OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

Entwicklung einer künstlichen Intelligenz für Brettspiele und deren Anbindung an eine Touch-Hardware über mobiele Endgeräte

An der Fakultät für Informatik und Mathematik der
Ostbayerischen Technischen Hochschule Regensburg
im Studiengang
Techniche Informatik

eingereichte

Bachelorarbeit

zur Erlangung des akademischen Grades des
Bachelor of Science (B.Sc.)

Vorgelegt von: Korbinian Federholzner
Matrikelnummer: 3114621

Erstgutachter: Prof. Dr. Carsten Kern
Zweitgutachter: Prof. Dr. Daniel Jobst

Abgabedatum: 31.08.2020

Erklärung zur Bachelorarbeit

1. Mir ist bekannt, dass dieses Exemplar der Abschlussarbeit als Prüfungsleistung in das Eigentum der Ostbayerischen Technischen Hochschule Regensburg übergeht.
2. Ich erkläre hiermit, dass ich diese Abschlussarbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Regensburg, den 31. August 2020

_____

Korbinian Federholzner

Zusammenfassung

In der folgenden Arbeit wird ...

Inhaltsverzeichnis

I Abkürzungsverzeichnis **VII**

1 Einleitung **1**
1.1 Motivation . . . 1
1.2 Aufgabenstellung . . . 1
1.3 Struktur dieser Arbeit . . . 1

2 Grundlagen **2**
2.1 Dame . . . 2
2.1.1 Internationale Dame . . . 2
2.2 Minimax . . . 3
2.3 Iterative Deepening . . . 5
2.4 Alpha-Beta Pruning . . . 6
2.5 Zugsortierung . . . 7
2.6 Monte Carlo Tree Search (MCTS) . . . 8

3 Architektur der Software **10**
3.1 Überblick . . . 10
3.2 Kommunikation der Komponenten . . . 10
3.2.1 Kommunikations Ablauf Benutzer gegen Benutzer . . . 11
3.2.2 Kommunikationsprotokoll . . . 12
3.3 Gameserver . . . 12
3.3.1 Softwareaufbau des Gameservers . . . 12
3.4 ReversiXT GUI . . . 14
3.4.1 Softwareaufbau der ReversiXT GUI . . . 14
3.5 KI Client . . . 16
3.5.1 Softwareaufbau des KI Clients . . . 16

4 Hardware **17**
4.1 Raspberry Pi . . . 18
4.2 Touch Monitor . . . 18

5 Implementierung **18**
5.1 Eingesetzte Sofwarekomponenten . . . 18
5.1.1 Programmiersprachen und Frameworks . . . 18
5.1.2 Datentransferprotokolle . . . 18
5.2 Gameserver . . . 18
5.2.1 Netzwerkspezifikation des Gameservers . . . 18
5.3 Graphische Oberfläche . . . 18

5.3.1 Gegebene React Anwendung 18
5.3.2 Erweiterungen 18
5.4 KI Client 18
5.4.1 Vergleich der KI Algorithmen 18

6 Testing **18**
6.1 Integrationstest 18
6.2 Ergebnisse 18

7 Fazit und Ausblick **19**

Anhang **I**

A Domändenmodell **I**

Abbildungsverzeichnis

1 DameSpielfeld 3
2 Minimax-Baum mit Spielfeldzustand als Knoten 4
3 Minimax-Baum mit Bewertung der Stellungen 5
4 Ablauf des Iterativen Deepenings 6
5 Gewinn durch Alpha-Beta Pruning 7
6 Beispiel einer Zugsortierung 8
7 Minimax-Baum mit Bewertung der Stellungen 9
8 SequenceDiagram 11
9 ClassDiagram 13
10 ReversiXTGUI 15
11 KIClient 17

Tabellenverzeichnis

I Abkürzungsverzeichnis

1 Einleitung

Das Thema dieser Arbeit ist die Implementierung einer künstlichen Intelligenz für diverse Brettspiele. Dabei soll die resultierende Software auf einem von einem Raspberry Pi gesteuerten Touch-Bildschirm laufen, durch welchen ein Benutzer die künstliche Intelligenz herausfordern kann. Außerdem gibt es die Möglichkeit, sich mit der Hardware über ein mobiles Endgerät wie einem Smartphone zu verbinden, um die KI oder den Spieler, der den Touch-Bildschirm bedient, herauszufodern.

1.1 Motivation

Das Feld der künstlichen Intelligenz ist momentan eines der sich am schnellsten entwickelnden Felder der Informatik. Dabei spielt die Spieltheorie schon seit Anfang eine große Rolle. So bieten klassische Brettspiele wie z. B. Schach, Dame oder Mühle nicht nur eine klar definierte Abstraktion von Problemen der realen Welt, sondern sie können auch von dem Großteil der Bevölkerung verstanden und gespielt werden. Das Meistern einer dieser Brettspiele wird auch oft mit hohem Grad an Intelligenz gleichgesetzt. Viele Algorithmen, die in der Spieltheorie entwickelt wurden, haben sich auch erfolgreich auf andere Felder der Informatik übertragen lassen. Ebenso hat sich die Art, wie Brettspiele gespielt werden, durch das Verwenden von künstlicher Intelligenz auch verändert, da die KI Züge in Betracht zieht, die auf den ersten Blick recht ungewöhnlich und nachteilhaft aussehen, sich aber als extrem stark herausstellen. Diese Arbeit versucht eine künstliche Intelligenz für Brettspiele, wie Dame, zu implementieren.

1.2 Aufgabenstellung

Im Rahmen der Bachelorarbeit soll eine künstliche Intelligenz entwickelt werden, welche Brettspiele, wie Dame, spielen kann. Gegeben ist eine Software, bei welcher man in der Lage ist, das Spiel ReversiXT (Reversi Extreme) gegen eine KI, sowie sich selbst zu spielen. Diese Software soll um einen Game Server, die KI und das neue Spiel in der GUI, erweitert werden. Außerdem soll ein Benutzer in der Lage sein, sich mit seinem Smarthphone mit der Hardware zu verbinden, um neue, sowie die alte KI herausfordern zu können.

1.3 Struktur dieser Arbeit

Die Arbeit ist folgendermaßen aufgebaut:
Kapitel 2 befasst sich mit den Grundlagen zu Dame, sowie den verwendeten künstliche Intelligenz Algorithmen. Dabei werden zu Dame auch die Grundregeln der verwendeten Variante erklärt. Bei den KI Algorithmen handelt es sich um die Algorithmen, die in der Arbeit verwendet und miteinander verglichen werden.
In Kapitel 3 werden die Anforderungen, welche gefordert sind, vorgestellt.

2 Grundlagen

Dieses Kapitel gibt einen Überblick über die theoretischen Grundlagen, die für das Verständnis dieser Arbeit notwendig sind. Zunächst werden die Grundregeln des behandelten Brettspieles vorgestellt. Darauf folgt eine Erklärung der künstlichen Intelligenz Algorithmen, welche für folgenden Kapitel von großer Relevanz sind. [Rus12]

2.1 Dame

Dame ist eines der ältesten Brettspiele, wobei erste Varianten 3000 v. Chr. im irakischen Ur entdeckt wurden. In der heutigen Zeit werden verschiedene Varianten desselben Spieles weltweit gespielt. So wird in vielen englischsprachigen Ländern eine andere Version gespielt als im Rest der Welt, was auch als Internationale Dame bekannt ist. Unterschiede sind z. B. bei diesen Varianten, dass bei internationaler Dame, Damen beliebig viele Felder in alle Richtungen springen dürfen, in anderen Varianten jedoch nur 1 Feld. Da die Regeln der internationalen Dame weiter verbreitet sind und auch im Vereinssport praktiziert werden, wird sich diese Arbeit im folgenden auf diese Regeln fokussieren. [Dra]

2.1.1 Internationale Dame

In dieser Variante des Spieles wird auf einem 10x10 Brett mit Schachbrett-Muster gespielt. Siehe Abbildung 1. Die Spielsteine sind scheibenförmig und in zwei Farben vorhanden, meist schwarz und weiß und dürfen nur auf den dunklen Feldern des Schachbrettes bewegt werden. Es gibt zwei Arten von Spielsteinen. Normale Spielsteine, welche nur in Richtung des Gegners bewegt werden, aber Rückwerts schlagen dürfen, und Damen, welche in alle Richtungen beliebig viele Felder fahren und schlagen dürfen. Allgemein herrscht Schlagzwang, was bedeutet, dass falls ein Spieler die Möglichkeit hat zu schlagen, er auch schlagen muss. Ein normaler Spielstein wird zur Dame, falls er in die hinterste Reihe des Gegners kommt. Ziel des Spieles ist es, entweder alle Steine des Gegners zu schlagen, oder den Gegner in eine Situation zu zwingen, in der er keine Züge mehr machen kann. [Int]

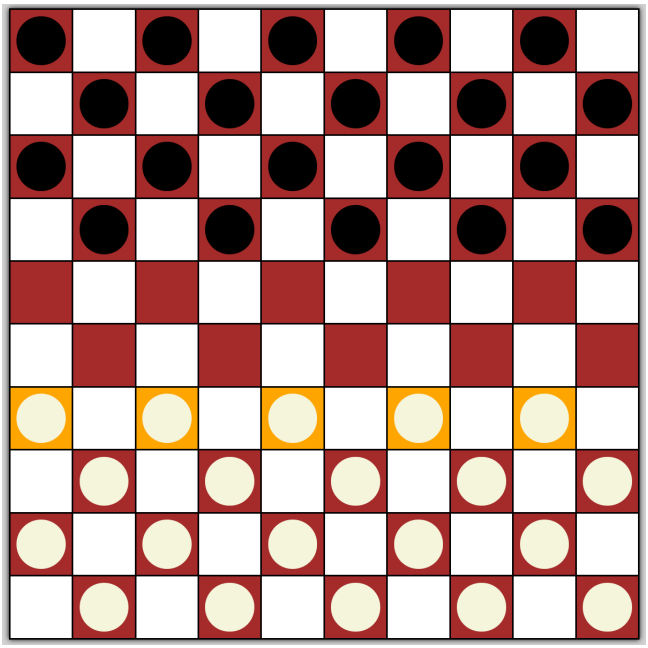

Abbildung 1: Das 10x10 Spielfeld aus der Anwendung

2.2 Minimax

Der Minimax Algorithmus wird verwendet, um einen optimalen Spielzug in Spielen mit perfekter Information, bei zwei Spielern zu finden. Dazu wird ein eine Baumstruktur verwendet, welche den Zustand des Spielbrettes als Knoten hat, siehe Abbildung 2. Alle Züge, die von einer Stellung aus möglich sind, werden in den Kindknoten des jeweiligen Knotens gespeichert. Der Wurzelknoten beschreibt den momentanen Zustand des, Spieles bei dem der Algorithmus aufgerufen wird. Die Blattknoten am Ende des Baumes entsprechen entweder einer Stellung in der das Spiel beendet wurde, oder der Stellung bei einer Tiefe, bei der der Algorithmus aufgehört hat zu suchen. Die Angabe einer Tiefe ist nötig, da Spiele wie Schach oder Dame einen extrem großen Suchbaum zur Folge hätten und das Suchen eines Endzustandes in diesen sehr viel Zeit beansprucht. [MSC82]

Zum Beispiel kann die Abbildung 2 durch den Baum aus der Grafik 3, mit einer Suchtiefe von vier, dargestellt werden. Dabei wird jeder Terminal-Knoten, ein Knoten bei dem das Spiel vorbei ist oder die Endtiefe erreicht wurde (im Beispiel Grün makiert), mit einer Bewertungsfunktion bewertet:

- Normale Figur: +1 für Weiß und -1 für Schwarz
- Dame: +3 für Weiß und -3 für Schwarz
- Spielende: +40 für Weiß und -40 für Schwarz

Ein Dreieck mit der lange Seite nach unten, steht für eine Maximierung der Kindknotenwerte, das andere Dreieck für eine Minimierung. Die Bewertungen in den Endzuständen werden nach oben durchgereicht und je nachdem ob der Elternknoten ein Maximierer oder ein Minimierer

ist bekommt er einen neuen Wert zugewiesen. Im Beispiel kann man sehen, dass egal welche Züge gewählt werden es immer zu einem Materialverlust von Weiß kommt. Würde Weiß die zweite Option wählen, so ist das Spiel nach dem Nächsten Zug von Schwarz schon entschieden und Weiß verliert.

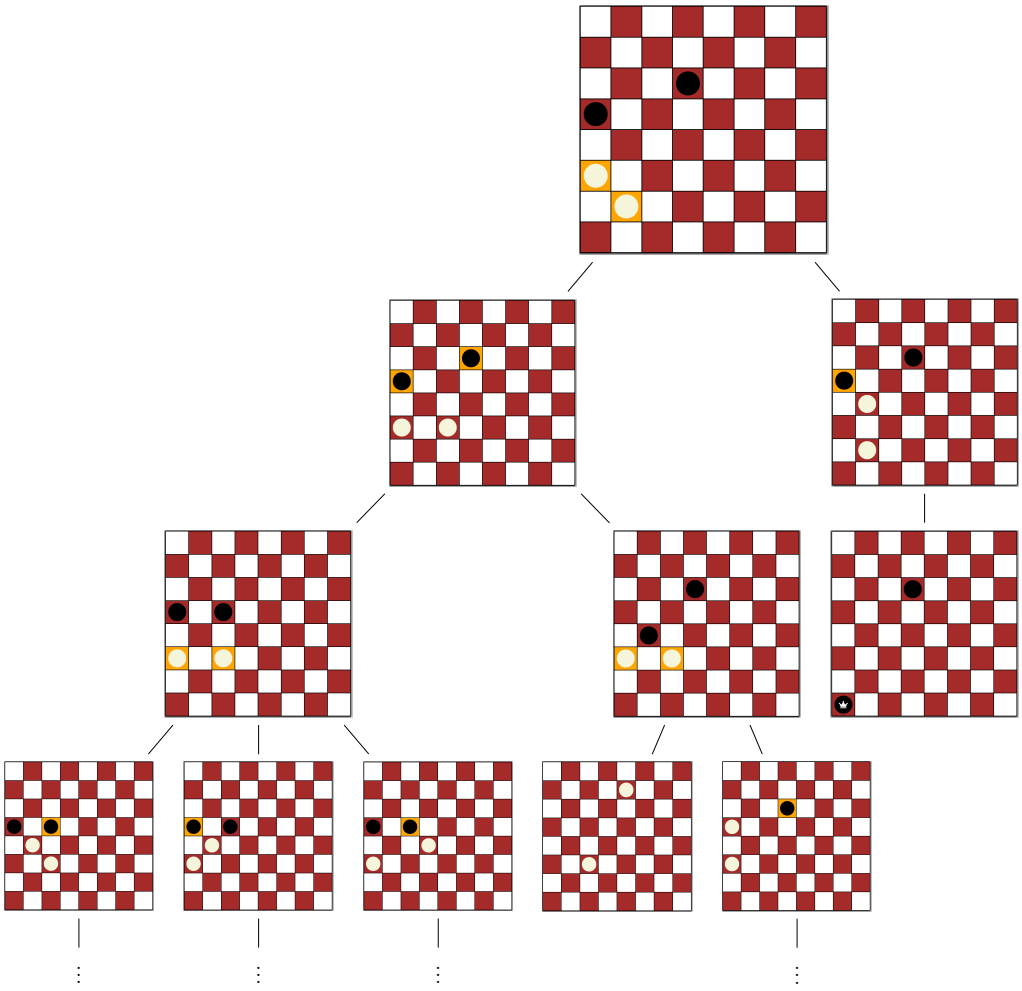

Abbildung 2: Minimax-Baum mit Spielfeldzustand als Knoten

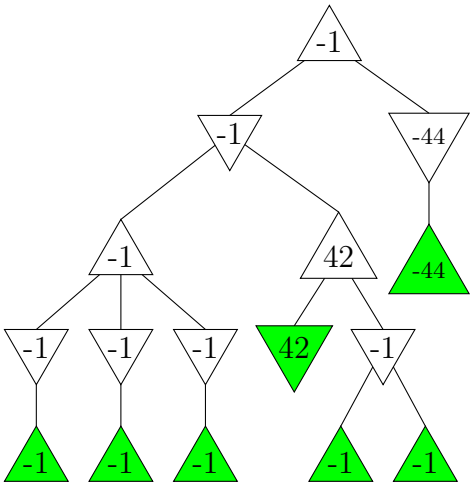

Abbildung 3: Minimax-Baum mit Bewertung der Stellungen

2.3 Iterative Deepening

In komplexeren Spielen wie Go, Schach oder Dame ist es wegen dem Rechenaufwand sehr schwer den kompletten Baum von Minimax bis zu den Endzuständen aufzubauen. Deswegen wird in diesen Spielen der Baum nur bis zu einer gewissen Tiefe aufgebaut. Da es aber bei einer gleichen Tiefe für verschiedene Stellungen zu unterschiedlichen Dauern der Suche kommen kann, ist es problematisch eine fixe Suchtiefe anzugeben, vor allem wenn mit Zeitlimits gearbeitet wird. Iterative Deepening hilft hierbei, der Ablauf des Algorithmus ist wie folgt: Zuerst führe Minimax für eine Tiefe von eins aus. Danach, verwerfe alle generierten Knoten des Baumes und starte erneut von Anfang, aber dieses Mal bis zu einer Tiefe von zwei. Dieses verwerfen und neu starteten wird so oft wiederholt bis ein Zeitlimit erreicht wird, siehe Abbildung 4. Der letzte aufgebaute Baum, bevor neugestartet wird, wird zwischengespeichert und falls Minimax bis zum Ablauf des Zeitlimits nicht fertig ist, wird die momentane Berechnung verworfen und der letzte gespeicherte Baum verwendet. Ein Nachteil dieser Methode ist, dass der Rechenaufwand der ersten Tiefen verschwendet wird, da diese Ergebnisse verworfen werden. Jedoch beeinflusst diese verschwendete Rechenzeit nicht die Asymptotische Laufzeit des Algorithmus, da die meiste Arbeit in der untersten Tiefe der Suche gebraucht wird. [Kor85].

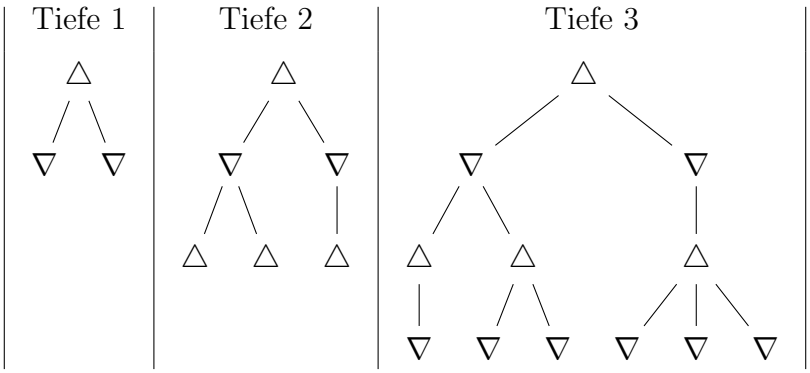

Abbildung 4: Ablauf des Iterativen Deepenings

2.4 Alpha-Beta Pruning

Das Alpha-Beta Pruning ist eine Optimierung zum Minimax Algorithmus. Die Idee des Algorithmus ist, dass manche Zweige des Suchbaums nicht untersucht werden müssen, da für den anderen Spieler diese Züge nicht in Frage kommen. Hierbei ist α der Wert für den Spieler, für den die niedrigen Werte besser sind und β für den anderen Spieler. Für jeden Knoten, je nachdem, ob er ein maximierender oder ein minimierender Knoten ist, wird überprüft, ob ein Kind-Knoten, welcher einen neuen Wert erhalten hat, nicht mehr vom Knoten beachtet werden muss. Der Vorteil des Alpha-Beta Prunings zu Minimax ist, dass der verbrauchte Speicher weniger wird, da vom Baum Zweige nicht beachtet werden müssen. Was wiederum zur Folge hat das die Ausführungszeit des Algorithmus schneller ist und gleichzeitig auch dasselbe Ergbnis wie Minimax zur Folge hat. [SPS]

Wenn man Abbildung 2 und 3 als Beispiel nimmt und den Alpha-Beta Pruning Algorithmus anwendet, so kann der Zweig mit dem Wert 42 ignoriert werden, siehe Abbildung 5. Der gelbe Knoten bekommt eine -1 durch seinen linken Zweig vorübergehend zugewiesen. Da der Knoten des Rechten Zweiges ein Maximierer ist, also immer den Wert des Kindknotens mit dem höchsten Wert nimmt, und dieser bereits einen Knoten mit dem Wert 42 gefunden hat, wird sein Wert definitiv mindestens 42 sein. Der restliche Rechte Zweig des gelben Knotens kann nun ignoriert werden, da der linke Zweig mit -1 definitiv kleiner sein wird.

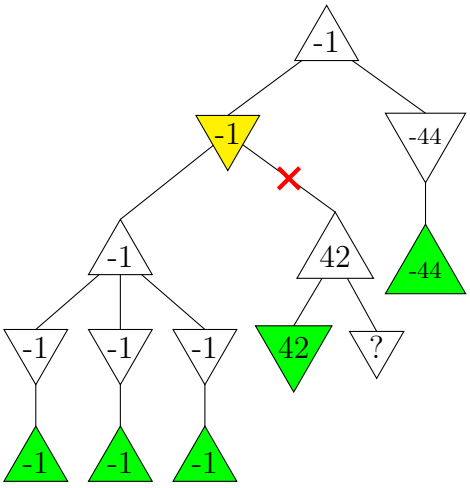

Abbildung 5: Gewinn durch Alpha-Beta Pruning

2.5 Zugsortierung

Zugsortierung ist ein Zusatz zur Alpha-Beta Suche. Da Alpha-Beta Pruning abhängig von der Reihenfolge, in der die Zustände untersucht werden ist, ist es sinnvoll die Nachfolger zu wählen, die die besten Werte erbringen. Den besten Nachfolger findet man, in dem man eine weitere Bewertungsfunktion einbaut, welche nicht so genau wie die Bewertungsfunktion an den Terminalknoten sein muss. Wenn ein Knoten also alle möglichen Nachfolgezüge als Kinder bekommt, werden auf diese die vereinfachte Bewertungsfunktion angewandt und, je nach Ergebnis der Funktion werden die Nachfolger sortiert. Dadruch, dass der beste Zug nun sehr weit am Anfang steht, ist es sehr wahrscheinlich das die anderen Züge durch Alpha-Beta Pruning ignoriert werden. [Rus12]

Auf der Linken Seite der Abbildung 6 kann man erkennen, dass Alpha-Beta Pruning keinen Effekt auf die beiden gelben Knoten hätte. Ändert man jedoch die Reihenfolge der Knoten, so kann der Knoten mit dem Wert 42 ignoriert werden. Bei der Bewertungsfunktion könnte man die bedrohte Figuren als Faktor haben, um auf dieses Ergbnis zu kommen.

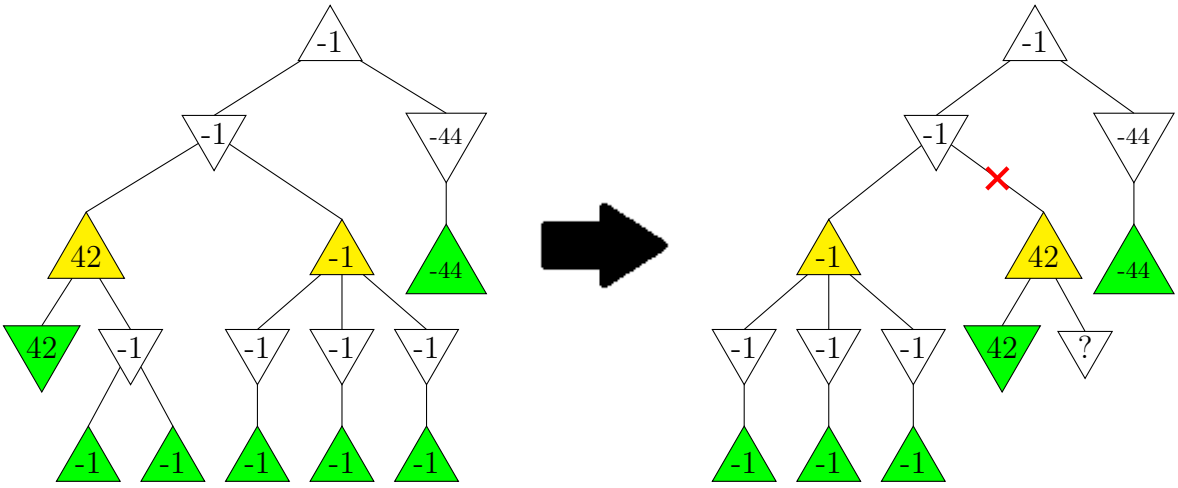

Abbildung 6: Beispiel einer Zugsortierung

2.6 Monte Carlo Tree Search (MCTS)

Der Monte Carlo Tree Search Algorithmus, ist ein heuristischer Algorithmus, bei welchem von einem Zustand eines Spieles zufällig endlich viele Simulationen durchgeführt werden. Die Simulation endet, wenn ein Ergebnis des simulierten Spieles feststeht. Das Wiederholen der Simulationen aus verschiedenen Knoten hat zur Folge, dass das Ergebnis immer genauer wird. Am Ende wird der Knoten gewählt, bei dem die Simulationen die besten Ergebnisse für den momentanen Spieler gezeigt haben. Ein Vorteil des MCTS-Algorithmus gegenüber Minimax ist, dass erst am Ende eines Durchlaufs eine Bewertungsfunktion benötigt wird. Allgemin besteht der Algorithmus aus vier Schritten:

- *Selektion*: Versucht wird, einen Zustand zu finden der noch erweiterbar ist, also einen Zustand zu finden, der kein Endzustand ist und noch nicht besuchte Züge hat.
- *Expansion*: Der Spielbaum wird zufällig um einen noch nicht besuchten Zug erweitert.
- *Simulation*: Von dem gewählten Knoten aus wird nun ein Spiel zufällig bis zum Ende simuliert.
- *Backpropagation*: Das Ergebnis der Simulation wird den vorhergehenden Knoten mitgeteilt und diese werden mit diesem aktualisiert.

Da man im Normalfall nicht beliebig viel Zeit hat alle Möglichkeiten zu simulieren, versucht man die limitierte Zeit so gut wie möglich zu nutzen und die richten Knoten zum Expandieren zu wählen. Dazu wird der *upper confidence bound for trees* (UCT) verwendet. Die UCT Formel lautet:

$$w + c\sqrt{\frac{\log N}{n}} \tag{1}$$

Wobei w die prozentuale Anzahl an Gewinnen des Knotens, N die Anzahl der gesamten Expansionen und n die Expansionen nur an dem betrachteten Knoten sind. Die Aufgabe der UCT Formel ist das Erreichen von zwei im Konflikt stehenden Zielen. Das erste Ziel ist es die Knoten die bisher die höchsten Chancen auf den Gewinn haben tiefer zu simulieren, um eine bessere Genauigkeit des besten Zuges zu haben. Das zweite Ziel ist Knoten, die noch nicht sehr oft besucht worden sind genauer zu untersuchen, da diese vielversprechender sein könnten als gedacht. Für die Balancierung der beiden Ziele gibt es den Parameter c [MP19].

Als Beispiel wird die zuvor verwendete Stellung aus 2 benutzt, aus welcher der MCTS-Baum in Abbildung 7 entsteht. „B“ steht für die Siege aus Schwarzer und „W“ für Siege aus Weißen Sicht, nach der Beendigung von 33 Simulationen. Weiß entscheidet sich in diesem Baum für den gelben Knoten, da dieser bei Betrachtung der Gewichtung von Siege eine höhere Gewinnchance für ihn hat.

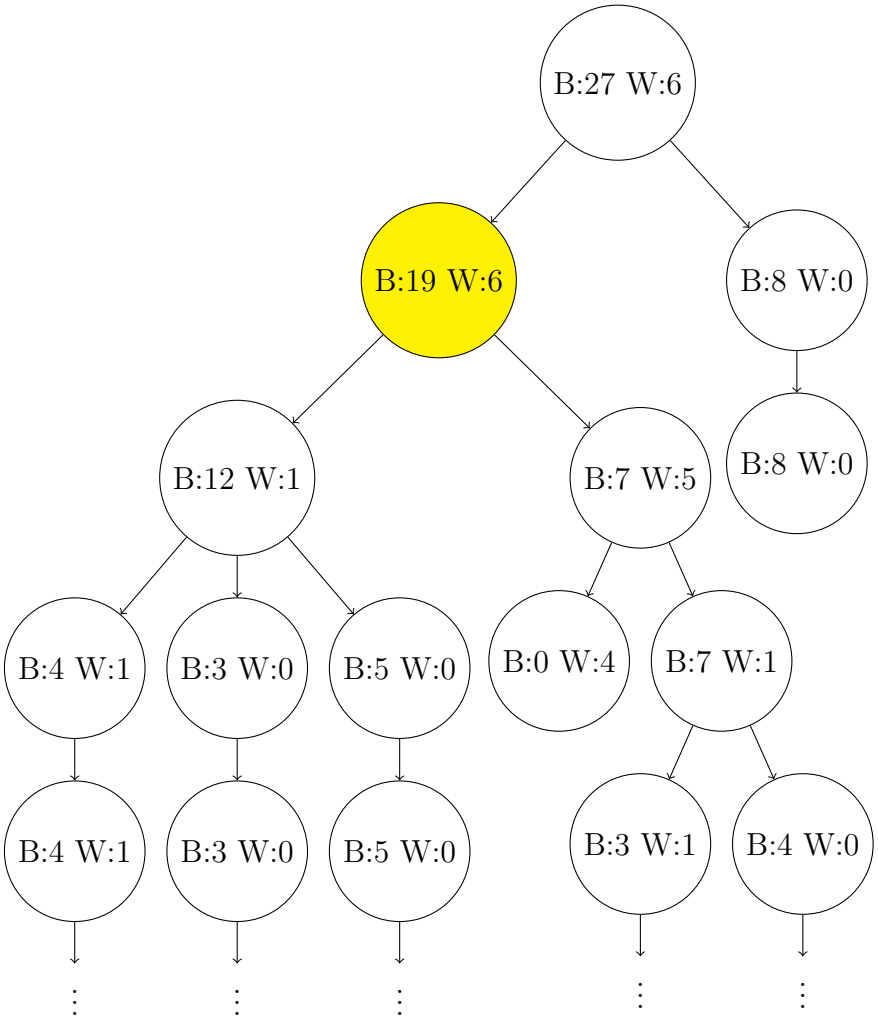

Abbildung 7: Minimax-Baum mit Bewertung der Stellungen

3 Architektur der Software

Dieses Kapitel setzt sich mit der Softwarearchitktur auseinander.

3.1 Überblick

Um für eine bessere Modularität und Erweiterbarkeit zu sorgen, ist die Software auf drei Teile aufgeteilt. Der Gameserver hält die Spielelogik und den Zustand des Spieles. Über die ReversiXT Graphische Oberfläche kann man über den Gameserver gegen die Dame KI des KI Clients spielen. Der KI Client ist eine Dame künstliche Intelligenz, welche sich mit dem Gameserver verbindet und die berechneten Züge an diesen sendet.

3.2 Kommunikation der Komponenten

Ein Spiel kann nur über die GUI der Applikation gestartet werden, dabei entstehen verschiedene Szenarien, abhängig davon was der User vor dem Spiel einstellt:

- Benutzer gegen Benutzer
- Benutzer gegen KI
- KI gegen KI

Egal welche Option gewählt wird, der Gameserver wird immer gestartet, da dieser die Züge überprüft und Sieg oder Niederlage auswertet. Bei Benutzer gegen Benutzer wird die KI Komponente der Software nicht gestartet, es Kommuniziert die GUI-Komponente direkt mit Gameserver. Wird Benutzer gegen KI gewählt, wird eine KI Komponente gestartet. Der Gameserver verbindet sich dann mit dieser und der GUI und verteilt abwechselnd Zugaufforderungen an beide. Wird sich für zwei KI Clients die gegeneinander spielen entschieden, so werden auch zwei gestartet. Die Kommunikation findet nur mehr von Gameserver mit den beiden KI-Clients statt, jedoch hat die GUI-Komponente eine Man-in-the-Middle-Funktion wodurch sie die Kommunikation abhört und die gespielten Züge darstellen kann.

3.2.1 Kommunikations Ablauf Benutzer gegen Benutzer

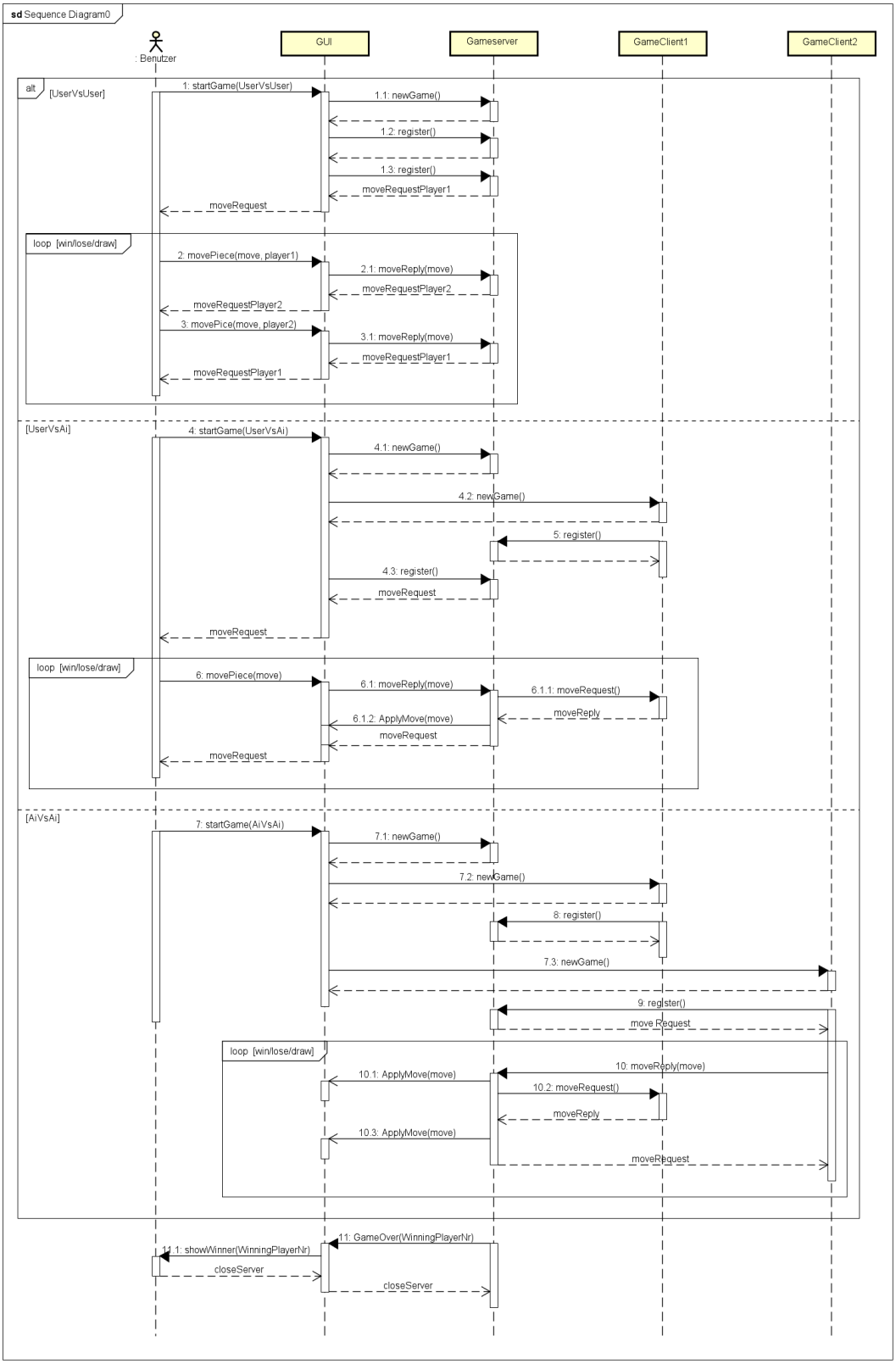

Abbildung 8: Das UML Sequenzdiagramm der Kommunikation

3.2.2 Kommunikationsprotokoll

Im Abschnitt 3.2 wird eine Kommunikation zwischen dem Gameserver und seinen Clients beschrieben, bei welcher der Server nur Anfragen eines festgelegten Protokolles Akzeptiert. Das Protokoll für den Server orientiert sich stark nach dem Protokoll des Reversi-Gameservers.

Typ (8-Bit-Integer)	Länge der Nachricht n (32-Bit-Integer)	Nachricht (n Bytes)

3.3 Gameserver

Als Gameserver wird der Teil der Software betrachtet, welcher für das einhalten der Spielregeln und die Verwaltung der Spielfeldzustandes, verantwortlich ist.

3.3.1 Softwareaufbau des Gameservers

Die Software des Gameservers ist in zwei Pakete aufgeteilt, in das Gamerulbook- und Serverconnection-Paket. Der Serverconnection Teil ist für die Verbindung und das Dekodieren der Nachrichten der Clients verantwortlich. Nachdem ein Client verbunden ist und seine Nachrichten schickt, wird diese dekodiert und falls das Protokoll korrekt eingehalten wird, an den Gamerulbook Teil weitergeleitet. Das Gamerulbook Paket verwaltet den Spielzustand, sowie die Spielregeln des laufenden Spieles. Es ist über das GameRules Interface mit dem Serverconnection Paket verbunden. Ein beliebiges Spiel kann mittels dieses Interfaces als eigenes Paket festgelegt werden, was zu einer Möglichkeit führt den Gameserver um beliebig viele Spiele erweitern zu können. Die Board-Klasse beeinhaltet das Spielbrett, welches beim start des Servers übergeben wird. Die AllPossibleMoves-Klasse ist verantwortlich alle möglichen Züge die in einer Stellung möglich sind dazustellen. Beide Klassen sind unabhängig von der Implementierung des Spieles über das GamesRules-Interface.

Abbildung 9 zeigt den Aufbau als UML-Klassendiagramm, dabei sind Dame und Mühle as Spiele über das GameRules Interface als Pakete Beispielhaft gezeigt.

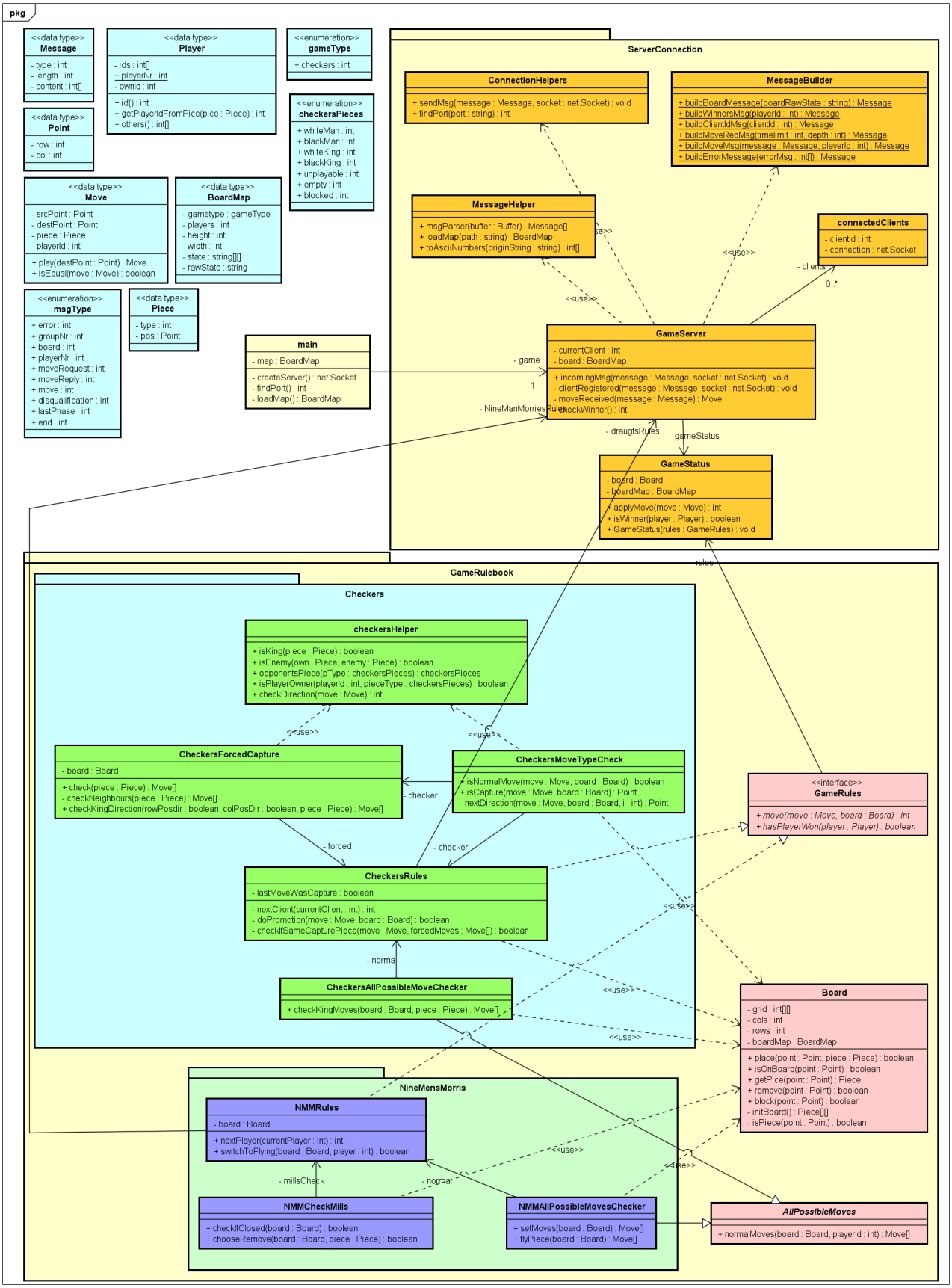

Abbildung 9: Das UML Klassendiagramm des Gameservers

3.4 ReversiXT GUI

Die ReversiXT Graphische Oberfläche (GUI) ist der Teil der Software der für die eigentliche Userinteraktion gedacht ist. User Können über die GUI spiele starten, bei welchen sie gegen KI's oder andere User spielen können.

3.4.1 Softwareaufbau der ReversiXT GUI

Die ReversiXT GUI besteht aus zwei Teilen, dem Server und dem Webapp-Paket, siehe 10. Dabei behandelt der Server, die Verbindung zu den anderen Komponenten. Er startet den Gameserver und je nachdem wie viele KI's als Spieler gewählt werden, startet er auch die Gameclients dafür. Sind die anderen Komponenten gestartet, kommunziert der Server der ReversiXT GUI nur noch mit Gameserver und benutzt dabei das Protokoll welches dieser vorschreibt. Die Webapp behinhaltet das Benutzer Interface, über welches die Software für den Endnutzer bedient werden kann. Über das Menü kann kann die Gameselection augerufen werden, bei welcher die verfügbaren Spiele ausgewählt werden können. Wird ein Spiel ausgewählt, gibt es ein weiteres Menü über dass Spielspezifische Parameter einstellbar sind. Die Spiele sind als eigene Pakete eingegliedert und behinhalten, je nach Spiel ein Spielbrett und Möglichkeiten mittels Touch-Berührungen Züge auszuführen. Durch diese Untergliederung in weitere Pakete kann die Software leicht um weitere Spiele erweitert werden.

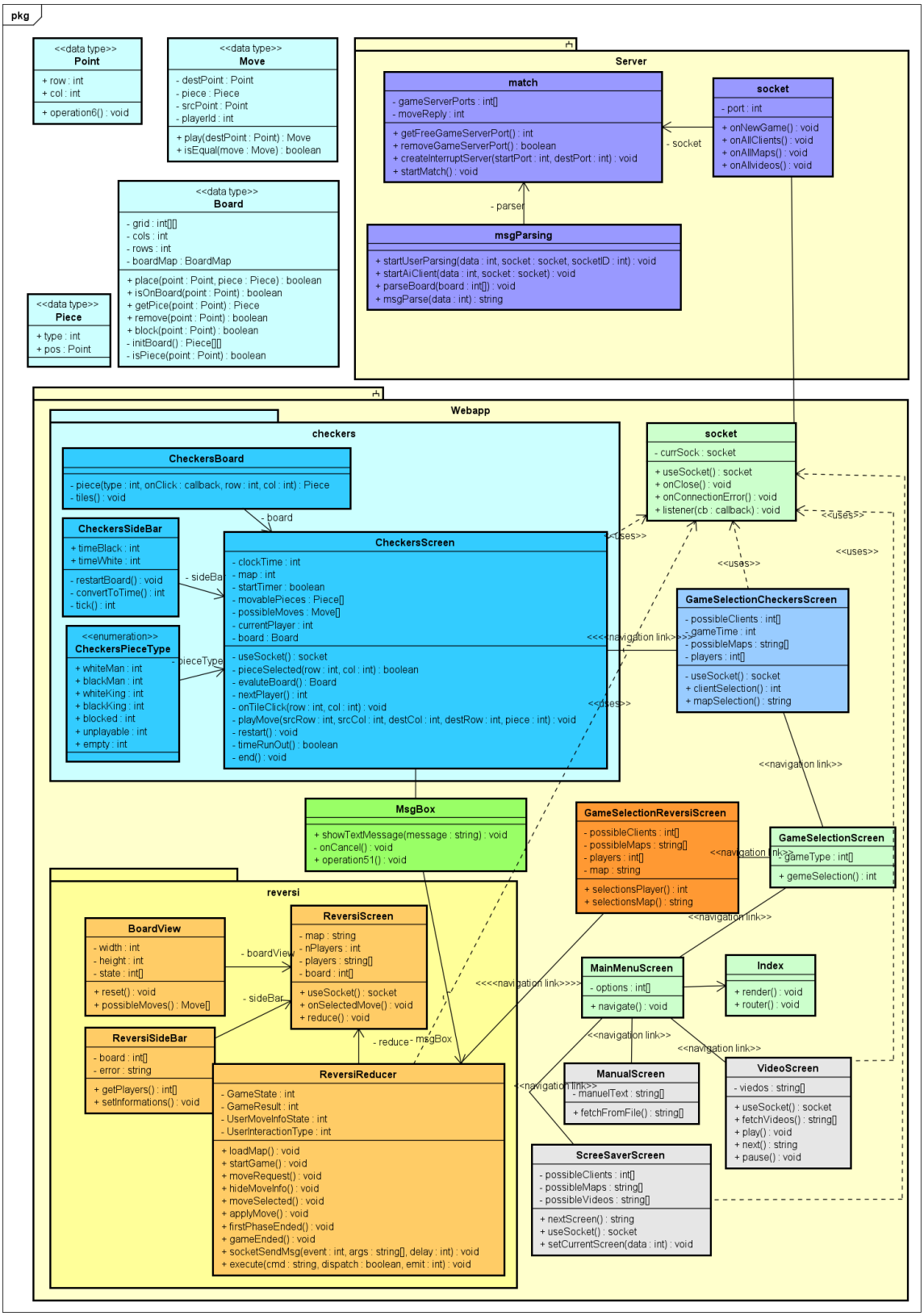

Abbildung 10: Das UML Klassendiagramm der ReversiXT GUI

3.5 KI Client

Der KI Client beinhaltet die Logik der Künstlichen Intelligenz der Applikation. Wird ein Spiel gegen einen KI Client gestartet, so wird dieser gestartet und agiert als Gegenspieler zum User. Der User kann auch ein Spiel bei welchem zwei KI's gegeneinander spielen starten, wodurch zwei KI Clients gestartet werden.

3.5.1 Softwareaufbau des KI Clients

Der KI Client besteht aus drei Paketen, der KI-Logik, der Spielelogik und der Serververbindungslogik, siehe 11. Die Spielelogik, beeinhaltet ein Momentanzustand des Spielbrettes, sowie Möglichkeiten dieses nach belieben zu modifizieren. Das KI-Paket benutzt diese Logik um Züge auf auszuwerten und den bestmöglichen Zug zu wählen. Um verschiedene KI-Algorithmen verwenden zu können wird ein Interface bereit gestellt, wird die Applikation um einen weiteren KI-Algorithmus erweitert, so muss dieser das Interface verwenden. Wird eine Auswertung des Nächsten besten Zuges abeschlossen, so sendet das KI-Paket sein Ergebnis an die das Server-Paket. Dieses ist verantwortlich für das Versenden und kodieren der Nachrichten. Nachrichten werden vom KI-Client zum Gameserver geschickt und müssen dabei das Protokoll von diesem einhalten.

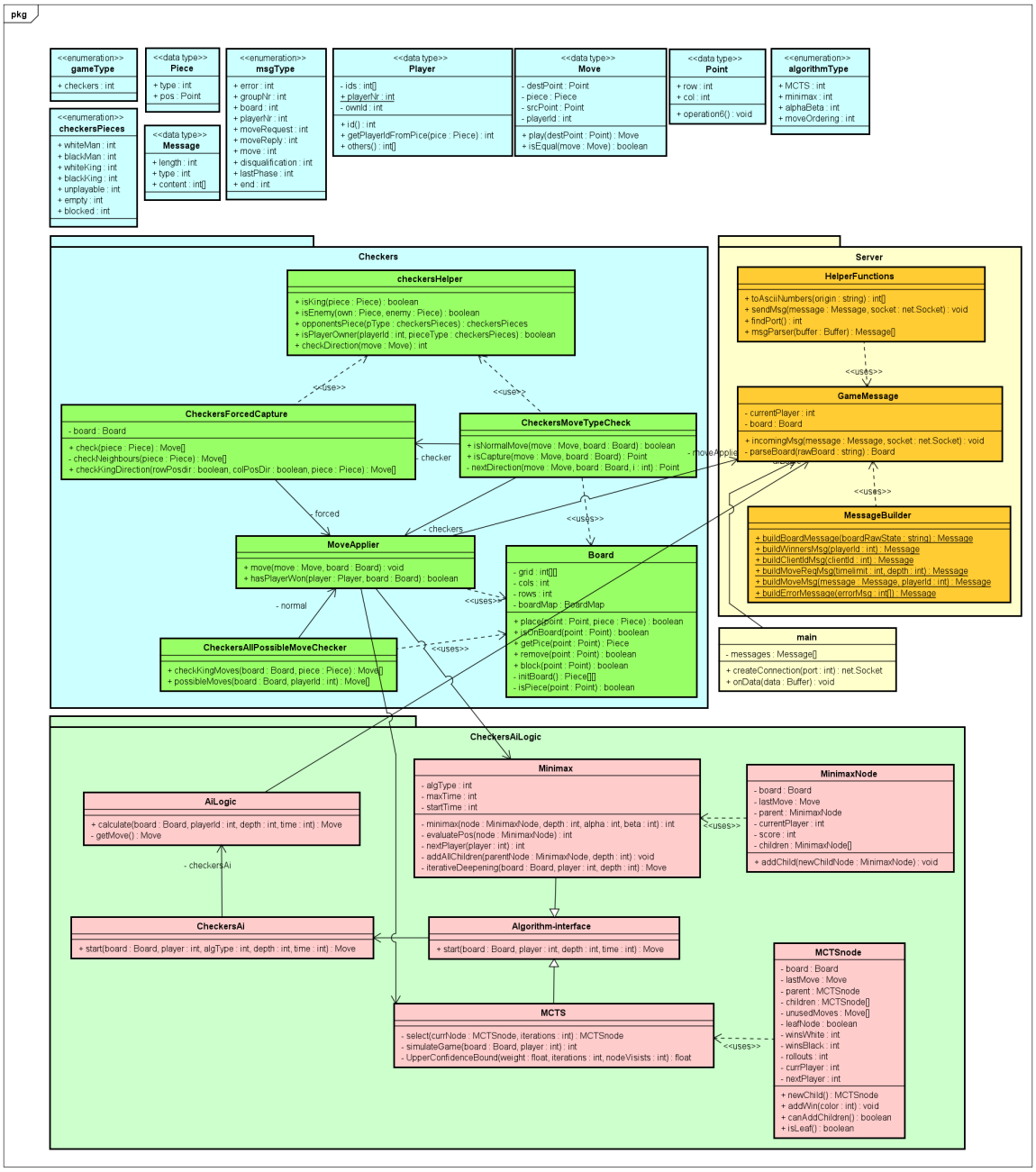

Abbildung 11: Das UML Klassendiagramm des KI Clients

4 Hardware

Dieses Kapitel handelt von der verwendeten Hardware auf der die Software zum läuft. Die Software wird auf einem Raspberry Pi, welcher an einem Touch Monitor anschlossen ist ausgeführt.

4.1 Raspberry Pi

4.2 Touch Monitor

5 Implementierung

5.1 Eingesetzte Sofwarekomponenten

5.1.1 Programmiersprachen und Frameworks

5.1.2 Datentransferprotokolle

5.2 Gameserver

5.2.1 Netzwerkspezifikation des Gameservers

5.3 Graphische Oberfläche

5.3.1 Gegebene React Anwendung

5.3.2 Erweiterungen

5.4 KI Client

5.4.1 Vergleich der KI Algorithmen

6 Testing

6.1 Integrationstest

6.2 Ergebnisse

7 Fazit und Ausblick

Literaturverzeichnis

[Kor85] Richard E. Korf. *Depth-First Iterative-Deepening: An Optimal Admissible Tree Search*. Techn. Ber. University of Columbia, 1985.

[MP19] Kevin Ferguson Max Pumperla. *Deep Learning and the Game of Go*. Manning, 2019.

[MSC82] T. A. Marsland M. S. Campbell. *A Comparison of Minimax Tree Search Algorithms*. Techn. Ber. University of Alberta, 1982.

[Rus12] Stuart Russell. *Künstliche Intelligenz ein moderner Ansatz*. Pearson, 2012.

[SPS] M. Sridevi Shubhendra Pal Singhal. *Comparative study of performance of parallel Alpha Beta Pruning for different architectures*. Techn. Ber. National Institute of Technology, Tiruchirappalli.

Quellenverzeichnis

[Dra] *Mindsports*. Die Geschichte von Dame. URL: http://www.mindsports.nl/index.php/arena/draughts/478-the-history-of-draughts.

[Int] *World Draughts Federation*. official FMJD rules for international draughts 100. URL: http://www.fmjd.org/?p=v-100.

Anhang

Inhalt des beigefügten Datenträgers:

- ...
- ...

A Domändenmodell

Ein toller Anhang, der nicht nur als „*Müllhalde*“ genutzt wird, sondern in dem Bilder und Inhalte auch mit eigenen Worten erklärt werden und den man auch für sich alleine lesen kann. Es sollten auch Referenzen auf die zugehörige ausführliche Behandlung im Hauptteil inklusive Seitenangabe mit `\pageref` gegeben werden.

Screenshot

Unterkategorie, die nicht im Inhaltsverzeichnis auftaucht.