Informatik Dokumentation "Schach.py"

Julian Sehbaoui

29. März 2021

https://github.com/JSehbaoui/Chess_Python

Made with LaTeX

Inhaltsverzeichnis

1 Vorwort **3**

2 Ideenfindung **3**

3 Die Regeln im Schach **3**
3.1 Das Spielprinzip . . . 3
3.2 Anfangsstellung der Figuren . . . 3
3.3 Gangart der Figuren . . . 4
3.3.1 Läufer . . . 4
3.3.2 Turm . . . 5
3.3.3 Dame . . . 5
3.3.4 Springer . . . 5
3.3.5 Bauer . . . 5
3.3.6 König . . . 5
3.4 Ende der Partie . . . 6

4 Benutze Pakete **6**
4.1 pygame . . . 6
4.1.1 Warum pygame? . . . 6
4.2 Stockfish . . . 6
4.2.1 Warum Stockfish? . . . 7

5 Installation **7**

6 Bedienung des Programmes **7**
6.1 Wie spiele ich? . . . 7
6.2 Overlays . . . 7
6.2.1 Spielerinformationstabs . . . 7
6.2.2 Spielhistorie . . . 8
6.2.3 Buttons . . . 8

7 Code **8**
7.1 Klassenbeziehungen . . . 8
7.2 Probleme und Problemlösungen . . . 9
7.2.1 Der König, das Problemkind . . . 9
7.2.2 Foresight . . . 9
7.2.3 Stockfish . . . 11

8 Danke an... **11**

9 Schlusswort und Fazit **11**
9.1 Zukünftige Aussichten . . . 12

1 Vorwort

Dieses Projekt findet im Rahmen des Informatikunterrichts des Luisengymnasiums Bergedorf statt und wird als zusätzliche Lernleistung entsprechend gewertet. Ich bin Julian Y. Sehbaoui und habe ein Schachprogramm „from scratch“ in python3 geschrieben

2 Ideenfindung

Ich persönlich habe mich schon seit dem ich klein bin für Schach interessiert. Es ist ein recht simples Spiel, welches aber durch seine Zahlreichen Zugmöglichkeiten zu einem sehr tiefgründigen Spiel werden kann. Als ich mich nach einem neuen Projekt umgesehen habe, kam mir direkt Schach in den Sinn, weil es an strikte Regeln gebunden ist, nicht viele Ausnahmen existieren und ich es wie schon erwähnt sehr gerne spiele.

3 Die Regeln im Schach

Im Folgenden finden Sie alle offiziellen Regeln des standartmäßigen Schachspiels

3.1 Das Spielprinzip

Ein Schachspiel wird zwischen zwei Gegnern ausgetragen, die ihre Spielfiguren auf einem quadratischen Spielbrett ziehen. Das Ziel des Spiels für beide Spieler ist es den gegnerischen König „Schachmatt“ zu setzten. Dieser Zustand ist erreicht, wenn der gegnerische Spieler keine Möglichkeit hat, den König mit dem nächsten Zug aus dem „Schach“ (angegriffener Zustand des Königs) zu bewegen. Falls es für keinen der Beiden Spieler möglich ist den Gegner Schachmatt zu setzten, wird das Spiel mit einem Remis (=Unentschieden/Gleichstand) beendet.

3.2 Anfangsstellung der Figuren

Das Schachbrett besteht aus einem 8x8 Gitter mit 64 gleichgroßen Quadraten, welche abwechselnd hell und dunkel sind (meist weiß und schwarz). Die Spieler positionieren sich gegenüber voneinander, sodass bei beiden Spielern ein weißes Feld in der unteren rechten Ecke vorhanden ist. Zu Beginn des Spiels hat jeder Spieler 16 Figuren; Ein Spieler die die Hellen (weiß) und einer die Dunklen (schwarz). Beide Spieler starten mit 8 Bauern, 2 Türmen, 2 Springern, 2 Läufern, einer Dame und einem König. Die Figuren sind wie folgt anzuordnen:

Die senkrechten Spalten sind die „Linien“, die waagerechten Zeilen sind die „Reihen“ und die gradlinige Folge der Felder, die sich an den Ecken Berühren sind die „Diagonalen“.

Abbildung 1: Anfangsaufstellung der Schachfiguren im Standardschach

3.3 Gangart der Figuren

Allgemeingültige Regeln sind folgende:

- Eine Figur darf nicht auf das Feld einer verbündeten Figur ziehen
- Es ist nicht möglich eine Figur durch ein Feld hindurch zu ziehen, welches von einer anderen Figur besetzt ist (Ausnahme siehe 3.3.4.)
- Wenn eine Figur auf ein Feld zieht, welches von einer gegnerischen Figur besetzt ist, dann wird die Gegnerfigur geschlagen (=entfernt)
- Keine Figur darf einen Zug machen, welcher den eigenen König einem Schachgebot aussetzt oder ihn in einem Schachgebot stehen lässt.

3.3.1 Läufer

Abbildung 2: Der Läufer darf auf ein beliebiges Feld entlang der Diagonale ziehen, auf der er steht.

3.3.2 Turm

Abbildung 3: Der Turm darf auf ein beliebiges Feld entlang der Linie oder der Reihe ziehen, auf der er steht.

3.3.3 Dame

Abbildung 4: Die Dame darf auf ein beliebiges Feld entlang der Linie, Reihe oder Diagonalen ziehen, auf der die steht.

3.3.4 Springer

Abbildung 5: Der Springer darf auf eines der Felder ziehen, die seinem Standfeld am nächsten, aber nicht auf gleicher Linie, Reihe oder Diagonalen mit diesem liegen.

3.3.5 Bauer

3.3.6 König

Es gibt zwei verschiedene Arten den König zu ziehen:

1. Er zieht auf ein beliebiges angrenzendes Feld, das nicht von einer oder mehreren gegnerischen Figuren angegriffen wird. Von den gegnerischen Figuren gilt, dass sie ein Feld auch dann angreifen, wenn sie selbst nicht ziehen können. Oder
2. er «rochiert». Die «Rochade» ist ein Zug des Königs und eines gleichfarbigen Turmes auf der gleichen Reihe. Sie gilt als ein einziger Zug und wird folgendermaßen ausgeführt: Der König wird von seinem Ursprungsfeld um zwei Felder in Richtung des Turmes hin versetzt, dann wird dieser Turm auf das Feld gesetzt, das der König soeben überquert hat.
   a) Die Rochade ist regelwidrig:
      i. wenn der König bereits gezogen hat, oder

ii. mit einem Turm, der bereits gezogen hat.

b) Die Rochade ist vorübergehend verhindert,

i. wenn das Standfeld des Königs oder das Feld, das er überqueren muss, oder sein Zielfeld von einer oder mehreren gegnerischen Figuren angegriffen wird,

ii. wenn sich zwischen dem König und dem Turm, mit dem rochiert werden soll, irgendeine Figur befindet.

Ein König «steht im Schach», wenn er von einer oder mehreren gegnerischen Figuren angegriffen wird, auch wenn diese selbst nicht ziehen können.

3.4 Ende der Partie

Eine Partie endet entweder durch ein „Schachmatt“ oder durch ein „Remis“ (siehe 3.1.). Eine Partie kann auch durch eine Übereinstimmung beider Spieler mit einem Remis entschieden werden. Ein Ende Partie ist auch durch eine dreifach wiederholte identische Zugfolge mit einem Remis zu beenden. Falls 50 Züge ohne einen Bauernzug oder das schlagen einer Figur vergehen, ist dies ebenfalls ein Unentschieden.

4 Benutze Pakete

Im Folgenden finden sie die verwendeten Pakete, sowie die Vorteile, welche mich zu der Entscheidung geführt haben.

4.1 pygame

„pygame“ ist eine Ansammlung von Python-Modulen welche auf das programmieren von Videospielen ausgelegt sind. Es ist auf sämtlichen OS verfügbar und hat schon mehrere Millionen Benutzer.

4.1.1 Warum pygame?

„pygame“ ist aufgrund der großen Community sehr gut dokumentiert. Außerdem bietet es eine wundervolle SDL (Simple Direct Media) Bibliothek. Ein weiterer Grund für die Verwendung von pygame war, dass ich schon ein wenig Erfahrung mit dem Modul sammeln konnte und daher für mich die erste oberflächliche Benutzung erleichtert wurde.

4.2 Stockfish

Stockfish ist eine berühmte Schachengine, welche von vielen großen Schachanalysten und Schachwebsites benutzt wird.

4.2.1 Warum Stockfish?

Stockfish hat eine gute Python Unterstützung. Außerdem ließ es sich nach ein paar wenigen Komplikationen auch gut benutzen und implementieren.

5 Installation

Mein Skript ist kostenlos von der Seite github herutergeladen werden. Offensichtlicher Weise muss man um das Python-Skript auszuführen, python3 herunter laden. Um dieses dann zu starten sind die Pakete „Stockfish“, „pygame“ und "pygame-widgets"über pip herunter zuladen. Dies ist über die Konsole mit den Befehlen

```
pip install pygame
```

```
pip install stockfish
```

```
pip install pygame-widgets
```

zu erreichen.

6 Bedienung des Programmes

6.1 Wie spiele ich?

Um das Schachprogramm zu starten muss man die init.py datei in python starten. Anschließend öffnet sich ein Fenster, in welchem man zwischen verschiedenen Spielmodi wählen kann. Die beiden Modi die hier zur Auswahl stehen sind einmal die Standard-Schach-Variante und die Chess 960 Variante, bei der die Figuren in der hinteren Reihe zufällig angeordnet werden. Für jeden der beiden Modi lässt sich optional ein zweiter virtueller Spieler hinzufügen, falls man keinen realen Gegner bei sich hat. Nachdem man sich für einen Modus entschieden hat, lässt sich der Spielername und je nach Option die Computerschwierigkeit oder der Name des zweiten Spielers einstellen. Nun kann das Spiel durch den Knopf unterhalb der Einstellungen gestartet werden und das tatsächliche Spiel kann beginnen. Sobald man auf eine Figur klickt, öffnen sich auf dem Feld alle legalen Zugoptionen. Wenn man anschließend auf ein Feld klickt, welches die Figur betreten kann, bewegt sich die ausgewählte Figur auf das gewünschte Feld.

6.2 Overlays

6.2.1 Spielerinformationstabs

Im oberen Bereich des Spielfelds befindet sich zum einen eine Uhr, welche die aktuelle Spielzeit anzeigt. Links und rechts danaben befinden sich die Spielerinformationen. In diesen werden die geschlagenen Figuren sowie der Spielername angezeigt. Außerdem

wird das Spielerinformationsfeld von dem Spieler hervorgehoben, welcher daram ist, den nächsten Zug zu machen.

6.2.2 Spielhistorie

In dem rechten Teil des Fensters befindet sich die Spielhistorie. Diese zeigt die bereits gezogenen Spielzüge an.

6.2.3 Buttons

Oben im rechten Eck finden sich insgesammt drei Buttons mit folgenden Funktionen:

- "QuitButton
  - Der Quit-Button beendet sofort das Spiel und schließt das Fenster.
- "ResignButton
  - Beendet das Spiel mit einem Sieg für den anderen Spieler
- "TakebackButton
  - Macht den letzten Zug rückgängig
  - Funktioniert nur, wenn keine Figur im letzten zug geschlagen wurde, um große Patzer zu bestrafen.

7 Code

In dem folgenden Kapitel werde ich das Konzept meines Codes erläutern und die Entwicklung darstellen.

7.1 Klassenbeziehungen

Neben einzelnen unabhängigen Klassen, wie zum Beispiel „Board“„Button“oder„Entry-Box“ gibt es die Figurenklassen. Angefangen bei dem Parent aller folgenden Figuren, der "PiecesClass. Diese enthält Methoden, welche von allen anderen Figuren benötigt werden, wie zum Beispiel

- move()
- draw()
- animate()
- foresight()

Alle spezifischen Figuren, also Dame, König, Läufer, Springer, Turm und Bauern, erben von Pieces. Die einzelnen Figuren haben, neben der init()-Methode, nur die Methoden um die möglichen Züge auszugeben. Der Output der Methode wird dann an Pieces.move() weiter gegeben, welche zunächst die Felder hervorhebt, auf welche man gehen kann und anschließend die Events verarbeitet um zu bestimmen, ob man auf ein legales Feld geclickt hat.

7.2 Probleme und Problemlösungen

Während des Programierens bin ich auf viele Probleme gestoßen, die mich an meine Grenzen trieben, welche ich aber immer wieder überwinden konnte. Das wohl größte Problem war, dass viele Regeln Voraussetzungen mit sich bringen, die man beim normalen Spielen gar nicht aktiv bedenkt, welche aber beim implementieren eine Hürde dar stellten.

7.2.1 Der König, das Problemkind

Das wohl größte Problem war der König. Dieser muss nämlich abhängig von der Gegnerischen Position der Figuren in seinen Bewegungen eingeschränkt werden. Die Lösung der Felderbegrenzung, konnte durch einen simplen Filter gelöst werden. Das erste größe Problem kam nun aber mit dem Blocken eines Schachs. Wenn eine Figur zwischen König und der angreifenden Figur steht, dann ist das Schach aufgelöst. Nur was bedeutet nun dazwischen? Um dieses Problem zu lösen musste ich den König entlang der Angriffslinien "tracken". Als ich nach langer Zeit das Problem gelöst hatte kam nun leider ein weiteres Problem einher. Der König konnte sich selbst blockieren bzw. das Schach im nächsten Zug verhindern, was wie folgt aussah:

Um dieses Problem zu lösen hatte ich zunächst die Idee den König zu einer "transparenten Figurßu machen, sodass der Angriffszug auch durch diesen hindurch wirkt, doch ich sah immer mehr Probleme kommen, die wieder aufwand gemacht hätten zu lösen, wie zum Beispiel, dass man die Figur, die vor dem König steht und das Schach verhindert nicht wegbewegt werden darf. Dann kam mir aber die Idee, wie ich all diese Probleme auf einen Schlag lösen konnte.

7.2.2 Foresight

Wenn man es schaffen könnte, in die Zukunft nach dem nächsten Zug zu sehen und erkennen könnte, ob der Zug als Konequenz hat, dass der König im Schach steht, könnte man diesen Zug verbieten. Genau das war der Gedanke hinter dieser Lösung. Ich lasse, bevor ich die möglichen Züge einer Figur exportiere, die Liste durch einen Filter laufen, der oben Erwähntes prüft. Der Filter sieht folgendermaßen aus:

```
def filter_method_foresight(self, move):
        self.x, self.y = move #setting the position of the piece to the move,
            that you want to check
        ignoring_piece = None #currently no piece has to be ignored
```

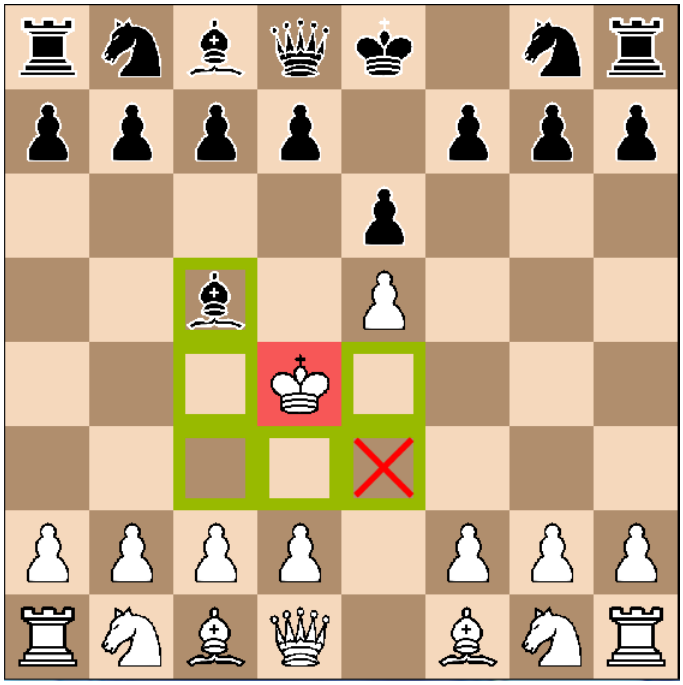

Abbildung 6: Bug, König verhindert Schach

```
### if you want to simulate to take a piece,
# without actually taking it, you can just
### ignore it in the .detectingCheck method
for piece in Pieces.all_pieces_list:
if (self.x, self.y) == (piece.x, piece.y) and not (piece == self):
       ignoring_piece = piece

#checking if the king is checked, after the move
Pieces.detectingCheck(ignoring_piece = ignoring_piece)

white_check = bool(Pieces.white_is_checked)
black_check = bool(Pieces.black_is_checked)
white_bool = bool(self.farbe == (255,255,255))
black_bool = bool(self.farbe == (0,0,0))

#returning if the move is legal or not
return (white_bool and not white_check) or (black_bool and not
    black_check)
```

Nachdem ich diesem Filter für jede Figur angewendet hatte, hatte ich keinerlei Probleme mehr mit illegalen Zügen.

7.2.3 Stockfish

Zunächst hatte ich noch an die Leinfachheit der Implementierung von Stockfish in python geglaubt, aber ich sollte falsch liegen. Zunächst hatte ich Stockfish wie in der Dokumentation beschrieben installiert und importiert. Als ich dann ein Objekt der Klasse „Stockfish“ erstellen wollte bekam ich folgende Fehlermeldung:

```
Traceback (most recent call last):
File "D:\Julian\Coding\Sprachen\python\lib\site-packages\
    stockfish\models.py", line 33, in __init__
self.stockfish = subprocess.Popen(
File "D:\Julian\Coding\Sprachen\python\lib\subprocess.py", line 947,
    in __init__
self._execute_child(args, executable, preexec_fn, close_fds,
File "D:\Julian\Coding\Sprachen\python\lib\subprocess.py", line 1416,
    in _execute_child
hp, ht, pid, tid = _winapi.CreateProcess(executable, args,
FileNotFoundError: [WinError 2] Das System kann die angegebene Datei
    nicht finden
Exception ignored in: <function Stockfish.__del__ at
    0x000001F7B592D820>
Traceback (most recent call last):
File "D:\Julian\Coding\Sprachen\python\lib\site-packages\
    stockfish\models.py", line 270, in __del__
self.stockfish.kill()
AttributeError: 'Stockfish' object has no attribute 'stockfish'
```

Als ich nach Lösungen gesucht hatte, war ich in einem Forum auf User gestoßen, die das selbe Problem hatten. Glücklicher Weise gab es die Möglichkeit eine .exe Datei herunterzuladen und den Bot von dort aus zu starten.

Ich fand die Lösung war nun nicht die Eleganteste, aber da das Programm für jeden auszuführen sein soll, wollte ich nichts bei mir lokal bei dem Paket „Stockfish“ ändern.

8 Danke an...

Ich möchte zunächst dem Luisengymnasium und dem Kollegium, insbesondere Frau Swiebodzinski, dafür danken, dass ich das Projekt in diesem Rahmen bearbeiten durfte. Ein besonderer Dank geht an zwei Mitschüler, Gero Beckmann und Vincent Piegsa, da ich mich bei Fragen immer die Möglichkeit hatte mich an sie zu wenden.

Außerdem möchte ich den Usern der Medien Youtube, Reddit, Stackoverflow und GitHub bedanken, da ich mir auch dort Hilfe holen konnte.

9 Schlusswort und Fazit

Dieses Projekt war sehr anspruchsvoll, da ich das Schachprogramm „from Scratch“ gestartet habe und auf jegliche Hilfen von API's verzichtet habe. Dennoch war dieses

Projekt sehr Umfangreich und spannend, da ich das erste Mal im Frontend entwickelt habe und ich, trotz vieler Hürden, meinen Spaß daran hatte.

Ich bin mit dem Endresultat sehr zufrieden, obwohl mir es bis Dato () nicht gelungen ist, die Regel „en passant “ zu implementieren.

9.1 Zukünftige Aussichten

Wie eben erwähnt habe ich vor die mir fehlende Regel einzubauen. Ebenfalls hatte ich vor den Code nocheinmal aufzuräumen und zu kommentieren, um ihn für Interessierte nachvollziebarer zu gestallten. Eine weitere Idee von mir war, dass ich eventuell das Programm auf meine eigene Website hochlade und somit ein Onlinespiel zu schaffen.