DOKUMENTACIJA ZA TRECU FAZU PROJEKTA

```
def minimax_alpha_beta(tablica, vrste, kolone,dubina, iks, alpha=((None, None), -10), beta=((None, None), 10)):
    if iks:
        return max_value(tablica,vrste, kolone, dubina, alpha, beta)
    else:
        return min_value(tablica,vrste, kolone, dubina, alpha, beta)
```

Funkcija minimax_alpha_beta se koristi za odabir min ili max algoritma sa alfa-beta odsecanjem.

X igrac je max igrac, dok je O igrac min igrac.

```
def max_value(tablica,vrste, kolone, dubina, alpha, beta, potez=None):
    if proveriDaLiJeKraj(True, vrste, kolone, tablica) == 10:
        return(potez, -10)
    listaPoteza=formirajMogucaStanja(True , vrste, kolone, tablica)
    if listaPoteza is None or dubina==0 or len(listaPoteza)==0:
        return (potez, heuristika(tablica, vrste, kolone))
    else:
        for p in listaPoteza:
            p1=(int(p[0][0]), p[0][1])
            p2=(int(p[1][0]), p[1][1])
            alpha=max(alpha, min_value(upisiPotezMinmax(True, tablica, p1, p2), vrste, kolone, dubina - 1, alpha, beta, p if potez is None else potez), key=lambda x: x[1])
        if alpha[1] >= beta[1]:
            return beta
    return alpha
```

Funkcija max_value je max algoritam sa alfa-beta odsecanjem koja za zadatu igru na osnovu stanja problema, dubine pretrazivanja i heuristike odredjuje najbolji moguci potez.

```
def min_value(tablica, vrste, kolone,dubina, alpha, beta, potez=None):
    if proveriDaLiJeKraj(False, vrste, kolone, tablica) == 10:
        return(potez, 10)
    listaPoteza=formirajMogucaStanja(False, vrste,kolone, tablica)
    if listaPoteza is None or dubina==0 or len(listaPoteza)==0:
        return (potez, heuristika(tablica, vrste,kolone))
    else:
        for p in listaPoteza:
            p1=(int(p[0][0]), p[0][1])
            p2=(int(p[1][0]), p[1][1])
            beta=min(beta, max_value(upisiPotezMinmax(False, tablica,p1, p2), vrste, kolone, dubina - 1, alpha, beta, p if potez is None else potez), key=lambda x: x[1])

        if beta[1] <= alpha[1]:
            return alpha
    return beta
```

Funkcija min_value je min algoritam sa alfa-beta odsecanjem koja za zadatu igru na osnovu stanja problema, dubine pretrazivanja i heuristike odredjuje najbolji moguci potez.

```
def heuristika(tablica, vrste, kolone):
        pomTabla=copy.deepcopy(tablica)
        brXPoteza=len(formirajMogucaStanja(True,vrste, kolone,pomTabla))
        brOPoteza=len(formirajMogucaStanja(False,vrste, kolone,pomTabla))
        return brXPoteza - brOPoteza
```

Funkcija heuristika vrsi procenu stanja kada se dostigne odredjena dubina trazenja.

```
def vrati_potez_racunara(tabla, vrste, kolone, iks):
        tablica= copy.deepcopy(tabla)
        rezultat = minimax_alpha_beta(tablica,vrste, kolone, 2, iks, alpha=((None,None), -10), beta=((None,None), 10))
        if(rezultat[0][0] == None):
            rezultat = minimax_alpha_beta(tablica,vrste, kolone, 1, iks, alpha=((None,None), -10), beta=((None,None), 10))

        return rezultat
```

Funkcija vrati_potez_racunara sluzi za poziv minimax_alpha_beta funkcije i vraca najbolji potez.

```
def main():
    iks=True
    (vrste, kolone)=unesiBrojVrstaIKolona()
    matrica=kreirajMatricu(vrste, kolone)
    prviIgrac=izaberiKoIgraPrvi()
    prikaziTrenutnoStanje(matrica, kolone, vrste)
    while True:
        if prviIgrac:
            if iks:
                unesiPotez(vrste,kolone,matrica,iks)
            else:
                igraPotez=vrati_potez_racunara(matrica, vrste, kolone,iks)
                igraPotez1=(int(igraPotez[0][0][0]), igraPotez[0][0][1])
                igraPotez2=(int(igraPotez[0][1][0]), igraPotez[0][1][1])
                upisiPotez(iks, matrica, igraPotez1, igraPotez2)
            prikaziTrenutnoStanje(matrica, kolone, vrste)
            if proveriDaLiJeKraj(iks, vrste, kolone, matrica):
                break
            iks = not iks
        else:
            if iks:
                igraPotez=vrati_potez_racunara(matrica, vrste, kolone,iks)
                igraPotez1=(int(igraPotez[0][0][0]), igraPotez[0][0][1])
                igraPotez2=(int(igraPotez[0][1][0]), igraPotez[0][1][1])
                upisiPotez(iks, matrica, igraPotez1, igraPotez2)
            else:
                unesiPotez(vrste,kolone,matrica,iks)
            prikaziTrenutnoStanje(matrica, kolone, vrste)
            if proveriDaLiJeKraj(iks, vrste, kolone, matrica):
                break
            iks = not iks
```

Funkcija main omogucava igru izmedju coveka i racunara u zavisnosti od toga ko prvi igra.

Marija Trajkovic, 18020

Katarina Cvetkovic, 18038

Marija Cvetkovic, 18039