SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2741

SIMULACIJA EKOSUSTAVA

Marin Njirić

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2741

SIMULACIJA EKOSUSTAVA

Marin Njirić

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 11. ožujka 2022.

DIPLOMSKI ZADATAK br. 2741

Pristupnik:	**Marin Njirić (0036506204)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Marin Golub

Zadatak: **Simulacija ekosustava**

Opis zadatka:

Osmisliti ekosustav u umjetnom okolišu s dvije vrste jedinki: s predatorima i plijenom. Programski simulirati suživot predatora i plijena u osmišljenom umjetnom okolišu u okruženju za razvoj računalnih igara Unity. Odrediti skup pravila kretanja plijena i predatora tako da plijen s jedne strane nastoji održati vrstu zadovoljavajući vlastite osnovne potrebe kao što su glad, žeđ i razmnožavanje te izbjegavati predatore, a predatori s druge strane nastoje uloviti plijen da ne bi uginuli od gladi. Prilikom određivanja skupa pravila kretanja uzeti u obzir vlastitu brzinu, vidokrug s ostalim jedinkama, tj. uzeti u obzir i tuđu brzinu i smjer kretanja te poželjnost za reprodukciju s tim da veća brzina kretanja podrazumijeva veću potrošnju energije pa posljedično i veću potrebu za hranom i pićem. Tijekom razmnožavanja jedinki iste vrste parametri jedinki neka se prenose s roditelja na djecu na jednak način kao i kod genetskog algoritma. Grafički prikazati rezultate simulacije tj. održivost pojedine vrste s obzirom na parametre i skup pravila.

Rok za predaju rada: 27. lipnja 2022.

Sadržaj

Uvod

Računalne simulacije koriste matematički opis ili model stvarnog sustava u obliku računalnog programa. Koriste se za istraživanje dinamičkog ponašanja objekata ili sustava ograničenima određenim uvjetima koji se ne bi mogli lako primijeniti u stvarnome životu. Simulacije su posebno korisne jer omogućavaju promatračima da izmjere i predvide kako na funkcioniranje cijelog sustava može utjecati promjena pojedinih komponenti unutar tog sustava.

Ekosustav je geografsko područje gdje biljke, životinje i drugi organizmi, kao i vrijeme i okoliš rade zajedno da stvore svijet života. Ekosustavi se sastoje od živih faktora kao što su biljke i životinje, i od neživih faktora kao što su kamenje, temperatura i vlažnost. Svi faktori međusobno utječu jedni na druge, bilo izravno ili neizravno. Promjena temperature će često utjecati na rast biljaka na primjer. Životinje koje ovise o biljkama za hranu i zaklon, moraju se prilagoditi promjenama, preseliti u drugi ekosustav ili izumrijeti.

Ovaj rad bavi se simulacijom ekosustava ostvarenom unutar računalnog okruženja Unity. Predstavljene su dvije vrste jedinki, predator i plijen. Zečevi su predstavnici plijena koji nastoji preživjeti i održati vrstu istovremeno bježeći od predatora, lisica, i zadovoljavajući životne potrebe za hranom i pićem kao i potrebe za reprodukcijom. Lisice nastoje uloviti zečeve jer su im oni jedini izvor hrane u ovom okruženju. Parametri po kojima se životinje razlikuju su brzina, širina vidokruga i veličina. Ti parametri se prenose na potomke križanjem roditelja uz dodavanje mutacije, kao što je slučaj kod genetskog algoritma.

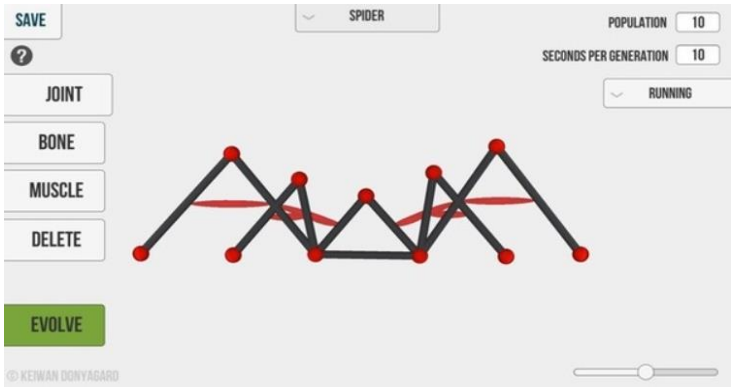
Na početku su opisani već postojeći simulatori života. Nakon toga slijedi opis izrade okoline u kojoj se odvija suživot predatora i plijena. Sljedeće poglavlje se bavi pravilima kretanja životinja i načinom donošenja odluka. Slijede rezultati simulacije pokrenute različitim parametrima. Rezultati su prikazani grafovima koji prikazuju odnos kretanja veličine populacije predatora i plijena kroz vrijeme, kao i kretanje njihovih parametara. Na kraju slijedi zaključak, popis literature i sažetak.

1. Postojeći simulatori

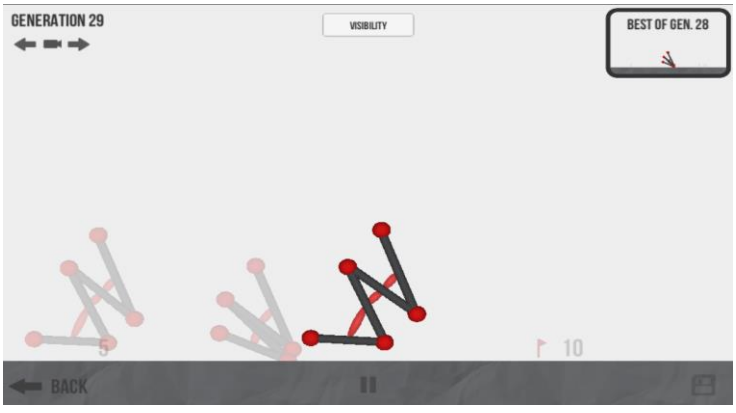
1.1. Mobilna igra Evolution

Evolution je igra u kojoj se stvaraju virtualna stvorenja korištenjem kosti, zglobova i mišića. Učita se jedan od nekoliko dostupnih scenarija gdje stvorenje mora naučiti obaviti neki zadatak kao što je kretanje unaprijed, skakanje ili penjanje uz stepenice. Ovo je omogućeno kroz proces evolucije unutar neuronskih mreža. Stvori se određeni broj jedinki, svaka sa neuronskom mrežom malo modificiranom u odnosu na prošlu. Nakon unaprijed određenog vremenskog perioda odabire se stvorenje koje ima najbolji rezultat na danom zadatku. Zatim se stvori nova generacija njezinim kloniranjem uz manje mutacije. Ciklus se zatim ponavlja.

Na sljedeće dvije slike prikazan je izgled ekrana u kojem se može oblikovati stvorenje, kao i proces treniranja populacije.



Slika 1.1. Konfiguracija pauka



Slika 1.2. Treniranje populacije

1.2. Računalna igra Species ALRE

Species: Artificial Life, Real Evolution je znanstveno točna video igra prirodne selekcije. Omogućava da se iskusi evolucija simulirana kroz sljedeće principe:

- varijacija – svako stvorenje je jedinstveno određeno svojim genima;
- mutacija – stvorenje dijete je nasumična modifikacija svojih roditelja;
- prirodna selekcija – okolina utječe na svako stvorenje u njihovoj borbi za preživljavanje.

Ovim mehanizmom igre vođenim navedenim principima, Species ponovno stvara uvjete koji su doveli do razvoja života na Zemlji u posljednjih 4 milijarde godina. Igru se može igrati tako da se samo promatra, gleda i proučava nastanak stabla života od jedne jedinke, a može se i aktivno utjecati na razvoj vrsta unutar igre.

Slika 1.3. Svojstva vrste

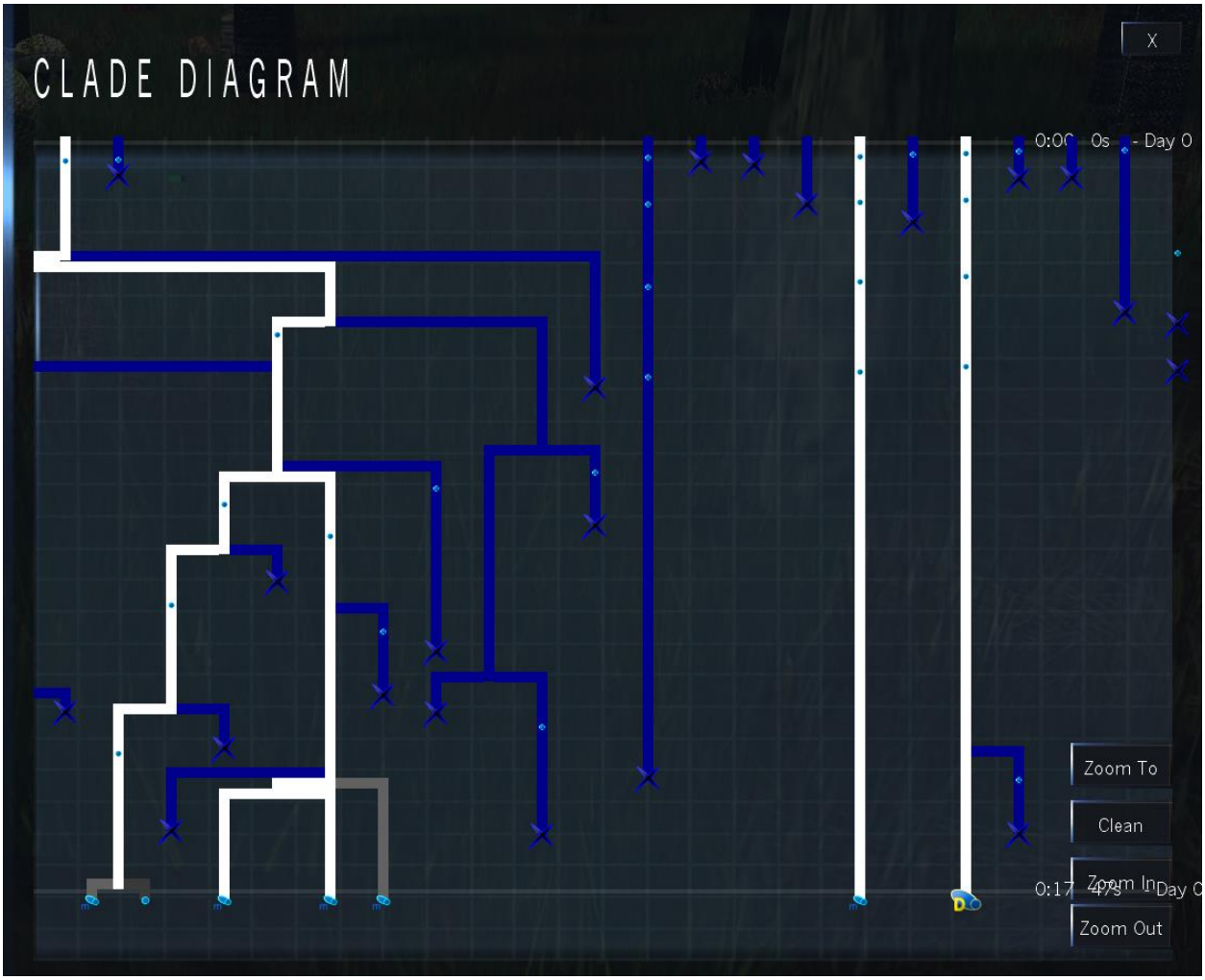
Igra daje vrlo detaljan opis svake vrste na fizičkoj i psihološkoj razini. Tako na primjer možemo vidjeti razinu zdravlja, energije i izdržljivosti kao i razinu proždrljivosti, lijenosti, agresivnosti, osjetljivosti na klimu i ostale podatke. Također je prikazan i genetski kod vrste kao niz ATCG nukleotida.

U igri postoji i uzgajalište (*eng. nursery*) u kojem je moguće evoluirati vrste. Jedinka se dovuče u prostor uzgajališta i odrede se poželjna svojstva te vrste kao što je tip kože, broj udova, veličina, agresivnost, osjetljivost i ostale. Unutar uzgajališta primijenjena je jača radijacija nego u izvan njega. Simulator uklanja one jedinke koje imaju lošija svojstva (manje slične željenoj vrsti), a ostavlja jedinke koje evoluiraju prema određenim svojstvima.

Slika 1.4. Uzgajalište

Osim na svojstva pojedinih vrsta, moguće je utjecati i na svojstva okoliša. Tako je omogućeno mijenjanje temperature tla, razine vode te plodnosti zemlje i vode. Te promjene se ne događaju odjednom već postupno kroz generacije. Može se postaviti i uređaj za kontrolu klime koji služi za grijanje, hlađenje ili fertilizaciju lokalnog područja i ne utječe na ostatak mape. U igri postoje mesojedi, biljojedi i svejedi. Moguće je i postavljanje ograde tako da bi se zaštitilo neke vrste od predatora.

Još jedan zanimljiv detalj igre je kladogram (*eng. Clade diagram*) koji pokazuje veze među organizmima. Može se proučiti svaki korak evolucije u detalje korištenjem ovog dijagrama. Kladogram koristi linije koje se granaju u različitim smjerovima i završavaju na grupama organizama. Plavim linijama na slici 1.5. prikazane su vrste koje su izumrle u procesu evolucije, a vrste koje su preživjele su prikazane bijelim linijama.



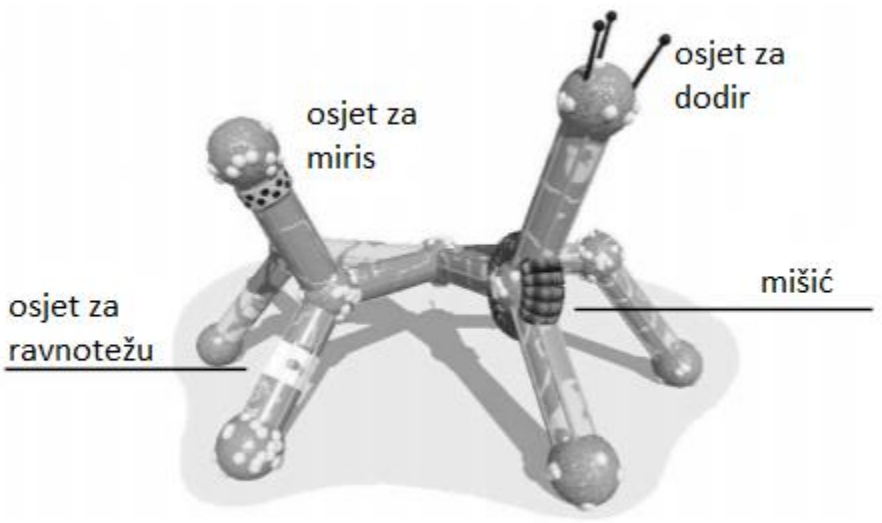
Slika 1.5. Kladogram

1.3. Simulator Framsticks

Framsticks simulira 3D svijet i stvorenja koja su modelirana mehaničkim strukturama (tijelo) i kontrolnim sustavima (mozak). Razni oblici interakcije između fizičkih objekata se uzimaju u obzir: statičko i dinamičko trenje, sile akcije i reakcije, gubitak energije pri deformacijama, gravitacija, pritisak, uzgon i ostali.

Uvijek postoji kompromis između točnosti simulacije i vremena izvođenja simulacije. Potrebna je brza simulacija za izvođenje evolucije, a s druge strane sustav bi trebao biti što detaljniji da bi proizveo realistična ponašanja.

Tijelo stvorenja sastoji se od materijalnih dijelova (štapića) povezanih elastičnim zglobovima. Mozak je napravljen od neurona i neuronskih veza. Postoje dva tipa mišića: savijajući i rotirajući. Promjene signala u kontroliranju mišića pomiču stvorenje u nekom smjeru. Snaga mišića određuje efektivnu sposobnost kretanja i brzine. Jedan primjer takvog stvorenja s mišićima i receptorima prikazan je na sljedećoj slici.

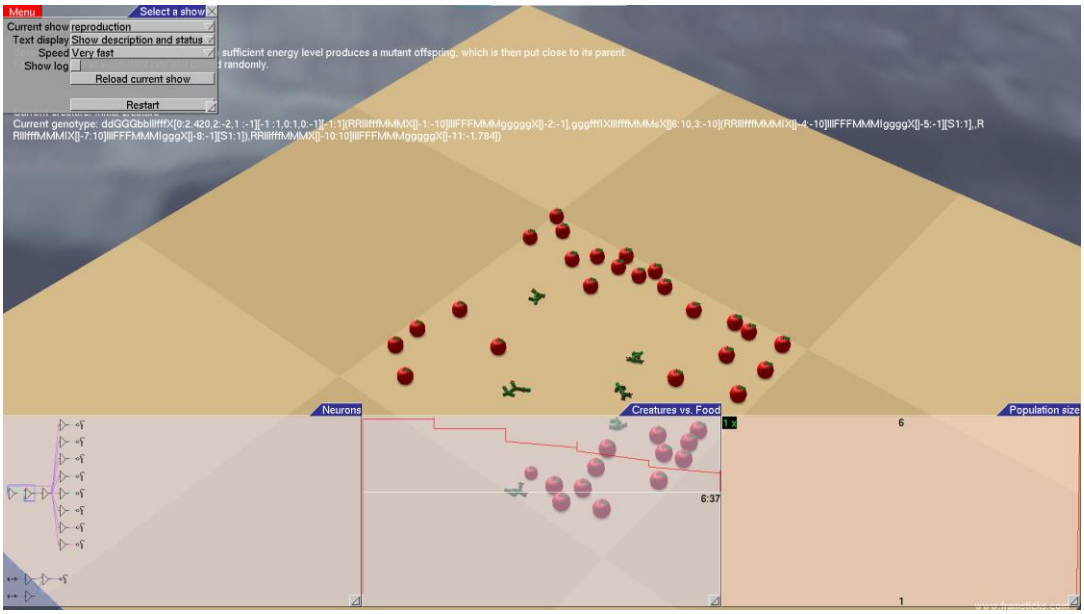


Slika 1.6. Receptori i mišići u Framsticksu

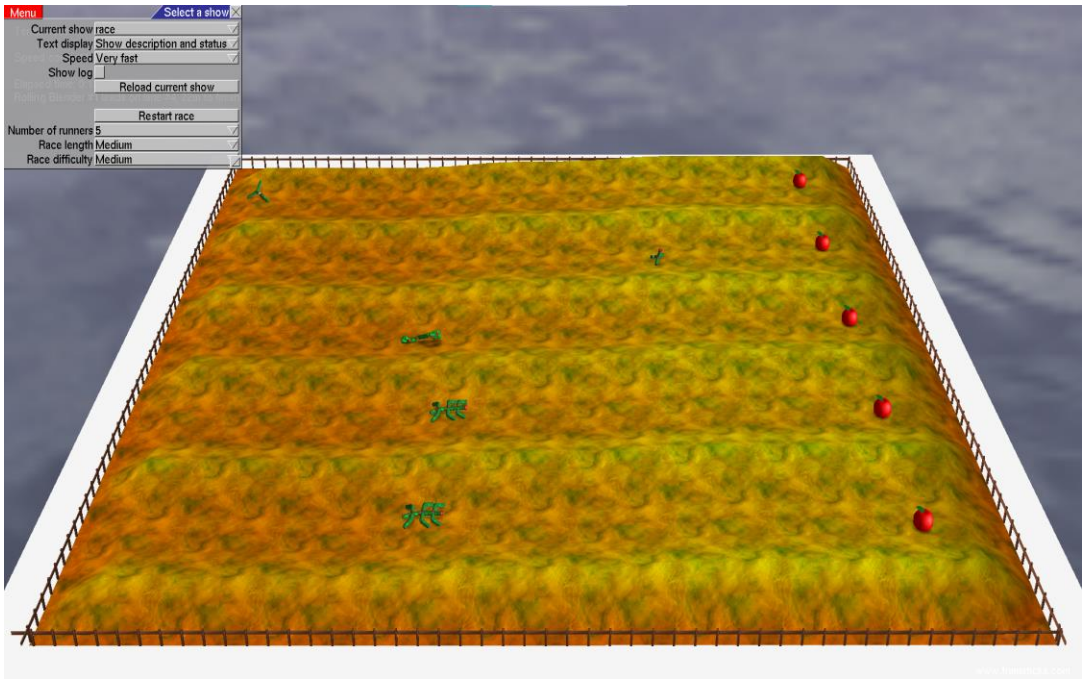
Programski sustav Framsticks Theater prikazuje neke osnovne pojave kao što su genetika, mutacija, evolucija, umjetna selekcija, hodanje, plivanje, virtualne interakcije i druge. Sadrži više različitih „predstava“ i korisnik sam može dodati nove.

Može se koristiti u edukacijske svrhe (npr. biologija, optimizacija, simulacija, robotika) kao i za razne ilustracije, atraktivne grafičke pozadine, reklamiranje, zabavu itd.

Dvije takve predstave prikazane su na slikama 1.7. i 1.8.



Slika 1.7. Predstava reprodukcija



Slika 1.8. Predstava utrka

2. Okruženje za simulaciju

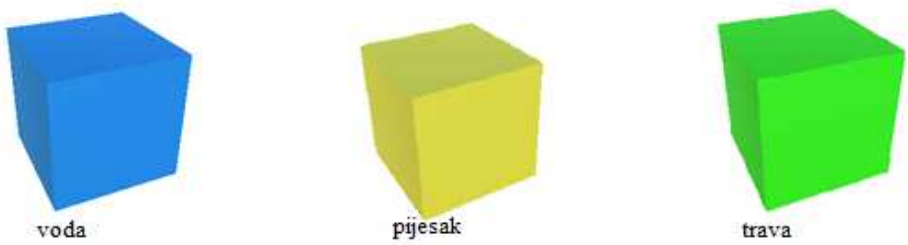
Za ostvarenje simulacije suživota predatora i plijena, najprije je potrebno osmisliti okruženje u kojemu će se ta simulacija izvršavati. Za obje vrste jedinki, žeđ je osnovna životna potreba koju moraju zadovoljiti da bi preživjeli. U tu svrhu, jedan dio okoliša je voda u obliku jezera ili rijeka bez kojih život u ovoj simulaciji ne bi bio moguć. Ti vodeni dijelovi su okruženi pješčanim pojasom koji služi kao prijelaz između vode i trave. Na travnatim dijelovima rastu biljke koje zečevima predstavljaju izvor hrane. Uz biljke, po okolišu su rasprostranjene i različite vrste stabala, grmova, kamenja i raznih dekoracija. Osim za ukras, neke od njih služe i kao prepreke za životinje koje trebaju paziti na njih pri kretanju.

Određivanje vrste tla na pojedinim mjestima na mapi vrši se uz pomoć algoritma *Perlin noise*. To je algoritam koji proizvodi pseudo-slučajne brojeve i ima mogućnost stvaranja prirodnog izgleda objekata kao što su oblaci, krajolici i uzorkovane teksture kao što je emramor. Nasumični generatori proizvode brojeve koji nemaju nikakve poveznice jedni s drugim. *Perlin noise* stvara prirodno poredane sekvence brojeva koji rezultiraju zaglađenijim prijelazima kakvi se mogu očekivati u prirodi.

Za prikaz željenog okoliša, predstavljena su tri načina njegovog stvaranja. Treći način se pokazao najboljim i najprikladnijim za provedbu simulacije.

2.1. Prvi okoliš

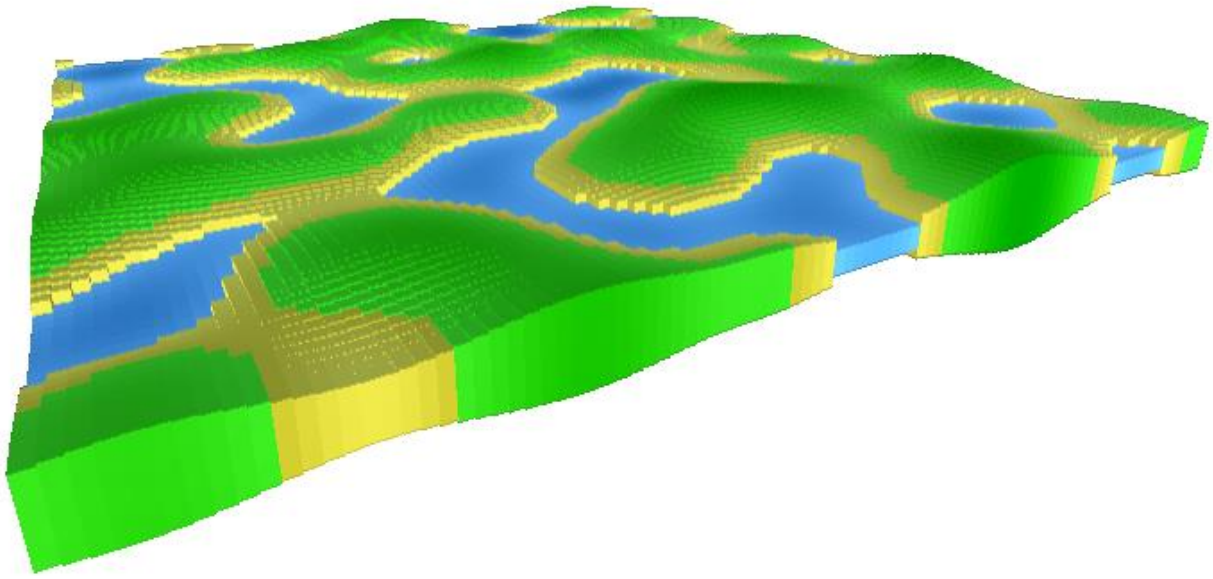
Kao prvi pokušaj stvaranja okoliša, nametnula se ideja generiranja mape korištenjem tri vrste blokova (kockica), po jedan za svaki tip tla, vodu, pijesak i travu. Mapa je zamišljena kao mreža od 100 * 100 blokova čiji je tip određen pomoću *Perlin noise*-a. Tipovi blokova prikazani su na sljedećoj slici.



Slika 2.1 Tipovi blokova u prvom okolišu

U ovisnosti o iznosu *Perlin noise*-a za svako mjesto na mapi, pojedini blok se skalira u vertikalnom smjeru tako da se stvori dojam trodimenzionalnosti. Vodeni blokovi se ne skaliraju tako da razina vode ostane konstantna na cijeloj mapi. Iako je ovaj način generiranja terena dao dobre vizualne rezultate, po performansama se pokazao vrlo lošim. Održavanje 10000 blokova na mapi se pokazalo računalno zahtjevnim i razina FPS padne ispod 10 čak prije izvođenja ikakvih simulacija i dodavanja ostalih objekata unutar scene.

Ovaj primjer pokazao je da će biti potrebno koristiti druge metode za stvaranje terena za ovu simulaciju. Rješenje je pronađeno korištenjem *shadera*, programa koji uzima razne oblike i teksture kao ulaz i stvara sliku kao izlaz.



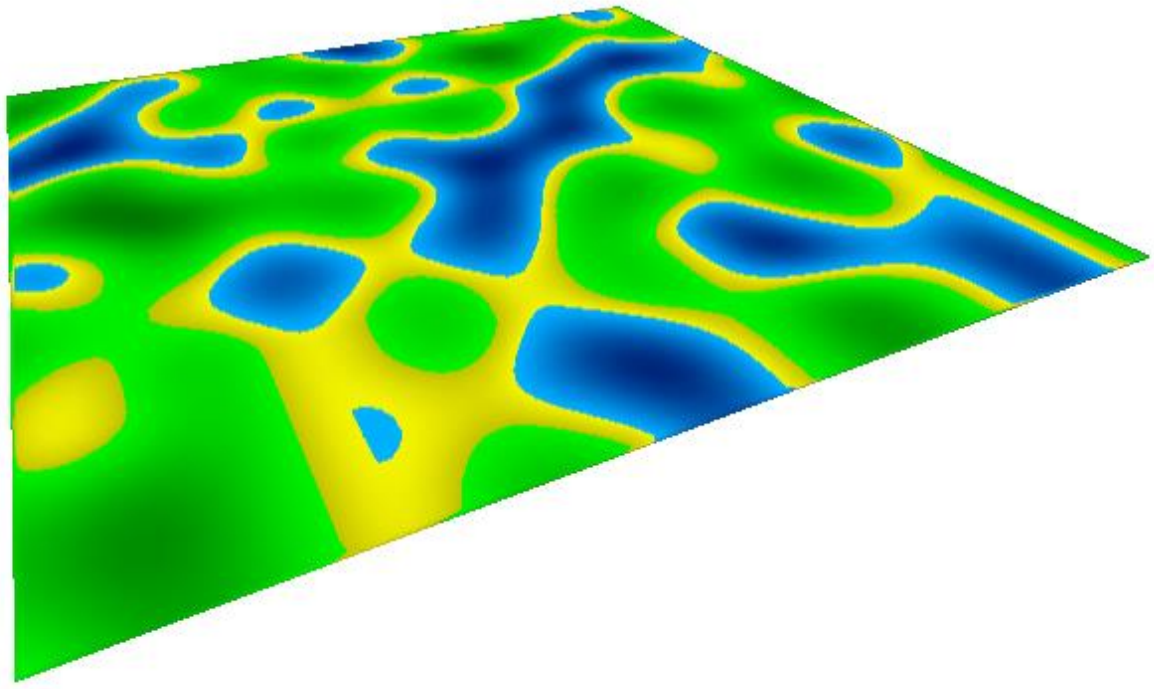
Slika 2.2. Prvi okoliš

2.2 Drugi okoliš

Korištenje *shadera* za renderiranje pokazalo se puno učinkovitijim što se odmah vidjelo po razini FPS-a koja prelazi 100. *Shaderi* izračunavaju prikladne razine osvjetljenja, sjene i boje unutar 3D scene u procesu zvanom sjenčanje. Oni su jednostavni programi koji opisuju značajke vrhova ili točaka. Svaki vrh se renderira kao niz točaka na površinu (blok memorije) koji će biti iscrtan na ekranu.

Iako ovaj okoliš ima puno bolje rezultate po performansama, vizualno daje lošije rezultate. Okoliš izgleda neprirodno jer se sve nalazi u ravnini. Prirodniji izgled bi se dobio uzdizanjem travnatih dijelova stvarajući dojam brežuljaka i na taj način dobivanje 3D izgleda.

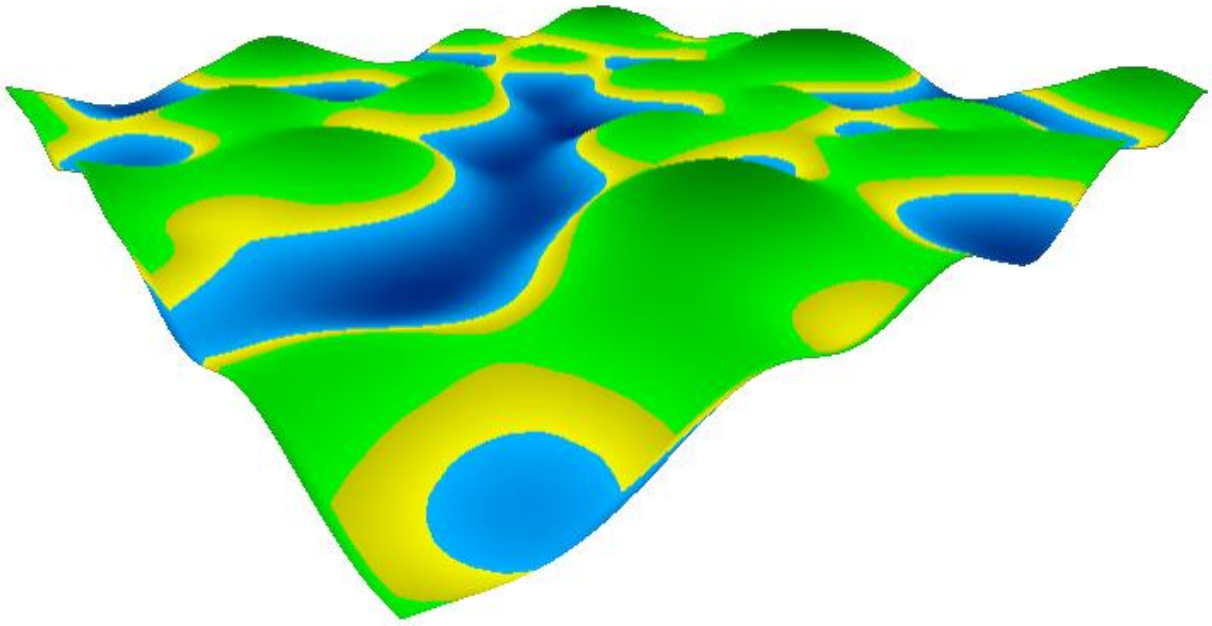
Najprikladniji izgled okoliša dobit će se kombiniranjem ova dva primjera. Stvaranje 3D mreže vrhova za bolji vizualni dojam uz korištenje *shadera* za postizanje boljih performansi.



Slika 2.3. Drugi okoliš

2.3. Treći okoliš

Za konačni izgled okoliša najprije je bilo potrebno izgraditi mrežu vrhova (*eng. mesh*). Mreža se sastoji od 100 * 100 vrhova pravilno raspoređenih u rešetkastu mrežu u XZ ravnini. Iznos Y komponente određen je uz pomoć *Perlin noise*-a. Vrijednosti te funkcije koje su bliže 1 predstavljaju više dijelove mape (brežuljci), a vrijednosti bliže 0 su niži, odnosno dublji dijelovi (jezera). Nad stvorenom mrežom zatim se koristi *shader* kao i u prethodnom primjeru. Ovakav način generiranja okoliša daje vrlo dobre rezultate prikazane na sljedećoj slici.

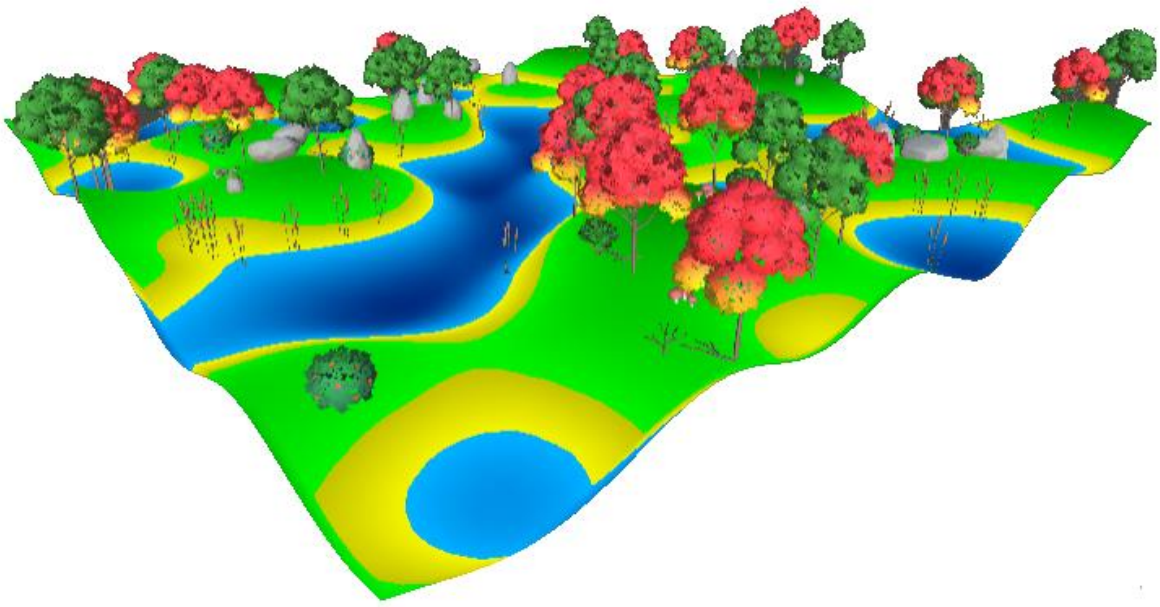


Slika 2.4. Treći okoliš

Za postizanje kompletnog izgleda okoliša, sljedeći korak koji je potreban je postavljanje dekoracija. One imaju određen interval visina na kojima se mogu stvarati. On je uglavnom za sve jednak (zeleni dijelovi mape), osim za trsku koja se stvara samo uz područja s vodom. Za simulaciju su izabrane dvije vrste stabala i moguće je unutar Unity editora odrediti željeni broj stabala u okolišu. Isto vrijedi i za broj trski. Ostale dekoracije uključuju razne vrste kamenja, grana, grmova i gljiva. Korisniku je omogućeno odrediti ukupan broj tih dekoracija, a broj pojedinih vrsta određen je nasumično.

Jedan primjer takvog okoliša s postavljenim dekoracijama prikazan je na slici ispod. Svakim pokretanjem programa generira se novi okoliš s različitim rasporedom voda, brežuljaka, stabala i ostalih dekoracija. Ovo predstavlja dobru podlogu za simulaciju suživota plijena i predatora.

Neke vrste dekoracija (stabla i veća kamenja) imaju definiran i vlastiti sudarač (*eng. collider*). Životinje ga pri kretanju moraju uzeti u obzir jer ne mogu prolaziti kroz njega. Time te dekoracije nemaju samo ulogu ukrasa u okolišu, već i prepreka za kretanje životinja.



Slika 2.5. Okoliš s dekoracijama

3. Svojstva subjekata i objekata u okolišu

Glavni cilj zečeva i lisica je preživljavanje u danom okolišu i razmnožavanje, odnosno prenošenje genetskog materijala na buduće generacije. Životinje moraju zadovoljiti osnovne životne potrebe, glad i žeđ, prije nego uzmu u obzir potrebu za reprodukcijom. Simulira se mali hranidbeni lanac. Na dnu hijerarhije su biljke koje služe kao hrana zečevima, a ti zečevi su hrana lisicama.

Najprije je objašnjeno početno postavljanje i daljnje generiranje biljaka, zatim ponašanje zečeva, i na kraju ponašanje lisica.

3.1. Biljke

Za stvaranje i upravljanje svim biljkama u okolišu zadužena je skripta *PlantsManager*. U editoru se definira početni željeni broj biljaka na mapi. Prije početka simulacije, unutar *Start* funkcije stvaraju se biljke na nasumičnim pozicijama na mapi uz uvjet da to moraju biti travnata područja. Ostale biljke na sličan način se stvaraju za vrijeme simulacije. Korisnik može definirati vremenski raspon unutar kojeg želi da se biljke stvaraju (npr. sljedeća biljka se pojavi nakon nasumičnog broja sekundi između 1 i 5). Nakon *Start* funkcije pozove se funkcija *GrowAPlant* koja rekurzivno zove samu sebe tijekom simulacije, pričeka određeno vrijeme i stvori novu biljku. Model biljke prikazan je na sljedećoj slici.

Slika 3.1. Model biljke

Nije nužno da će svaka biljka koja se generira uspjeti. Nekad se nasumičnim postavljanjem biljke može dogoditi da se ona stvori na nekim nedostupnim mjestima (npr. ispod, odnosno unutar velikog kamena). To se provjerava slanjem zrake iz pozicije biljke u vertikalnom smjeru (*eng. raycasting*). Ako ta zraka pogodi određene objekte (sloj dekoracija), ta biljka se uništava.

Ponašanje pojedinog primjerka biljke određeno je skriptom *PlantController*. Glavna funkcionalnost opisana je unutar funkcije *OnTriggerEnter* koja se poziva kad dođe do sudara s drugim objektom. U toj funkciji provjerava li se je li taj objekt zec koji je u tom trenutku u potrazi za hranom. Ako jest, tom zecu se na određeni način smanjuje potreba za hranom, a biljci se pokrene animacija nestajanja. Nakon što animacija završi, objekt se uništava. Tijekom simulacije, *PlantsManager* skripta čuva sve aktivne biljke u listi i prati kretanje broja biljaka na mapi.

3.2. Zečevi

RabbitManager je skripta koja je zadužena za upravljanje populacijom zečeva u okolišu. Unutar nje se zadaje početni broj zečeva, čuvaju se podaci o svim aktivnim zečevima i pamti se broj smrti zečeva od različitih uzroka (glad, žeđ ili lisica). Kako bi se razlikovali u okolišu, muški i ženski zečevi su predstavljeni s dva različita modela prikazana na sljedećoj slici.

Slika 3.2. Modeli zečeva

Parametri po kojima se zečevi razlikuju su brzina, širina vidokruga i veličina. Veća brzina omogućava dolazak do hrane prije ostalih i veću vjerojatnost za bježanje od lisice. Širina vidokruga je radijus kugle koja predstavlja dio okoline koje je zec svjestan oko sebe. Objekte koji se nalaze izvan te kugle zec ne vidi. Veličina tijela nema posebnu funkcionalnost u danom okolišu. Jedina svrha joj može biti veća ili manja poželjnost za reprodukciju koja ovisi od jedinke do jedinke.

Navedeni parametri utječu na brzinu potrošnje energije zeca. Što je veća potrošnja energije, brže će rasti potreba za hranom i vodom. Ovisnost brzine zeca i potrošnje energije je kvadratna, a za parametre širine vidokruga i veličine zeca je linearna. Za svakog zeca je brzina kojom raste potreba za vodom dvostruko veća od brzine rasta potrebe za hranom.

Pri stvaranju početne populacije, polovica zadanog početnog broja zečeva se popuni muškim zečevima, a druga polovica ženskim. Svaki od modela ima unaprijed određene iznose brzine, širine vidokruga i veličine. Prije početka simulacije uvodi se određena doza mutacije za te parametre kako bi populacija bila raznolikija. Brzina i širina vidokruga se promijene na nasumičan iznos između 90% i 110% zadanog, a veličina zeca se promjeni za nasumičan iznos između -5% i +5% od početne veličine.

Ponašanje pojedinog zeca je početno bilo zamišljeno unutar skripte *RabbitController*. U toj skripti trebalo je biti definirano cjelokupno ponašanje zeca u svakoj situaciji. Takav pristup pokazao se previše složenim i neskalabilnim. Stvorila se potreba za korištenjem velikog broja zastavica (zec traži hranu, zec se kreće prema vodi, zec bježi od lisice…) koje je postalo teško popratiti. Uz to, dodavanje svake nove funkcionalnosti za zeca bi bilo sve složenije jer bi se za svaku novu promjenu ponašanja trebala provjeravati dodatna zastavica.

Rješenje je pronađeno u korištenju oblikovnog obrasca stanje. Obrazac stanje je blizak konceptu stroja s konačnim brojem stanja. Glavna ideja je da u svakom trenutku postoji konačan broj stanja u kojemu se program može nalaziti. Unutar svakog stanja, program se ponaša drugačije i promjena s jednog stanja na drugo se može dogoditi u trenutku. U ovisnosti o trenutnom stanju, program može, ali i ne mora se prebaciti na neko od drugih stanja. Ti prijelazi su također konačni i unaprijed određeni.

Glavna skripta koja upravlja stanjima je *AnimalStateManager*. Ona sadrži varijablu koja predstavlja trenutno stanje, *currentState,* i obavlja prijelaze iz jednog stanja u drugo u ovisnosti o zadanim pravilima. Tu se također računaju iznosi razine gladi, žeđi i potrebe za reprodukcijom. Za svaku životinju se tijekom simulacije prikazuju trenutne razine tih potreba, kao i stanje u kojem se trenutno nalazi.

Slika 3.3. Prikaz statusa zeca

Svako konkretno stanje definirano je u svojoj skripti koja je zadužena za obavljanje tih specifičnih funkcija. Ta stanja su:

- stanje potrage za hranom,
- stanje potrage za vodom,
- stanje traženja partnera i
- stanje bježanja od predatora.

Unutar glavne skripte (*AnimalStateManager*) se u svakom vremenskom okviru (*eng*. *frame*) izračunava trenutno prioritetno stanje i ovisno o raznim uvjetima obavlja se prijelaz u to stanje. Glavni parametar za izračun prioritetnog stanja je razina svake pojedine potrebe. Pseudokod te funkcije izgleda ovako:

```
ako u blizini su predatori, a glad i žeđ su < 90% {
	vrati stanje bježanja
}
ako glad ili žeđ > 70% {
	vrati stanje s većom potrebom
}
ako je muško i spreman za pronalazak partnera i spreman za parenje {
	vrati stanje traženja partnera
}
inače vrati prioritetnije stanje između potrage za hranom ili vodom
```

Slika 3.4. Pseudokod za izračun prioritetnog stanja

Nije nužno da će se odmah uvijek dogoditi prijelaz u prioritetno stanje. S obzirom na to da se pri pijenju vode žeđ smanjuje linearno, prijelaz u npr. stanje potrage za hranom bi se dogodio čim se izjednače potrebe za hranom i pićem što u simulaciji izgleda neprirodno. Zbog toga je dodan uvjet da se promjena iz stanja potrage za vodom izvrši tek nakon što se zec dovoljno napije. U nastavku su opisana pojedina stanja.

3.2.1. Stanje potrage za hranom

Kad se nalazi u stanju potrage za hranom, zec provjerava sve biljke koje se nalaze unutar njegovog vidokruga. Ako ne postoji nijedna biljka koju on može vidjeti, nasumično će se kretati dok se to ne promijeni. Ako postoje biljke do kojih zec može izravno doći, izabrat će onu najbližu i kretati se ravno prema njoj kako je prikazano na sljedećoj slici.

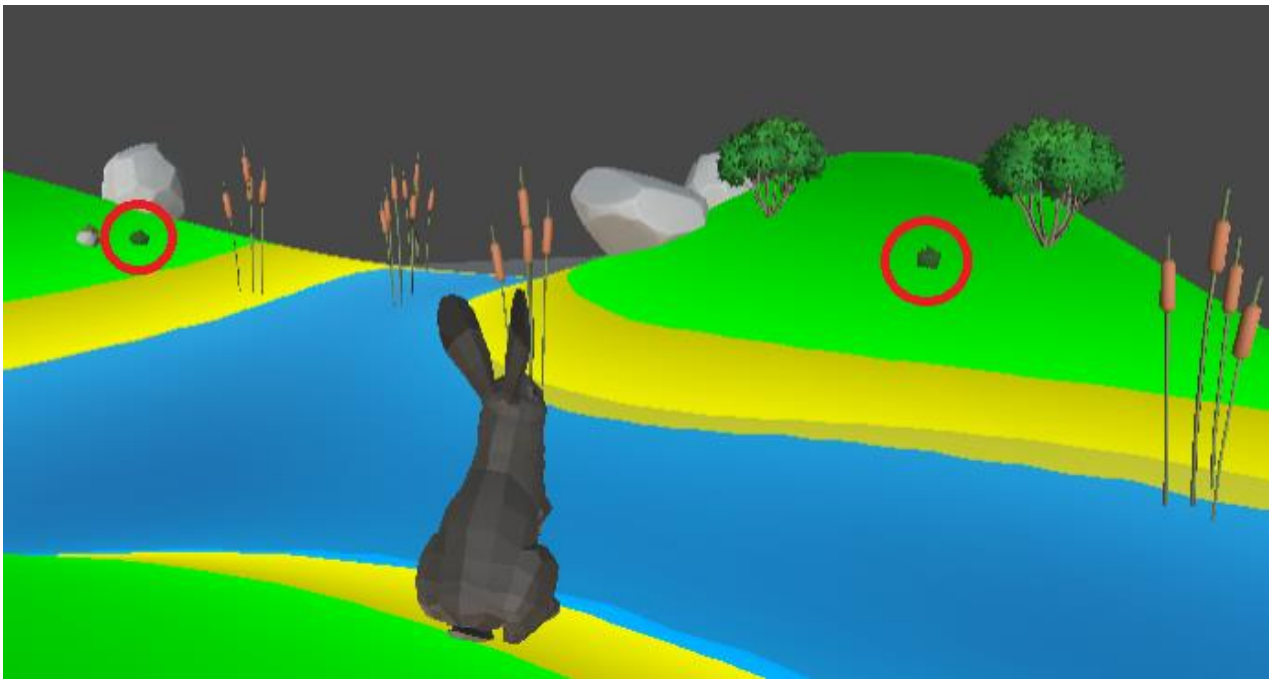
Slika 3.5. Potraga za hranom

Ponekad se može dogoditi da su neke biljke koje su bliže zecu, nedostupne. Zbog toga pri odluci prema kojoj se biljki želi kretati, zec mora uzeti u obzir može li uopće doći do nje. To se provjerava unutar funkcije *CheckIfReachable* gdje se šalju zrake (*raycast*) u rasponu -45 do +45 stupnjeva u odnosu na položaj zeca. Ukoliko se svaka od tih zraka sudara s vodom, zec će označiti gledanu biljku kao nedohvatljivom.

Kada ima skup svih dohvatljivih biljaka unutar svog vidokruga, zec se kreće prema onoj koja mu je najbliža. Često će se dogoditi slučaj to kretanje ne može biti jednostavno pravocrtno, već se mora izračunati smjer kretanja prema toj biljci uzimajući u obzir prepreke kao što su voda, stabla, kamenja i ostali zečevi.

Izračun smjera kretanja prema hrani obavlja funkcija *CalculateMovingDirection* koja prima danog zeca i određenu najbližu biljku. Ako postoji prepreka na izravnom putu između zeca i hrane, ona ispituje pod koji se kutom zec treba kretati kako bi izbjegao prepreke.

Primjer kada zec nije u mogućnosti doći do biljaka koje jesu unutar njegovog vidokruga prikazan je na sljedećoj slici. Takve biljke zec zanemaruje i nastavlja s potragom za drugim biljkama.



Slika 3.6. Nedohvatljive biljke

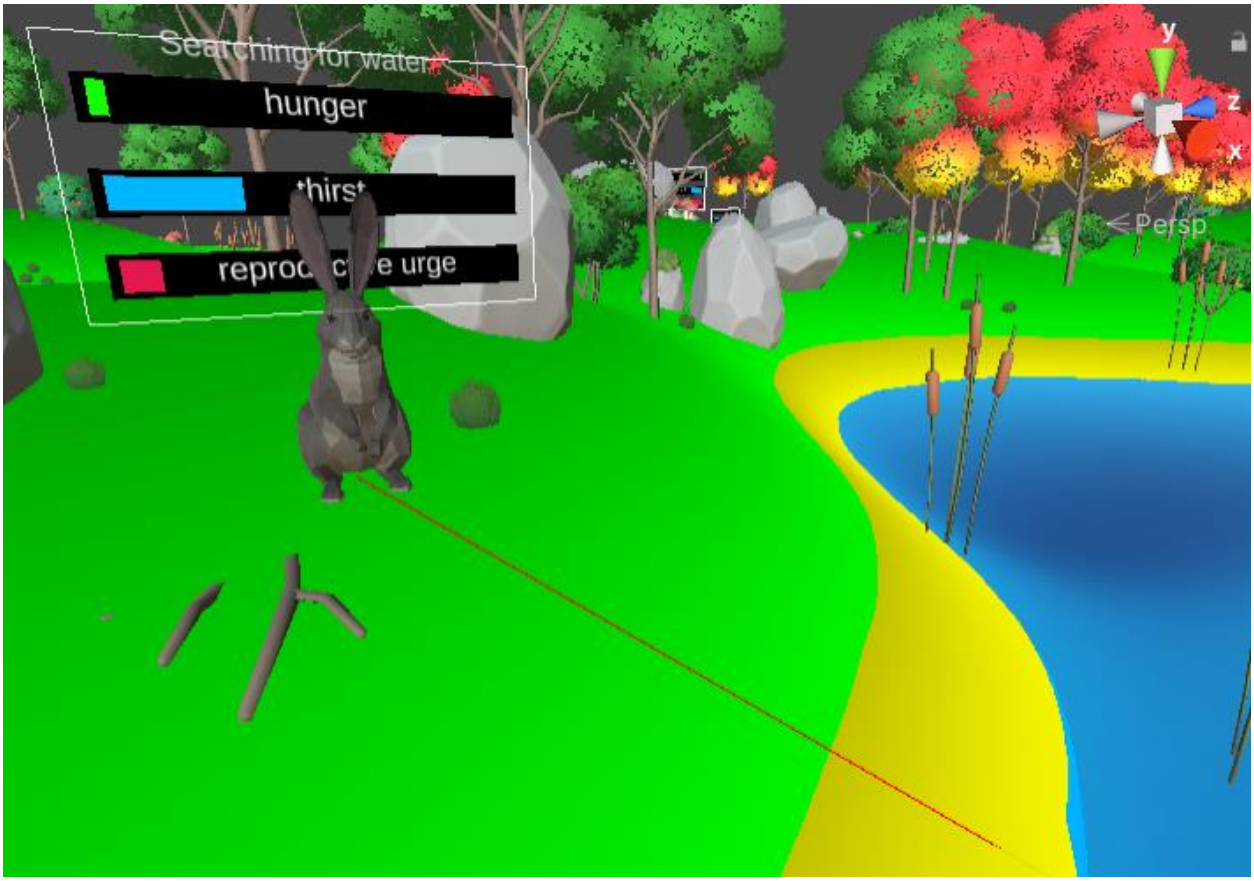
Jednom kad zec dođe do željene hrane, u *AnimalStateManager* skripti pokreće se funkcija *EatFood*. Zec se tada zaustavlja na dvije sekunde kako bi se simulirao proces jedenja hrane i izvršila animacija u kojoj biljka postepeno nestaje. Nakon toga se iznos gladi smanji za nasumični iznos između 50% i 100% od maksimalnog iznosa.

Osim izbjegavanja vode, zec mora paziti i da se ne sudara s ostalim preprekama kao što su dekoracije u okolišu, granica mape i drugi zečevi. Za izbjegavanje granice definirana je funkcija *CalculateBorderDirection* koja pazi da se zec ne kreće prema područjima van mape. Funkcija *Jump* omogućava izbjegavanje sudaranja sa stablima, kamenjima i ostalim zečevima. Unutar nje se odredi nasumični kut smjera u kojem zec skače i doda se impuls sile u tom smjeru. Na taj način može se izbjeći zaglavljivanje zeca na nekom mjestu.

3.2.2. Stanje potrage za vodom

S obzirom na to da je u prosjeku 35% mape prekriveno vodom, zecu ne bi trebao predstavljati velik problem pronalaska mjesta vode. Također, zalihe vode u ovoj simulaciji su neograničene za razliku od hrane te nema nadmetanja s ostalim zečevima tko će prvi doći do nje.

Kad traži vodu, zec najprije odredi poziciju vode koja je najbliža njemu. To se događa unutar funkcije *CalculateWaterDirection* koja šalje zrake u svim smjerovima oko zeca s korakom od 5 stupnjeva. Smjer kretanja je onaj gdje je udaljenost sudara poslane zrake s vodom najmanja. Ako slučajno ne postoji voda u njegovom vidokrugu, zec se kreće nasumično dok se to ne promijeni.



Slika 3.7. Potraga za vodom

Smatra se da je zec došao do vode kada udaljenost njegove pozicije i tražene pozicije vode postane manja od 0.3. Tada se razina žeđi smanjuje 10 puta većom brzinom nego što se inače povećava. Kad se promijeni stanje, brzina kojom se žeđ puni se vraća na iznos koji je određen na početku simulacije.

3.2.3. Stanje traženja partnera

Stanja potrebe za hranom i vodom, kao i stanje bježanja od predatora od važnosti su za preživljavanje pojedinog zeca. Stanje traženja partnera se događa kad se zadovolje te osnovne životne potrebe. Iako je ono manjeg prioriteta za jedinku, ono je od iznimne važnosti za razvoj i održavanje cijele populacije u okolišu.

Potreba za reprodukcijom je parametar svakog zeca koji se vremenom povećava na sličan način kao i glad i žeđ. Razlika je ta što ako taj parametar dosegne maksimalnu vrijednost, zec neće umrijeti kao što je to slučaj kod gladi i žeđi. Parametar potrebe za reprodukcijom odnosi se na vrijeme koje će taj zec provesti u traženju poželjnog partnera za reprodukciju. Što je taj parametar manji, zec će biti izbirljiviji. Kako vrijeme ide, povećava se ta potreba, sve više jedinki će ulaziti u obzir kao poželjni partner.

Svaki zec ima preferenciju kakav želi da mu bude partner po svim parametrima: brzini, širini vidokruga i veličini. Te preferencije određuju se posebno za svakog zeca pri njegovom stvaranju. Za razmnožavanje dva zeca potrebno je da se dogodi sljedeći scenarij. Muški zec u svom vidokrugu primijeti ženskog zeca. Ako ona, uz njegov trenutni iznos potrebe za reprodukciju, zadovoljava njegove preferencije, čeka se njen odgovor na mogućnost parenja. Tek ako i muški zec zadovoljava njezine preferencije, dolazi do razmnožavanja. Navedeni scenarij se odvija po sljedećem pseudokodu za muškog zeca.

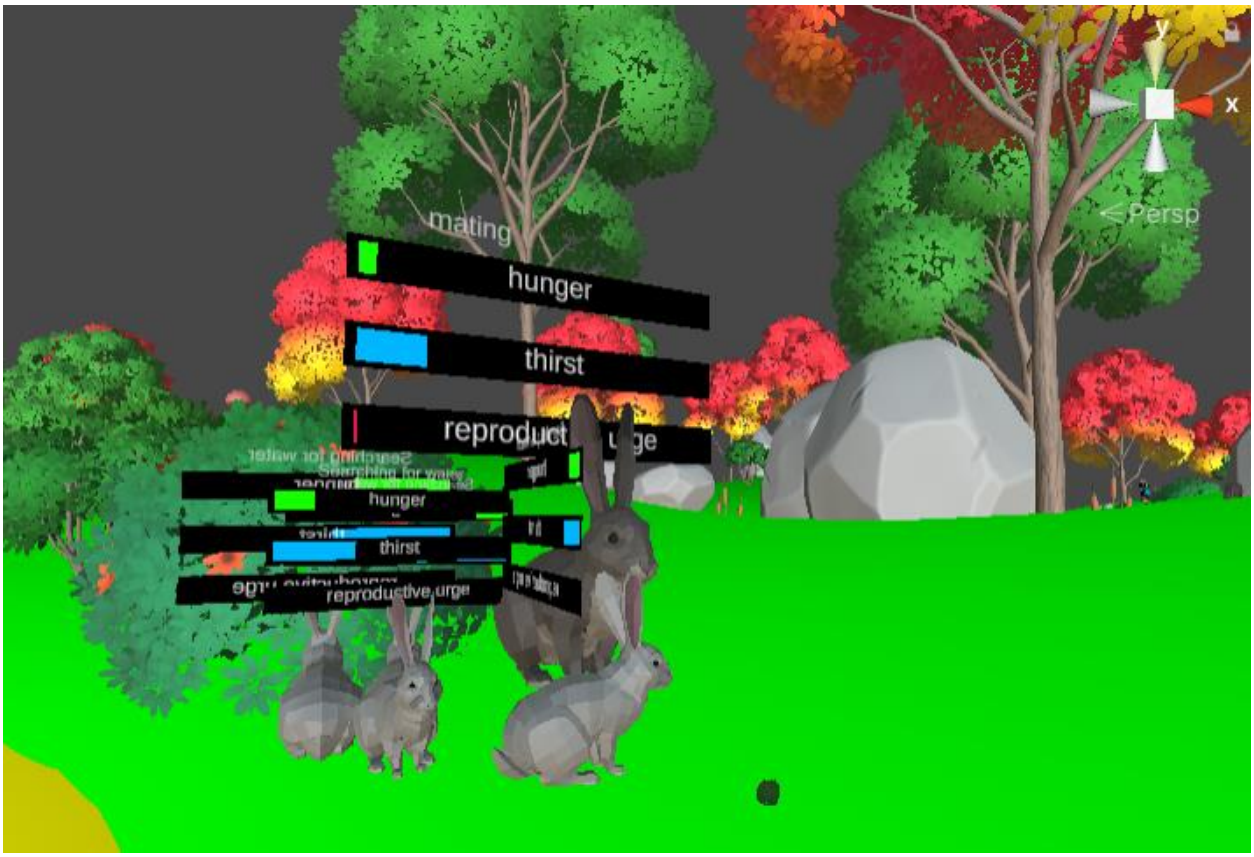
```
za svaki zec u vidokrugu {
	ako to je ženski zec {
		razlika brzine = (preferencija brzine – njena brzina) / preferencija brzine;
		razlika osjeta = (preferencija osjeta - njen osjet) / preferencija osjeta;
		razlika veličine = (preferencija veličine - njena veličina) / preferencija veličine;

		koeficijent poželjnosti = 1 - razlika brzine - razlika osjeta - razlika veličine;
		ako random(0,1) < koeficijent poželjnosti * potreba za reprodukcijom {
			ako ženka odgovori pozitivno
				stvori potomke
	}	}
}
```

Slika 3.8. Pseudokod za ostvarenje reprodukcije

Provjera odgovora ženskog zeca obavlja se na jednak način kao i kod muškog zeca. Kako ne bi neprekidno pitali istog zeca za mogućnost parenja, uveden je oblik „hlađenja“ (*eng. cooldown*). Nakon što muški zec bude odbijen, čeka 5 sekundi prije nego što je spreman tražiti novog partnera.

Slična situacija se događa nakon što se dogodi parenje. Nakon parenja, i muški i ženski zec čekaju određeno vrijeme prije nego su spremni za nove potomke. To vrijeme može se postaviti proizvoljno, a unaprijed određeno je na 30 sekundi.

Stvaranje potomaka odvija se u funkciji *MakeOffsprings*. Za zečeve, najprije se odredi broj potomaka koji je nasumičan broj između 1 i 10. Za svakog potomka, nasumično se odredi hoće li biti muško ili žensko. Odvija se križanje s jednom točkom prekida. Nasumično se odredi koliki postotak će potomak naslijediti od kojeg roditelja pri definiranju svakog parametra. Nakon toga se primijeni mutacija za svaki parametar koja ih promijeni za maksimalno 10%. Zatim se poziva funkcija *RabbitManager*-a koja stvara novi objekt u sceni sa zadanim parametrima i dodaje ga u listu svih zečeva.



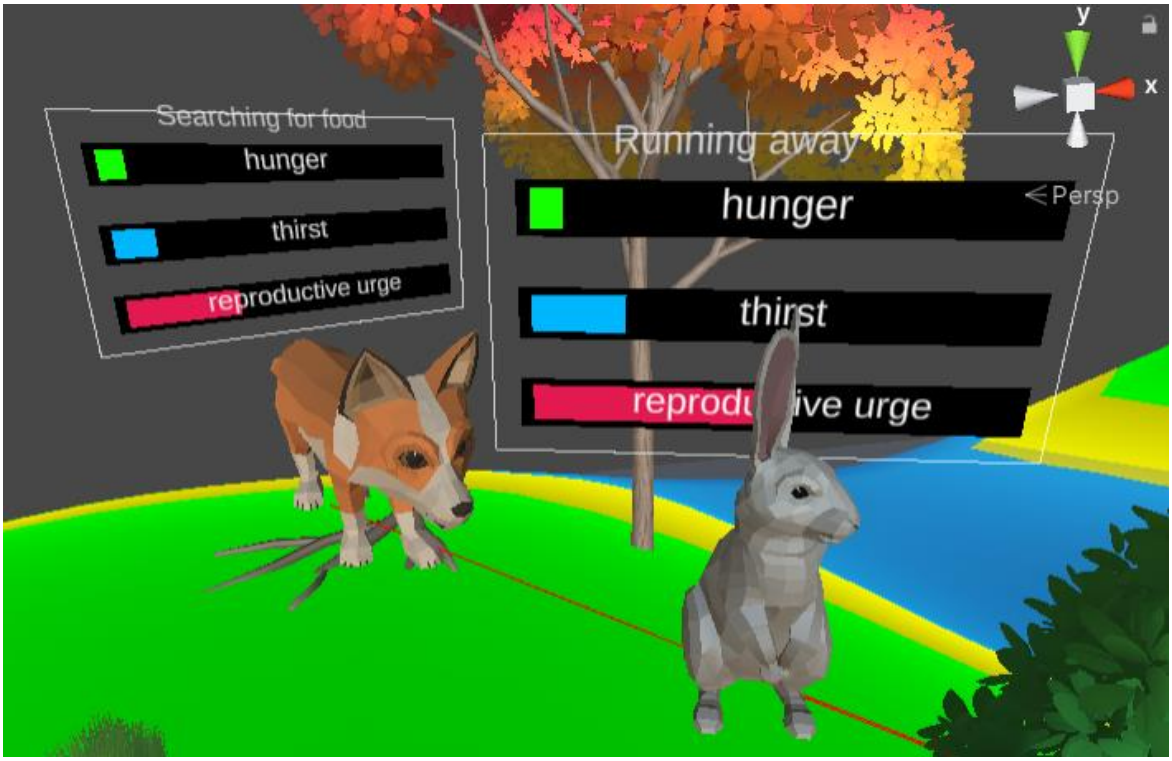
Slika 3.9. Stvaranje potomaka zečeva

Kad se stvore, potrebno je neko vrijeme da se potomci razviju i dosegnu svoj puni kapacitet. To vrijeme postavljeno je na 15 sekundi, ali se također može proizvoljno mijenjati. Dok su u nerazvijenoj fazi, zečevi imaju dvostruko manji volumen, brzinu i širinu vidokruga. Zbog toga mogu postati laka meta za lisice. Također, u toj fazi potreba za reprodukcijom je postavljena na 0 i ne povećava se.

Jednom kad prođe vrijeme razvoja potomka, svi parametri se postavljaju na zadanu vrijednost dobivenu razmnožavanjem (sve se udvostručava). Počinje rasti potreba za reprodukcijom i ti zečevi postaju sposobni za stvaranje novih potomaka.

3.2.4. Stanje bježanja od predatora

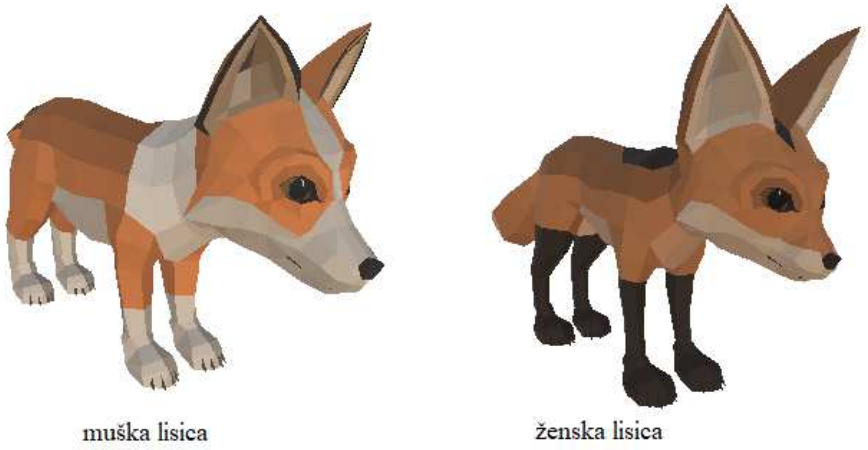
Bježanje od predatora ima najviši prioritet pri odabiru stanja jer može biti smrtonosno za zeca ako se odmah ne uzme u obzir. Implementirano je na način da ako zec primijeti lisicu u blizini, počne bježati pravocrtno u smjeru suprotnom od nje. Smatra se da je lisica blizu zecu ako se nalazi unutar 40% njegovog vidokruga. Zec bježi u nadi da lisica nije trenutno gladna, da će usput skrenuti zbog veće potrebe za vodom ili da će naići na nekog drugog zeca koji joj je bliže.



Slika 3.10. Bježanje od predatora

3.3. Lisice

FoxManager je skripta koja je zadužena za upravljanje populacijom lisica u okolišu. U njoj se definira početni broj lisica i vremenski odmak kada se te lisice stvaraju. Korisniku je omogućeno definiranje tih parametara. Tako je lisice moguće stvoriti odmah na početku zajedno sa zečevima, ili nakon određenog broja sekundi. Stvaranje pojedine lisice se ostvaruje na jednak način kao i kod zečeva. Unaprijed određene vrijednosti za brzinu i širinu vidokruga kod lisice su nešto veće nego kod zečeva (0.5 u odnosu na 0.4 za brzinu i 3 u odnosu na 2 za vidokrug). Modeli lisica prikazani su sljedećom slikom.



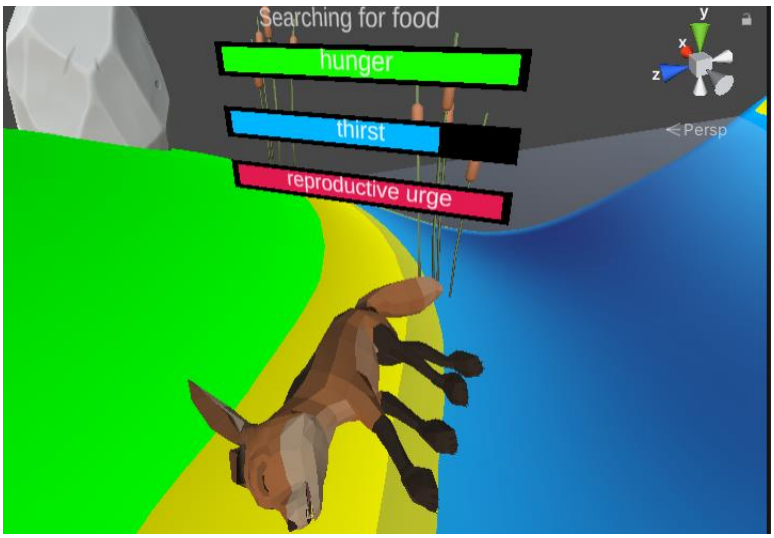
Slika 3.11. Modeli lisica

Ponašanje lisica zapravo je istovjetno ponašanju zečeva. Oboje imaju potrebu za hranom i vodom, kao i potrebu za reprodukcijom. Iste skripte koje su se koristile za definiranje stanja zečeva mogu se koristiti i za lisice. Glavna razlika je u stanju potrage za hranom. Kao što zečevi traže obližnje biljke za hranu, tako lisice traže obližnje zečeve. Ista skripta, *FoodSearchState*, se koristi za obje vrste životinja, samo se razlikuje tip hrane koji se traži. On je definiran za svaki model lisice i zeca pomoću slojeva. Kod zečeva je sloj za hranu postavljen na biljke, a kod lisica je postavljen na zečeve.

Još jedna mala razlika je pri razmnožavanju. Kao što to i u prirodi biva, lisice imaju nešto manji broj potomaka od zečeva, tako da je u ovoj simulaciji definiran maksimalan broj potomaka od 7, za razliku od zečeva gdje je taj broj 10.

Slika 3.12.Stvaranje potomaka lisica

S obzirom na to da lisice u prosjeku imaju veću brzinu i širi vidokrug od zečeva, njihova potreba za hranom i pićem brže će rasti. Uz to, njihov jedini izvor hrane su zečevi koji bježe od lisica kada su u blizini. Zbog tog lisice moraju biti vrlo efikasne u lovu ukoliko žele preživjeti, odnosno ne umrijeti od gladi. S druge strane, one su na vrhu hranidbenog lanca i nisu ugrožene od neke druge vrste. Lov na zečeve je njihova glavna briga. Ukoliko ne pronađu zečeve ili su ih već istrijebili, lisice umiru od gladi u potrazi za hranom kako prikazuje slika 3.13. Nakon smrti poziva se funkcija *BodyDisappear* koja čeka 10 sekundi i nakon toga uklanja objekt iz scene.



Slika 3.13. Lisica uginula od gladi

4. Rezultati simulacije

S obzirom na to da ovakve simulacije mogu potrajati, definiran je objekt u sceni, *SimulationManager*, koji omogućava ubrzavanje simulacije do 4 puta. Unutar skripte koja njega kontrolira se periodički (svako 3 sekunde) u datoteke zapisuju podaci o broju lisica i zečeva u sceni, kao i prosječni iznosi parametara (brzina, vidokrug i veličina).

Za svaku vrstu objekta korisnik ima slobodu definirati iznose pojedinih parametara. Tako za biljke može odrediti početnu količinu, minimalni i maksimalni vremenski odmak za stvaranje nove biljke. Za zečeve također definira proizvoljan broj početne populacije koja se stvara na početku simulacije. Za lisice, uz početni broj, omogućeno je i definiranje vremenskog odmaka kad će se one stvoriti u sceni. To može biti odmah na početku zajedno sa zečevima (vremenski odmak je 0) ili se njihova početna populacija može stvoriti nakon definiranog broja sekundi.

Prvo će se prikazati rezultati simulacije kada su samo zečevi u okolišu, a zatim se uvode lisice s različitim vremenom pojavljivanja.

4.1. Samo zečevi

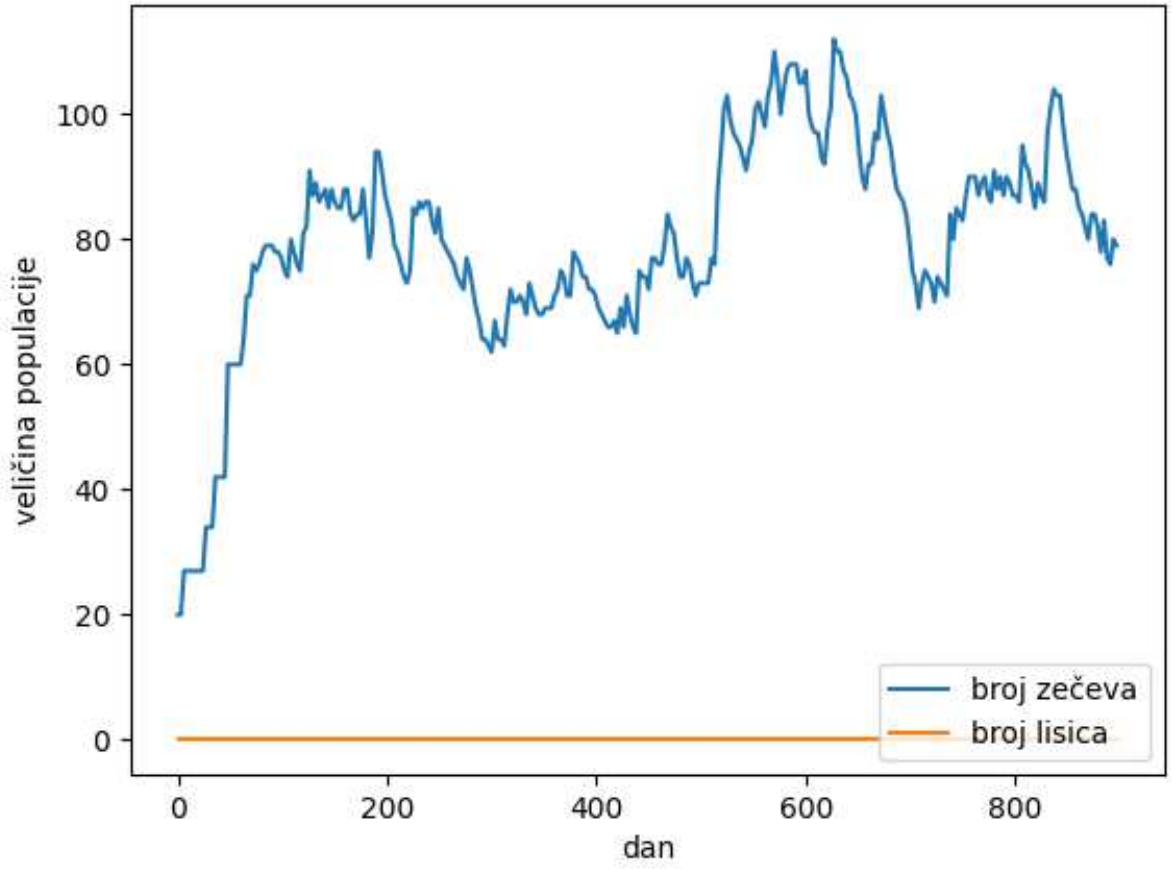
Broj zečeva u okolišu uvelike ovisi o broju biljaka jer su one njihov jedini izvor hrane. Dva parametra tu dolaze u obzir, početni broj biljaka i brzina stvaranja novih biljaka. Ti parametri mogu se proizvoljno namještati, a unaprijed određene vrijednosti su 100 početnih biljaka i stvaranje nove biljke nakon nasumičnog broja sekundi između 0.5 i 1.

Na slici ispod prikazani su rezultati simulacije s početnih 20 zečeva u sceni uz ostale unaprijed određene postavke. Ovdje nije bilo lisica tako da je broj smrti od njih 0. Broj smrti od gladi je puno veći nego broj smrti od žeđi jer je izvor hrane ograničen i zečevi se moraju natjecati za njega. Vode ima u izobilju, ali se svejedno dogodi situacija da zec nije u mogućnosti doći na vrijeme do nje i umre od žeđi.

Slika 4.1. Broj smrti kod zečeva

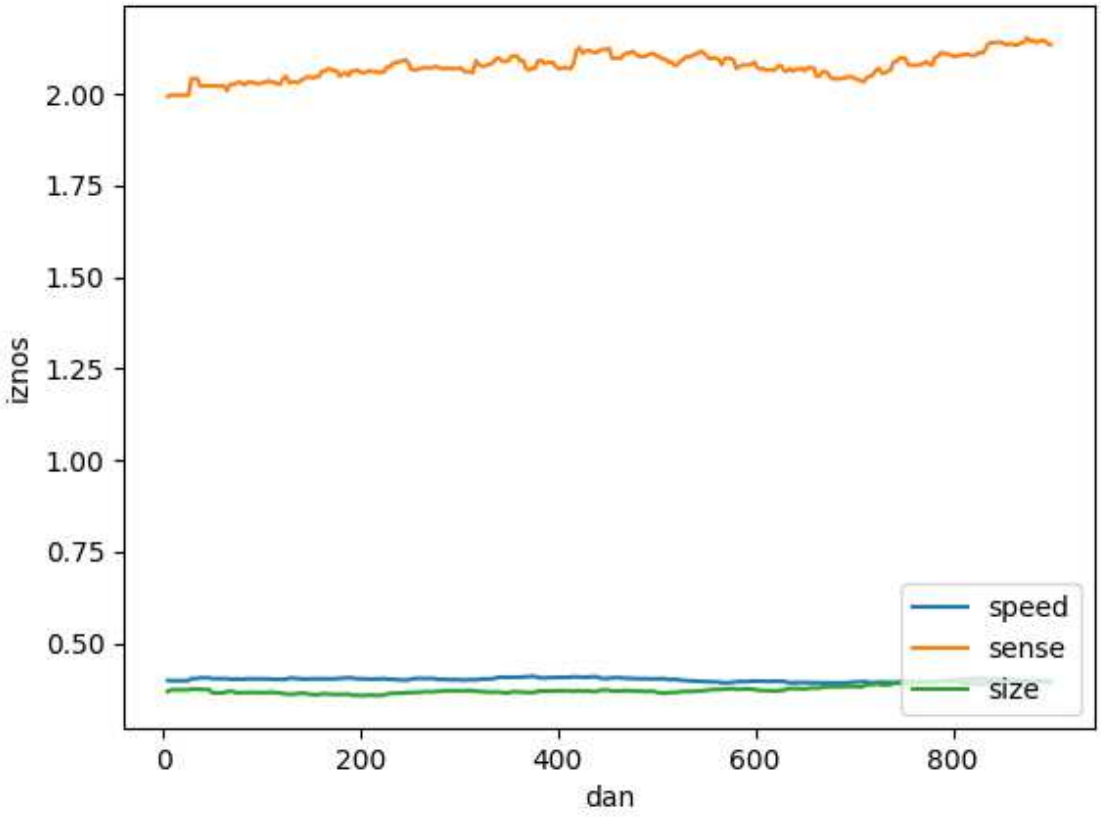
U ovoj simulaciji, broj zečeva se vrlo brzo učetverostručio s prvim valom potomaka. Nakon tog razdoblja, roditelji imaju određeno vrijeme hlađenja (*eng. cooldown*) za parenje i tada ne stvaraju nove potomke. S obzirom na to da je tada u sceni puno nerazvijenih zečeva koji imaju dvostruko manju brzinu i širinu vidokruga, te roditelja koji nisu spremni za stvaranje novih potomaka, veličina populacije se smanji (ovdje s oko 80 na oko 60).

Nakon što prođe određeno vrijeme, prvi potomci se razviju i roditelji su spremni za nove, događa se drugi val potomaka (na slici ispod oko 500. dana) i veličina populacije prijeđe 100. Ovaj ciklus se nastavlja periodički, a veličina populacije se stabilizira na oko 80 zečeva.



Slika 4.2. Populacija zečeva uz unaprijed određene postavke

Parametri u prikazanoj simulaciji ne mijenjaju previše svoju početnu vrijednost. Na slici ispod može se vidjeti blagi rast prosječnog iznosa širine vidokruga, ali nije značajan. Ostale vrijednosti također ne pokazuju neku veću promjenu. Uz dane postavke okoliša, ovi zečevi su očito dovoljno prilagođeni za preživljavanje i održavanje vrste.

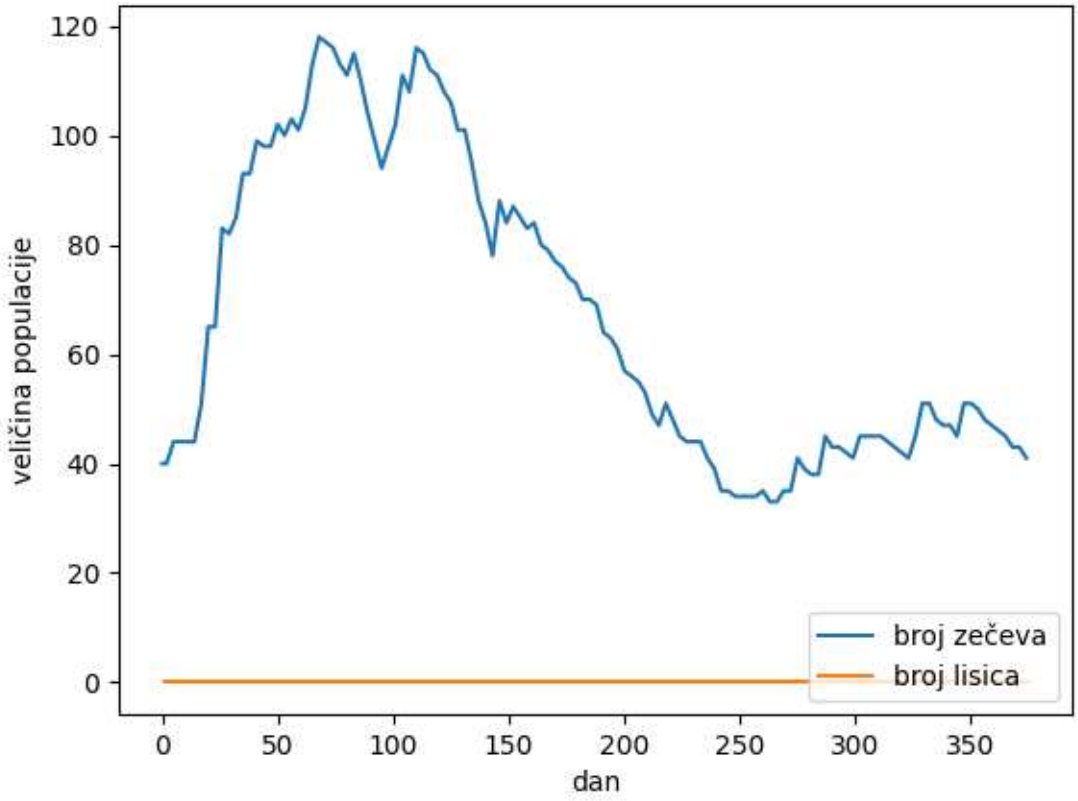


Slika 4.3. Prosječni parametri zečeva

Za očekivati je da će promjena postavki rasta biljaka utjecati na veličinu populacije zečeva. Sljedeća simulacija pokrenuta je uz dvostruko manji broj biljaka na početku (50 umjesto 100) i uz dvostruko sporije stvaranje novih biljaka (1-2s). Uz to, početni broj zečeva postavljen je na 40.

Na sličan način kao i u prethodnoj simulaciji, na početku populacija vrlo brzo naraste, ovdje čak do 120 (slika 4.1.4.). Međutim, ovaj okoliš nije postavljen tako da podržava toliku količinu zečeva istovremeno na mapi. Iz tog razloga, veličina populacije vrlo brzo opadne i vrati se na početnu razinu. Broj zečeva se stabilizira na približno 40.

Može se zaključiti izravna poveznica broja biljaka i broja zečeva. Dvostruko veći broj biljaka rezultira dvostruko većom populacijom zečeva u stabilnom stanju.



Slika 4.4. Kretanje populacije zečeva uz manji broj biljaka

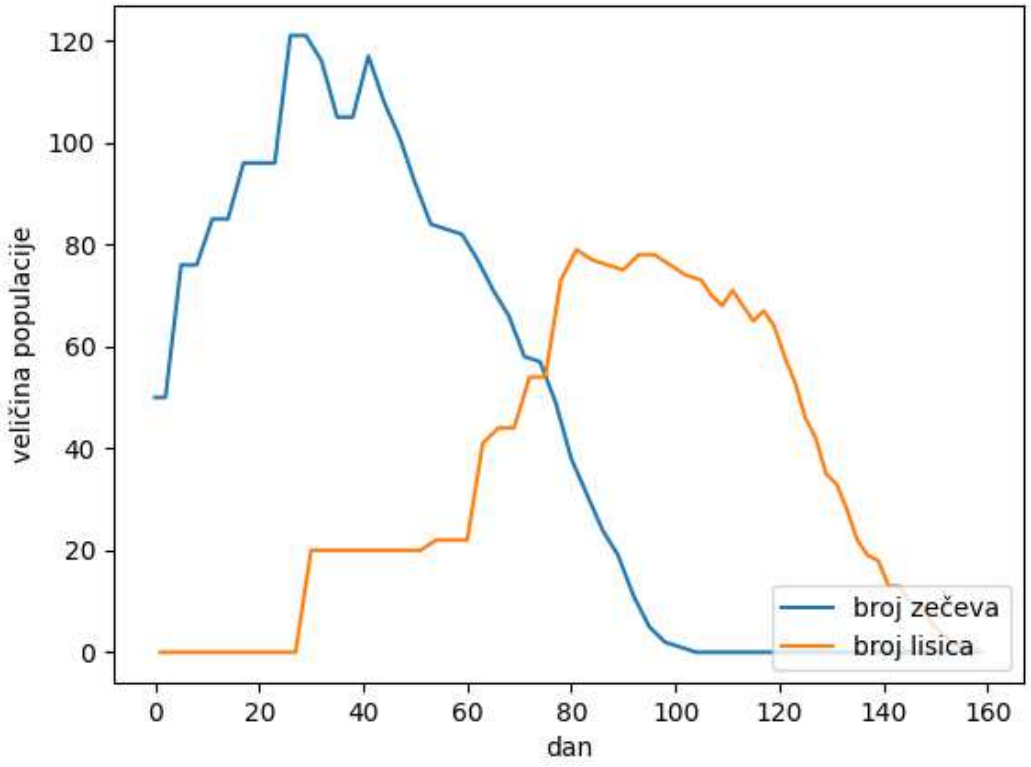
U ovakvom tipu okoliša gdje je zec jedina vrsta životinje, sve oko čega se treba brinuti je zadovoljavanje vlastitih potreba za glad i žeđ uz mogućnost pronalaska partnera i prosljeđivanja svojih gena na potomke.

Uvođenje lisica u igru sigurno će promijeniti način života zečeva. Osim navedenih funkcija, zec mora biti na konstantnom oprezu zbog opasnosti od predatora. Bježanje od lisice često će biti važnije od ostalih potreba ako zec nastoji preživjeti. Također, potreba za reprodukcijom bi mogla imati puno važniju ulogu uvođenjem lisica jer će zečevi biti u opasnosti od izumiranja. Sljedeće poglavlje prikazuje rezultate simulacije suživota zečeva i lisica pokrenutih različitim početnim postavkama.

4.2. Zečevi i lisice

Za očekivati je da će se kretanje populacije zečeva poprilično razlikovati pojavom lisica u okolišu u odnosu na situaciju kada su zečevi sami. Preživljavanje i održavanje vrste postaje puno teže jer su oni lovina. Donošenje odluka je također složenije jer svaki zec mora balansirati između vlastitih životnih potreba (glad i žeđ), ali i potreba cijele populacije (opasnost od izumiranja zbog lisica).

Uobičajeni scenarij kretanja veličine populacije zečeva i lisica je prikazan na slici ispod. Simulacija je pokrenuta s početnih 50 zečeva i 20 lisica koje se stvore nakon 30 sekundi (30 dana u simulaciji). Kao i u prethodnim simulacijama, broj zečeva vrlo brzo naraste (ovdje do 120) dok nema lisica. Jedan kratak period od otprilike 20-30 dana obje populacije su stabilne dok se lisice ne počnu razmnožavati. U ovom primjeru to se događa u 60. danu kada broj lisica naglo poraste, a broj zečeva počne naglo padati. Oko 100. dana lisice su istrijebile sve zečeve i shvatile da su napravile veliku grešku jer nemaju više izvora hrane. Na kraju, iz navedenog razloga, populacija lisica izumre i okoliš ostane prazan.



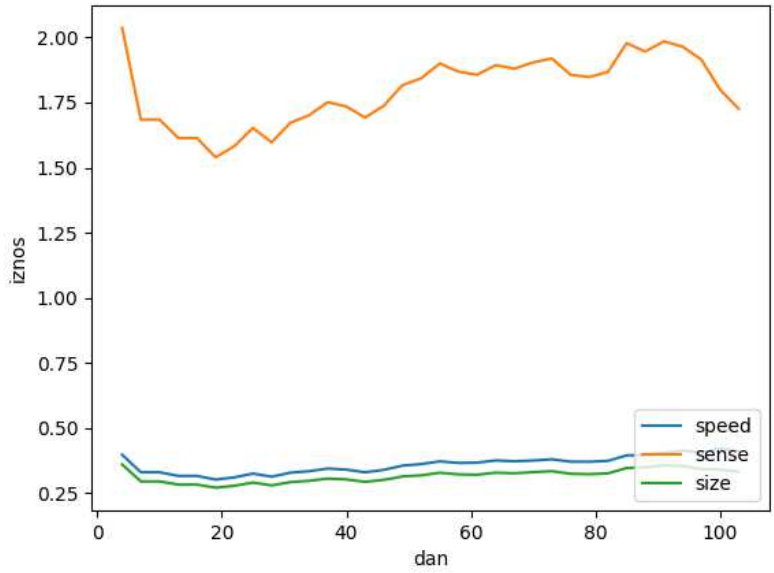
Slika 4.5. Uobičajeni scenarij populacije zečeva i lisica

Dok se lisice još ne pojave u okolišu, jedini način na koji zečevi mogu umrijeti je od gladi ili od žeđi. Broj takvih smrti je zanemariv prema broju smrti od lisica (ukupno 24 smrti od gladi i žeđi u usporedbi sa 146 smrti od lisica). Naravno, događa se i prvi tip smrti dok su lisice prisutne. Na primjer, dok bježi od lisice, zec umre od gladi jer ne stigne pronaći hranu.

Kad se lisice stvore, one imaju hrane u izobilju jer se populacija zečeva već razvila (ovdje oko 120 zečeva na početnih 20 lisica). Iz tog razloga, u početku će većina lisica umirati zbog žeđi. Nakon što istrijebe sve zečeve, lisice počnu masovno umirati od gladi (95 takvih smrti u ovoj simulaciji).

Slika 4.6. Rezultati simulacije

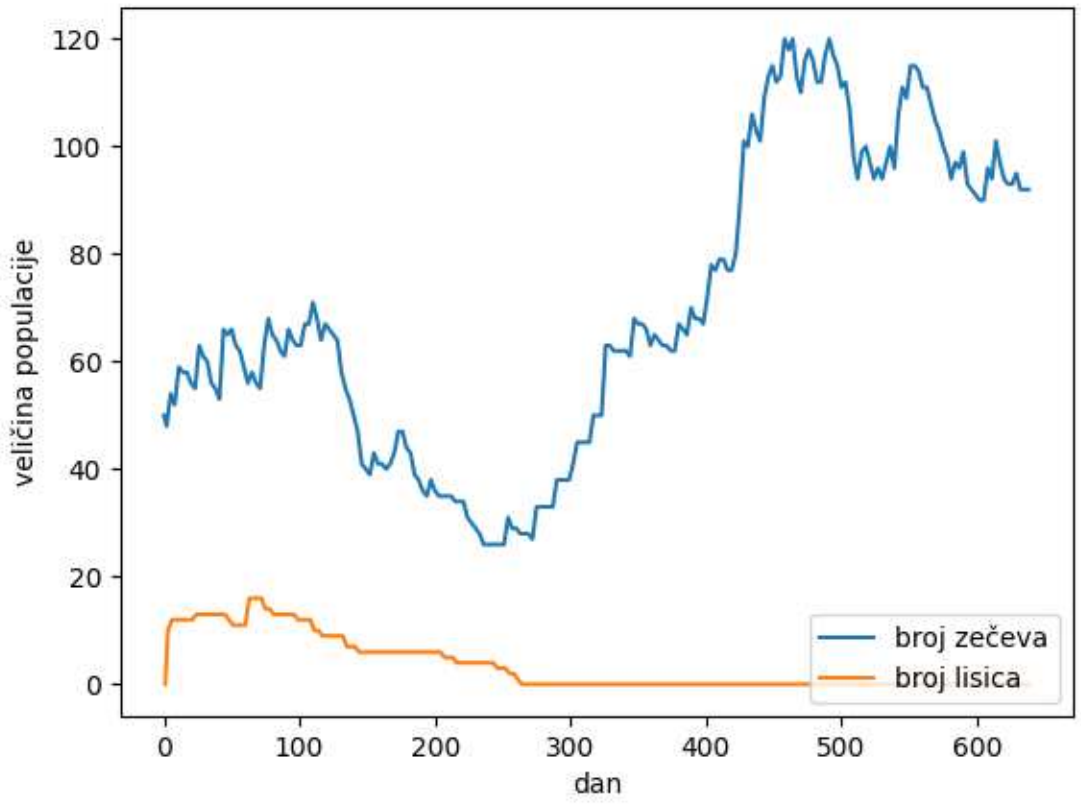
Iznosi parametara se u prosjeku blago povećavaju tijekom simulacije. U početku se čak smanje određeni iznos prije pojavljivanja lisica. Nakon toga se događa rast prosječnog iznosa svih parametrima s posebnim naglaskom na širinu vidokruga (od 1.5 do 2). Pokazalo se da uvođenjem lisica u okoliš, taj parametar postane najvažniji kod zečeva. Veći prostor kojeg je zec svjestan oko sebe mu omogućava da na vrijeme donese pravu odluku.



Slika 4.7. Kretanje parametara

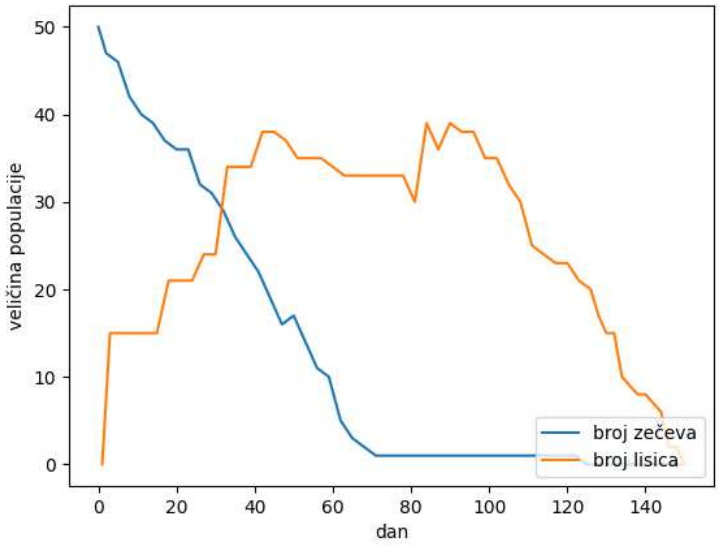
Iako je prethodno navedeni scenarij najčešći kad su u pitanju obje vrste životinja, ponekad se dogode i alternativni slučajevi kada se ne dogodi izumiranje svih životinja. Lisice nikada ne mogu preživjeti bez zečeva, ali obratni scenarij je moguć. Slika 4.2.4. prikazuje taj slučaj.

Lisice su u okolišu od početka, ali ih je samo 10. Nakon blagog rasta populacije zečeva u početku, broj zečeva se naglo smanjuje dok su lisice prisutne. Broj zečeva padne na oko 20 i tada su u opasnosti za izumiranje. Međutim, populacija lisica se nije uspjela dovoljno razviti u danom periodu. Početnih 10 lisica pokazalo se premalim za razvijanje populacije. Puno je manja vjerojatnost pronalaska zadovoljavajućeg partnera. U ovoj simulaciji dogodila su se samo dva parenja lisica. Uz manji broj zečeva u sceni, oko 250. dana, lisice su izumrle. Nakon što prestane opasnost od predatora, populacija zečeva se naglo obnovi (od 20 do 120). Broj zečeva se na kraju stabilizira na oko 100 zečeva koliko ih okoliš podržava sa zadanim postavkama.



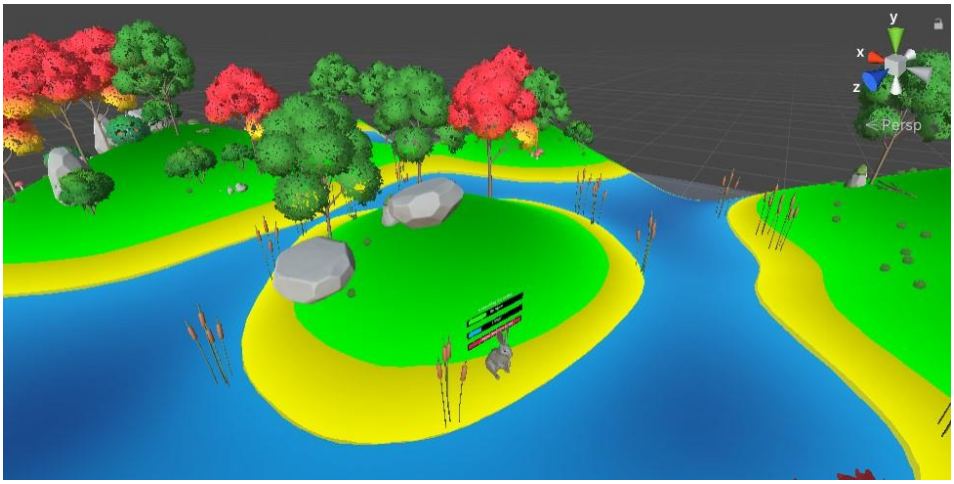
Slika 4.8. Obnavljanje populacije zečeva

Najčešći slučaj kada se lisice postave u scenu odmah na početku simulacije prikazan je na donjoj slici. Uz velik broj zečeva na početku, lisice vrlo brzo povećaju svoju populaciju i istrijebe zečeve. Nedugo nakon toga i one izumru jer nemaju više izvora hrane.



Slika 4.9. Naglo izumiranje zečeva

Ponekad se dogodi da neki zečevi slučajno prežive u okolišu iako bi trebali izumrijeti zbog velike populacije lisica. Takvi slučajevi se dogode kada lisice ne mogu pronaći preostale zečeve ili su im oni fizički nedostupni. To prikazuje sljedeća slika gdje zec cijelo vrijeme živi na sigurnom, na otoku koji je nedostupan lisicama. Jedino se moglo dogoditi da se početnim postavljanjem lisica neke od njih stvore na tom otoku, ali taj slučaj se ovdje nije dogodio i zec ostaje na sigurnom.

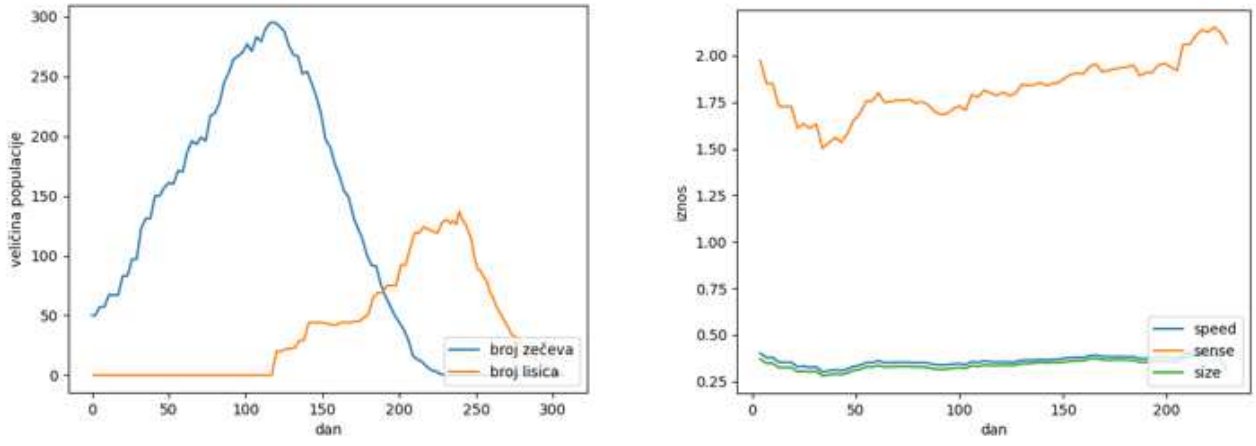


Slika 4.10. Zec na otoku

Izgled grafova ne pokazuje velike promjene kada se namjeste postavke tako da okoliš podržava veći broj životinja istovremeno. U sljedećoj simulaciji nove biljke se stvaraju unutar vremenskog raspona od 0.3 do 0.5 sekundi. Zbog bržeg stvaranja biljaka, u okolišu ima dovoljno hrane za velik broj zečeva. Taj broj dođe do 300 tijekom simulacije. Zečevi i dalje umiru najviše od strane lisica. Većina smrti od gladi (ukupno 159) se dogodi prije pojave lisica. Veličina populacije lisica u vrhuncu dolazi do skoro 150 u ovakvom okolišu. Nakon istrebljenja zečeva, većina ih umire od gladi (ukupno 190).

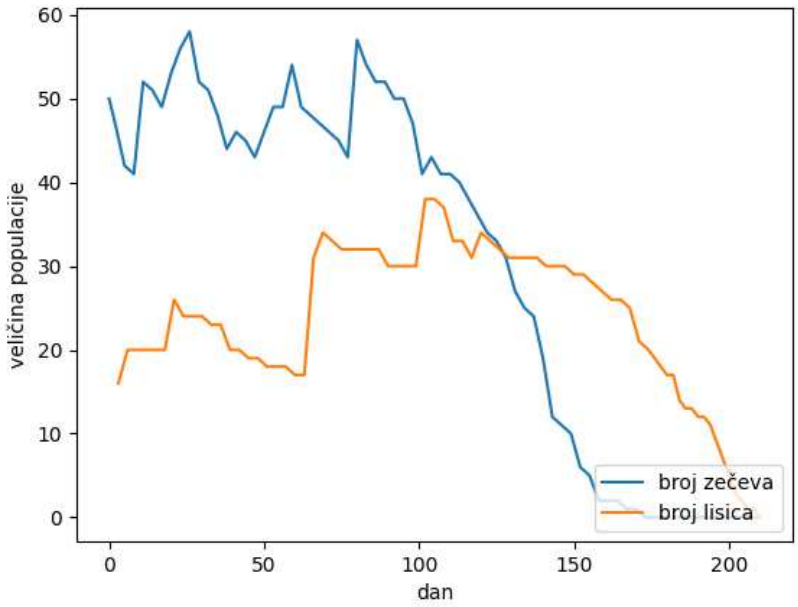
Slika 4.11. Rezultati simulacije

Populacija zečeva je u ovoj simulaciji narasla čak 6 puta (od početnih 50 do 300). Nakon pojave lisica, taj broj počne naglo padati. Lisice su se razmnožile i zečevi više nisu imali šanse za obnovu populacije. U trenutku kada broj lisica dosegne svoj maksimum što se događa oko 240. dana, zečevi su u potpunosti istrijebljeni. Od tada, broj lisica počne naglo opadati i one također izumiru. Kretanje veličine populacije zečeva i lisica, kao i parametara zečeva prikazani su grafovima ispod. Parametri pokazuju već viđenu situaciju. Nakon početnog pada, iznosi parametara u prosjeku rastu vremenom što se najviše vidi na iznosu širine vidokruga.

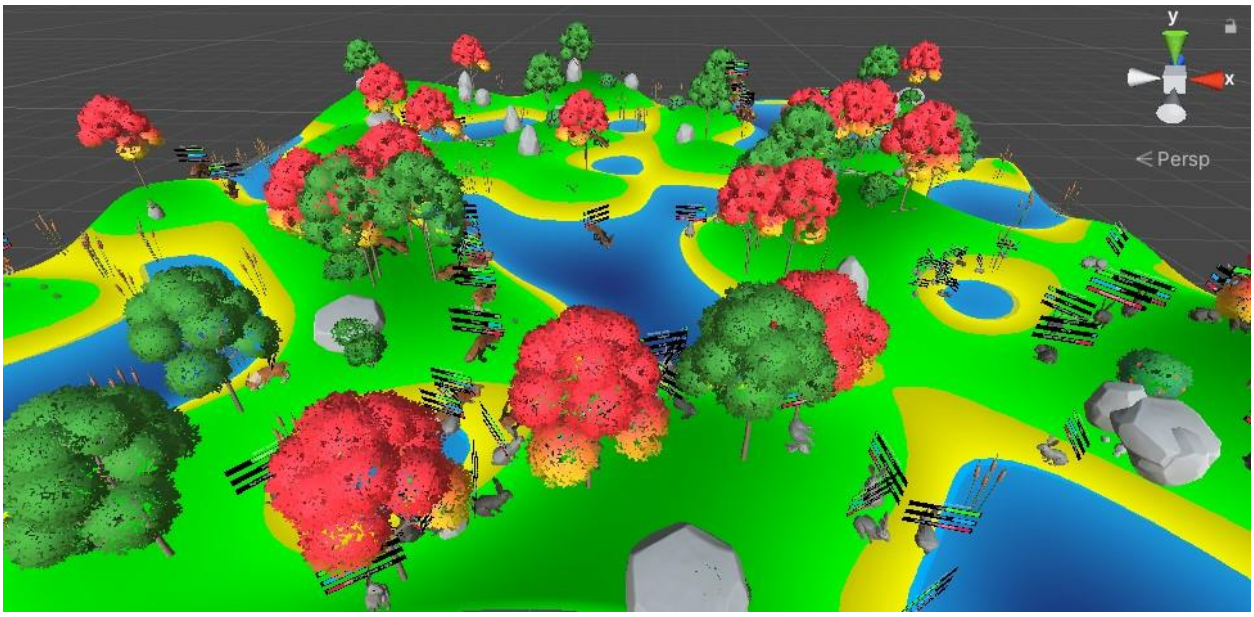


Slika 4.12. Kretanje populacija i parametara

Simulacija u kojoj se najdulji period mogle održati relativno stabilna populacija zečeva i lisica prikazan je na slici ispod. Nešto više od 100 dana obje su populacije održavale broj jedinki ili ga povećavale u određenim periodima. Oko 120. dana zečevi se više ipak ne mogu boriti protiv lisica i postepeno izumiru. Lisice zatim slijede isti uzorak. Izgled okoliša za navedenu simulaciju prikazan je slikom 4.2.10.



Slika 4.13. Djelomično održiva populacija



Slika 4.14. Stanje na mapi

4.3. Budući rad

Simulacija ekosustava je vrlo složen zadatak i uvijek je moguće dodavanje novih funkcionalnosti kako bi taj prikaz bio što realističniji. U ovom radu implementirane su samo dvije vrste životinja: zečevi kao predstavnici plijena i lisice kao predstavnici predatora. U daljnjoj nadogradnji programa mogle bi se dodati i druge vrste životinja i tako proširiti hijerarhiju hranidbenog lanca.

Slika 4.15. Potencijalne dodatne vrste životinja

Trenutna simulacija ima uzrok smrti samo od gladi, žeđi i od ulova lisice. Mogao bi se implementirati i životni vijek životinja tako da one mogu umrijeti i prirodnom smrću.

Odluke u kojem će se stanju nalaziti u kojem trenutku je određeno prirodnim nagonima (potrebama za glad, žeđ i reprodukciju). Potencijalno se može definirati jedna vrsta mozga upravljana neuronskom mrežom ili neizrazitim sustavom za donošenje odluka. Ti parametri bi se onda prenosili na potomke i pratio bi se razvoj tog mozga kroz generacije.

Zečevi trenutno samo pravocrtno bježe od lisica i nemaju se gdje sakriti ako se žele spasiti. Implementacija mjesta za skrivanje, kao što su rupe do kojih lisice ne bi mogle doći, moglo bi dovesti do realističnijeg i složenijeg prikaza okoliša.

Dodatno stanje mirovanja bi također imalo smisla za svaku životinju. Nije nužno da ona stalno bude u pokretu i u potrazi za hranom, vodom ili partnerom. Uvođenje stanja spavanja ili odmaranja u kojem se energija obnavlja dalo bi prirodniji izgled simulaciji.

Još jedna zanimljiva funkcionalnost bila bi uvođenje komunikacije među životinjama. Na primjer, zec je svjestan samo svojega vidokruga i ne zna što se događa van njega, ali ako vidi nekog drugog zeca, on mu može prenijeti bitne informacije koje ovaj zna i na taj način mu pomoći. Uz to, korisno bi bilo implementirati i neku vrstu kratkoročnog pamćenja za sve vrste životinja.

5. Zaključak

Računalne simulacije su u svojim početcima bile korištenje za meteorologiju i nuklearnu fiziku u razdoblju nakon Drugog svjetskog rata. Od tada su postale neophodne za rastući broj disciplina kao što su astrofizika, mehanika fluida, klimatske znanosti, evolucijska biologija, ekologija, medicina, sociologija, epidemiologija i mnoge druge.

U širokom smislu, računalne simulacije mogu se shvatiti kao metoda za proučavanje sustava. Ovaj proces uključuje pronalazak modela, način implementacije tog modela na način da može biti pokrenut na računalu te vizualizacije i analize rezultata.

Ovaj rad opisuje programski sustav ostvaren unutar okruženja Unity sa skriptama pisanim u programskom jeziku C#. Opisano je ponašanje zečeva kao predstavnika plijena i lisica kao predstavnika predatora. Biljke u okolišu predstavljaju hranu za zečeve, a ti zečevi su hrana lisicama. Na taj način ostvaren je mali hranidbeni lanac kojeg ovaj program simulira.

Mehanizam koji upravlja ponašanjem životinja je stroj s konačnim brojem stanja. Ta stanja mogu biti potraga za hranom, potraga za vodom, traženje partnera za razmnožavanje i bijeg od predatora. Životinje su vođene nagonima, odnosno iznosima navedenih potreba. One zapravo nemaju neku vrstu inteligencije ni pamćenja. Glavni cilj koji one žele postići je preživljavanje u danom okolišu i proširenje svoje populacije.

Program se može pokrenuti uz različite početne postavke. Te postavke uključuju parametre koji određuju broj početnih biljaka, zečeva i lisica u okolišu. Također, može se mijenjati i brzina stvaranja novih biljaka kao i vrijeme kada se lisice uvode u scenu. Omogućeno je i mijenjanje sporednih parametara kao što su vrijeme razvoja pojedine životinje od trenutka rođenja do odrasle dobi i vrijeme pauze između dva uzastopna parenja.

Rezultati simulacija pokazali su da je vrlo teško postići da obje vrste životinja imaju stabilnu populaciju za bilo kakve parametre. Najčešće se dogodi slučaj da se populacija zečeva naglo razvije u početku. Iz tog razloga se i populacija lisica razvije što dovodi do izumiranja zečeva. Nakon toga i lisice izumru pa okoliš ostane prazan. Rijetki slučajevi kada se ovaj scenarij ne dogodi jest kada lisice ne mogu pronaći preostale zečeve ili kada ih nema dovoljno za razvoj populacije. Lisice tada izumru, a populacija zečeva nastavlja sa životom.

Literatura

[1] http://www.speciesgame.com/

[2] https://www.britannica.com/technology/artificial-life

[3] http://www.framsticks.com/

[4] http://www.ifa.hawaii.edu/~meech/a740/2004/fall/papers/Framsticks.pdf

[5] https://store.steampowered.com/app/774541/Species_Artificial_Life_Real_Evolution/

[6] https://link.springer.com/chapter/10.1007%2F3-540-45016-5_20

[7] https://www.britannica.com/technology/computer-simulation

[8] https://www.nationalgeographic.org/encyclopedia/ecosystem

[9] https://www.khanacademy.org/computing/computer-programming/programming-natural-simulations/programming-noise/a/perlin-noise

[10] https://en.wikipedia.org/wiki/Shader

[11] https://refactoring.guru/design-patterns/state

[12] https://plato.stanford.edu/entries/simulations-science/

[13] https://www.youtube.com/watch?v=r_It_X7v-1E&ab_channel=SebastianLague

Simulacija ekosustava

Sažetak

Ovaj rad bavi se simulacijom ekosustava napravljenog u okruženju za razvoj digitalnih igara Unity. U početku su opisani postojeći simulatori: mobilna igra Evolution, računalna igra Species ALRE i simulator Framsticks.

Nakon toga u praktičnom dijelu najprije je opisano stvaranje okoliša u kojemu će se životinje kretati. Implementirane su dvije vrste životinja, zečevi i lisice, kao i stvaranje biljaka koje su hrana za zečeve. Ponašanje životinja implementirano je koristeći oblikovni obrazac stanje. Rezultati simulacija prikazani su grafovima. Na njima se može vidjeti kretanje veličine populacija kroz vrijeme, kao i kretanje prosječnog iznosa parametara: brzine, širine vidokruga i veličine.

Ključne riječi: simulacija, ekosustav, simulator, predator, plijen, ponašanje, odluke, stanje, populacija, genetski algoritam

Simulating an ecosystem

Abstract

This paper describes simulation of an ecosystem made within cross-platform game engine Unity. Existing simulators are described at the beginning: mobile game Evolution, computer game Species ALRE and Framsticks simulator.

After that, in practical part, generation of the environment which animals will inhabit is described first. Two types of animals are implemented, rabbits and foxes, as well as generation of plants which present the food for the rabbits. Behavior of the animals is implemented using the State design pattern. Results of the simulation are presented using graphs. Connection between size of the populations and time is shown, as well as change in average amount of parameters: speed, sense and size.

Key words: simulation, ecosystem, simulator, predator, prey, behavior, decisions, state, population, genetic algorithm