**SKRIPTNI JEZICI PROJEKT**

**CONNECT 4 IGRA**

**Sadržaj**

1.Uvod

1.1. Connect 4 pravila i opis igre

1.2. Python i zašto je najbolji izbor za kreiranje ovakvog tipa video igara

2. Glavni dio programa

2.2. Importi i definiranje veličina igraće ploče

3. Funckije

3.1. Funkcija za kreiranje virtualne igraće ploče

3.2. Funkcija za postavljanje diskova u polju unutar igraće ploče

3.3. Funckija koja provjerava valjanost lokacije

3.4. Funckija koja dobavlja novi slobodni red

3.5. Funckija koja iscrtava igraću ploču

3.6. Funckija koja prepoznaje pobjednički potez

3.7. Funkcija koja crta po igraćoj ploči

4. Literatura

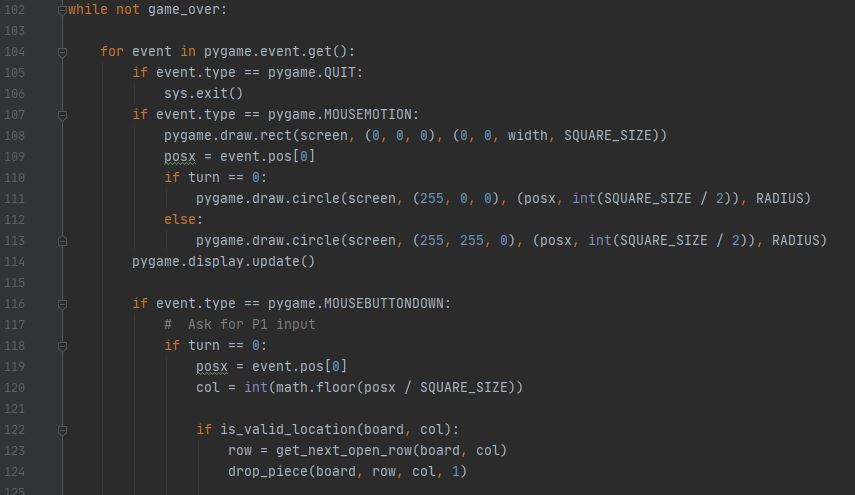
5. Popis slika i tablica

**1.UVOD**

U ovom projektu je poznata dječija igra Connect 4 prikazana i funkcionalno obrađena za igranje na računalu.

Sama igra je kompletno napisana u python programskom jeziku iz razloga što je python sam po sebi vrlo moćan program kojeg danas ne koriste samo programeri već i matematičari,analitičari podataka,znanstvenici itd

Python omogućava ogroman broj biblioteka i alata, 'friendly' sintaksu za što lakše kreiranje ovakve ,a i kompliciranijih video-igara/programa. Sam program se sastoji od 150 linija koda, 7 funkcija(funckije za kreiranje igraće tablice, printanje igraće tablice, postavljanje diskova, prelaska u novi red, funckija za određivanje pobjednika, crtanje igraće tablice) i 4 importa među kojima je pygame import koji ima najviše zasluga za kreiranje ove video igre.



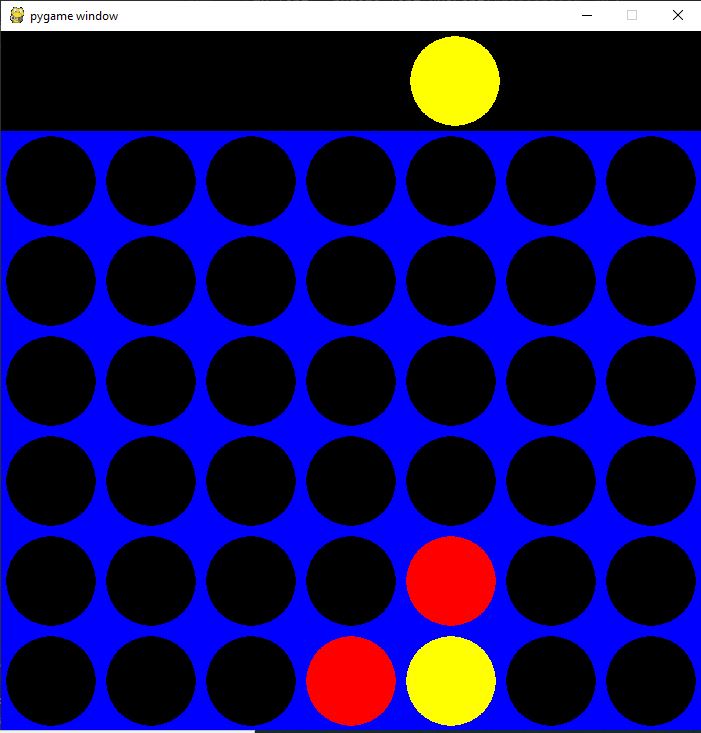
1.1. slika-Primjer python sintakse

**1.1. CONNECT 4 PRAVILA I OPIS IGRE**

Igra je namjenjena za minimalno dva igrača, pobjednik je onaj igrač koji prvi složi 4 polja iste boje u horizontalnom,vertikalnom ili kosom smjeru.

Pokretanjem programa prvi igrač ima pravo odabira jednog od sedam stupaca odnosno polja na kojem će postaviti svoju disk, odabir polja se vrši pomicanjem miša ulijevo ili udesno ,nakon što se disk pozicionira iznad željenog stupca igrač lijevim ili desnim klikom pokreće naredbu koja će postaviti njegov disk na dno tog stupca , nakon toga boja diska se automatski mjenja pa je na redu drugi igrač koji na isti način bira stupac u kojem će pozicionirat svoj disk; ako se drugi igrač odluči za polje na kojem je prvi igrač predhodno pozicionirao svoj disk, disk drugog igrača će se pozicionirati u istom stupcu, ali jedan red iznad reda u kojem se nalazi disk protivničkog igrača.

Programska traka na vrhu sadrži u krajnjem lijevom kutu natpis „pygame window“ i ikonu, a u krajnjem desnom kutu se nalazi ikona za izlazak iz programa i ikona za minimalizaciju prozora.



1.2. slika-Izgled Connect4 video igre

**1.2. PYTHON I ZAŠTO JE NAJBOLJI IZBOR ZA KREIRANJE OVAKVOG TIPA VIDEO IGARA**

Sama igra je kompletno napisana u python jeziku iz razloga što je python sam po sebi vrlo moćan program kojeg danas ne koriste samo programeri već i matematičari,analitičari podataka,znanstvenici itd.

Sa pythonom možemo rješavati kompleksne probleme u kraćem vremenu i sa mnogo manje linija koda za razliku od ostalih programskih jezika, cross-platforma je što znači da programe možemo koristiti na različitim platformama kao što su windows,mac,linux; ima velik eco-system što znači da posjeduje veliki broj knjižnica,datoteka,alata i jednostavnu „beginner-friendly“ sintaksu što sve ukupno omogućava lakši i brži rad, što na kraju rezultira sve veći utjecaj pythona na programerskoj sceni.

**2.GLAVNI(MAIN) DIO PROGRAMA**

Iz glavnog dijela pozivamo funckije koje su nam potrebne za noramalan rad programa, u prvoj liniji main koda imamo varijablu game\_over koju smo postavili na false što će se kasnije promjeniti u true ako neki od igrača uspije postaviti 4 diska iste boje u vertikalnom,horizontalnom ili dijagonalnom smjeru.

Varijablu board je izjednačena sa create\_board() funkcijom koja će kreirati tablicu(matricu) dimenzija 6x7 ispunjenu nulama.

Varijabla turn označava broj ukupnih poteza u jednom meču,inicijalizirana je na nula.

Naredba pygame.init() inicijalizira sve važne pygame module

Init()->(numpass,numfail)

Nijedan izuzetak neće biti podignut ako modul padne, ali ukupan broj uspiješnih i neuspješnih initova će se vratiti kao tuple(tuple je kolekcija koja je posložena i nepromjenjiva).

Uvijek možemo inicijalizirati pojedinačne module ručno, ali pygame.init() inicijalizacija svih važnih pygame modula je prikladan način da se sve pokrene.

U kodu na linijama od 90 do 95:

SQUARE\_SIZE = 100  
width = COLUMN\_COUNT \* SQUARE\_SIZE  
height = (ROW\_COUNT + 1) \* SQUARE\_SIZE  
size = (width, height)  
RADIUS = int(SQUARE\_SIZE / 2 - 5)

Postavljamo vrijednosti za kvadratnu veličinu koja će biti 100, širinu koja čini umnožak kvadratne veličine i broja stupaca, visinu koja čini umnožak broja redova plus 1(broj redova+1) i kvadratne veličine , radius koji dobivamo dijeljenjem kvadratne veličine sa dva i oduzimanjem tog kvocijenta sa pet.

Sljedeći dio koda u glavnom služi kao prikaz tablice odnosno matrice na ekranu :

screen = pygame.display.set\_mode(size)  
draw\_board(board)  
pygame.display.update()

Varijabli screen smo inicijalizirali prozor ili ekran za prikaz, veličine size(širina i visina) preko naredble pygame.display\_set\_mode(size). Kreirani prozor ili ekran će biti najbolje moguće podudaranje podržano od strane sustava.

Sljedeća naredba draw\_board poziva istoimenu funkciju sa parametrom board.

Ponovo koristimo pygame.display.update() koji će ažurirati prikaz na ekranu.

Sljedeći dio koda je više dizajnerskog tipa u kojem varijabli my\_font preko naredbe pygame.font.SysFont dodjeljujemo tip slova u prvom parametru, a u drugom veličinu slova.

My\_font će se kasnije koristit za ispis pobjednika na ekranu.

my\_font = pygame.font.SysFont("monospace", 75)

Nakon inicijalizacije nekih osnovnih dijelova koristimo while not petlju koja će se „vrtjeti“ sve dok ne dobijemo pobjednika.

Unutar while not petlje imamo for petlju koja će prolaziti kroz strukturu podataka red preko naredbe pygame.event.get(). Laički rečeno provjeravat ćemo akcije igrača odnosno što su pritisnuli i gdje, a te njhove akcije će se pohranjivati u strukturu podataka red.

Unutar for petlje imamo if uvjete ,prvi if uvijet provjerava dali je igrač pritisnuo x na prozori što u kodu izgleda ovako:

if event.type == pygame.QUIT:

ako je ova usporedba istinita odnosno true, zatvarmo trenutni prozor preko naredbe sys.exit.

Sljedeća if petlja provjerava dali smo napravili neki pokret sa mišem:

if event.type == pygame.MOUSEMOTION:

ako je jednakost true, crta se pravokutnik sa naredbom:

pygame.draw.rect(screen, (0, 0, 0), (0, 0, width, SQUARE\_SIZE))

gdje screen označava površinu gdje će se nacrtati,drugi parametar označava boju,a posljednji parametar označava poziciju i dimenzije pravokutnika.

U varijablu posx pohranjujemo trenutnu poziciju miša odnosno onu poziciju u odnosu na x kordinatu os:

event.pos[0]

Y kordinatu nije potrebno pohranjivati.

Sljedeći if uvijet unutar trenutnog uvijeta provjerava dali je meč trenutno u prvom krugu:

if turn == 0:

Ako je ovaj uvijet istinit crta se disk crvene boje na poziciju koja je pohranjena u varijabli posx.

Zatim imamo else uvijet koji će nacrtati disk žute boje na poziciju posx ako ovo nije prvi krug.

Zatim se prozor osvježava naredbom pygame.display.update() kako bi korisnikova akcija bila vidljiva.

Sljedeći if provjerava dali je igrač pritisnuo lijevi ili desni klik miša, ako jeste provjeravamo dali je trenutni krug prvi:

if turn == 0:  
 posx = event.pos[0]  
 col = int(math.floor(posx / SQUARE\_SIZE))

ako je true, pohranjujemo trenutnu poziciju miša u varijablu posx, a u varijablu col se upisuje jednaki ili manji broj od broja kojeg dobijemo nakon dijeljenja posx sa SQUARE\_SIZE.

U sljedećoj liniji koda provjeravamo ispravnost lokacije pozivom funkcije is\_valid\_location kojoj dajemo argumente board i col.

if is\_valid\_location(board, col):  
 row = get\_next\_open\_row(board, col)  
 drop\_piece(board, row, col, 1)

Unutar ovog if-a u varijablu row pohranjujemo return iz funkcije get\_next\_open\_row sa argumentima board i col, ova funkcija će vratiti lokaciju prvog slobodnog reda u stupcu col.

Sljedeća pozvana funkcija je drop\_piece koja kao parametre ima board,row,col i 1(piece), ova funkcija će u polje kordinata row i col umjesto nula postaviti jedan što u praksi prestavlja popunjeno polje na koje dolazi disk.

if winning\_move(board, 1):  
 pygame.draw.rect(screen, (0, 0, 0), (0, 0, width, SQUARE\_SIZE))  
 label = my\_font.render("Player 1 WINS!!", 1, (255, 0, 0))  
 screen.blit(label, (40, 10))  
 game\_over = True

Sljedeči uvijet poziva funkciju winning\_move kojoj prosljeđujemo parametar board i 1(piece) ako je uvijet true crta se pravokutnik naredbom pygame.draw.rect.

Nakon toga varijablu label izjednačavamo sa my\_font.render koja će ispisati tekst „Player 1 WINS“ crvenim slovima.

Naredba screen.blit će prikazati vrijednost iz labela na zadanoj x,y lokaciji (40,10).

Varijablu game\_over izjednačavamo sa true jer je igra gotova i imamo pobjednika.

U else dijelu posx je jednaka poziciji miša u odnosu na x os, zatim varijabla col je jednaka manjem ili jednakom broju nakon dijeljenja posx sa SQUARE\_SIZE.

Unutar else dijela imamo pod if uvjete, prvi provjerava valjanost lokacije sa is\_valid\_location kao u predhodnom slučaju samo što će se ovaj put umjest jedan(1) kao argument u drop\_piece poslati 2 jer je igrao drugi igrač.

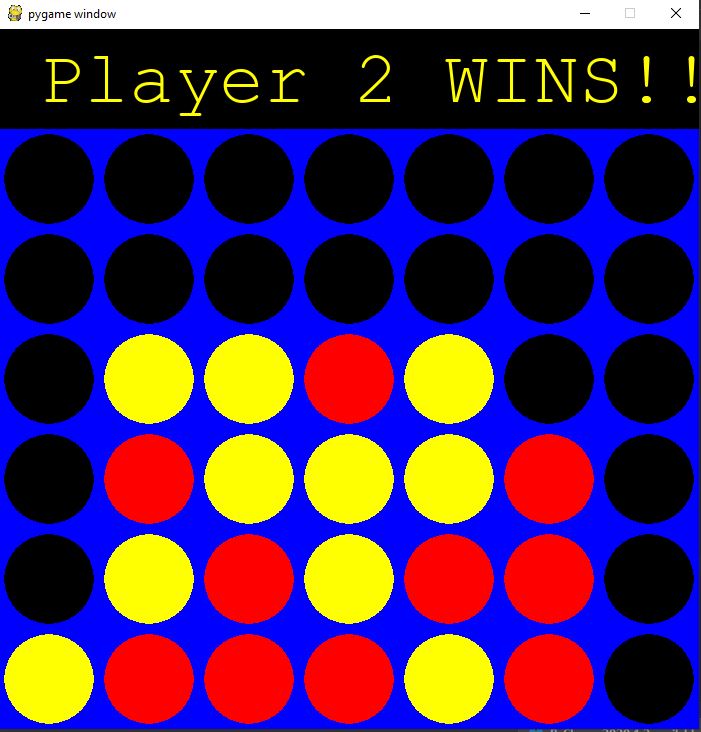
Nakon toga provjeravamo dali je to bio pobjednički potez preko funkcije winning\_move kojoj kao parametar dajemo board i 2( jer je igrao drugi igrač). U slučaju da je funckija winning\_move vratila true naredbom screen.blit se ispisuje „Player 2 WINS!!“ i varijabla game\_over se inicijalizira sa true čime je igra gotova.

Zadnjih šest linija koda:

draw\_board(board)  
turn += 1  
turn = turn % 2  
  
if game\_over:  
 pygame.time.wait(3000)

Kodom draw\_board kreiramo tablicu, krugove se povećavaju za jedan svaki put kada jedan od igrača odigra potez i uzima se ostatak dijeljenja sa 2.

Zadnji if provjerava dali je game\_over true ako jeste pokrece se naredba pygame.time.wait koja će pauzirati igru na određeni broj milisekunda u ovom slučaju 3000 milisekunde.



2.1. slika-Izgled nakon pobjedničkog poteza

**2.2. IMPORTI I DEFINIRANJE VELIČINE IGRAĆE TABLICE**

Program Connect4.py sastoji se od 151 linije koda, u prvih 4 linije koda nalaze se import naredbe. Import naredba služi za pristup informacijama, funckijama, metodama iz nekih klasa koje se nalaze u nekom drugom paketu(package) ,a želimo ih koristiti u trenutnom paketu(package).

Import math- omogućava nam korištenje raznih matematičkih funkcija

Import sys- pruža informacije o konstantima, funkcijama i metodama od strane pythonovog interpretera.

Import numpy as np- pruža visoko performanse multidimenzionalne mreže i alate za rad sa tim mrežama. U Connect 4 programu značaj ovog importa se najviše može vidjeti u kreiranju polja gdje se postavljaju okruglice.

Import pygame- otvoreni modul za Python programski jezik , koji služi za kreiranje video-igara i drugih multimedijalnih aplikacija.

Nakon import dijela programa dolazimo u dio u kojemu definiramo veličinu odnosno broj polja koji će biti dostupni za popunjavanje diskovima. Broj redova inicijaliziramo u varijabli ROW\_COUNT,čija vrijednost iznosi 6 ,a broj stupaca inicijaliziramo u varijabli COLUMN\_COUNT koja će iznositi 7.

**3. FUNKCIJE**

**3.1.FUNKCIJA ZA KREIRANJE VIRTUALNE IGRAĆE PLOČE**

Prva funkcija u programu je create\_board koja služi za kreiranje površine za igranje. Ova funkcija nema parametre,a kao rezultat vraća(return) tablicu. Unutar ove funkcije imamo naredbu:

\_board = np.zeros((ROW\_COUNT, COLUMN\_COUNT))

Koja će kreirati mrežu sa dimenzijama unutar zagrade ,čija će vrijednost svakog polja biti nula, u ovom slučaju dimenzije će biti 6x7.

**3.2. FUNKCIJA ZA POSTAVLJANJE DISKOVA U POLJU UNUTAR IGRAĆE PLOČE**

Sljedeća funckija je drop\_piece koja za parametre uzima tablicu, red, stupac i disk(piece). Naredba unutar ove funckije je:

board[\_row][\_col] = \_piece

koja će postaviti okruglicu u red i stupac koji su određeni sa vrijednostima unutar \_row(red) i \_col(stupac).

**3.3. FUNKCIJA KOJA PROVJERAVA VALJANOST LOKACIJE**

Sljedeća funkcija je is\_valid\_location koja provjerava i vraća dali je neko polje prazno odnosno na tom mjestu ne postoji disk ili je puno, odnosno na tom mjestu već postoji disk/ovi neke boje.

Kod unutar ove funkcije:

return \_board[ROW\_COUNT - 1][\_col] == 0

**3.4. FUNKCIJA KOJA DOBAVLJA NOVI SLOBODNI RED**

Funkcija get\_next\_open\_row, za parametre uzima tablicu i vrijednost stupca. Kod unutar ove funckije:

for r in range(ROW\_COUNT):  
 if \_board[r][\_col] == 0:  
 return r

U ovom dijelu koda prolazimo kroz sve retke u nekom stupcu pomoću for petlje koja unutar sebe ima uvijet if koji kaže da kada dođemo do prvog retka koji je prazan vraćamo r odnosno broj tog retka preko return naredbe.

ROW\_COUNT označava ukupan broj redova unutar tablice.

**3.5. FUNKCIJA KOJA ISCRTAVA IGRAĆU PLOČU**

Sljedeća funckija je print\_board, koja služi za obrnuti prikaz tablice na način da će se elementi iz prvog reda na ekranu prikazati u zadnjem redu,a elementi iz zadnjeg reda prikazati u prvom redu , bez ovog dijela koda igrači bi se teško orijentirali i izgled same tablice bi bio nerealan.

print(np.flip(board, 0))

Naredbla np.flip omogućava taj obrnuti prikaz tablice.

**3.6. FUNKCIJA KOJA PREPOZNAJE POBJEDNIČKI POTEZ**

Funckija winning\_move kao parametre koristi tablicu i diskove.

U ovoj funkciji provjeravamo dali su 4 polja ispunjena u horizontalnom,vertikalnom ili dijagonalnom smjeru.Prvo provjeravamo horiznotalno,kod za to je:

for c in range(COLUMN\_COUNT - 3):  
 for r in range(ROW\_COUNT):  
 if \_board[r][c] == \_piece and \_board[r][c + 1] == \_piece and \_board[r][c + 2] == \_piece and \  
 \_board[r][c + 3] == \_piece:  
 return True

COLUMN\_COUNT-3 je iz razloga što provjeravamo samo po 4 stupca npr. Stupci 1,2,3,4 pa nakon toga provjeravamo 2,3,4,5 itd.

Slovo r označava redove,a c stupce.

U if dijelu koda provjeravamo dali su u nekom redu r, stupci c,c+1,c+2,c+3 ispunjeni diskovima, ako uvijet vrijedi za sve, vraća se boolean true.

Nakon horizontalne,provjeravamo i vertikalnu mogućnost pobjede:

for c in range(COLUMN\_COUNT):  
 for r in range(ROW\_COUNT - 3):  
 if \_board[r][c] == \_piece and \_board[r + 1][c] == \_piece and \_board[r + 2][c] == \_piece and \  
 \_board[r + 3][c] == \_piece:  
 return True

ROW\_COUNT-3 je iz razloga što provjeravamo samo po 4 reda npr. Redovi 1,2,3,4 pa nakon toga provjeravamo 2,3,4,5 itd.

Slovo r označava redove,a c stupce.

U if dijelu koda provjeravamo dali su u nekom stupcu c, redovi r,r+1,r+2,r+3 ispunjeni diskovima, ako uvijet vrijedi za sve, vraća se boolean true.

Nakon vertikalne,provjeravamo dijagonalno:

for c in range(COLUMN\_COUNT - 3):  
 for r in range(ROW\_COUNT - 3):  
 if \_board[r][c] == \_piece and \_board[r + 1][c + 1] == \_piece and \_board[r + 2][c + 2] == \_piece and \  
 \_board[r + 3][c + 3] == \_piece:  
 return True

Preko ROW\_COUNT-3 i COLUMN\_COUNT-3 smanjujemo raspon provjere npr. Provjeravaju se stupci od 1 do 5 i redovi od 1 do 5 zatim stupci od 2 do 6 i redovi od 1 do 5 itd.

Slovo r označava redove,a c stupce.

U if dijelu koda dijagonalnu provjeri vršimo tako što ćemo prvoj provjeri za r i c dodati 1, drugoj provjeri 2, trecoj 3 i četvrtoj provjeri za r i c ćemo dodati 4 . Na ovaj način smo provjerili sva dijagonalna polja.

Ako imamo dijagonalna podudaranja funkcija vraća boolean True.

Nakon dijagonalne provjere provjeravamo negativnu dijagonalu:

for c in range(COLUMN\_COUNT - 3):  
 for r in range(3, ROW\_COUNT):  
 if \_board[r][c] == \_piece and \_board[r - 1][c + 1] == \_piece and \_board[r - 2][c + 2] == \_piece and \  
 \_board[r - 3][c + 3] == \_piece:  
 return True

Pod negativnom dijagonalom se misli na onu dijagonalu koja se smanjuje od desna ka lijevo(/).

for r in range(3, ROW\_COUNT):

Tri u ovom dijelu koda označava startnu poziciju koja je 3 ,a ROW\_COUNT označava stop odnosno kraj. Iz razloga što provjeravamo negativnu dijagonalu nema potrebe da provjeravamo redove 1 i 2 već odmah možemo krenuti od trećeg reda.

U if dijelu koda dijagonalnu provjeri vršimo tako što ćemo prvoj provjeri za r i c dodati 1, drugoj provjeri 2, trecoj 3 i četvrtoj provjeri za r i c ćemo dodati 4 . Na ovaj način smo provjerili sva dijagonalna polja.

Ako svi uvijet podudaraju odnosno ako su diskobi postavljeni dijagonalno ,funkcija vraća boolean True.

**3.7. FUNKCIJA KOJA CRTA PO IGRAĆOJ PLOČI**

Funkcija draw\_board ima samo jedan parametar ,a to je \_board(tablica 6x7 ispunjena nulama).

Prvih šest linija koda draw\_board funckije:

for c in range(COLUMN\_COUNT):  
 for r in range(ROW\_COUNT):  
 pygame.draw.rect(screen, (0, 0, 255),  
 (c \* SQUARE\_SIZE, r \* SQUARE\_SIZE + SQUARE\_SIZE, SQUARE\_SIZE, SQUARE\_SIZE))  
 pygame.draw.circle(screen, (0, 0, 0), (int(c \* SQUARE\_SIZE + SQUARE\_SIZE / 2),  
 int(r \* SQUARE\_SIZE + SQUARE\_SIZE + SQUARE\_SIZE / 2)), RADIUS)

Prolazimo kroz sva polja tablice odnosno matrice 6x7 i upisujemo kvadrate i krugove. Pygame.draw.rect je naredba koja služi za crtanje pravokutnika, prvi parametar označava površinu na kojoj će se ispisati pravokutnik, u ovom slučaju će to biti ekran , drugi parametar označava boju koja će biti plava ,a u trećem parametru određujemo veličinu tog pravokutnika.

Pygame.draw.circle je naredba koja će unutar tih pravokutnika nacrtati krugove crne boje. Prvi parametar označava površinu na kojoj će se ispisati krug, u ovom slučaju će to biti ekran , drugi parametar je boja, crna, u trećem parametru označavamo veličinu kruga ,a u četvrtom parametru ćemo odrediti debljinu, koja je u ovom slučaju debljina radiusa.

Nakon kreiranja igraće površine , kreiramo okruglice:

for c in range(COLUMN\_COUNT):  
 for r in range(ROW\_COUNT):  
 if \_board[r][c] == 1:  
 pygame.draw.circle(screen, (255, 0, 0), (int(c \* SQUARE\_SIZE + SQUARE\_SIZE / 2),  
 height - int(r \* SQUARE\_SIZE + SQUARE\_SIZE / 2)),  
 RADIUS)  
 elif \_board[r][c] == 2:  
 pygame.draw.circle(screen, (255, 255, 0), (int(c \* SQUARE\_SIZE + SQUARE\_SIZE / 2),  
 height - int(  
 r \* SQUARE\_SIZE + SQUARE\_SIZE / 2)),  
 RADIUS)  
 pygame.display.update()

Prvim if provjeravamo dali je vrijednost nekog polja [r][c] jednaka 1 ili 2, u slučaju da je jednaka jedan ,naredbom pygame.draw.circle postavljamo krug koji je vidljiv na ekranu, crvene boje i nekih dimenzija i debljine.

U slučaju da je vrijednost polja[r][c] jednaka 2, postavlja se okruglica koja je vidljiva na ekranu ,žute boje i nekih dimenzija i debljine.

Pygame.display.update() je naredba koja ažurira prikaz na ekranu, tako kada igrač odigra svoj potez,njegov odabir će se prikazati na ekranu u obliku kruga u nekoj od dvije ponuđene boje.

**4. LITERATURA**

<https://www.w3schools.com/python/>

<https://www.pygame.org/>

<https://www.freecodecamp.org/>

**5. POPIS SLIKA I TABLICA**

Slika 1. Primjer Python sintakse-stranica 3

Slika 2. Izgled Connect 4 igre-stranica 5

Slika 3. Izgled prozora nakon pobjedničkog poteza-stranica 11