroberfläche

5.3 Systemarchitektur

Der Ausgangspunkt bei dem Design des Programms ist das 'MVC Pattern'-Prinzip der Objektorientierung. Es bedeutet, dass Daten, Visualisierung und Programmlogik von einander getrennte werden. Das 'M' steht für Model (Daten), in diesem Fall sind es die Spieler. 'V' steht für View (Visualisierung), sprich die GUI. Das 'C' steht für Controller (Programmlogik), dies ist dementsprechend die Steuerung. Getrennt werden diese einzelnen Bereiche mittels Packages, welche die Klassen beinhalten.

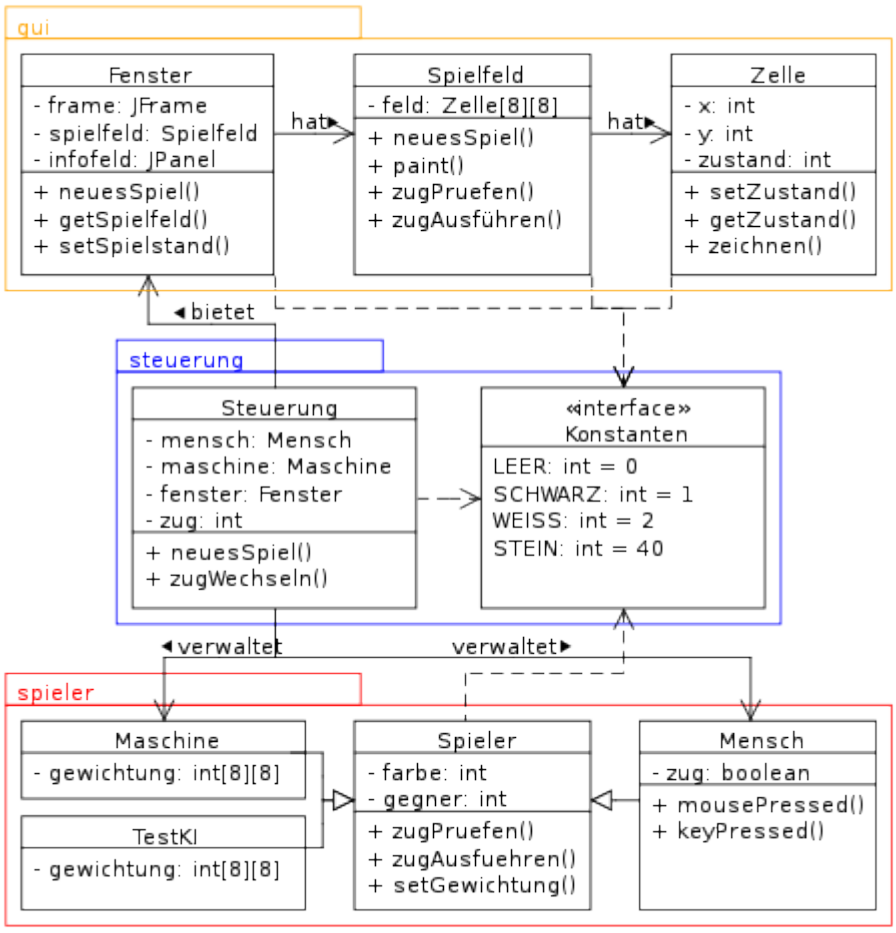

Abbildung 10: Systemarchitektur des Computerspiels Reversi

Die zentrale Klasse, welche die Main-Methode beinhaltet, und das ganze Spiel verwaltet ist die Klasse Steuerung. Sie hat zum einen die Spieler, zum bietet sie diesen das Fenster, und damit die Benutzeroberfläche. Alle notwendigen Konstanten, wie beispielsweise die Farben der Spieler und Steine, sowie die Breite der Steine werden über das Interface Konstanten allen Klassen zugänglich gemacht. Die Klasse Fenster beinhaltet den Rahmen der Benutzeroberfläche, und ein das Infofeld, auf welchem der aktuelle Spielstand angezeigt wird, und mittels eines Buttons ein neues Spiel gestartet werden kann. Das Spielfeld ist eine eigene Klasse, welche von JPanel erbt und in der Fensterklasse instanziiert wird. Das Spielfeld besteht aus einem zweidimensionalen Feld aus einzelnen Zellen. Die Spielfeldgröße beträgt 8*8 Zellen. Die einzelnen Zellen sind Objekte der Klasse Zelle. Diese Klasse hat alle notwendigen Attribute und Methoden für eine Zelle. Die Eigenschaften sind die x- und y-Koordinate, und ihr Zustand, sprich ob diese Zelle mit einem Stein belegt ist, und wenn ja welche Farbe dieser Stein hat. Die wichtigsten Methoden für das Spielfeld sind die Methode 'zugPruefen' und 'zugAusfuehren'. Der 'zugPruefen'-Methode muss eine x- und eine y-Koordinate, sowie die entsprechende Farbe übergeben werden. Der Rückgabewert ist ein acht Zellen großes, ganzzahliges Feld, in welchem an jeder Stelle die Anzahl der möglichen gewonnenen Steine angegeben wird. Das heißt an der ersten Stelle steht wie viele Steine nach rechts gewonnen werden können, die zweite Stelle gibt das gleiche für rechts unten an und so weiter. Die Klasse Steuerung beinhaltet neben dem Fenster auch die Spieler. Diese sind jeweils eine Klasse, welche von der Superclass Spieler erben. Der menschliche Spieler wird durch die Klasse Mensch repräsentiert, der Computer durch die Klasse Maschine. Das heißt Mensch und Maschine sind die beiden Gegenspieler. In der Oberklasse Spieler sind alle Methoden und Attribute der einzelnen Spieler vorgegeben. Dadurch ist es möglich für testzwecke zwei verschiedene KI's gegeneinader spielen zu lassen, ohne den kompletten Programmcode ändern zu müssen.

Die Eingaben des menschlichen Spielers werden über die Interfaces MouseListener, und ActionListener ermöglicht. Um eine ordentliche Verwaltung des Spiels zu ermöglichen, hat die Klasse Steuerung das Attribut Zug. Je nach dem welcher Spieler an der Reihe ist, wird Zug auf WEISS oder SCHWARZ geschaltet. Wird in der Klasse Mensch durch einen Mausklick des Spielers die Methode 'mousePressed' aufgerufen, wird als erstes mittels des Zugs überprüft, ob der Spieler überhaupt an der Reihe ist. Wenn dem so ist, wird der Klick ausgewertet und seine x- und y-Koordinate, mitsamt der Farbe, der Spielfeldmethode 'zugPruefen' übergeben. Ist an der angeklickten Stelle ein Zug möglich, wird dieser darauf hin über die 'zugAusfuehren'-Methode ausgeführt. Sobald die Leertaste gedrückt wird, löst dies ein 'actionEvent' aus, und in der 'actionPerformed'-Methode des ActionListeners wird die Methode 'zugWechseln' der Steuerungsklasse aufgerufen. Diese Methode ist die zentrale Schaltstelle zwischen den Zügen. Bei ihrem Aufruf wird überprüft, ob das Spiel zu Ende ist, und wenn ja eine entsprechende Meldung ausgegeben. Wenn nicht wird der Zug umgestellt, je nachdem, von wem die Methode aufgerufen wurde. Ist daraufhin die Maschine an der Reihe, wird ihre Methode 'zugAusfuehren' aufgerufen.

5.3.1 Die KI

5.3.1.1 Entwurfsentscheidung

Die künstliche Intelligenz des Spiels Reversi wird mittels einer Ablaufbeschreibung umgesetzt. Ein Neuronales Netzwerk wäre zwar möglich, aber das Problem ist einfach genug, dass sich die Problemlösung formalisieren lässt. Ein Neuronales Netzwerk ist also in erster Linie nicht nötig. Eine Mathematische Beschreibung wäre sehr aufwendig, wenn nicht sogar unmöglich, da man zu jeder möglichen Spielposition den besten Zug ermitteln müsste. Auf einem 8*8 Spielfeld, sprich mit 64 Feldern, welche jeweils drei unterschiedliche Zustände haben können entspricht dies 3^64, also 3,43*(10^30) Möglichkeiten. Dies könnte man zwar auf weniger beschränken, da das Spielfeld radialsymmetrisch, und viele Spielstände dadurch gleich, nur eben gespiegelt oder gedreht sind, aber es wäre immer noch sehr viel Aufwand. Es bleibt also noch die Ablaufbeschreibung, und diese macht auch am meisten Sinn, da die Problemlösung für jeden Zug im Grunde die selbe ist.

5.3.1.2 Umsetzung

Die KI wird ist mittels der Klasse Maschine verwirklicht. Das Problem, welches es zu lösen gilt, besteht darin, den sinnvollsten Zug gegen den echten Spieler zu finden. Da am Ende des Spiels derjenige gewonnen hat, der die meisten Steine seiner Farbe hat, ist es sinnvoll zu versuchen mit seinen Zügen zum einen so viele Steine wie möglich zu bekommen, zum anderen aber auch dem Gegner möglichst wenig Möglichkeiten bieten, das gleiche zu tun. Dazu hat die KI ein 8*8 großes Feld, welches das Spielfeld symbolisiert. Jedes der einzelnen Zellen hat einen bestimmten Wert, eine Gewichtung. Die Eckzellen sind am wertvollsten, da sie einem nicht mehr weggenommen werden können. Danach kommen die Randzellen, allerdings nicht diejenigen, die direkt neben einer Ecke liegen, denn diese sind der Zugang für den Gegner zu den Eckzellen. Die Problematik ist hier, einen guten Kompromiss zu finden. Ähnlich verhält es sich bei denen, die einen Zugang zum Rand ermöglichen. Die restlichen Zellen in der Mitte haben eine Gewichtung dazwischen.

5	1	4	4	4	4	1	5
1	1	2	2	2	2	1	1
4	2	3	3	3	3	2	4
4	2	3	3	3	3	2	4
4	2	3	3	3	3	2	4
4	2	3	3	3	3	2	4
1	1	2	2	2	2	1	1
5	1	4	4	4	4	1	5

Abbildung 11: Die Gewichtung der einzelnen Zellen für die KI

Wird die Methode 'zugAusfuehren' nun aufgerufen, geht die KI mittels zweier Schleifen jede einzelne Zelle durch, und überprüft ob ein Zug möglich ist. Wenn dem so ist werden die Steine, welche mit diesem Zug gewonnen werden können, zusammengezählt und mit der Gewichtung des jeweiligen Feldes multipliziert. Dadurch hat die KI wenn sie alle 64 Zellen durchgegangen ist, für jede Zelle einen bestimmten Wert, und umso höher dieser Wert ist, desto besser ist diese Zelle für den nächsten Zug geeignet. Das heißt, die KI entscheidet sich am Ende für die Zelle mit dem höchsten Wert.

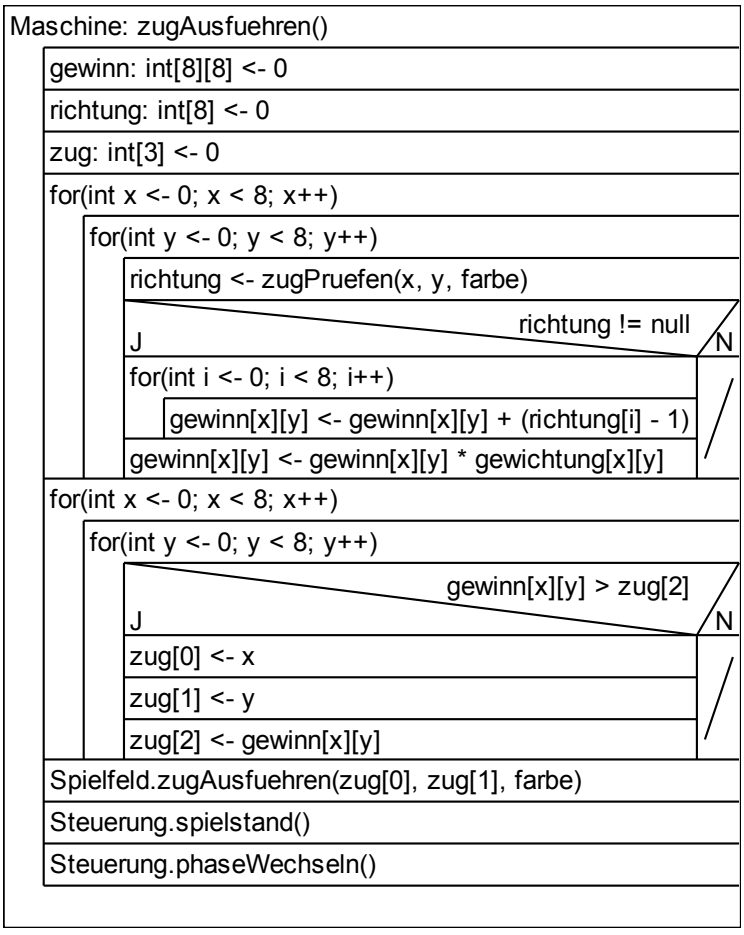

Abbildung 12: So findet die KI den optimalen Zug

5.3.1.3 Test

Um die KI zu testen kann man zwei verschiedene KI's gegeneinander spielen lassen. Das heißt statt der Klasse Mensch wird eine weitere KI-Klasse instanziiert. Dort kann man verschiedene Gewichtungen oder Algorithmen zur Entscheidungsfindung ausprobieren. Die KI welche gewinnt, ist offensichtlich die bessere. Somit kann man jede Veränderung an der KI testen, ob sie dadurch wirklich besser wird. Vielversprechen scheint dabei eine dynamische Anpassung der Gewichtung zu sein. Eine KI, die einen Zug vorausdenkt, ist bis jetzt nicht sehr stark.

6 Zusammenfassung

6.1 Reflexion

Das Projekt konnte erfolgreich durchgeführt werden. Besonders hilfreich war dabei das Vorwissen, aus den letzten beiden Schuljahren in dem Bereich Objektorientierte Programmierung in Java. Das Projekt könnte dahingehend weitergeführt werden, dass die KI weiter verbessert wird, unter Umständen auch mittels eines Neuralen Netzes. Ebenso könnten Benutzeraccounts und Statistiken für die Spieler hinzugefügt werden. Es wäre auch möglich das Programm um einen Mehrspieler Modus zu erweitern.

6.2 Zeitlicher Verlauf des Projekt

Die Projektdokumentation entstand aus dem Pflichtenheft des Projekts. In der nachfolgenden Tabelle sieht man den zeitlichen Verlauf bei der Entstehung des Pflichtenhefts.

Version	Datum	Änderungen
0.1	23.02.13	Erstellung
0.2	20.03.13	Änderungen entsprechend des Feedbacks
0.3	22.03.13	Strukturierung, Künstliche Intelligenz, Klassenbildung, die KI, Bilder / Skizzen
0.4	27.05.13	Überarbeitung entsprechend des Feedbacks
0.5	03.06.13	Überarbeitung entsprechend des Feedbacks
1.0	15.06.13	Umwandlung in Projektdokumentation
1.1	21.06.13	Überarbeitung entsprechend des Feedbacks
1.2	23.06.13	Optimierung allgemein

7 Erklärung

„Hiermit versichere ich (David Härer), die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt zu haben“

8 Anhang

8.1 Quellen

[1] Bremer, Lars,: Von MIPS zu Grips, c't 2007, Heft 18, S. 170 - 175

[2] Internet: „Künstliche Intelligenz“ [http://de.wikipedia.org/wiki/K%C3%BCnstliche_Intelligenz] (Stand: 22.3.13, 19:45 Uhr)

[3] Internet: „Neuronales Netz“ [http://de.wikipedia.org/wiki/Neuronales_Netz] (Stand: 22.3.13, 21:00 Uhr)

[4] Internet: „Expertensystem“ [http://de.wikipedia.org/wiki/Expertensystem] (Stand: 22.3.13, 21:00 Uhr)

8.2 Abbildungsverzeichnis