

# Software - Projekt

## Mini Chess

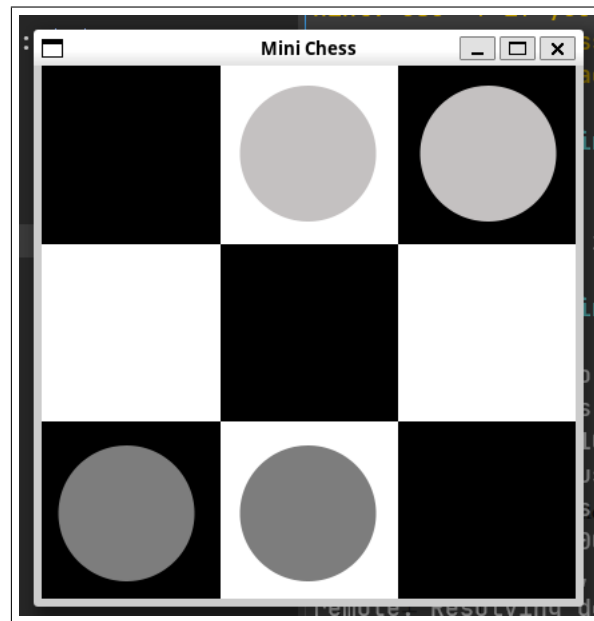
### Contents

<b>1</b>	<b>Ziele</b>	<b>2</b>
1.1	Minimalanforderung . . . . .	2
1.2	Zusatzanforderung . . . . .	2
<b>2</b>	<b>Systemanforderung</b>	<b>3</b>
2.1	Hardware . . . . .	3
2.2	Software . . . . .	3
2.3	Merkmale . . . . .	3
<b>3</b>	<b>Produktumgebung</b>	<b>4</b>
3.1	Benutzeroberfläche . . . . .	4
3.1.1	Minimalanforderung . . . . .	4
3.1.2	Zusatzanforderung . . . . .	5
3.2	Klasendiagramm . . . . .	6
3.2.1	Minimalanforderung . . . . .	6
3.2.2	Zusatzanforderung . . . . .	6
3.3	Spezifikationen . . . . .	7
3.3.1	Game . . . . .	7
3.3.2	Position . . . . .	7
3.3.3	Player . . . . .	7
3.3.4	AI . . . . .	7
3.3.5	Node . . . . .	7
<b>4</b>	<b>Arbeitstagebuch</b>	<b>8</b>
4.1	Chronologie . . . . .	8
4.2	Testläufe . . . . .	8

# 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

- Algorithmus zur Vertiefung der Suchtiefe des "Zugbaum"
- 4x4; 5x5
- spezielle Figuren
- 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 Effizienz
  - Python libraries → 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	<i>o</i>
Validität	++

### 3 Produktumgebung

#### 3.1 Benutzeroberfläche

##### 3.1.1 Minimalanforderung

Startscreen:



Gamescreen:

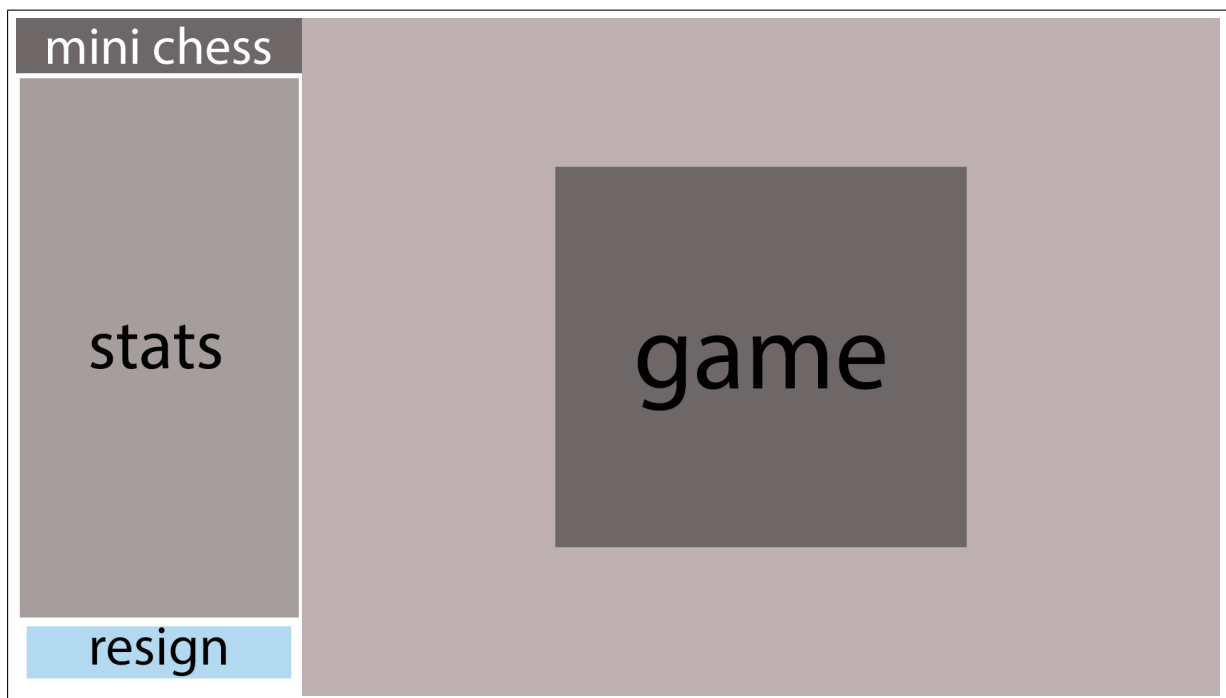


### 3.1.2 Zusatzanforderung

Startscreen:



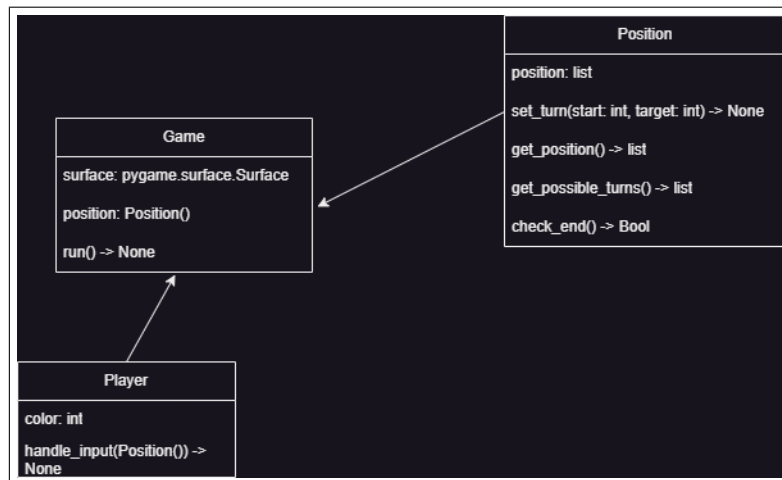
Gamescreen:



## 3.2 Klasendiagramm

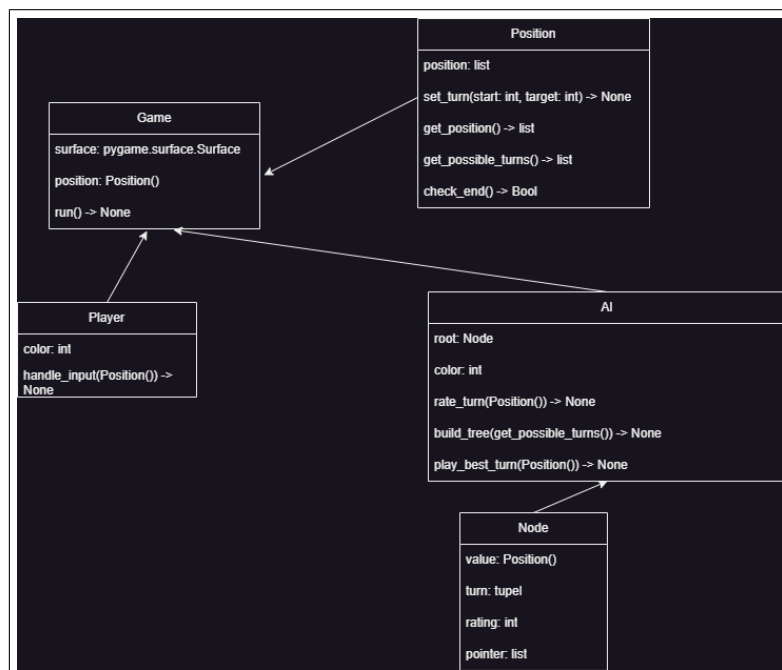
### 3.2.1 Minimalanforderung

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: Algorithmus 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**