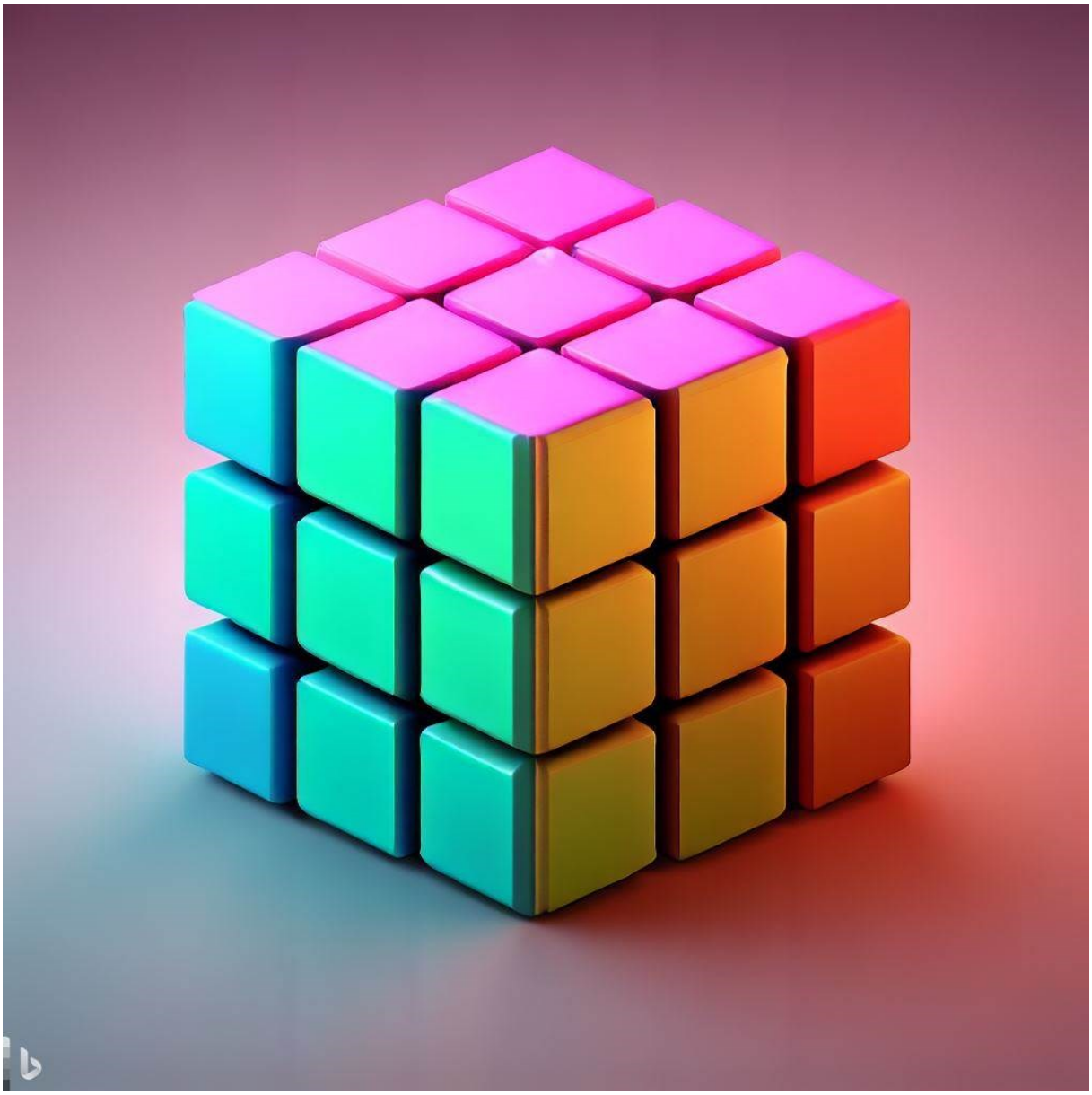
Softwareprojekt: Qubic

Pflichtenheft



Name: Qubic

Mitglieder: Timo Gisder, Arne Pfüller, Valentin Schaar

Datum: 11.05.2023

Inhaltsverzeichnis

1 Ziele

1.1 Minimalanforderung

Minimalanforderung:

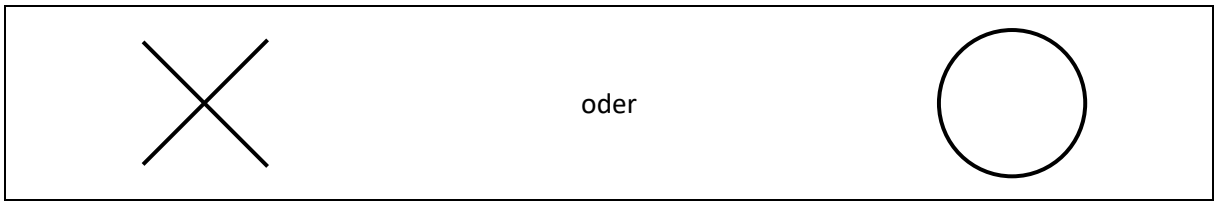
Bei dem Spiel „Tic-Tac-Toe“ handelt es sich um das bekannte Strategiespiel, bei dem abwechselnd Kreise und Kreuze gesetzt werden. Gewonnen hat derjenige, der zuerst eine Reihe des eigenen Symbols gebildet hat. Bei der Erweiterung Qubic handelt es sich um eine 3-dimensionale Variante von Tic-Tac-Toe.

Es besitzt eine funktionelle graphische Oberfläche, die die Seiten des Felds als 2D-Graphik darstellt. In die Tiefe hinein kann man die Ebenen wechseln. Zudem spielt man gegen eine KI, die bestmöglich Züge voraussagt.

Spielprinzip:

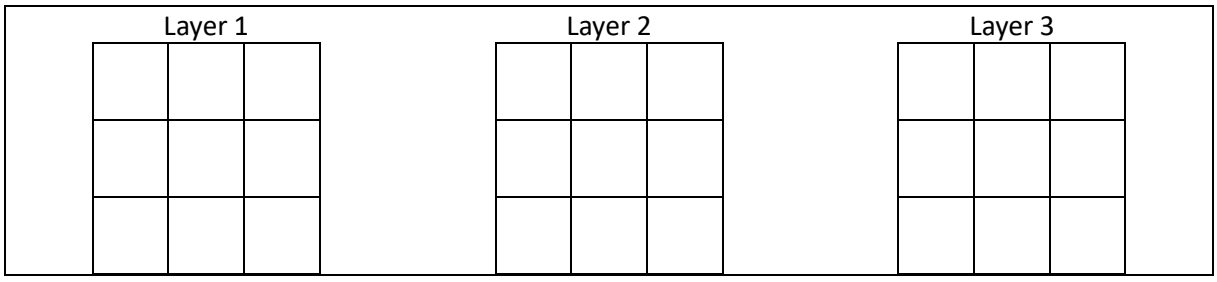
Anfangs wählt der Spieler ein Symbol, entweder Kreis oder Kreuz.

Mögliche Symbole:



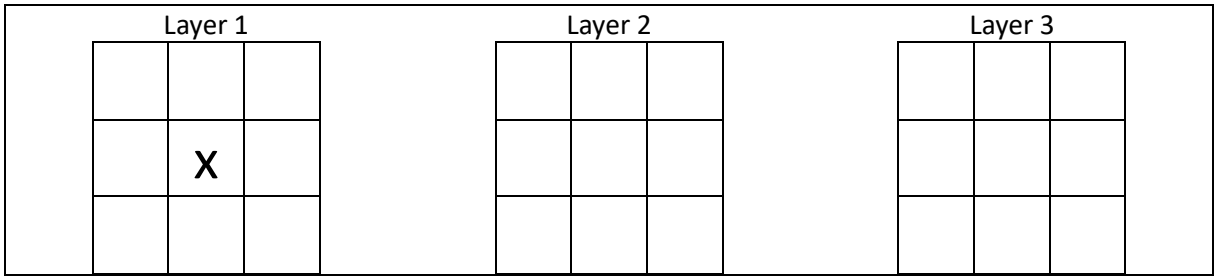
Es gibt ein leeres Spielfeld von der Größe 3 x 3 x 3 Felder.

Anfängliches Spielfeld:

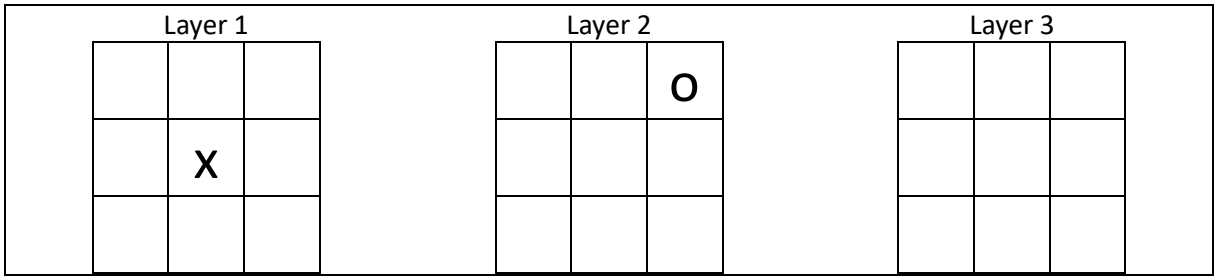


In jedem freien Feld kann ein Spieler sein Symbol platzieren. Abwechselnd wird nun das eigene Symbol gesetzt. Der Spieler beginnt.

(Beispielhafter) erster Spielzug:

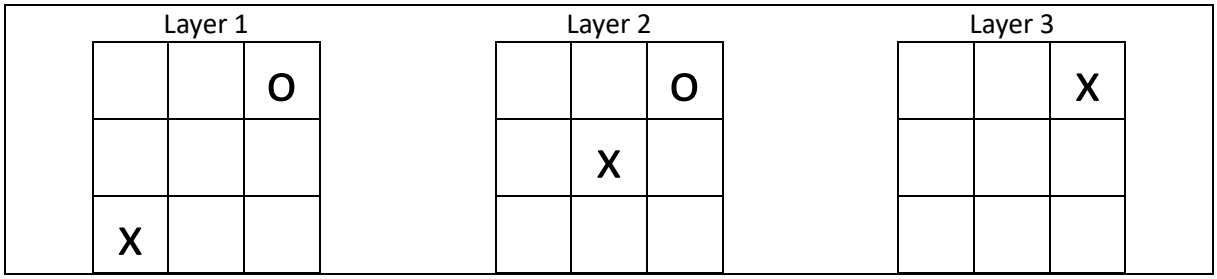


(Beispielhafter) zweiter Spielzug:



Gewonnen hat derjenige, der zuerst in seinem Zug eine Reihe seines Symbols gebildet hat.

(Beispielhafte) Siegesposition für x:



1.2 Zusatzanforderung

Zusatzanforderung:

1. Die 2D-Oberfläche wird mit einer möglichen Rotation erweitert. Mit den Pfeiltasten der Tastatur kann das Spielfeld gedreht werden.
2. Statt einer reinen 2D-Oberfläche wird das Spielfeld dreidimensional virtualisiert. Hierfür kann die Bibliothek OpenGL verwendet.
3. Das Spiel ist online verfügbar, sodass es nicht mehr auf dem eigenen PC ausgeführt werden muss. Das kann über einen Raspberry-Pi-Homeserver erreicht werden.
4. Das Spiel kann nicht nur gegen eine KI, sondern auch gegen einen anderen Spieler gespielt werden. Die Spieler wechseln sich an einem Computer mit den Zügen ab.

2 Systemanforderung

2.1 Hardware

Hardwareanforderungen:

Es wird ein durchschnittlicher Arbeitscomputer benötigt. Erforderlich für einen flüssigen Spielverlauf sind mindestens 8 GB erforderlich. Um das Spiel zu installieren sind mindestens 1 GB freier Festplattenspeicher erforderlich. Zudem wird für die Spielsteuerung eine Tastatur und eine Maus benötigt.

Für die Zusatzanforderungen ist dazu noch eine Internetverbindung über WLAN oder LAN erforderlich.

2.2 Software

Softwareanforderung:

Der Computer muss Windows 10 als Betriebssystem installiert haben. Des Weiteren wird für die Ausführung des Programmcodes mindestens Python 3.8 benötigt. Für die GUI-Oberfläche wird außerdem die Bibliothek *PyGame* verwendet.

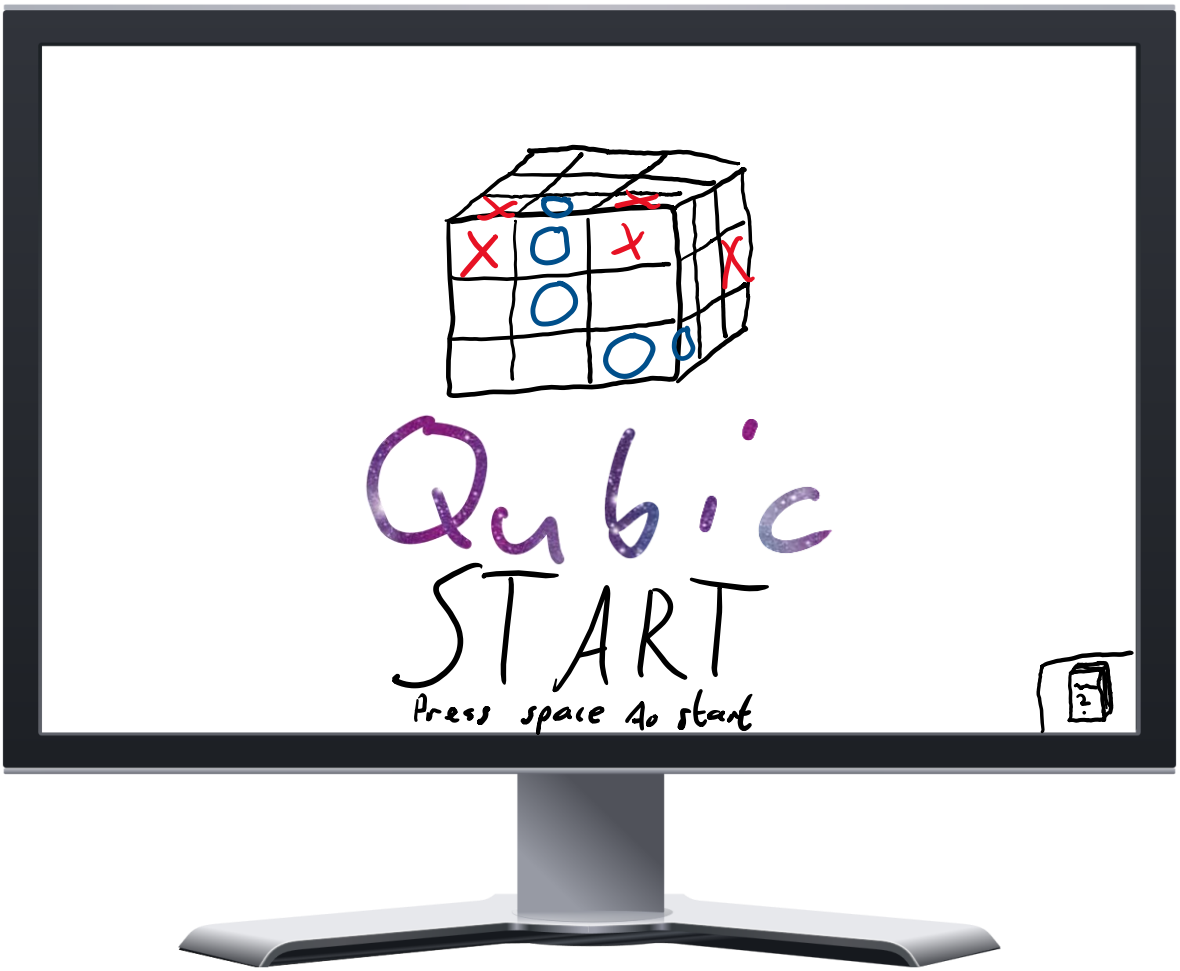
Merkmale:

Merkmal	**--**	**-**	**o**	**+**	**++**
Benutzerfreundlichkeit					x
Korrektheit				x	
Wartungsfreundlichkeit			x		
Zuverlässigkeit		x			
Effizienz	x				

3 Produktumgebung

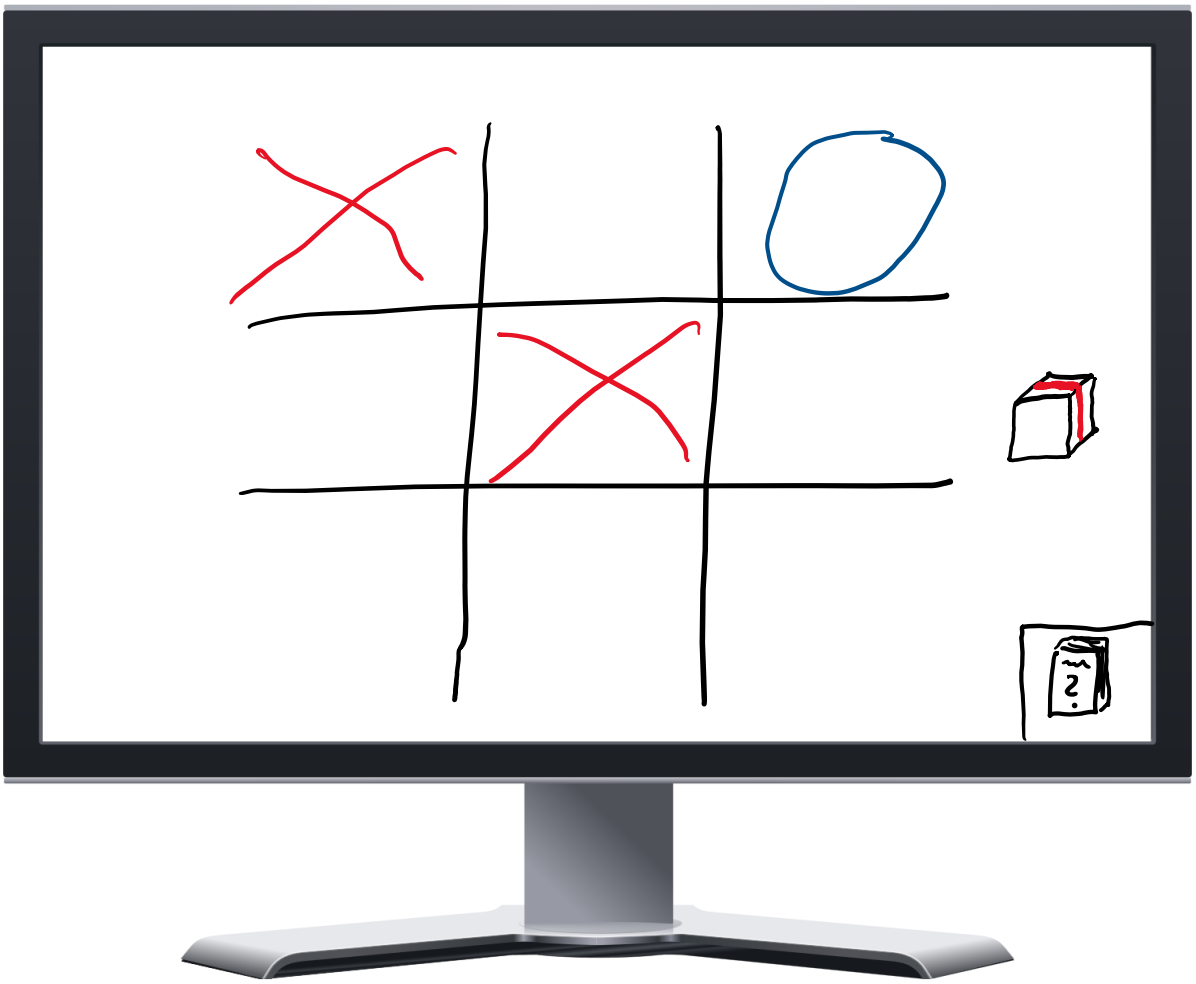
3.1 Benutzeroberfläche

Startbildschirm:



Auf dem Startbildschirm wird dem Benutzer eine einladende Oberfläche in HD (1280×720 Pixel) präsentiert. Sie ist Damit das Spiel gestartet wird, muss die Leertaste gedrückt werden. Über das Buch-Icon unten rechts kann eine Spielanleitung in Form einer PDF geöffnet werden.

Spielansicht:



Das Spieldesign ist simpel gehalten. In der Mitte befindet sich das Tic-Tac-Toe-Feld aus der Front-Perspektive. Zwischen den drei Schichten kann mithilfe des Mausrads gewechselt werden. Ein Würfel auf der rechten Seite gibt diese aktuelle Position mit einer farbigen Markierung an. Der Hilfe-Button aus dem Startmenü bleibt bestehen. Mit der Esc-Taste kann zum Hauptmenü zurückgekehrt werden.

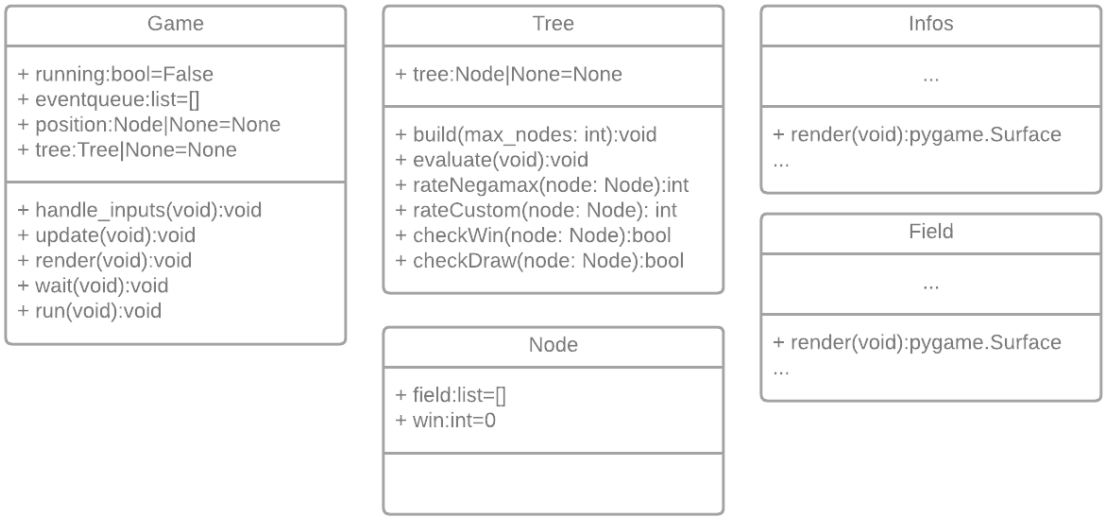
Verlust-, Unentschieden- bzw. Siegesposition:

Hat der Spieler gewonnen, verloren oder unentschieden gespielt, erscheint eine jeweilige Meldung in der Mitte des Bildschirms. Mithilfe der Esc-Taste kann zur Spielansicht zurückgekehrt werden. Möchte man zum Hauptmenü gelangen, muss Esc noch einmal gedrückt werden. Bei einem Gewinn/ Verlust werden die jeweiligen Symbole farbig hervorgehoben.

3.2 Klassendiagramm

Klassendiagramm:

Spielablauf	Logik	Grafik



Ausführung:

Der Spielablauf stellt den Hauptprozess des Spiels dar. Mit ihm ist es überhaupt möglich, es zu spielen. Eingaben werden eingelesen, verarbeitet und entsprechend reagiert. Die Spielschleife sorgt dafür, dass das Spiel solange läuft, bis es beendet wird.

Bei der Logik handelt es sich um einen Spielbaum, der von der KI benutzt wird, um den nächsten Zug zu bestimmen. Die nächsten möglichen Positionen werden festgestellt und bewertet. Es wird überprüft, ob man gewonnen hat.

Die Grafik-Klassen werden von der Spielablaufklasse angefordert und deren Inhalte auf dem Bildschirm angezeigt. Jede dieser Klassen besitzt je nach Zweck eine individuelle Renderfunktion.

KI:

Die KI ist eine Funktion, die nur eine (für den Spieler) Verlustposition aus dem Baum auswählt. Sie ist keine eigene Klasse.

3.3 Spezifikation

Game:

handle_inputs(void):void

Voraussetzung: keine

Effekt: Eingaben werden eingelesen und gespeichert.

update(void):void

Voraussetzung: keine

Effekt: Die Logik-Klassen werden anhand der Eingaben geupdated.

render(void):void

Voraussetzung: keine

Effekt: Aktualisierung und visualisiert das Spiel auf dem Bildschirm.

wait(void):void

Voraussetzung: keine

Effekt: Auf den nächsten Frame wird gewartet.

run(void):void

Voraussetzung: keine

Effekt: Die Hauptschleife des Spiels wird ausgeführt.

Tree:

build(max_nodes: int):void

Voraussetzung: *max_nodes* gibt die Anzahl der Knoten an, die an den Baum ergänzt werden sollen.

Effekt: Der Baum wird um die Anzahl der Knoten erweitert.

evaluate(void):void

Voraussetzung: keine

Effekt: Die Spielpositionen des Baums werden bewertet.

rateNegamax(node: Node):int

Voraussetzung: *node* ist ein Knoten mit einer Spielposition.

Effekt: Mithilfe des Negamax-Algorithmus wird ein Knoten bewertet.

rateCustom(node: Node):int

Voraussetzung: *node* ist ein Knoten mit einer Spielposition.

Effekt: Mithilfe eigener Algorithmen wird ein Knoten bewertet.

checkWin(node: Node):bool

Voraussetzung: *node* ist ein Knoten mit einer Spielposition.

Effekt: Es wird überprüft, ob der Spieler gewonnen hat.

checkDraw(node: Node):bool

Voraussetzung: *node* ist ein Knoten mit einer Spielposition.

Effekt: Es wird überprüft, ob das Spiel unentschieden ausgegangen ist.