### *Veštačka Inteligencija Izveštaj I faze projekta*

# - Projekat Slaganje -

*Tim: Shallow Blue*

*Sara Prušević 19027*

*Mihajlo Bencun 18081*

*Milena Mijucić 18249*

**Opis problema**

Igra je Slaganje (Byte). To je strateška igra gomilanja figura unapred postavljenih na tabli. Dva igrača (crni i beli) naizmenično odigravaju po jedan potez. Figure se nalaze na crnim poljima table i kreću se samo dijagonalno za jedno polje.

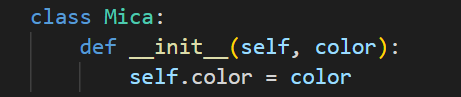
Pobednik je igrač koji složi više stekova od 8 figura na čijem je vrhu figura njegove boje.

Redosled boja figura na steku može biti proizvoljan. Kada se napravi stek od 8 figura on se uklanja sa table. Ako igrač pomera neku figuru koja je u steku, onda se sa njom pomeraju i sve figure koje se nalaze na njoj i postavljaju na poslednju figuru na steku susednog polja.

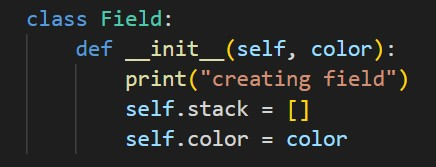
**state.py**

U fajlu **state.py** pisan je back-end deo aplikacije koji obezbeđuje logiku za rad i klase koje služe da predstave trenutno stanje igre.

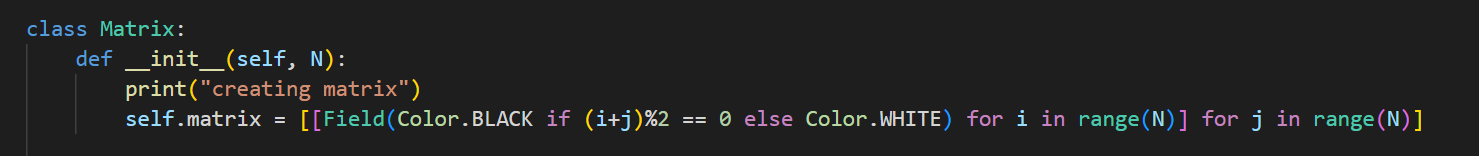
* Klasa *Mica* predstavlja objekat koji se pomera po poljima (i tako prazni ili puni stekove koje svako polje ima), sadrže boju tipa enumeracija (*Color*).



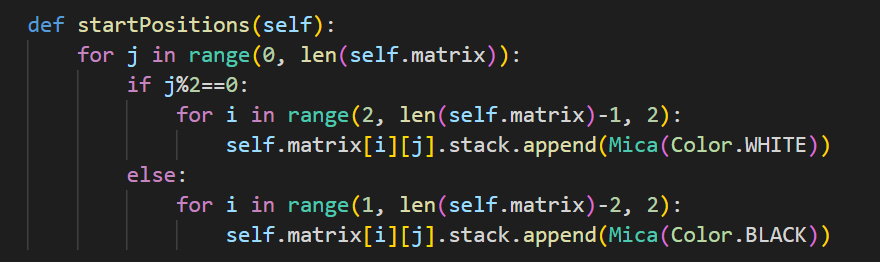
* Polja su klase *Field* i pored steka (implementiranog kao lista), sadrže još i boju tipa enumeracija (*Color*).



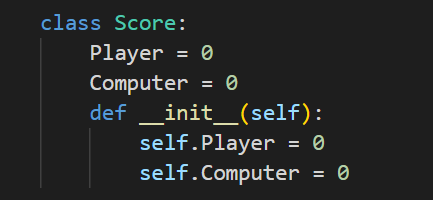
* Klasa *Matrix* pri svojoj inicijalizaciji kreira tablu sa crnim i belim poljima, uzimajući u obzir dimenziju koju je korisnik prethodno odabrao.



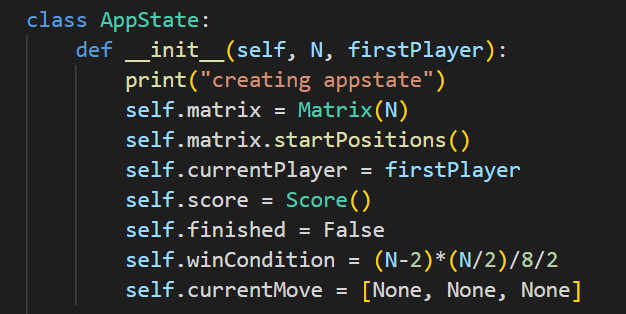
* Ukoliko je početak igre u pitanju, funkcija *startPositions* koja pripada klasi *Matrix* će postaviti mice u inicijalni položaj koji se računa na osnovu dimenzija matrice.



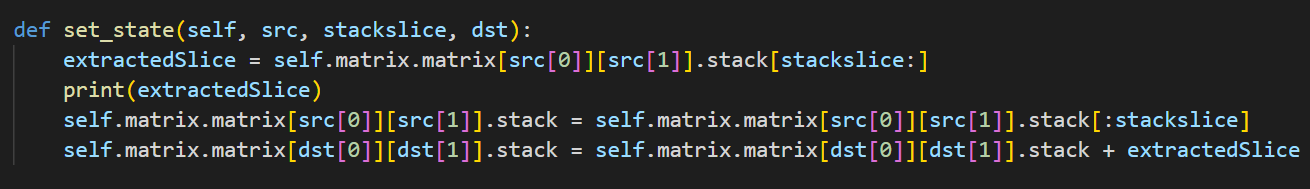
* Klasa *Score* služi da pamti broj osvojenih stekova po oba učesnika, tj. u ovom slučaju za korisnika (Player) i za računar (Computer), predstavljenih enumeracijom (*Player*).



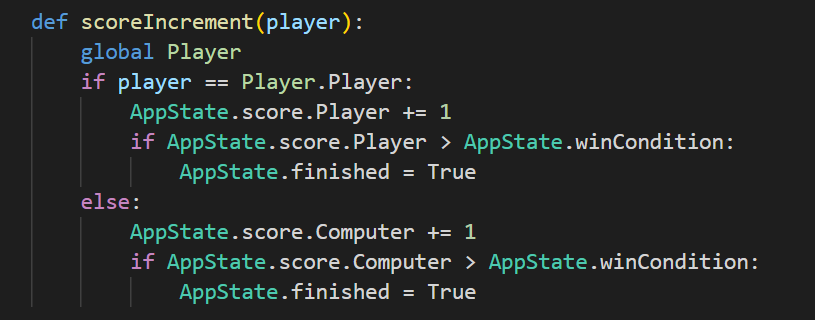
* Glavna klasa jeste *AppState* i ona pri svojoj inicijalizaciji kreira i matricu polja, raspoređuje mice po početnim pozicijama, postavlja odabir korisnika koji će igrač prvi igrati, inicijalizuje skor, #winCondition#, kreira currentMove za trenutni potez.



* Klasa *AppState* ima funkciju *set\_state*. Njoj kao parametre prosleđujemo pozicije polja u matrici sa kog uzimamo micu/e, indeks mice u steku na tom polju, i pozicije polja u matrici na koje prenosimo micu/e), ona premesti deo steka izvornog polja koji počinje od odabrane mice na stek koji pripada ciljnom polju.



* Funkcija *scoreIncrement* (u set\_state uslov len(stek)==8, ako da zovemo ovu fju za igraca čija je mica na vrhu steka i skidamo stek sa tog polja, mora da odredimo koja boja za koga ide) inkrementira broj osvojenih stekova za prosleđenog igrača i uz pomoć #winCondition# proverava da li se stiglo do kraja igre (na osnovu dimenzija, u igri može biti određen br. stekova, pa ako je neki igrač osvojio više od pola, pobedio je).



Funkcija get\_state? Funkcija setStateImmutable? Funkcija add?

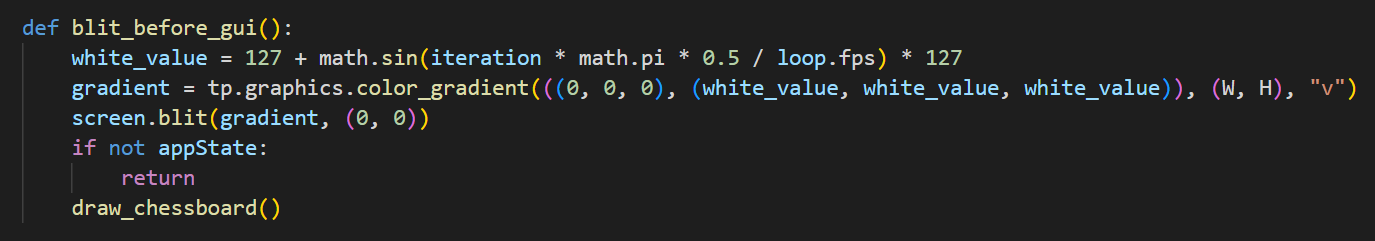
**drawing.py**

U **drawing.py** je odrađen interfejs, prikazana je šahovska tabla sa micama naređanim jedna na drugu za svako polje u kome se nalaze. Igrač ima mogućnost da izabere dimenzije matrice od ponuđenih, i da izabere ko će igrati prvi (player ili computer).

Postojeće funkcije:

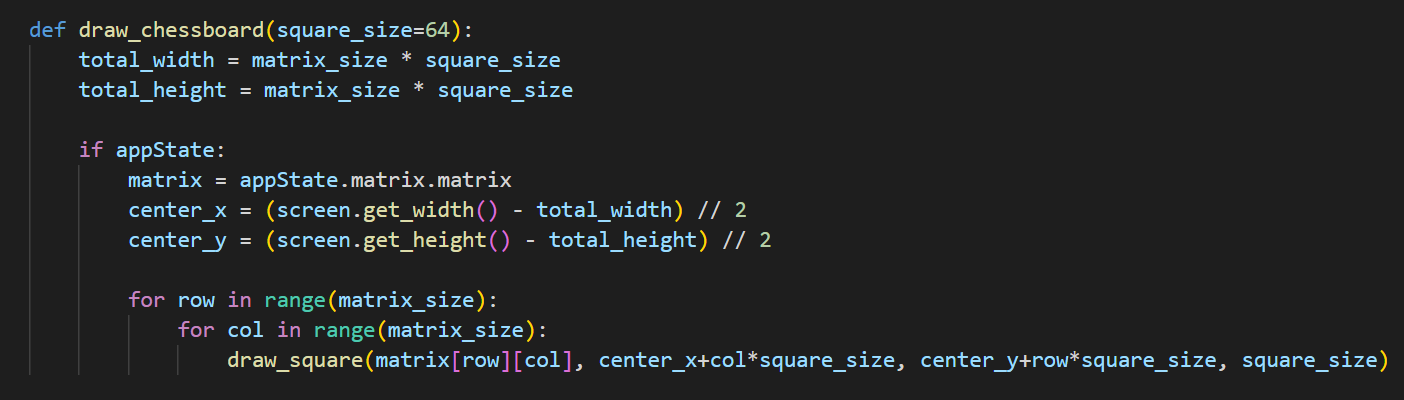
* **blit\_before\_gui()**

Ova funkcija crta gradijent na ekranu, i ukoliko promenljiva *appState* nije None, iscrtava i šahovsku tablu.



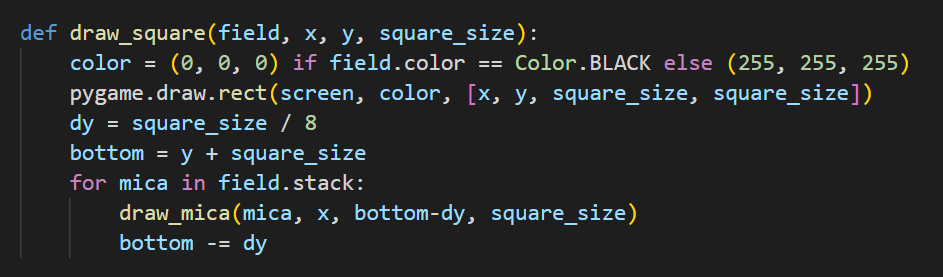
* **draw\_chessboard(square\_size=64)**

Crta kompletnu šahovsku tablu po sredini ekrana, zajedno sa micama koje se po njoj nalaze u trenutnom stanju.



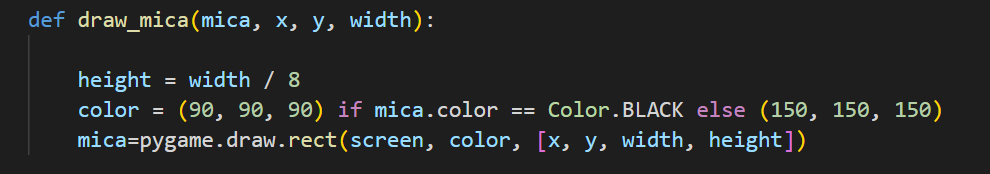
* **draw\_square(field, x, y, square\_size)**

Ukoliko pozicija sa x i y koordinatama u matrici ima crnu boju, ova funkcija to polje predstavlja crnim pravougaonikom, isto važi i za bela polja. U sebi poziva funkciju za crtanje mica, pa tako pruža celokupni prikaz trenutnog stanja šahovske table.



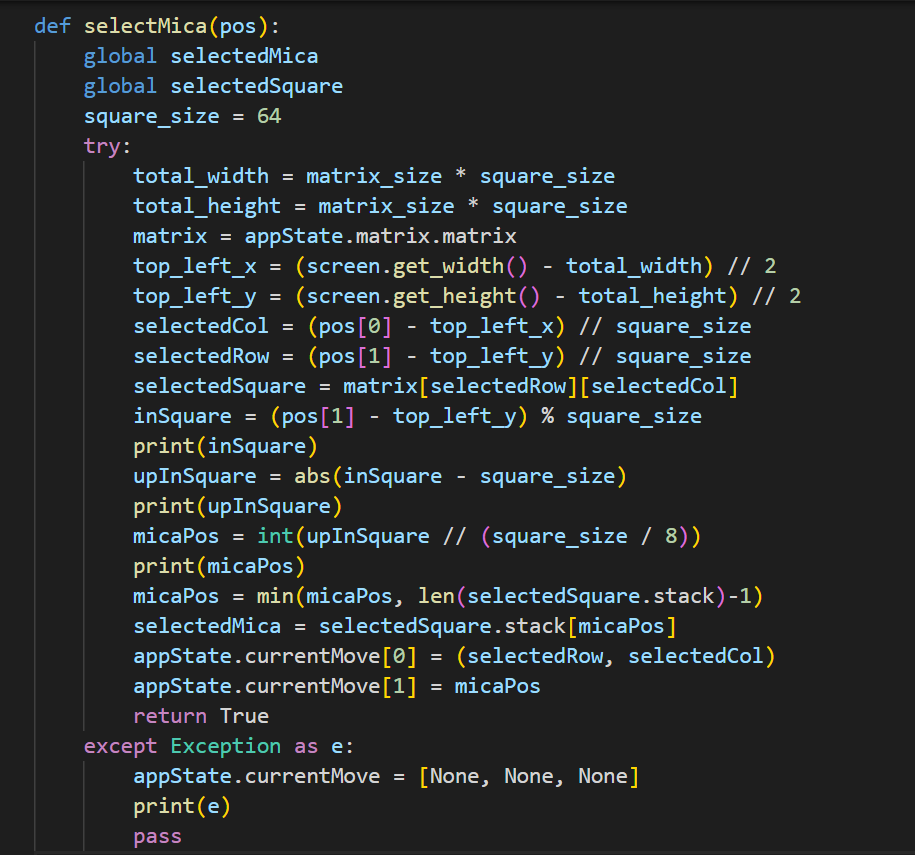
* **draw\_mica(mica, x, y, width)**

Iscrtava mice koje pripadaju nekom polju tako što ih predstavlja kao pravougaonike.



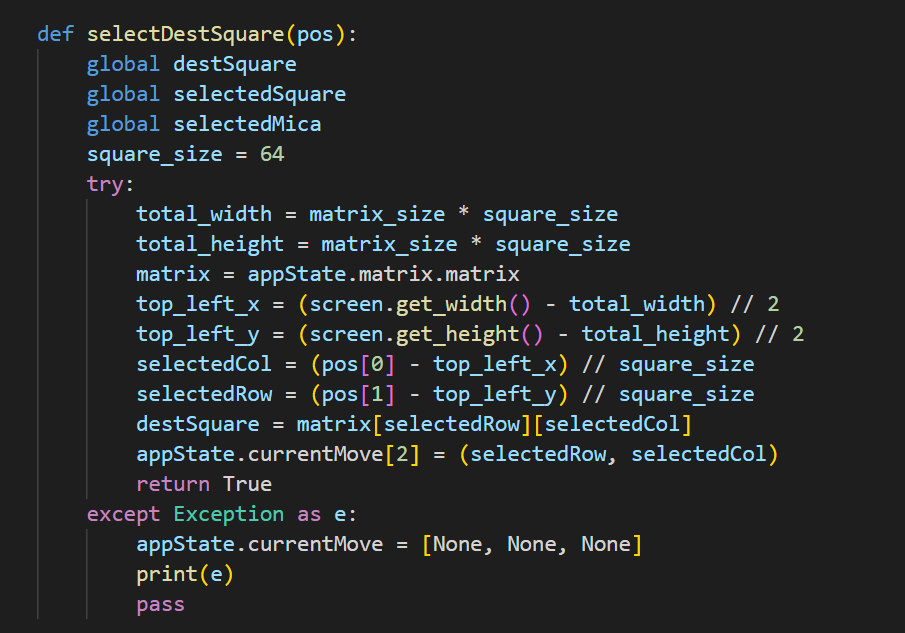
* **selectMica(pos)**

Funkcija selectMica kao parametar *pos* uzima poziciju kursora i precizno nalazi tačnu poziciju mice u nekom polju, odnosno govori nam o položaju mice u steku mica za to polje. U *currentMove* kao prva dva člana liste od ukupno tri, smešta indekse polja sa kog uzimamo micu (ili mice ako nije na vrhu), i indeks mice u steku tog polja.



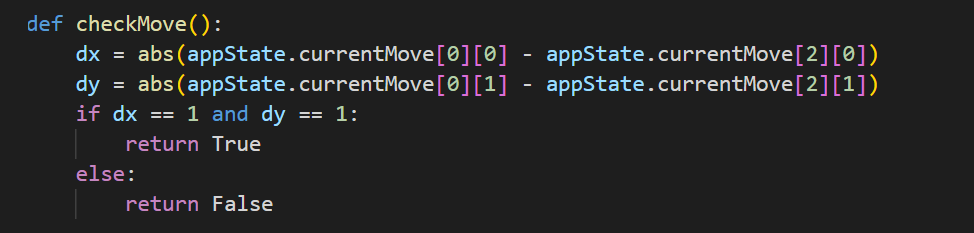
* **selectDestSquare(pos)**

Kao parametar *pos* uzima poziciju kursora i kao treći član *currentMove*-a postavlja polje na koje želimo da premestimo micu/e.



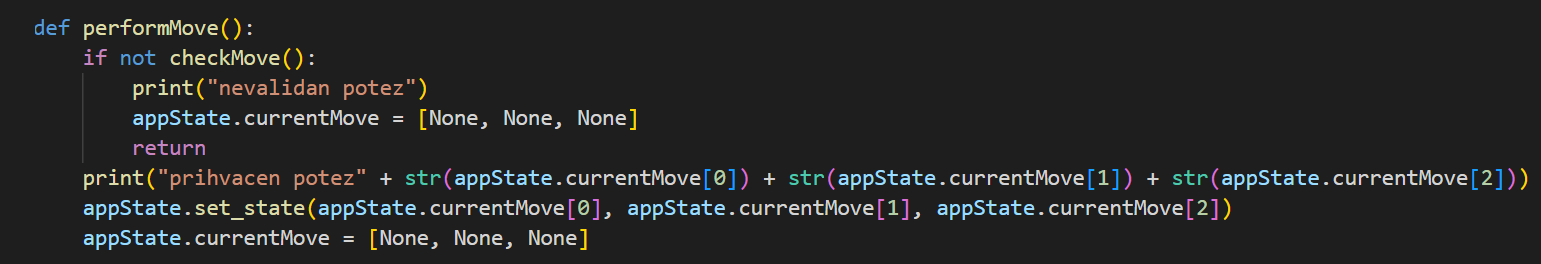
* **checkMove()**

Proverava da li je potez ispravan, tj. da li je ciljno polje dijagonalno i odmah pored početnog polja.



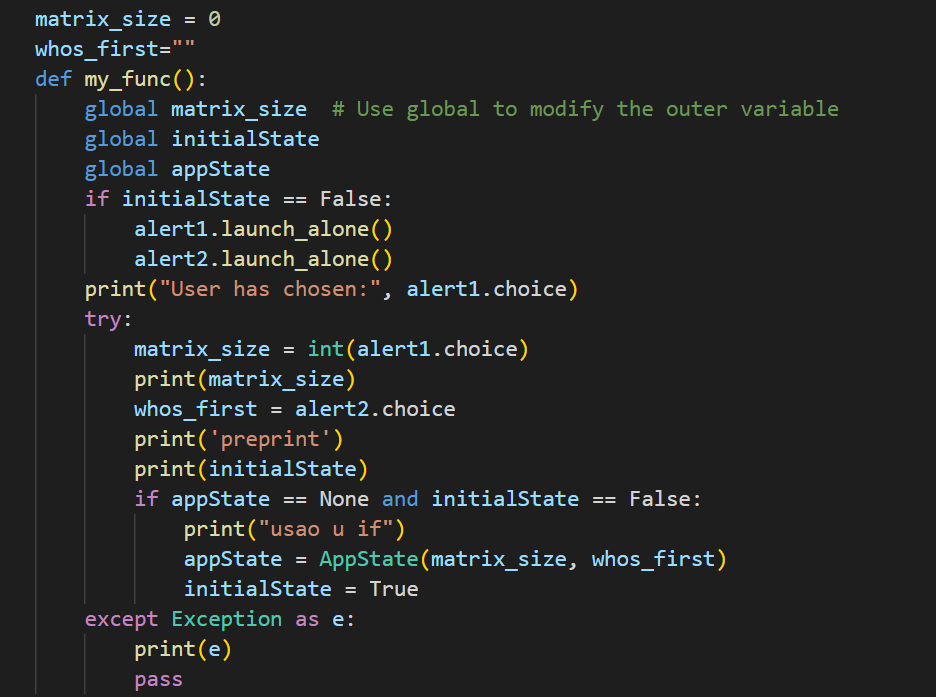
* **performMove()**

Ako je potez ispravan, poziva se funkcija set\_state i deo steka koji počinje od odabrane mice se premešta sa početnog na ciljno polje.



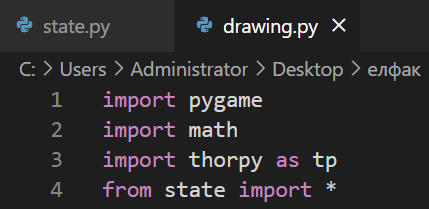
* **my\_func()**

Ukoliko *initialState* nije True, odnosno ako nije instancirana klasa AppState, pitaćemo korisnika da unese dimenziju šahovske table i odabere ko će prvi igrati. Na osnovu toga ćemo kreirati *appState.*



**Komunikacija**

U **drawing.py** je importovan modul **state.py** i omogućeno nam je korišćenje “deljivih” promenljivih, funkcija i klasa.



**Reference**

**\*\*\* Materijal sa Web-a i iz knjiga i radova možete da koristite, ali samo pod uslovom da za sve delove koda ili rešenja koje ste preuzeli navedete referencu!**