Software - Projekt
Mini Chess

Contents

1 Ziele

1.1 Minimalanforderung

- simple Oberfläche → Spielfläche, Start / Resign Button
- rating algo → Player vs. CPU
- 3x3
- pygame GUI
- simples 3x3 Schach → Ziel: als erstes die andere Seite erreichen

1.2 Zusatzanforderung

- Algorythmus zur Vertiefung der Suchtiefe des "Zugbaum"
- 4x4; 5x5
- Player vs. Player
- Multithreading

2 Systemanforderung

2.1 Hardware

- 8 GB RAM
- 32 MB Speicher
- Multicore CPU
- Maus und Tastatur
- Farbbildschirm (empfohlen)

2.2 Software

1. Ausführen via .exe → Windows 10 21H2 +
2. Ausführen via Python
   - Python 3.11+ → Python 3.11 für bessere Effizents
   - Python libarys → einfacher Installationsprozess via requirements.txt
     - PyQt6
     - pygame 2.4

2.3 Merkmale

sehr großer Fokus: ++ großer Fokus: + mittlerer Fokus: *o* kleiner Fokus: − sehr kleiner Fokus: −−

Merkmale	Gewichtung
Benutzerfreundlichkeit	++
Korrektheit	+
Wartungsfreundlichkeit	+
Zuverlässigkeit	++
Effizienz	*o*

3 Produktumgebung

3.1 Benutzeroberfläche

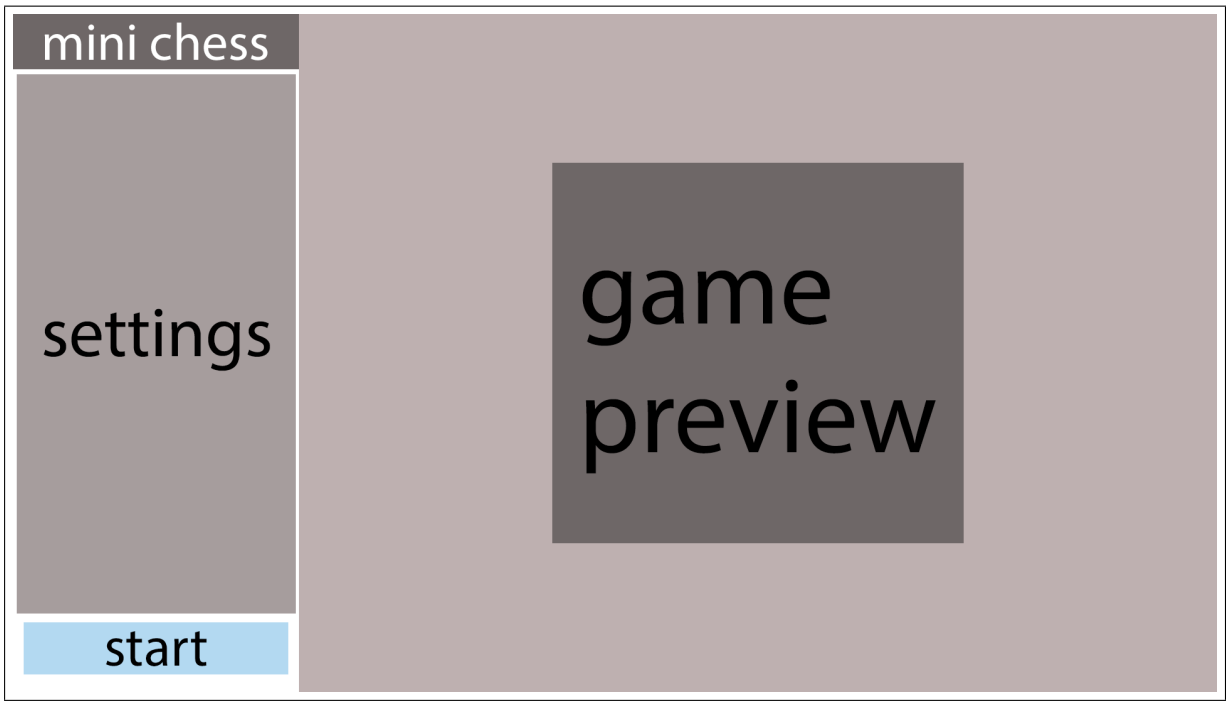
3.1.1 Minimalanforderung

Startscreen:

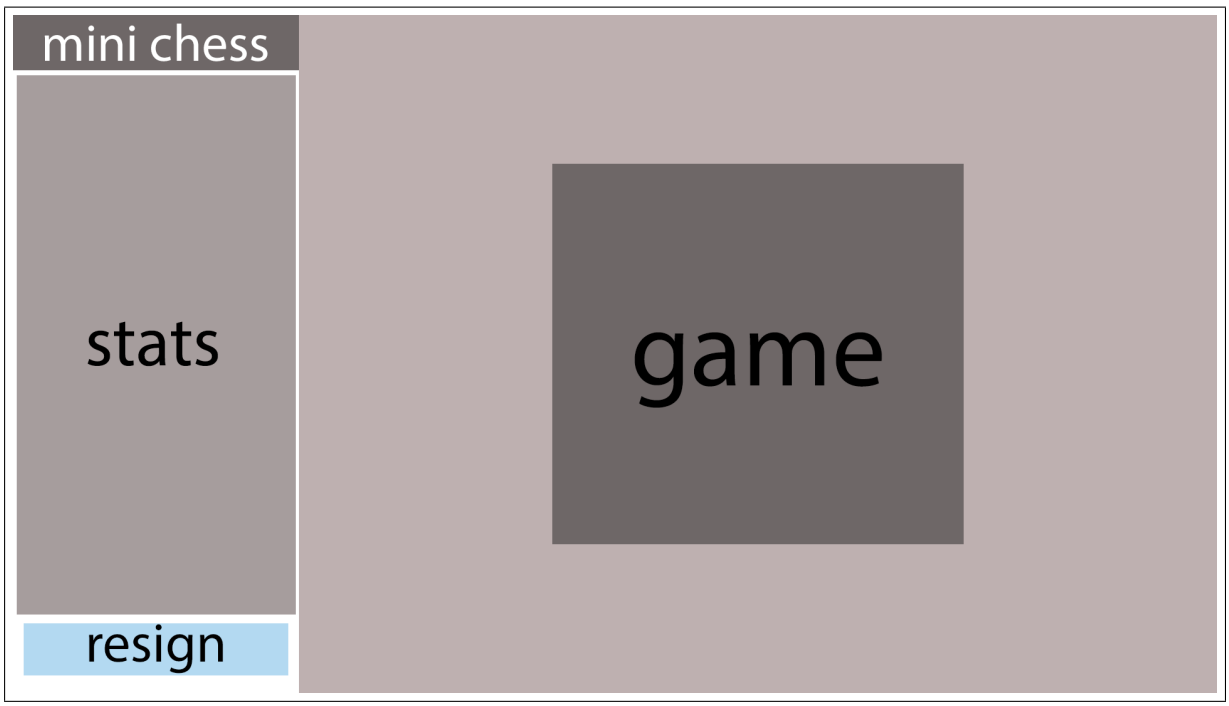
Gamescreen:

3.1.2 Zusatzanforderung

Startscreen:



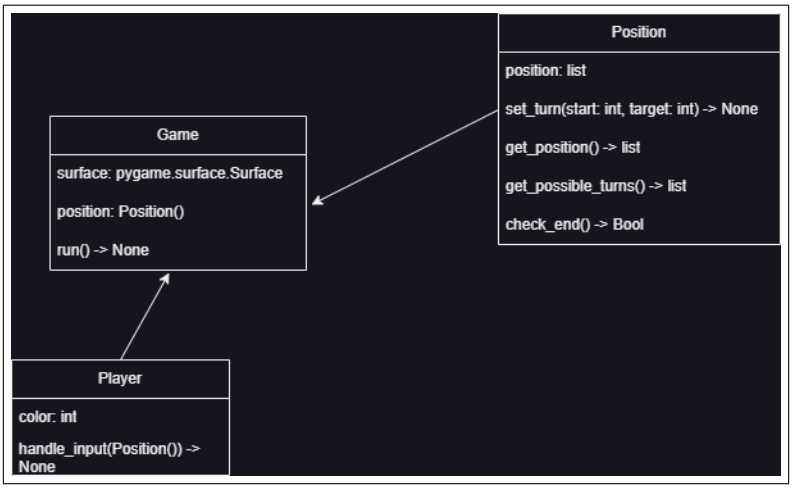
Gamescreen:



3.2 Klasendiagramm

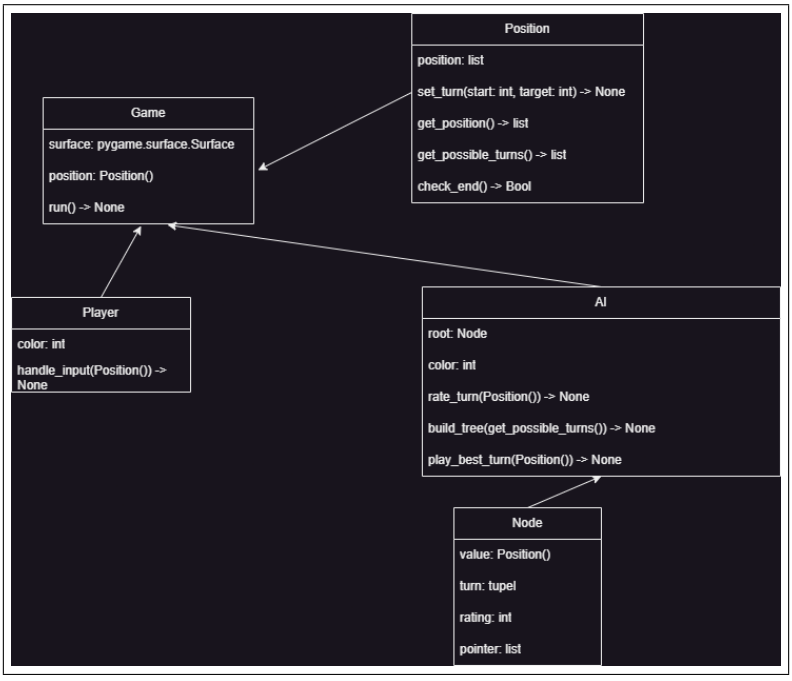
3.2.1 Minimalanforderung

ohne AI



3.2.2 Zusatzanforderung

mit AI



3.3 Spezifikationen

3.3.1 Game

run(): Beherbergt die main game loop, die den Spielzyklus ausführt, die GUI baut und das Spiel beendet.

3.3.2 Position

set_turn(start: int, target: int): Bekommt den Zug als Integer der Felder. Führt den Zug aus und updated den Spielzustand.
get_position(): Gibt das Spielfeld als Liste zurück.
get_possible_turns(): Gibt alle Möglichen Züge als Liste zurück.
check_end(): Gibt True zurück, wenn jemand gewonnen hat und merkt sich den Gewinner.

3.3.3 Player

handle_input(Position): Bekommt die aktuelle Position. Kann dem Spieler die möglichen Züge anzeigen, Wartet auf einen Input des Spielers, überprüft ob dieser valide ist und verarbeitet diesen.

3.3.4 AI

rate_turn(Position): Bewertungsfunktion für die angegebene Position.
build_tree(): Lässt alle möglichen Züge bewerten und baut dann den Baum.
Zusatz: Algorythmus zur Vertiefung der Suchtiefe des "Zugbaum"
play best turn(): Nimmt den besten Zug aus dem Baum und führt ihn aus.

3.3.5 Node

Knoten des Baumes, die alle Informationen über den Zug speichern

4 Arbeitstagebuch

4.1 Chronologie

4.2 Testläufe