Prirodoslovno matematički fakultet

Sveučilište u splitu

Seminarski rad

UVOD U UMJETNU INTELIGENCIJU

BJEŽANJE OD LOVACA

Student: Josip Goreta

Split, veljača, 2018.

SADRŽAJ

1. UVOD

Svrha ovoga rada je da se opiše projekt „Bježanje od lovaca“. Osnovna ideja projekta je da se kroz veliki labirint bježi od lovaca koji predstavljaju policajce. Igrač je lopov koji je upao u banku i krade novčiće po labirintu koji predstavlja banku. Agenti imaju oblik osobe, lopov je predstavljen zelenom bojom, a policajci plavom.

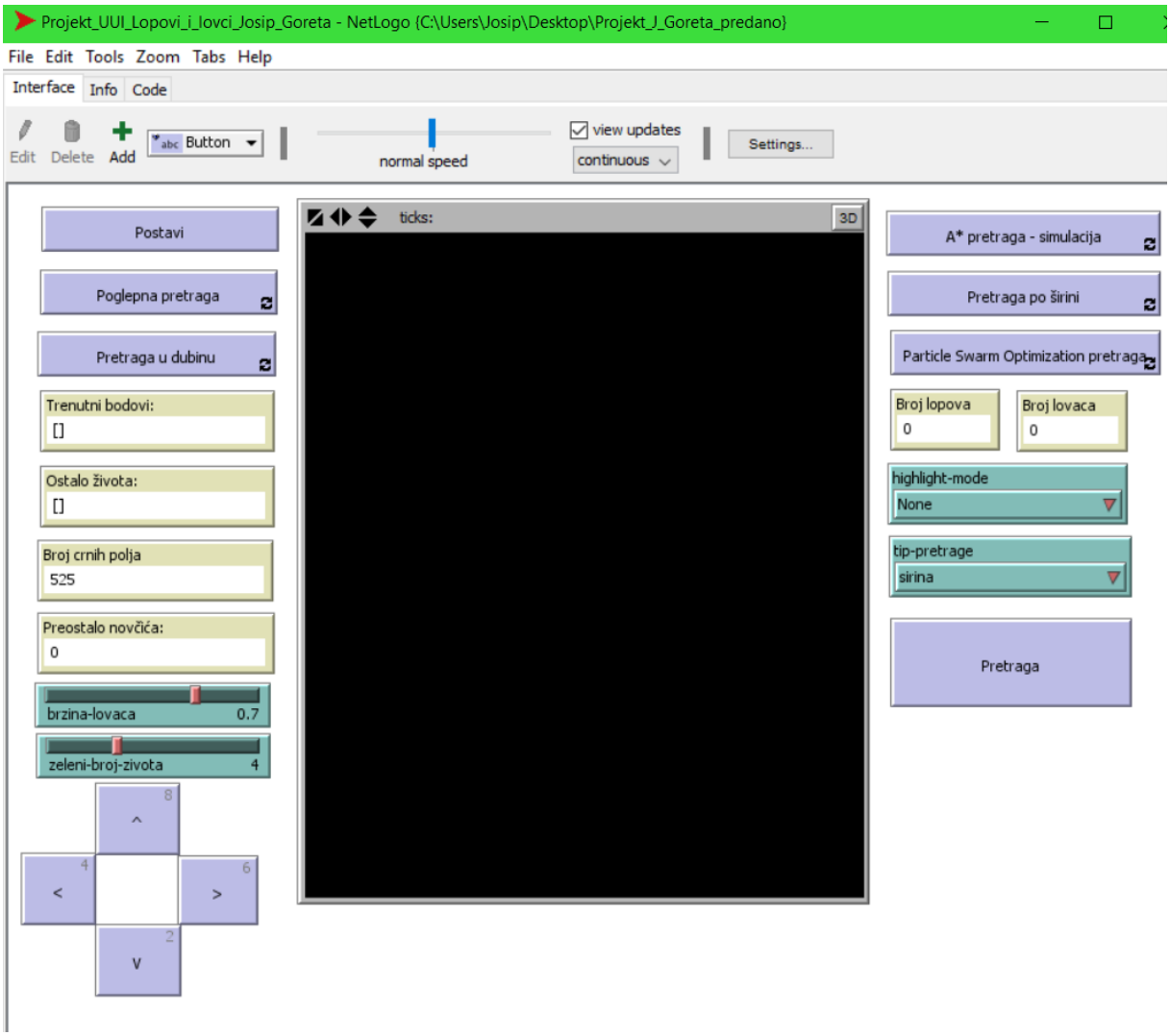
Labirint se sastoji od dvije vrste polja. Crvena polja predstavljaju zidove kroz koje ni lopov ni policajci ne mogu proći, te crna polja koja predstavljaju slobodan prostor i po njima su pobacani novčići.

Cilj igre je da lopov pokupi sve novčiće prije nego izgubi sve živote, a živote gubi kad ga policajci uhvate.

2. OPIS PROJEKTA

2.1. Izgled projekta

Izgled sučelja prije nego što se pokrene igra izgleda ovako:



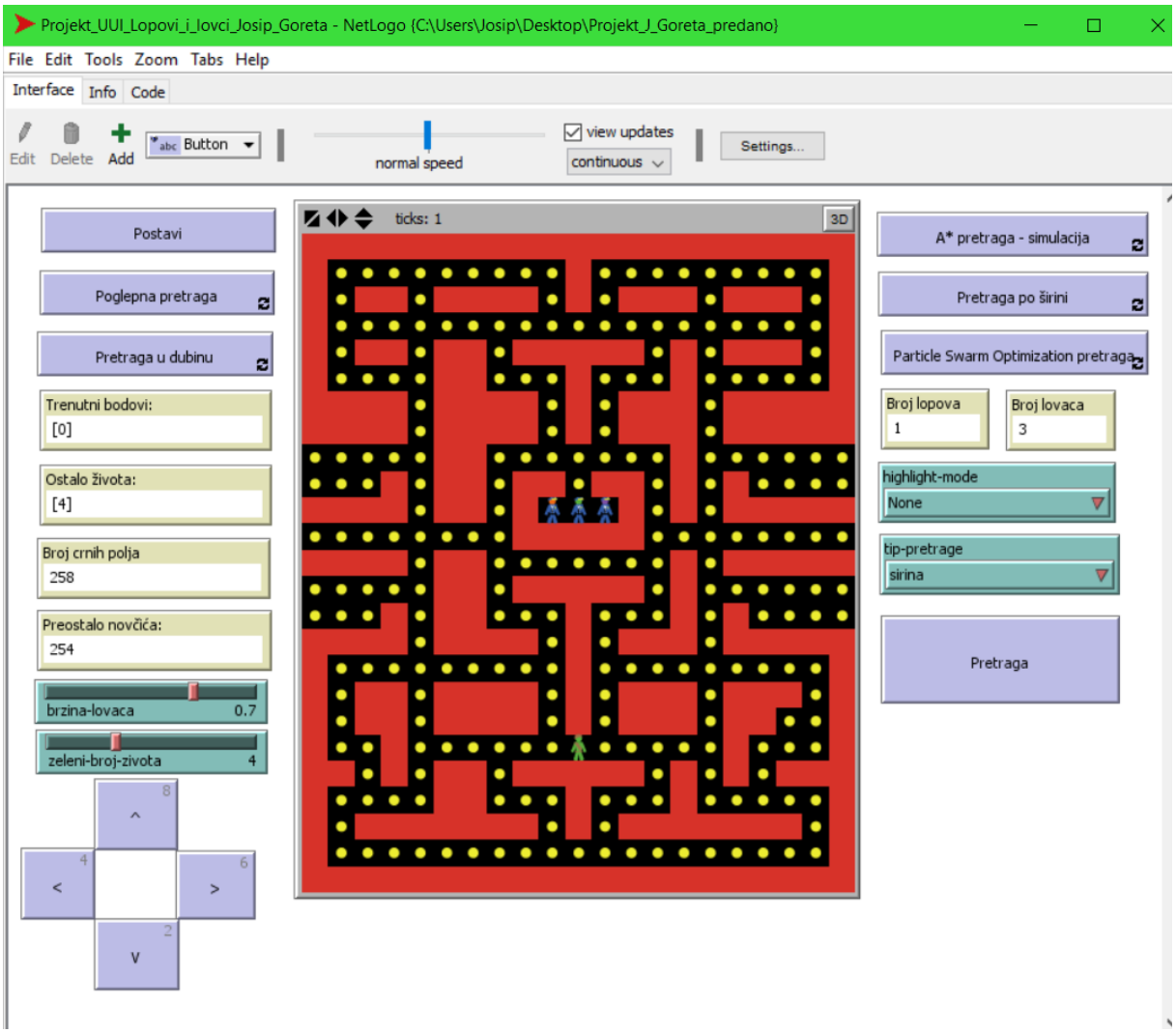
Slika 1. Sučelje igre prije pritiska na botun „Postavi“

Na sučelju se nalaze i dodatni botuni, monitori i slideri pomoću kojih igrač gleda i regulira tijek igre.

- botun „Postavi“
- botun „Pohlepna pretraga“
- botun „Pretraga u dubinu“
- monitor „Trenutni bodovi“
- monitor „Ostalo života“

- monitor „Broj crnih polja“
- monitor „Preostalo novčića“
- slider „brzina-lovaca“
- slider „zeleni-broj-zivota“
- botun „A* pretraga - simulacija“
- botun „Pretraga po širini“
- botun „Particle Swarm Optimization pretraga“
- monitor „Broj lopova“
- monitor „Broj lovaca“
- chooser „highlight-mode“
- te četiri botuna za svaki smjer.

2.2 Postavljanje svijeta



Slika 2. Sučelje igre nakon pritiska na botun „Postavi“

Stiskom na botun „*Postavi*“ poziva se procedura „*Setup*“ koja u sebi sadrži proceduru za kreiranje labirinta „*labirint*“ te procedure za stvaranje agenata.

Procedura „stvori-*lopova*“ stvara agenta lopova i postavlja ga uvijek na isto polje u labirintu, a to je patch 10 5.

```
to stvori-lopova
  create-lopovi 1
  [
    move-to one-of patches with [pxcor = 10 and pycor = 5]
    set heading 0
    set shape "person soldier"
    set color green
    set uk-bodova 0
    set cilj? false
    set uk-zivota zeleni-broj-zivota
    set tikovi 0
    ;pen-down
    ]
end
```

Primjer koda 1: Definicija procedure „stvori-lopova“

Procedura „*stvori-lovce*“ stvara agente policajce i postavlja ih uvijek na isto mjesto u labirintu tj. na patcheve 9 14, 10 14 i 11 14. Na početku se uvijek stvore tri lovca, a dalje tokom igre im se mijenja broj ovisno o kretanju po stazi i ovisno i pretrazi.

```
to stvori-lovce
  create-lovci 3
  [
    move-to one-of patches with [(pxcor = 9 and pycor = 14 )
or (pxcor = 10 and pycor = 14) or (pxcor = 11 and pycor = 14)
and not any? lovci-here
      ]
    set heading 0
    set shape "person police"
    let r 0
    ask one-of lovci [set color yellow]
    ask one-of lovci [set color white]
    ask one-of lovci [set color grey]
    ]
end
```

Primjer koda 2: Definicija procedure „stvori-lovce“

Procedura „*stvori-novcice*“ stvara agente koji predstavljaju novac. Pri postavljanju svijeta oni se rasporede po svim crnim poljima na kojima nema nijedan drugi agent, tj. rasporede se na 254 crna polja labirinta od mogućih 258 zato što se u početku stvore četiri agenta, tj. jedan lovac i tri policajca.

```
to stvori-novcice
  create-novcici count patches with [pcolor = black] - 4
  [
    move-to one-of patches with [pcolor = black and not any?
turtles-here]
    set color yellow
    set shape "dot"
    ]
end
```

Primjer koda 3: Definicija procedure „stvori-novcice“

Procedura „*stvori-bonuse2*“ stvara agenta koji ima simbol hrane na nekom random crnom polju i kad ga lopov pokupi poveća mu se život za 1. On nije vidljiv odmah na početku igre nego se stvori tek kada lopov sakupi 200 bodova, tj. 200 novčića.

```
to stvori-bonuse2
  create-bonusi2 1
  [
    move-to one-of patches with [pcolor = black]
    set shape "food"
    set color blue
    set hidden? true
    ]
end
```

Primjer koda 4: Definicija procedure „stvori-bonuse2“

Procedura „*stvori-bonuse*“ stvara agente koji imaju simbol zvjezice. Pobaca ih u kutove labirinta i postavi im bool varijablu „*hidden?*“ na „*true*“. Postanu vidljivi tek kada lopov sakupi 20 novčića.

```
to stvori-bonuse
  create-bonusi 4
  [
    move-to one-of patches with [pcolor = black and ((pxcor =
1 and pycor = 23 ) or (pxcor = 1 and pycor = 1 ) or (pxcor =
19 and pycor = 23 ) or (pxcor = 19 and pycor = 1 )) and not
any? bonusi-here]
    set shape "star"
    set color yellow
    set hidden? true
    ]
end
```

Primjer koda 5: Definicija procedure „stvori-bonuse“

Pomoću tipki „^“, „*V*“, „<“ i „>“ ručno mičemo agenta lopova po labirintu i to samo po crnim poljima.

2.3. Agenti

U ovom projektu postoji pet vrsta agenata. To su „*novcici*“, „*lopov*“, „*lovci*“, „*bonus2*“ i „*bonusi*“.

Glavni agenti u simulaciji su lovci i lopov, a sporedni novčići, i dvije vrste bonusa.

Lopov ima svojstvo „uk-*bodova*“ koje je na početku postavljeno na nulu i povećaje se za jedan kad god skupi novčić. Zatim ima svojstvo „*cilj?*“ koje govori da li je igra završena. Na početku je postavljeno na „*false*“, a kad lopov pokupi sve novčiće postavlja se na „*true*“ i igra je završena. Svojstvo „*uk-zivota*“ pokazuje koliko je još života preostalo lopovu. Kad „*uk-zivota*“ bude nula igra je gotova i lopov je izgubio.

Lovci od svojstava imaju „*procjena-puta*“ koja govori koliko je koji duh udaljen od lopova i ovisno o tome duhovi se stvaraju i umiru.

Agenti „*novcici*“ su žute boje i imaju oblik kovanice. Njih lopov prikuplja i tako pokušava prijeći igricu. „*bonus2*“ je agent u obliku hrane i plave je boje. Stvara se nasumično na nekom crnom polju kada lopov sakupi 200 bodova. Lopov ga tada može pokupiti tako da prijeđe preko njega i onda on nestaje zauvijek. „*bonus*“ su agenti koji se stvore kada lopov sakupi 20 bodova. Stvori se njih četiri i imaju oblik žute zvijezdice. Služe lopovu da dobije 2 života kad ih sakupi, ali mu dobro dođu i kada ga lovci stjeraju u kut pa kada pokupi zvijezdicu premjesti se na mjesto na kojem se u početku stvore lovci uz pretpostavku da tada tu neće biti nitko od lovaca pošto su svi ostali u onom kutu u kojem je lopov pokupio bonus zvjezdicu.

3. PRETRAGA

3.1 Dodatne postavke pretrage

Na početku igre se mogu namjestiti dodatne postavke ovisno da li igrač želi lakšu ili težu igru. To može učiniti pomoću dva slidera. Na slideru „*brzina-lovaca*“ igrač namješta kojom brzinom želi da se kreću lovci. Moguće je namjestiti brzinu od 0 do 1 sa povećanjem za 0.1. S obzirom da se lopov uvijek kreće brzinom 1, a lovci nikada ne mogu biti brži od njega, može se namjestiti da budu jednako brzi ili sporiji ili čak da se ne miču kad se postavi na 0.

Slider „*zeleni-broj-zivota*“ govori koliko će života na početku simulacije imati lopov. Igrač može namjestiti da lopov ima od 1 do 10 života sa povećanjem za 1.

3.2. Igranje

U projektu kada igrač igra protiv računala postoje dvije vrste pretrage koje koriste lovci da bi uhvatili lopova. To su *pretraga po dubini* i *pretraga pohlepnog algoritma*, a heuristika je *manhattan udaljenost*.

Za ovaj projekt je odabran *pohlepni algoritam* jer on može dati jako brzu aproksimaciju najboljeg rješenje jer u svakom koraku bira lokalno najbolje rješenje. Ali iako daje brzu aproksimaciju ipak ne daje uvijek najbolje rješenje.

Algoritam pretrage po dubini je korišten zbog specifične vrste labirinta i izgledao je kao povoljan algoritam za prolazak kroz „duge tunele“ ovog labirinta.

Sama pretraga započinje pritiskom na botun jedne od ponuđene dvije pretrage te korakom lopova. Lovci mogu napraviti korak tek kada ga lopov napravi i tako redom do kraja igre. To je realizirano uz pomoć svojstva „*tikovi*“ koje omogućuje da lovci mogu napraviti sljedeći korak tek kada lopov napravi korak. Korak agenata koji se kreću definiran je u proceduri „*korak*“.

Pretraga po dubini se radi pozivom procedure „*dubina*“, a pohlepna pretraga pozivom procedure „*pretraga2*“.

Pri svakom koraku lovaca gleda se da li su na nekom križanju. Ako su naišli na križanje onda se umnože u svakom smjeru (*engl. hatch*), ali je to umnožavanje ograničeno da lopova hvata najviše četiri lovca da se nebi stvorilo previše lovaca i otežalo igru igrača. Ako ih se umnoži više tada se ubija onoga koji ima najveću procjenu cijene puta do lopova što se može očitati u svojstvu lovaca „*procjena-puta*“. Ako se prilikom pretrage lovac i lopov nađu na istom mjestu, lopov gubi jedan život i svi se vraćaju na svoje početne pozicije.

Unutar projekta postoje još neke procedure koje su važne za tijek igre. To su procedura „*skupi-bonus2*“ koja lopovu omogućava kad skupi 200 bodova da skupi dodatni život, zatim procedura „*skupi-bonus*“ koja lopovu omogućava kada skupi 20 bodova da može dobijati dodatna dva života za svaku zvijezicu koju pokupi. Dalje postoji procedura „*provjeri-uk-zivota*“ koja služi da se lopovu smanji život za jedan kada ga uhvati neki od lovaca, te procedura „*manhattan-udaljenost2*“ koja prima dva parametra, x i y te računa udaljenost lopova od lovaca.

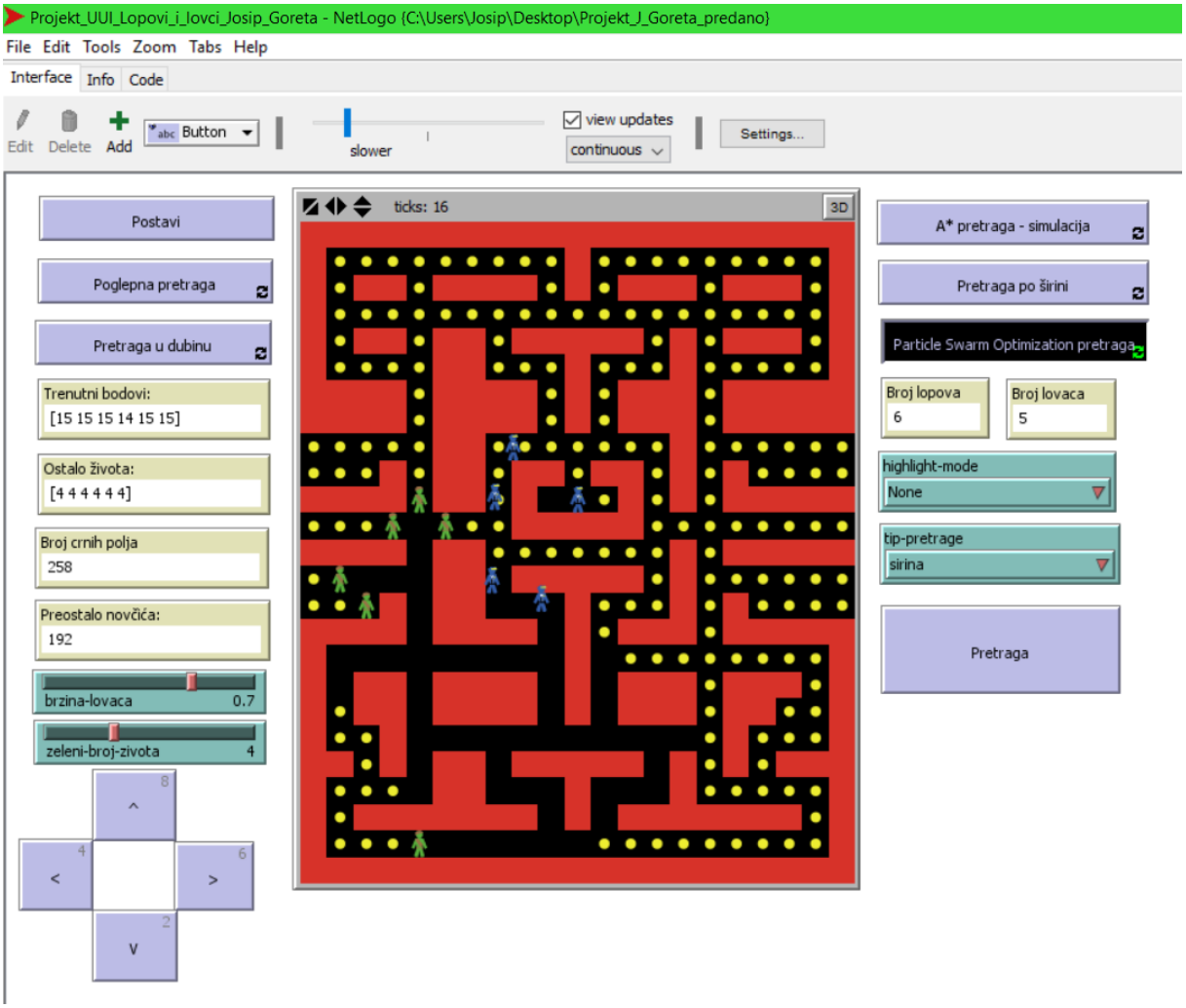
Kada se klikne na botun „*A* pretraga - simulacija*“ tada se poziva procedura „*pretraga3*“ koja lovce šalje pretragom pohlepnog algoritma, a lopove A* pretragom i odvija se simulacija.

Klikom na botun „*Pretraga po širini*“ poziva se procedura „*pretraga4*“ koja lopove šalje algoritmom pretrage po širini, a lovce pohlepnom pretragom.

Klikom na botun „*Particle Swarm Optimization pretraga*“ poziva se procedura „*PSO*“ koja lopove šalje pohlepnom pretragom, a lovce šalje jednim od „Clever“ algoritama, a to je Particle Swarm Optimization. To je optimizacija rojem čestica i spada u algoritme inteligencije mnoštva. Uzmemo za primjer jato ptica koje traži hranu. Ukoliko jedna ptica osjeti dobar izvor hrane vrlo je vjerojatno kako će je i ostale ptice slijediti, no ipak kako bi se omogućila i potraga za boljim hraništem , svaka ptica ima u sebi neki svoj instinkt za bolje hranište i time joj je omogućeno kratkotrajno odvajanje od jata u potrazi za boljim hraništem.

Tom logikom se vode i agenti u ovom projektu. Lovci imaju instinkt prema lopovima uz pomoć procjene puta koju dobiju korištenjem procedure za manhattan udaljenost. Prilikom određivanja

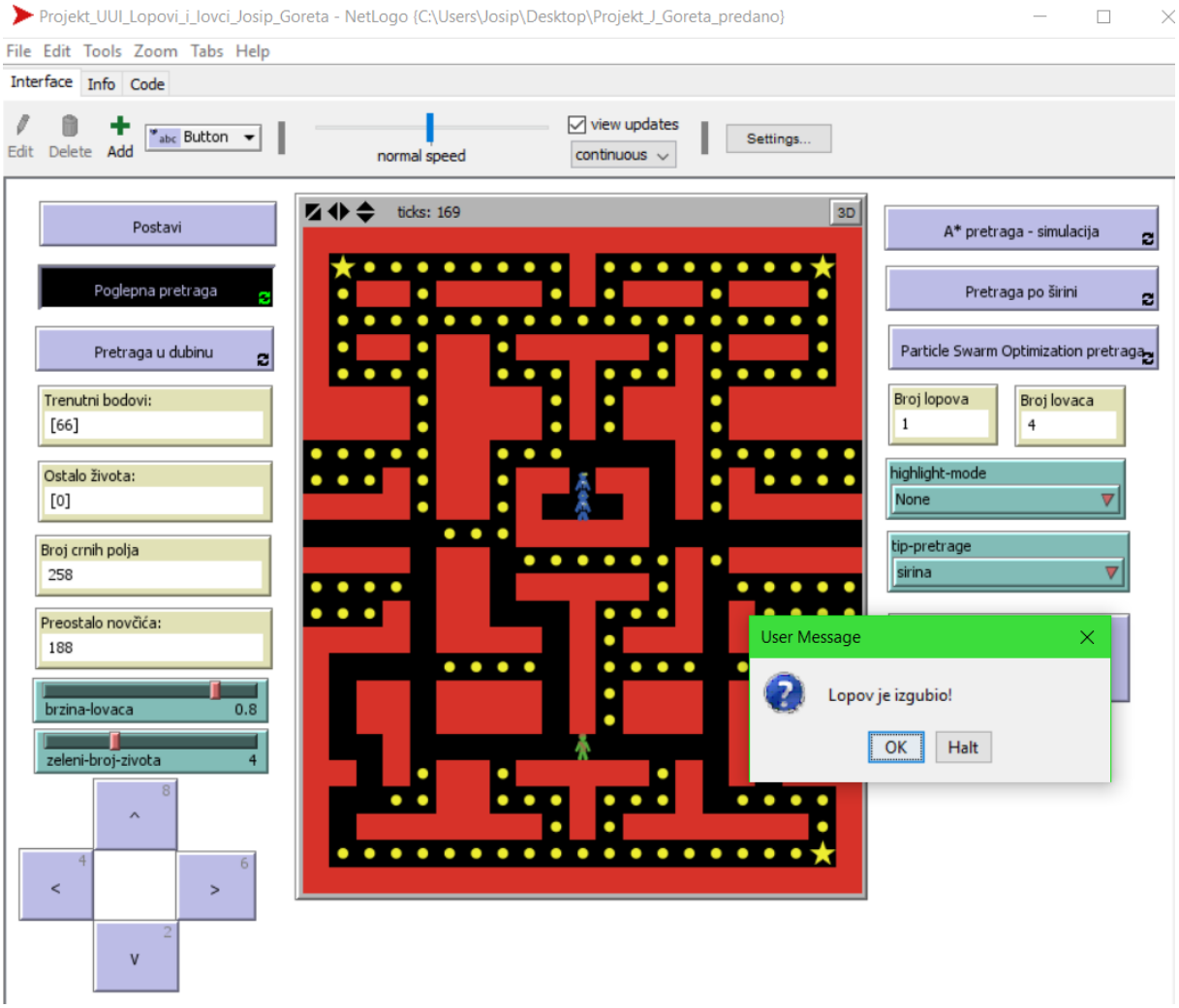
smjera kretanja svaki lovac uzima u obzir svoje do tada pronađeno najbolje rješenje te najbolje rješenje lovaca iz okoline.



Slika 3. Sučelje igre nakon nekoliko koraka

3.3 Cilj

Svaki od glavnih agenata ima svoj cilj. Lovci imaju za cilj uhvatiti lopova kojim ručno upravlja igrač, a lopovu je cilj prikupiti sve novčiće prije nego izgubi sve živote. Lopov pomoću agenata „*bonusi2*“ i „*bonusi*“ dobija dodatne živote što igraču olakšava igranje.

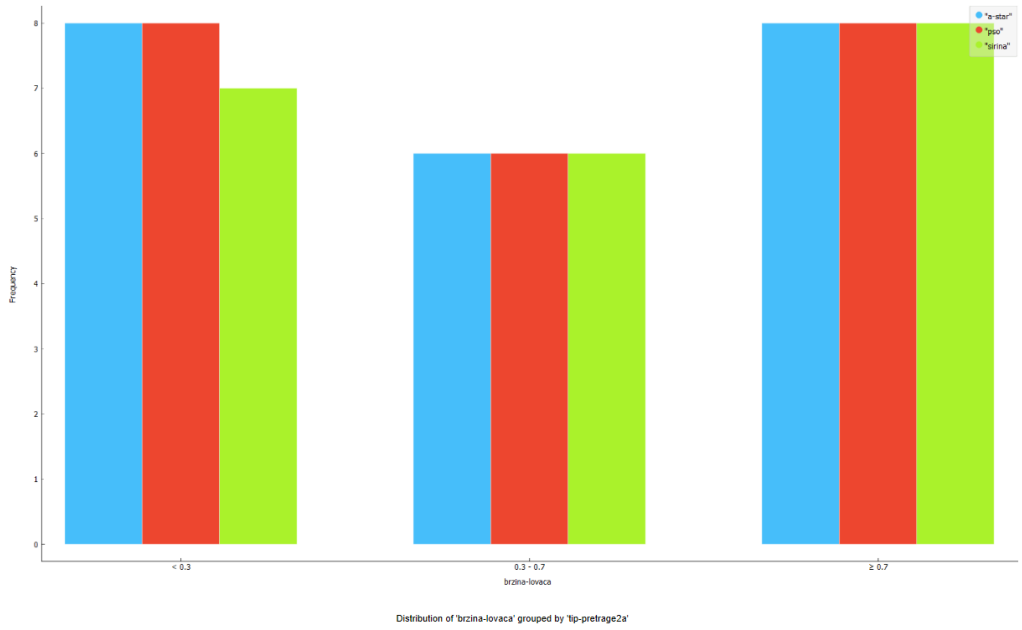


Slika 4. Kraj igre

4. STATISTIČKI PODACI

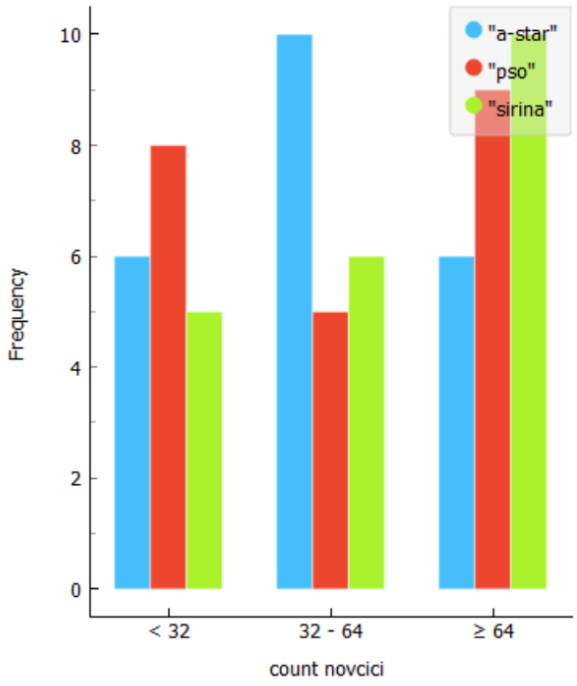
Statistički podaci za ovaj projekt su dobiveni pomoću *BehaviorSpace*-a ugrađenog u NetLogo i programa *Orange* koji je poslužio da se dobiju razni dijagrami.

U *BehaviorSpaceu* se izvrtio zadani eksperiment u kojem su se mijenjale pretraga po širini, a-star pretraga te pretraga po PSO algoritmu i uz njih su se mijenjale vrijednosti *slidera* unutar projekta kao što su *zeleni-br-zivota* te *brzina-lovaca*. Sve je to spremljeno u jednu *.csv* datoteku koja se poslije očitala pomoću programa *Orange* te su dobiveni sljedeći dijagrami:



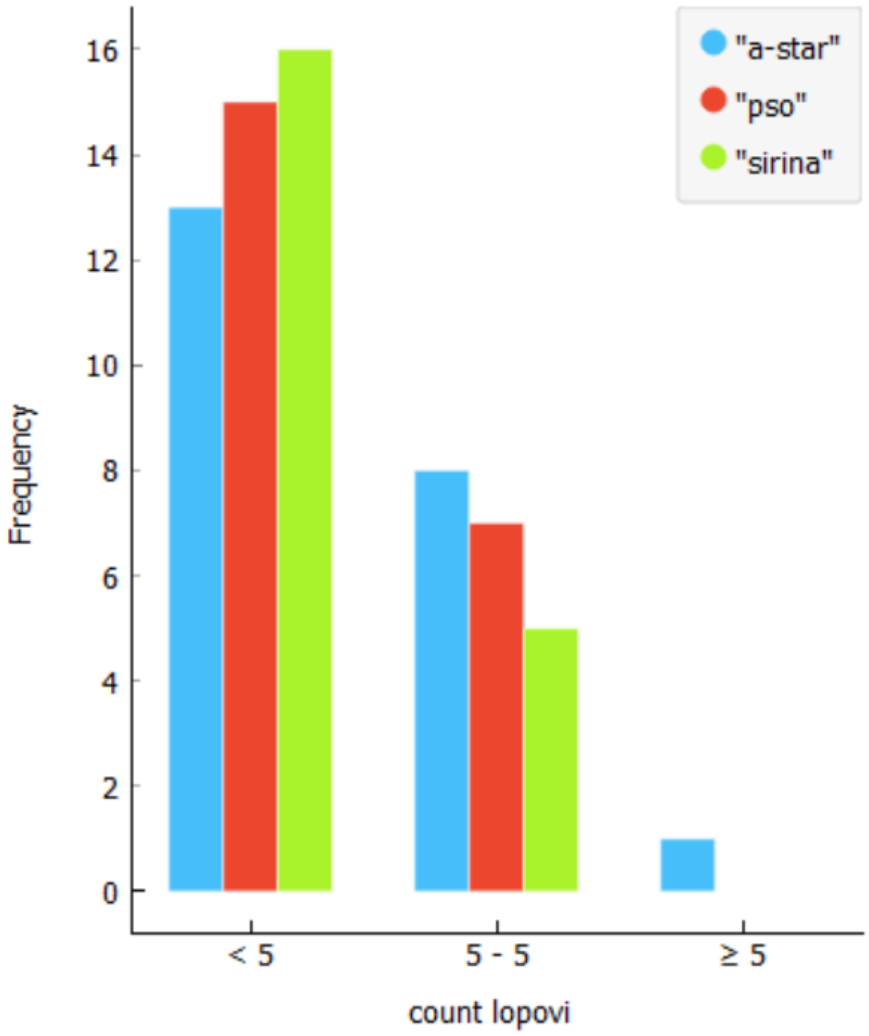
Slika 1 - Dijagram distribucije 1

Ovaj dijagram nam govori ako je brzina lovaca manja od 0.3 da je tada najlošiji algoritam pretrage po širini, a a-star i PSO su jednako dobri. Za brzine veće od 0.3 svi su algoritmi jednako učinkoviti.



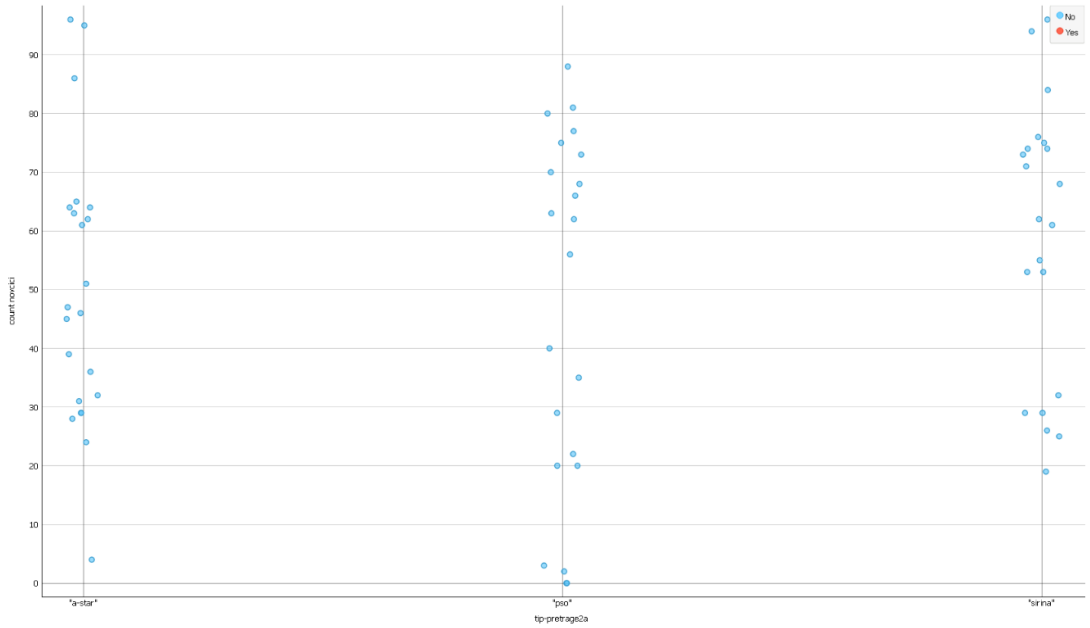
Slika 2 - Dijagram distribucije 2

Ovdje možemo vidjeti da kada imamo manje od 32 novčića najbolju pretragu nam daje PSO algoritam, a najlošiju algoritam pretrage po širini. Kad je broj novčića između 32 i 64 tada je najbolje koristiti a-star pretragu, a kao najlošiji se pokaže PSO algoritam. Kad je broj novčića veći od 64 stvari se drastično mijenjaju, tada se kao najbolji algoritam pokazuje algoritam pretrage po širini, a kao najlošiji a-star.



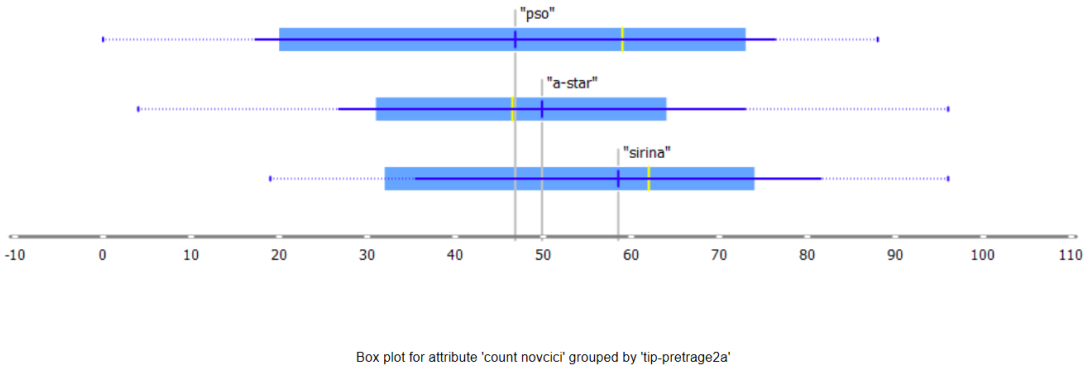
Slika 3 - Dijagram distribucije 3

Ovdje možemo vidjeti da kada se pretražuje PSO algoritmom ili algoritmom po širini nikad broj lopova neće biti veći od 5, dok kod a-star algoritma to nije slučaj.



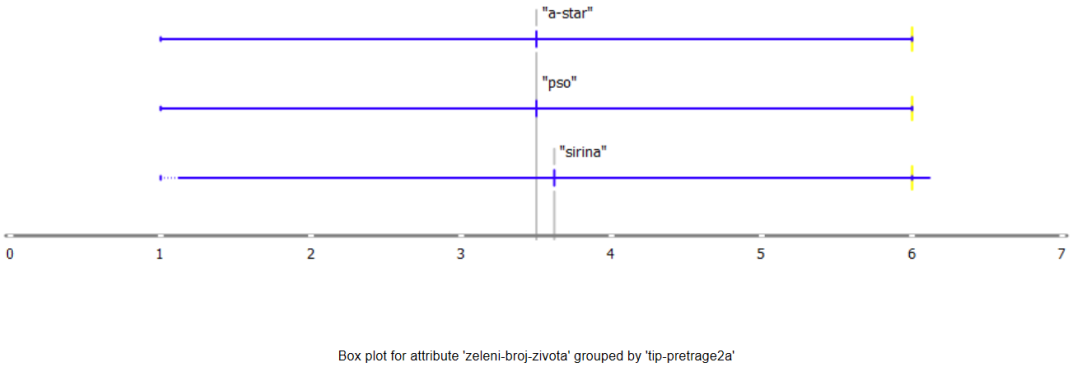
Slika 4 - Scatter dijagram 1

Ovaj dijagram nam pokazuje da a-star algoritam najbrže kupi novčiće kada je njihov broj između 60 i 70, dok je raspon kod PSO algoritma između 60 i 80. Algoritam pretrage po širini kupi najbolje kada je broj novčića između 70 i 80.



Slika 5 - Box plot 1

U ovom Box plot-u vidimo da PSO algoritam najučinkovitije radi kada je broj novčića u labirintu između 20 i 75. A-star je najbolji kada je broj novčića između 30 i 65, dok je pretraga po širini najučinkovitija kad je broj novčića između 30 i 75.



Slika 6 - Box plot 2

Ovaj dijagram nam govori kako dobro algoritmi odrade posao u odnosu na početni broj života agenta koji predstavlja lopova. Možemo vidjeti da su svi slični po tom pitanju i uglavnom je najbolje postaviti da lopov ima između 1 i 6 života.