Društvena igra Santorini

Opis

Problem je rešen u programskom jeziku Java, u okruženju Android Studio.

Realizovani projekat se sastoji iz dve aktivnosti. U prvoj se biraju parametri pokretanja igre, dok je u drugoj realizovana sama igra.

Dalje stavke ovog projekta realizovane su u skladu specifikacije teksta projektnog zadataka. Tu spadaju realizacije posebnih modula:

- Prikaz i biranje parametara igre u prvoj aktivnosti
- Čitanje početnih poteza iz txt fajla. (Inicijalni nazivi fajla: "Input_Santorini")
- Ispis odigranih poteza u txt fajl. (Inicijalni nazivi fajla: "Output_Santorini_main")
- Realizacija igre koju igra Čovek
- Realizacija igre koju igra Bot u tri nivoa težine:
    1. MinMax algoritam uz datu heurističku funkciju ($f = m + l$, gde je m broj pločica odredišnog polja, a l broj nivoa na koje se dodaje pločica pomnožen razlikom rastojanja sopstvenih i protivničkih igrača od tog polja). Prvi igrač je max igrač dok je drugi min igrač, pa su tako vrednosti ove funkcije množene sa 1 i -1 respektivno. Dalje je ovaj algoritam realizovan na opšti način (Svaka iretaracija rekurzije: sledeći igrač).
    2. MinMax algoritam sa AlphaBeta odsecanjem uz datu heurističku funkciju kao i u nivou 1.
    3. Custom algoritam koji je realizovan kao nadogradnja algoritma 2 gde je promenjena heuristička funkcija. Funkcija sadrži cenovnik poteza gde su za određene slučajeve dodeljivane konkretne vrednosti za koje je osmišljeno da daju što verniji prikaz trenutnog stanja table.

Algoritmi

1. MinMax algoritam

```
public static BoardState minmax(BoardState boardState,int depth,boolean maxPlayer,int
player)
```

2. MinMax AlphaBeta algoritam

```
public static BoardState minmaxAlphaBeta(BoardState boardState,int depth,int
a, int b, boolean maxPlayer,int player
```

3. Custom algoritam

```
public static BoardState minmaxNinja(BoardState boardState,int depth,int a, int b,
boolean maxPlayer,int player)
```

Heurističke funkcije za koje koriste ovi algoritmi:

```
• public static int heuristicFunction(BoardState boardState,int player)
• public static int heuristicFunctionNinja(BoardState boardState,int player)
```

Paket "algorithm"

Class Game (*Sadrži logiku kretanja figure*)

```
//Konstruktor
public Game(GameActivity myActivity,Cell[][] mCells, PrintWriter output)

//Logika odigravanja sledeceg poteza na koordinatama coordX,coordY
public String playNextMove(int coordX,int coordY)

//Promena stanja igre(Inicijalizija, Pomeri, Gradi)
public void nextState()

//Inicijalizacija figura na tabli matrix, na kordinate click_x,click_y za igrača player
public boolean setFigure(Cell [][] matrix, int click_x,int click_y,int player)
//Pomeranje figure na tabli matrix, igrača player na kordinate click_x, click_y
public boolean move(Cell [][] matrix,int click_x,int click_y,int player)
//Gradnja na tabli matrix, igrača player na kordinatama click_x, click_y
public boolean build(Cell [][] matrix,int click_x,int click_y,int player)
//Provera da li igrac player moze da pomeri neku od svojih figura

public static boolean canMove(Cell[][] matrix,int player)
//Vraca -1 ako nema pobednika na tabli, 0 ako je pobendik prvi igrac i 1 ako je pobednik
drugi igrac
public static int winnerWho(Cell[][] board)
//Upis koordinata u fajl(U obliku KarakterBroj [A1,B3...])
public void printPosOutputFile(int i, int j)
//Upis koordinata u fajl(U obliku KarakterBroj [A1,B3...]) u jednoj liniji
public void printLinePosOutputFile(int i,int j)
//Dekodovanje X koordinate (Iz oblika KarakterBroj [A1,B3...])
public static int decodeX(String data)
//Dekodovanje Y koordinate (Iz oblika KarakterBroj [A1,B3...])
public static int decodeY(String data)
//Vraca random slobodnu celiju
public Point randCell()
//Logika odigravanja sledeceg poteza, bot sa tezinama 1,2 ili 3, bira odredjene algoritme za
odabiranje iz MinMax klase
public void botPlayNextMove(int difficulty)
```

Class MinMax (*Sadrži logiku odabiranja sledećeg najboljeg poteza*)

```
//Funkcije vracaju max i min respektivno za date argumente
private static int max(int a,int b)
private static int min(int a,int b)
//Wrapper funkcija za minmaxNinja(...)
public static BoardState minmaxNinjaDecision(Cell[][] current_board,int depth,int player)
//Rekurzivna funkcija odabiranja sledeceg poteza tezine 3
public static BoardState minmaxNinja(BoardState boardState,int depth,int a, int b, boolean
maxPlayer,int player)
//Wrapper funkcija za minmaxAlphaBeta(...)
public static BoardState minmaxAlphaBetaDecision(Cell[][] current_board,int depth,int player)
//Rekurzivna funkcija odabiranja sledeceg poteza tezine 2
public static BoardState minmaxAlphaBeta(BoardState boardState,int depth,int a, int b,
boolean maxPlayer,int player)
//Wrapper funkcija za minmax(...)
public static BoardState minmaxDecision(Cell[][] current_board,int depth,int player)
//Rekurzivna funkcija odabiranja sledeceg poteza tezine 1
public static BoardState minmax(BoardState boardState,int depth,boolean maxPlayer,int player)
//Heuristicka funkcija za sledeci potez za tezine 1 i 2
public static int heuristicFunction(BoardState boardState,int player)
//Heuristicka funkcija za sledeci potez za tezinu 3
public static int heuristicFunctionNinja(BoardState boardState,int player)
```

Pomoćne clase:

- **Class Point**: struktura za čuvanje kordinata x i y (korišćena samo u MinMax klasi)
- **Class BoardState**: sve informacije o odigranom potezu (čvor u stablu minmax pretraživanja)

Paket "struct"

Class Cell(*Sadrži logiku jedne ćelije table*)

```
//Konstruktor
public Cell(int height, int player, int color, int x, int y)
//Geter i Seter za igraca koji je na toj celiji, -1 u suprotnom
public int getPlayer()
public void setPlayer(int player)
//Geter i Seter za visinu celije
public int getHeight()
public void setHeight(int height)
//Geter i Seter koordinata celije
public int getX()
public void setX(int x)
public int getY()
public void setY(int y)
//Geter i Seter boje celije
public int getColor()
public void setColor(int color)
//Inkrementiranje nivoa celije
public void incHeight()
```

Glavni paket (*view cele igre*)

Class MainActivity extends AppCompatActivity

```
//Funkcija koja se poziva na klik Start dugmeta i zapocinje igru (Poziva se sledeca
aktivnost)
public void onStartClick(View view)

//Dozvole za upis i citanje iz fajla androida
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults)
```

Class GameActivity extends AppCompatActivity

```
//Inicijalizacija table
private void initalizeBoard(Game myGame)
//Inicijalizacija izlaznog fajla
private PrintWriter initalizeOutputFile()
//Inicijalizacija igre na osnovu ulaznog fajla
private void readGameStateInputFile(String fileName)
//Postavljanje trenutnog naslova igre (ko igra i koji potez)
@SuppressLint("SetTextI18n")
private void setTitle()
//Postavljanje krajnjeg naslova igre (kada je igra gotova i kada je poznat pobednik)
public void setTitleWon()
//Poziva azuriranje table iz Adaptera
public void refreshBoard()
//Kompletna Wrapper funkcija(Poziva druge wrapper funkcije setTitle i refreshBoard)
azuriranja stanja tabele sa animacijama
public void nextMoveRefresh()
//Moze da vrati heuristicku funkciju za odredjeni potez (kada se igra covek protiv coveka)
@SuppressLint("SetTextI18n")
public void onTipsClick(View view)
//Wrapper funkcija odigravanja sledeceg poteza
public void onNextClick(View view)
```

Class MyAdapter extends RecyclerView.Adapter<MyAdapter.MojHolder>

```
//Azurira tablu vizuelno, dodeljujuci promenjene vrednosti parametara
public void refreshBoard()

//Potklasa sa funkcijom za setovanje vrednosti jedne celije
class MojHolder extends RecyclerView.ViewHolder
public void setValues(Cell cell)
```

Sadrže i implementirane klase iz kojih su izvedene(Specifiče za Andorid Studio okruženje)

Realizacija igre dostupna je na: https://github.com/marko-hudomal/SantoriniGame