Klase

Algorithm

Ovo je apstraktna klasa cije funkcije oslikavaju rad algoritma koji koristi vestacka inteligencija, tj. racunar. Najvaznija funkcija ove klase je apstraktna, jer je ideja da se iz ove klase izvedu konkretni algoritmi koji ce biti korisceni u radu programa.

public abstract Decision algorithm(int[][] matrix, Player[] players, int turn, int levelsToInspect, boolean isMax)

Ovo je najvaznija funkcija ove klase, jer predstavlja srz samog algoritma. Neke konkretne klase ce biti izvedene iz ove klase (MinMaxAlgorithm, AlphaBetaAlgorithm itd.) Ova funkcija vraca objekat tipa Decision koji predstavlja najbolju odluku za odigravanje poteza trenutnog igraca, na osnovu trenutnog stanja igre. Ulazni parametri su matrica stanja igre (matrix[i][j] oznacava nivo trenutne izgradnje polja [i][j]), niz igraca (konkretno u ovoj igri dva igraca), broj koji oznacava ko je trenutno na potezu (0 – prvi igrac, 1 – drugi igrac), broj koji oznacava koliko se duboko pretrazuje stablo igre i boolean promenljiva koja oznacava da li je igrac MIN ili MAX.

public static ArrayList<Point> findNeighbours(boolean move, int[][] matrix, int turn, int figure, Player p1, Player p2)

Metoda koja vraca listu polja-komsija od konkretne figure igraca koji je na potezu. Ukoliko argument move ima vrednost true, vraca se lista polja na koje data figura moze da se pomeri, a ukoliko ima vrednost false, vraca se lista polja na koje data figura moze da gradi.

public static boolean isSomebodyOnPosition(Point p, Player p1, Player p2)

Metoda koja vraca informaciju o tome da li se neka figura nalazi na poziciji p.

public static int findManhattanDinstanceSum(Point point, Player player)

Racuna sumu Menhetn rastojanja izmedju figura zadatog igraca i zadate tacke.

public static int findDistanceSum(Point point, Player player)

Racuna sumu rastojanja izmedju figura igraca do zadate tacke kao minimalan broj poteza pomeranja od figure do zadatog polja.

AlphaBetaAlgorithm

Konkretna klasa koja nasledjuje klasu Algorithm i implementira njenu apstraktnu metodu.

public Decision algorithm(int[][] matrix, Player[] players, int turn, int levelsToInspect, boolean isMax)

Override-ovana metoda klase roditeljske klase Algorithm koja samo poziva sledecu metodu.

public Decision algorithm(int[][] matrix, Player[] players, int turn, int levelsToInspect, boolean isMax, int alpha, int beta)

Metoda koja implemetira Alfa-Beta algoritam. Ima velike slicnosti sa MIN-MAX algoritmom, zapravo predstalja njegovu modifikaciju koja je optimalnija. Ideja je da za svaku figuru pronadjemo sva moguca polja gde ta figura moze da se pomeri, a zatim za svako to polje pronadjemo sva moguca polja gde figura moze da gradi. Par tacaka koji cine tacka pomeranja i tacka izgradnje smatramo odlukom i na osnovu toga pronalazimo odluke koje ce drugi igrac da donese na osnovu ove odluke (ukoliko je levelsToInspect > 0 ima smisla to raditi i tada rekurzivno zovemo funkciju, u suprotnom posmatramo samo tu jednu nasu odluku). Za svaku takvu odluku (bilo da je odluka trenutnog igraca ili odluka drugog igraca koja je doneta na osnovu odluke prvog igraca itd.) racunamo funkciju procene. Odluka koja ima najbolju funkciju procene (minimalnu ili maksimalnu, a to zavisi od toga da li je igrac MIN ili MAX) proglasavamo najboljom odlukom i vracamo kao rezultat funkcije. Ukoliko ne postoji takva odluka, to znaci da trenutni igrac ne moze nigde da se pomeri ili da gradi, tako da je on izgubio. Argumenti funkcije su promenljive alpha i beta. Alpha oznacava trenutno najbolju vrednost za MAX igraca, a beta za MIN igraca. Ukoliko se desi da je beta <= alfa, tada izlazimo iz petlje i na taj nacin smanjujemo broj grana stabla koje proveravamo i optimizujemo algoritam.

CompetitiveAlgorithm

Konkretna klasa koja nasledjuje klasu Algorithm i override-uje njenu metodu.

public Decision algorithm(int[][] matrix, Player[] players, int turn, int levelsToInspect, boolean isMax)

Predstavlja override-ovanu metodu roditeljske klase Algorithm. Telo se sastoji samo iz poziva sledece metode.

public Decision algorithm(int[][] matrix, Player[] players, int turn, int levelsToInspect, boolean isMax, int alpha, int beta)

Metoda koja ima prakticno isto telo kao metoda sa istim potpisom klase AlphaBetaAlgorithm. Glavna razlika je u metodi koja izracunava funkciju procene i koja je data u nastavku.

public static int getFunctionEvaluation(Point currentPoint, Point movePoint, Point buildPoint, int[][] matrix, int turn, Player p1, Player p2)

Kao argumente prima polje na kojem se trenutno trenutni igrac nalazi, polje na koje moze da se pomeri i polje na koje moze da gradi, kao i matricu i oba igraca. Ukoliko je polje pomeraja nivoa tri, funkcija bi trebalo da vrati maksimalnu vrednost, jer je to polje najveceg prioriteta, tj. pomeraj na to polje oznacava pobedu.

Sledeca stvar o kojoj bi trebalo voditi racuna jeste da li protivnik ima u blizini polje na koje moze da se premesti, a da je to polje nivoa 3. Ukoliko je tako, funkcija treba vratiti nesto manju vrednost u odnosu na proslu situaciju, jer bismo time sprecili protivnika da u sledecem potezu pobedi.

Sledeca taktika kojom se vodi ovaj igrac jeste da se uvek trudi da se popne na polje viseg nivoa i da gradi nad poljima viseg nivoa.

Controller

Klasa koja predstavlja kontrolera, posrednika izmedju View-a i Modela. Kada god se desi neki dogadjaj, npr. Odredjeni klik, Controller se obavesti, a zatim on obavesti Model-a koji vrati odredjeni kod i Controller na osnovu koda pozove odredjenu metoda za osvezavanje GUI-a.

public void clicked(int x, int y)

Kada god se desi neki klik, pozove se Controller, on pozove Model koji mu vrati kod i onda Controller pozove svoju metodu resolveCode() koja kao argument prima kod.

public void resolveCode(int code)

Na osnovu koda se odlucuje koja metoda ViewInterface-a se poziva, tj. na koji nacin se osvezava GUI.

public void currentIsTheWinner()

Oznacava da je trenutni igrac pobednik i na osnovu toga azurira GUI.

public void currentIsTheLoser()

Oznacava da je trenutni igrac izgubio i na osnovu toga azurira GUI.

public void saveToFile()

Zove istoimenu akciju Modela-a koji pamti stanje igre u izlaznom fajlu.

public void loadFromFile(String filename)

Zove istoimenu akciju Modela-a koji cita stanje igre iz ulaznog fajla.

DateTimeHelper

Ima samo jednu funkciju koja se koristi prilikom odredjivanja naziva fajla u koji se smesta stanje igre.

public static String getCurrentDateTimeString()

Vraca String koji predstavlja naziv fajla u kojem ce biti sacuvano stanje igre. Taj string se sastoji iz trenutnog datuma i trenutnog vremena i na taj nacin ce svaki fajl imati jedinstven naziv.

Decision

Predstavlja odluku, tj. potez igraca. Polja su: figura koja se koristi u potezu, polje na koje se figura pomera, polje nad kojim se gradi, vrednost funkcije za dati potez, kao i StepByStepResolver koji ce biti objasnjen u daljem tekstu. Sadrzi samo odgovarajuce konstruktore i get i set metode.

public Decision(Point movePoint, Point buildPoint, int figure)

public Decision(Point movePoint, Point buildPoint, int figure, int functionValue)

public Point getMovePoint()

public void setMovePoint(Point movePoint)

public Point getBuildPoint()

public void setBuildPoint(Point buildPoint)

public int getFigure()

public void setFigure(int figure)

public int getFunctionValue()

public void setFunctionValue(int functionValue)

public StepByStepResolver getResolver()

public void setResolver(StepByStepResolver resolver)

GameActivity

Klasa aktivnosti koja predstavlja samu igru. Moze biti pozvana od strane MainActivity aktivnosti ili LoadActivity aktivnosti.

protected void onCreate(Bundle savedInstanceState)

Metoda koja se poziva svaki put kada se kreira ova aktivnost. U ovoj metodi se vrsi inicijalizacija GUI-a i dohvataju se odgovarajuci GUI elementi koji ce kasnije biti korisceni i azurirani. Prave se objekti Controllera, Modela i ViewInterface-a i povezuju se. Traze se odgovarajuce dozvole, ukoliko vec nisu dobijene. Postavljaju se onClickListener-i na svako polje i definise se sta se desava kada se klikne. U svakom slucaju se poziva odgovarajuca akcija Controllera, a pritom se i vodi racuna u kojem je rezimu pokrenuta igra.

public void onNextClicked(View view)

Metoda koja se poziva svaki put kada se klikne na ‘Next’ dugme. Ovo dugme je aktivno jedino u modu kada igra racunar protiv racunara i to kada se odabere rezim ‘korak po korak’. Svaki put kada se klikne na ovo dugme prikaze se rezultat rada odredjenog dela algoritma.

public void onSaveClicked(View view)

Metoda koja se poziva kada se klikne na ‘Save’ dugme. U tom trenutku se u fajlu cuva trenutno stanje igre, tj. poziva se odredjena metoda Controllera koja to obavlja.

LoadActivity

Klasa aktivnosti koja implementira mehanizam za ucitavanje stanja igre. Ova aktivnost moze biti pozvana od strane MainActivity aktivnosti.

protected void onCreate(Bundle savedInstanceState)

Ova metoda poziva se svaki put kada se kreira ova aktivnost. U ovoj metodi se vrsi citanje podataka koji su poslati od strane MainActivity aktivnosti i ti podaci ce biti poslati kasnije GameActivity aktivnosti. Pronalaze se svi fajlovi u kojima je sacuvano neko stanje igre i nazivi tih fajlova se prikazuju na korisnickom interfejsu. Postavljaju se i odredjeni osluskivaci, tako da klikom na odredjeni naziv, zapravo selektujemo fajl koji cemo ucitati.

public void onStartClicked(View view)

Metoda koja se poziva kada se klikne na ‘Start’ dugme. ‘Start’ dugme moze biti jedino aktivno ako postoji barem jedan fajl koji moze da se ucita. U ovoj metodi se pravi objekat klase Intent koji zapravo sluzi za prenos podataka izmedju ove aktivnosti i GameActivity aktivnosti. Poziva se metoda koja kreira GameActivity aktivnost i graficki interfejs se menja u skladu sa tim.

MainActivity

Aktivnost koja se pokrece samim pokretanjem igre. Na interfejsu sadrzi odredjene opcije i dugmice. Moguce je odabrati mod igre, tj. da li igra covek protiv coveka, covek protiv racunara ili racunar protiv racunara. U slucaju da je barem jedan od igraca racunar, moguce je odabrati da li on treba da koristi MIN-MAX algoritam (jedan nivo blokova), Alfa-Beta algoritam (dva nivoa blokova) ili Takmicarski algoritam (tri nivoa blokova). U slucaju da je odabran mod racunar protiv racunara, moguce je i odabrati rezim 'Korak po korak' ili rezim koji predstavlja simulaciju igre sa odredjenim pauzama koje predstavljaju vreme razmisljanja racunara.

protected void onCreate(Bundle savedInstanceState)

Metoda koja se poziva svaki put kada se kreira ova aktivnost. Dohvataju se odredjeni elementi GUI-a kojima ce se pristupati u ostalim metodama.

public void onStartClicked(View view)

Ova metoda se poziva klikom na 'Start' dugme. Kreira se Intent objekat koji se puni odgovarajucim informacijama o konfiguraciji igre i kreira se GameActivity aktivnost.

public void onHumanVsHumanClicked(View view)

Metoda koja se poziva prilikom klika na GUI element gde su prikazana dva coveka i oznacava biranje moda u kojem igra covek protiv coveka. Taj GUI element onda dobija zelenu pozadinu.

public void onHumanVsRobotClicked(View view)

Metoda koja se poziva prilikom klika na GUI element gde su prikazani covek i robot i oznacava biranje moda u kojem igra covek protiv racunara. Taj GUI element onda dobija zelenu pozadinu. Na grafickom interfejsu se pojavi opcija gde moze da se izabere koji algoritam koristi robot i do koje dubine se vrsi pretrazivanje stabla.

public void onRobotVsRobotClicked(View view)

Metoda koja se poziva prilikom klika na GUI element gde su prikazana dva robota i oznacava biranje moda u kojem igra racunar protiv racunara. Taj GUI element onda dobija zelenu pozadinu. Na grafickom interfejsu se pojavi opcija gde moze da se izabere koji algoritam koristi odredjeni robot i do koje dubine se vrsi pretrazivanje stabla.

public void onLoadClicked(View view)

Metoda koja se poziva kada se klikne na ‘Load’ dugme. Napravi se Intent objekat i popuni odgovarajucim informacijama o konfiguraciji igre i pokrene se LoadActivity aktivnost.

public void onLevel1Clicked(View view)

Metoda koja se poziva kada se klikne na GUI element koji se odnosi na podesavanje algoritma koji koristi prvi racunar. (prvi klik – MIN MAX algoritam, drugi klik - Alfa Beta, treci klik – takmicarski algoritam).

public void onLevel2Clicked(View view)

Metoda koja se poziva kada se klikne na GUI element koji se odnosi na podesavanje algoritma koji koristi drugi racunar. (prvi klik – MIN MAX algoritam, drugi klik - Alfa Beta, treci klik – takmicarski algoritam).

public void onBranching1Clicked(View view)

Metoda koja se poziva kada se klikne na GUI element koji se odnosi na podesavanje nivoa pretrazivanja stabla igre za prvi racunar.

public void onBranching2Clicked(View view)

Metoda koja se poziva kada se klikne na GUI element koji se odnosi na podesavanje nivoa pretrazivanja stabla igre za drugi racunar.

public void onStepClicked(View view)

Metoda koja se poziva kada se klikne na GUI element koji se odnosi na rezim rada. Ova opcija je jedino aktivna i vidljiva kada se odabere mod u kojem igra racunar protiv racunara. Oznacava rezim ‘korak po korak’.

public void onFlowClicked(View view)

Metoda koja se poziva kada se klikne na GUI element koji se odnosi na rezim rada. Ova opcija je jedino aktivna i vidljiva kada se odabere mod u kojem igra racunar protiv racunara. Oznacava rezim koji predstavlja simulaciju igre sa odredjenim pauzama u potezima racunara.

Messages

Klasa bez metoda koja ima staticka polja koja predstavljaju poruke koje se prikazuju u odredjenim fazama igre, npr. ‘Select your figure’ itd.

MiniMaxAlgorithm

Konkretna klasa koja nasledjuje klasu Algorithm i implementira njenu apstraktnu metodu.

public Decision algorithm(int[][] matrix, Player[] players, int turn, int levelsToInspect, boolean isMax)

Ideja je da za svaku figuru pronadjemo sva moguca polja gde ta figura moze da se pomeri, a zatim za svako to polje pronadjemo sva moguca polja gde figura moze da gradi. Par tacaka koji cine tacka pomeranja i tacka izgradnje smatramo odlukom i na osnovu toga pronalazimo odluke koje ce drugi igrac da donese na osnovu ove odluke (ukoliko je levelsToInspect > 0 ima smisla to raditi i tada rekurzivno zovemo funkciju, u suprotnom posmatramo samo tu jednu nasu odluku). Za svaku takvu odluku (bilo da je odluka trenutnog igraca ili odluka drugog igraca koja je doneta na osnovu odluke prvog igraca itd.) racunamo funkciju procene. Odluka koja ima najbolju funkciju procene (minimalnu ili maksimalnu, a to zavisi od toga da li je igrac MIN ili MAX) proglasavamo najboljom odlukom i vracamo kao rezultat funkcije. Ukoliko ne postoji takva odluka, to znaci da trenutni igrac ne moze nigde da se pomeri ili da gradi, tako da je on izgubio.

Model

Klasa koja implementira svu problemsku logiku i na taj nacin je drzi odvojeno od View-a i Controllera.

public Model(int dimension)

Konstruktor koji kao argument prima dimenziju matrice, konkretno u igri je 5. Vrse se odredjene inicijalizacije.

public Controller getController()

Get metoda koja vraca objekat Controller klase.

public void setController(Controller controller)

Set metoda koja postavlja objekat Controller klase.

public void setFigurePosition(int player, int figure, int x, int y)

Postavlja zadatu figuru zadatog igraca na zadate coordinate, tj. poziciju.

public int clicked(int x, int y)

Ovu metodu poziva Controller svaki put kada se desi neki klik. Model vodi racuna o tome sta bi ovaj klik trebalo da predstavlja, npr. odabir figure prvog igraca, drugog igraca, pomeranje figure, izgradnju itd. i na osnovu toga azurira matricu i vraca odredjeni kod koji kasnije Controller tumaci i na osnovu toga azurira GUI.

public boolean isSomebodyOnPosition(int x, int y)

Vraca informaciju o tome da li se neka figura nalazi na zadatim koordinatama.

public boolean isPlayerOnPosition(int player, int x, int y)

Vraca informaciju o tome da li se neka od figura zadatog igraca nalazi na zadatim koordinatama.

public Point[] getPointsFromPlayer(int player)

Vraca pozicije na kojima se nalaze figure zadatog igraca.

public void findSelectedFigure(int x, int y)

Pronalazi figuru koju je selektovao igrac koji trenutno igra.

public int getFigureChosen()

Dohvata broj figure koja je selektovana.

public int getTurn()

Vraca informaciju o tome ko je trenutno na potezu.

public ArrayList<Point> findNeighbours(boolean move)

Pronalazi sva susedna polja na koja se moze premestiti ili nad kojima moze graditi igrac koji je na potezu (false vrednost za gradnju, true vrednost za premestanje).

public ArrayList<Point> findNotNeighbours(boolean move)

Pronalazi sva susedna polja na koja se ne moze premestiti ili nad kojima ne moze graditi igrac koji je na potezu (false vrednost za gradnju, true vrednost za premestanje).

public int[][] getMatrix()

Metoda koja vraca matricu.

public int getPrevX()

Metoda koja vraca X koordinatu koja je poslednji put bila izabrana.

public int getPrevY()

Metoda koja vraca Y koordinatu koja je poslednji put bila izabrana.

public Player[] getPlayers()

Metoda koja vraca igrace.

public void saveToFile()

Metoda koja sluzi za cuvanje stanja igre u izlazni fajl.

public void loadFromFile(String filename)

Ucitavanje stanja igre iz ulaznog fajla.

public int getValidClicks()

Metoda koja vraca broj validnih klikova

MoveObserver

Klasa koja sluzi za belezenje poteza od strane Modela kako bi se oni prepisali u izlazni fajl u slucaju cuvanja stanja igre ili ucitali iz fajla u slucaju ucitavanja stanja igre.

public MoveObserver()

Konstruktor koji samo napravi listu poteza.

public void addPoint(Point point)

Metoda za dodavanje poteza u listu poteza.

public String getString()

Metoda koja vrsi konverziju poteza u veliki string koji se zapisuje u izlazni fajl.

public void saveToFile()

Metoda koja pronalazi odgovarajuci direktorijum u kojem se cuvaju fajlovi vezani za ovu igru, pravi novi fajl(ukoliko prethodno igra nije sacuvana ili ucitana) i upisuje u fajl String objekat koji odgovara potezima igraca.

public void loadFromFile(String filename, Controller controller)

Metoda koja pronalazi odgovarajuci direktorijum u kojem se cuvaju fajlovi vezani za ovu igru, pronalazi fajl sa zadatim imenom, cita njegov sadrzaj, razlaze svaku liniju na odgovarajuce Stringove i njih konvertuje u potez, a zatim poziva clicked(int x, int y) metodu Controllera i na taj nacin ucitava stanje igre.

MyViewInterface

Klasa koja implementira ViewInterface interfejs.

public void drawPlayer0(int x, int y)

Metoda koja iscrtava figure prvog igraca na zadatoj poziciji.

public void drawPlayer1(int x, int y)

Metoda koja iscrtava drugog igraca na zadatoj poziciji.

public void setMessage(String message)

Metoda koja postavlja poruku, npr. ‘Select your figure’ itd.

public void setToast(String toast)

Metoda koja kreira Toast sa odredjenom porukom.

public void setPlayerName(String playerName)

Metoda koja postavlja informaciju o tome koji igrac je na potezu.

public void fadePlayer(int player, Point[] points)

Metoda koja izbledi prikaz figura zadatog igraca.

public void fadeFields(ArrayList<Point> points, int[][] matrix)

Metoda koja izbledi zadata polja matrice.

public void removePlayer(int x, int y)

Metoda koja uklanja bilo kakvu figuru na zadatom polju.

public void unfadeAllFields(int[][] matrix)

Metoda koja vraca izgled svih polja u originalno stanje.

public void unfadePlayer(int player, Point[] points)

Metoda koja vraca prikaz figura zadatog igraca u originalno stanje.

public void disableNextButton()

Metoda koja onemogucava klik na ‘Next’ dugme.

public void enableSaveButton()

Metoda koja aktivira mogucnost klika na ‘Save’ dugme.

Player

Klasa koja predstavlja igraca. Od polja sadrzi trenutne pozicije figura igraca.

public Player()

Javni konstruktor.

public void setFigurePoint(int figure, int x, int y)

Postavlja zadatu figuru na zadato polje

public Point getFigurePoint(int figure)

Vraca poziciju zadate figure.

public Point[] getFigurePoints()

Vraca pozicije figura.

Point

Klasa koja predstavlja poziciju odredjenu koordinatama X i Y.

public Point(int x, int y)

Konstruktor kojim se postavljaju x i y coordinate.

public int getX()

Vraca x koordinatu.

public void setX(int x)

Postavlja x koordinatu.

public int getY()

Vraca y koordinatu.

public void setY(int y)

Postavlja y koordinatu.

public boolean equals(Point p)

Proverava da li su ove pozicije iste, tj. da li se obe coordinate poklapaju.

StepByStepResolver

Klasa koja se koristi prilikom rezima rada ‘korak po korak’. Ideja je da kada u jednom trenutku kliknemo na next, na susednim poljima selektovane figure vidimo koje su vrednosti funkcije procene u slucaju kada se ta figura premesti na tu poziciju.

public void add(Point movePoint, Point buildPoint, int functionValue)

Metoda koja dodaje jedan od mogucih poteza koji se sastoji od pomeranja figure na movePoint i izgradnje na buildPoint, uz vrednost funkcije functionValue.

public ArrayList<Decision> getBestBuildDecisionsForEveryMove()

Kada se selektuje figura, sledeci klik na dugme ‘Next’ prikazuje najbolje funkcije procene za svako moguce pomeranje.

public ArrayList<Decision> getBestBuildDecisionsForMove(Point movePoint)

Kada se figura pomeri na movePoint, sledeci klik na dugme ‘Next’ prikazuje najbolje funkcije procene za svaku mogucu izgradnju.