schach_MAIN2_2_grafik

- die nötigen Informationen für das Spiel werden im Dictionary „**einstellungen**“ gespeichert (Herausfinden der Informationen mit der settings() Funktion)
- dieses Dokument beinhaltet außer die folgenden Funktionen, alle nötigen Befehle, um die graphische Oberfläche mithilfe Tkinters zu erstellen

button_Funktion(y,x)
- wird ausgeführt, wenn ein Button(des Spielfelds) gedrückt wird
- erhält y und x Koordinate des gedrückten Button im Feld
- Bedeutungen der benutzten Variablen:
  - ya, xa : von wo aus Zieht die Figur? -> Ausgangskoordinaten
  - farbe : Welche Farbe ist am Zug?
  - status_ablauf:
    - besitzt immer eine von zwei möglichen Werten:
      - 1 = zu ziehende Figur ist noch nicht ausgewählt
      - 2 = Ziel der Figur muss noch festgelegt werden

- wenn **status_ablauf == 1**:
  - ausgewählte Koordinaten werden in ya,xa gespeichert
  - status_ablauf wird auf 2 gesetzt, damit der zweite Teil des Zuges beim nächsten Drücken des Buttons beginnen kann
  - Überprüfung: gehört das ausgewählte feld an der Stelle ya,xa der Person, die gerade an der reihe ist
    - wenn nicht: Wiederholung des Vorgangs unter status_ablauf == 1, da status_ablauf auf 1 geändert wird
- wenn **status_ablauf == 2**:
  - es wird ein zugarray4 erstellt, welches mit den Informationen für einen Zug (Ausgangskoordinaten, Endkoordinaten) gefüllt wird
  - zugarray4 wird an z.zug_grafik(c_feld,farbe,zugarray) übergeben
  - wenn zug_grafik zurückgibt, dass der Zug korrekt ablief, dann wird das neue_feld(welches zug_grafik zurückgibt) gespeichert ; wenn dies nicht der Fall ist wird der Vorgang unter 2 wiederholt
- **NACH JEDEM ZUG**:
  - wird überprüft, ob ein Bauer das Ende des Spielfelds erreicht hat und umgewandelt werden muss (bauernumwandlung1())
  - ein Bauer wird notfalls umgeändert (bauernumwandlung2())
  - wird die Farbe geändert
  - config() wird ausgeführt
  - Überprüfung: hat der Gegner nun verloren, also gibt die Funktion partie_verloren(feld,farbe) True zurück?
- **wenn die Anzahl der Spieler == 1**:
  - **Computerspieler** wird durch cpu_main() abgerufen
  - diese Funktion speichert den Zustand des Spiels nach einem Zug des Computers im feld Array
  - die Schritte unter „NACH JEDEM ZUG“ werden ausgeführt

config()
- ändert das Aussehen des graphischen Spielfelds auf die werte des jetzigen feld-Arrays („0“ wird vorher durch Leerzeichen ersetzt)

action_get_info_dialog()
- zeigt Grundinformationen zum Spiel an

schach_SETTINGS_SONSTIGES

settings()
- gibt ein Dictionary zurück, in dem der Namen der Spieler, die Anzahl der Spieler und die Schwierigkeit des Spiels gespeichert sind

spielername()
- Fragt die Namen und die Anzahl der Spieler über die Konsole ab und gibt diese Informationen zurück

schwierigkeit_festlegen()
- Fragt die Schwierigkeit des Spiels über die Konsole ab und gibt diese zurück

bauernumwandlung_1(feld,Anzahl_Spieler)
- schaut ob ein Bauer im feld die letzte Zeile erreicht hat
  - wenn dieser Bauer dem Computer gehört, wird dieser in eine Dame geändert
  - wenn nicht wird schritt2 = True zurückgegeben, damit bauernumwandlung_2 im nächsten Schritt ausgeführt wird
  - gibt die Farbe des gefundenen Bauern b_farbe zurück
- gibt das aktuelle Spielfeld zurück

bauernumwandlung_2(feld,figur_wahl)
- wandelt den Bauern in die vom Spieler ausgewählte Figur um
- gibt das aktuelle Spielfeld zurück

partie_verloren(feld,farbe)
- schaut ob die ausgewählte farbe auf dem feld noch Figuren besitzt (ist die Länge des Arrays von cpu.alle_eigenen_figuren() == 0?) und ob der Spieler dieser farbe einen Zug machen kann (ist die Länge des Arrays von cpu.alle_moeglichen_zuge() == 0?)
- gibt True zurück (farbe hat das Spiel verloren), wenn eines dieser Eigenschaften nicht zutrifft

schach_ZUG

zug_grafik(feld,farbe,zugarray4)
- ist für die Durchführung eines Zuges zuständig (Zug ist im zugarray4 gespeichert: Form [ybegin,xbegin,yend,xend])
- führt die Funktion zugpruefung() aus und gibt den zurückgegebenen Wert zurück
- führt die Funktion zugdurchfuehren() aus und gibt das zurückgegebene feld zurück

zug(feld,farbe)
- wird in der jetzigen Version des Spiels nicht mehr benötigt

zugeingabe()
- wird in der jetzigen Version des Spiels nicht mehr benötigt

zugpruefung(feld,zugarray,farbe)
- überprüft ob der auszuführende Zug den Spielregeln entspricht; schaut ob die Zielkoordinaten im Array welches moeglichezuege() zurückgibt enthalten ist
- gibt True zurück, wenn dies der Fall ist

zugdurchfuehren(feld,zugarray)
- führt den Zug aus dem zugarray durch
- wandelt das Zielfeld in die Figur auf dem Ursprungsfeld um und ändert den Wert des Ursprungsfeld auf „0“
- gibt das neue Spielfeld zurück

zug_syntaxpruefung(zugarray2)
- wird in der jetzigen Version des Spiels nicht mehr benötigt

zuguebersetzung(zugarray2)
- wird in der jetzigen Version des Spiels nicht mehr benötigt

schach_zugMOEGLICHKEITEN

moeglichezuege(y,x,feld,farbe)
- sucht mithilfe der Schachregeln zu der Figur mit den Koordinaten y,x alle möglichen Zielkoordinaten
- gibt diese Koordinaten in Form des Arrays „m“ zurück (m besitzt die Form [y1,x1,y2,x2,y3,x3….yn,xn])

genugabstandkönige(y,x,farbe,feld)
- gibt True zurück, wenn die Könige einen ausreichenden Abstand (1) zueinander hätten, falls der König der übergebenen Farbe auf das Feld y,x gesetzt wird
- funktioniert aufgrund eines nicht bekannten Fehlers nicht mit dem Computergegner… daher gibt diese Funktion in der aktuellen Version immer True zurück

alle_ziele(feld,farbe)
- wird in der jetzigen Version des Spiels nicht mehr benötigt

schach_CPU

cpu_main(feld,farbe,schwierigkeit)
- erhält aus der Funktion alle_moeglichen_zuege() ein Array mit allen möglichen Zügen und deren Bewertungen
- ermittelt die am höchsten bewerteten Züge und sucht zufällig einen davon aus
- führt diesen Zug aus, indem die Werte des Feldes geändert werden und gibt das neue Feld zurück

alle_moeglichen_zuege(feld,farbe,schwierigkeit)
- erhält die Standorte aller eigenen Figuren von der Funktion alle_eigenen_figuren()
- erzeugt für jede Figur ein Array „mz“ mit allen möglichen Zielen dieser Figur (benutzt dazu die moeglichezuege())
- erhält nun für jeden dieser Züge eine Bewertungsarray „zb“ von der Funktion zug_bewertung_main() (Enthält Ausgangspunkt, Endpunkt, Typ der Figur im Ausgangspunkt, Typ der Figur im Endpunkt und Bewertung des Zuges)
- fasst die Informationen aus dem Array „zb“ im Array „amz_array“ zusammen und gibt dieses zurück (Form: [y_eigeneFigur1,x_eigeneFigur1,y_ziel1,x_ziel1,typ_eigeneFigur1,typ_ziel1,bewertung1,y_eigeneFigur2…])

zug_bewertung_main(ya,xa,ye,xe,feld,schwierigkeit)
- fügt bei der Schwierigkeit „leicht“ für jeden Zug die Bewertung 0 hinzu
- fügt bei der Schwierigkeit „normal“ für jeden Zug eine individuelle Bewertung hinzu, welche die Funktion zug_bewertung_entscheider() erstellt
- Schwierigkeit „schwer“ ist noch nicht fertig
- gibt die Eingegangenen Daten und die Bewertung in einem Array zurück

zug_bewertung_entscheider(ya,xa,ye,xe,feld)
- bewertet jeden Zug aufgrund vorgegebener Regeln individuell und gibt diese Bewertung als Integer zurück
- (dabei werden unter anderem die Informationen von feld_gedeckt() genutzt)

feld_gedeckt(ya,xa,y,x,feld,farbe)
- gibt Informationen darüber zurück, wie das Feld y,x gedeckt wird (die Figur auf der Position xa,ya wird dabei gelöscht)
- Diese Informationen sind:
  - status: gibt True zurück, wenn das Feld von einer Figur gedeckt wird
  - anzahl: gibt die Anzahl der Figuren an, die das Feld decken
  - art: gibt an welcher Typ von Figuren das Feld decken

alle_eigenen_figuren(feld,farbe)
- gibt zwei Arrays zurück
  - 1.: aef_array : enthält alle Positionen der Figuren mit der gegebenen Farbe
  - 2.:aef_typ_array: enthält alle Positionen und Informationen über den Typ der gegebenen Farbe