Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jakob Bambič

Proceduralna generacija terena za uporabo v računalniških igrah

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Narvika Bovcon

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo: Proceduralna generacija terena za uporabo v računalniških igrah

Tematika naloge:

Proceduralna generacija terena za računalniške igre prinaša nekatere prednosti na ravni uporabniške izkušnje. Zasnujte primer tako generiranega terena, pri tem pa upoštevajte več parametrov, ki bodo ob ustreznih nastavitvah omogočili dovolj raznoliko in hkrati prepoznavno oblikovanje terena. Optimizirajte delovanje vaše rešitve. Omogočite uporabniku, da s pomočjo lastnih nastavitev ustvari nove terene.

Kazalo

Povzetek

Naslov: Proceduralna generacija terena za uporabo v računalniških igrah

Avtor: Jakob Bambič

V diplomski nalogi opisujemo proceduralno generacijo terena, izvor, definicijo ter manipulacije Perlinovega šuma in uporabljeno programsko opremo - igralni pogon Unity. Opisujemo postopke, s katerimi smo optimizirali delovanje projekta. S ponovno uporabo delov terena ter z zmanjševanjem količine detajlov v daljavi smo drastično pohitrili delovanje igre. V diplomskem delu opisujemo tudi pristope, ki smo jih uporabili za vsak posamezen del proceduralne generacije terena. Pri generiranju terena je bilo potrebno generirati višino, vlažnost ter toploto. Nato smo iz dobljenih podatkov generirali ekosisteme. Naslednja proceduralna generacija, ki smo jo implementirali, je generacija zelenja ter naselij, ki vključuje generacijo cest in stavb. Prikazani in ovrednoteni so tudi končni rezultati proceduralne generacije terena. Na koncu diplomska naloga ponuja tudi navodila za uporabo generatorja terena v igralnem pogonu Unity. Opisane so vse funkcije ter nastavitve, ki jih je mogoče kontrolirati iz uporabniškega vmesnika igralnega pogona.

Ključne besede: Perlinov šum, Unity, proceduralna generacija, računalniška igra.

Abstract

Title: Procedural terrain generation for use in computer games

Author: Jakob Bambič

The diploma thesis describes the procedural generation of the terrain, the origin, definition and manipulation of the Perlin noise, and the software that we used - game engine Unity. The optimization of the performance of the project was achieved by reusing parts of the terrain and by reducing the details in the distance. In the diploma thesis we also describe methods we used for each layer of our terrain generation: we generated the terrain by first generating height, moisture and heat maps, and on the basis of the data we generated different ecosystems. The next generation we implemented is the generation of greenery and cities, which includes the generation of roads and buildings. We also present and evaluate the final results of the procedural generation of the terrain. At the end of the diploma thesis we give the instructions for use of our procedural terrain generation in Unity. We describe all the functions and parameters, which can be changed in the user interface of the game engine.

Keywords: Perlin noise, Unity, procedural content generation, computer game.

Poglavje 1

Uvod

Uporaba naključne generacije v računalniških igrah je čedalje bolj popularna. Z uporabo naključne generacije so igre igralcem zanimive dlje, saj nikoli ne vedo, kakšna vsebina jih čaka ob naslednjem igranju. V diplomskem delu bomo predstavili implementacijo generiranja sveta s pomočjo igralnega pogona Unity. Generirali bomo teren s pomočjo Perlinovega šuma. Uporabili bomo Perlinov šum, ki ga bomo generirali na različne načine - šum za višino, vlažnost ter toploto. Poleg tega bomo implementirali naključno generacijo zelenja, mest, stavb ter drugih elementov.
Za implementacijo naključne generacije terena bomo uporabili enega izmed najbolj popularnih igralnih pogonov, igralni pogon Unity. Vso kodo pa bomo napisali v programskem jeziku C#.
Raziskali ter opisali bomo tudi vso tematiko povezano s Perlinovim šumom, proceduralno generacijo vsebine ter igralnim pogonom Unity.
Cilj diplomske naloge je ustvariti zanesljivo, nadzorljivo ter raznoliko proceduralno generacijo terena, ki jo bodo lahko uporabili tudi drugi ustvarjalci računalniških iger kot predlogo.

Poglavje 2

Proceduralna generacija in Perlinov šum

2.1 Proceduralna generacija vsebine

Pojem proceduralna generacija vsebine se nanaša na avtomatsko generiranje vsebine igre s pomočjo raznih algoritmov. Pojem vsebine, ki jo generiramo, pa lahko definiramo na nešteto veliko načinov. V nekaterih igrah proceduralno generirana vsebina zajema kar cel svet, pri drugih pa le zelo majhen del igre. Najbolj slavni primeri proceduralne generacije vsebine so [4]:

- Diablo (Blizzard 1996) - proceduralno generiranje ječ
- Civilization (MicroPRose 1991) - proceduralno generiranje sveta
- Borderlands (Gearbox 2009) - proceduralno generiranje orožja
- SpeedTree (Interactive Data Visualization 2003) - proceduralno generiranje zelenja

Pri implementiranju proceduralne generacije vsebine so zaželjene naslednje lastnosti [3]:

- Hitrost generiranja - vpliva na to, ali se vsebina generira med igranjem ali med razvojem igre.

- Zanesljivost generacije - po navadi želimo, da naša implementacija vsakič jamči, da je generirana vsebina dovolj kvalitetna za našo igro.
- Nadzorljivost - možnost, da uporabnik ali nek algoritem določita nekaj lastnosti generirane vsebine.
- Izraznost in raznolikost - vsebina, ki jo večkrat generiramo, mora biti ob vsaki generaciji različna.
- Kreativnost in resničnost - po navadi želimo, da vsebina, ki jo proceduralno generiramo, izgleda, kot da smo jo izdelali ročno in ne generirali.

V naši diplomski nalogi bomo kot vsebino proceduralne generacije generirali teren, zelenje ter mesta.

2.1.1 Prednosti in slabosti

Green [1] navaja naslednje prednosti uporabe proceduralne generacije v igrah:

- generiranje dinamične vsebine,
- manjša uporaba pomnilnika, saj vsebina ni predhodno shranjena,
- prihrani čas in denar za razvoj,
- ustvari veliko število lastnosti, ki jih lahko spreminjamo,
- vzpodbudi ponovno igranje, saj je vsebina vsakič drugačna.

Kot negativne lastnosti uporabe proceduralne generacije pa navaja:

- lahko porabi veliko resursov strojne opreme,
- vsebina se lahko na videz ponavlja,
- težko je programirati v naprej določene dogodke,
- obstaja verjetnost generiranja neuporabne vsebine.

2.2 Perlinov šum

Perlinov šum je ustvaril Ken Perlin leta 1983. Z njim je hotel rešiti problem preveč umetnega izgleda računalniške grafike tistega časa. Svoje najdbe je formalno predstavil na konferenci SIGGRAPH leta 1985, v članku z naslovom "An image Synthesizer"[2].

2.2.1 Definicija

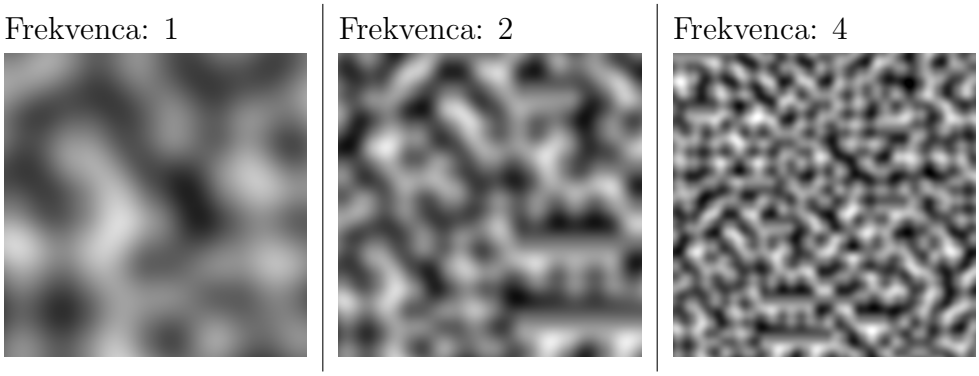
Perlinov šum je tip gradientnega šuma. Funkcija Perlinovega šuma ima naključen izgled, vendar so vse njene vizualne podrobnosti enake velikosti. Ta lastnost omogoča, da je funkcija zlahka nadzorovana. Več kopij Perlinovega šuma je mogoče združiti z matematičnimi manipulacijami v veliko različnih tekstur. Te teksture so uporabljene v računalniški grafiki za ustvarjanje računalniško generiranih vizualnih efektov, kot so površine, ogenj, dim ali oblaki. Perlinov šum pogosto uporabljamo, ko imamo omejen pomnilnik in želimo generirati teksture [2].

2.2.2 Manipuliranje

V spodnjih primerih so prikazane različne metode manipuliranja Perlinovega šuma. Vse grafične predstavitve uporabljajo enak izvirnik Perlinovega šuma.

Frekvenca

S frekvenco lahko Perlinov šum gostimo ali redčimo.
Grafični prikaz spreminjanja frekvence:



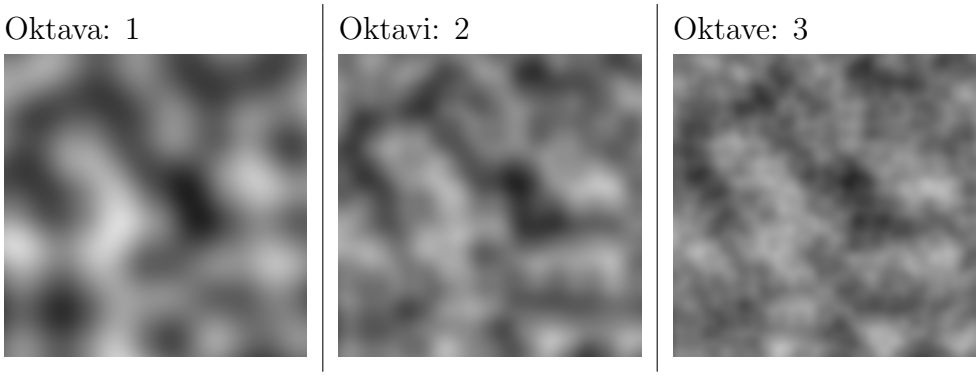
Slika 2.1: Slika prikazuje različne frekvence Perlinovega šuma. (Vir: lasten)

Unity (C# implementacija) za določitev vrednosti Perlinovega šuma na lokaciji (x, y):

```
float sample;
sample = Mathf.PerlinNoise(x * scale, y * scale);
```

Vsota oktav ali turbulenca

Pri vsoti oktav uteženo seštejemo več Perlinovih šumov različnih frekvenc. Grafični prikaz oktav:



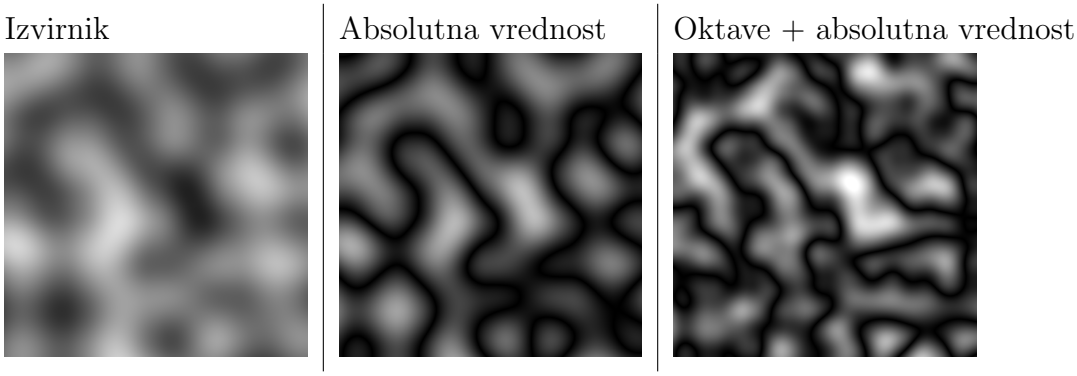
Slika 2.2: Slika prikazuje Perlinov šum različnih oktav. (Vir: lasten)

Unity (C# implementacija) za določitev vsote več oktav Perlinovega šuma na lokaciji (x, y):

```
float mult = 1.0f;
float mult2 = 1.0f;
float sample;
for (int i = 0; i < octaves; i++)
{
  sample += mult * (Mathf.PerlinNoise(mult2 * x,
    mult2 * y));
  mult /= 2;
  mult2 *= 2;
}
```

Absolutna vrednost

Absolutno vrednost šuma v programu Unity dosežemo tako, da šum skaliramo na nov interval in sicer iz intervala [0, 1] na interval [-1, 1]. Dobljene negativne vrednosti pa preslikamo v pozitivne.
Grafični prikaz absolutne vrednosti:



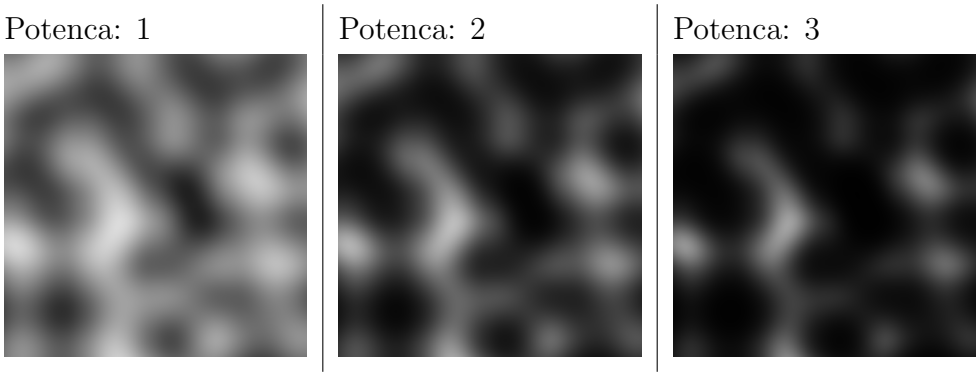
Slika 2.3: Slika prikazuje izvirnik, absolutno vrednost izvirnika ter absolutno vrednost izvirnika z oktavami. (Vir: lasten)

Unity (C# implementacija) za določitev absolutne vrednosti Perlinovega šuma na lokaciji (x, y):

```
float sample;
sample = (Mathf.PerlinNoise(x, y) * 2 - 1f);
sample = Mathf.Abs(sample);
```

Potenca

S potenciranjem šumu zmanjšujemo nižje vrednosti, višje pa ohranjamo. Grafični prikaz potenc:



Slika 2.4: Slika prikazuje različne potence Perlinovega šuma. (Vir: lasten)

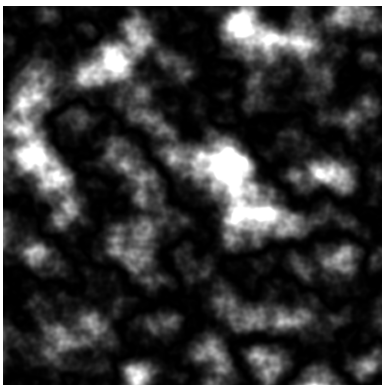
Unity (C# implementacija) za določitev vrednosti potence Perlinovega šuma na lokaciji (x, y):

```
float sample;
sample = Mathf.PerlinNoise(x, y);
sample = Mathf.Pow(sample, power);
```

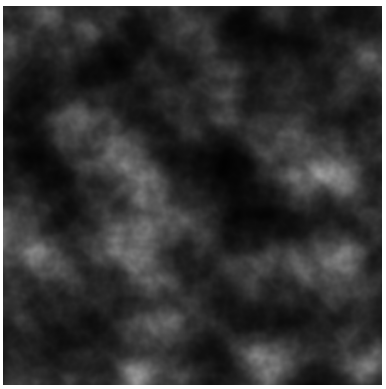
Kombinacije

Primeri rezultatov z uporabo več omenjenih manipulacij Perlinovega šuma. Grafični prikaz:

Frekvenca: 1, Oktave: 5, Potenca: 3, Absolutna vrednost



Frekvenca: 1, Oktave: 5, Potenca: 3



Slika 2.5: Slika prikazuje več kombinacij modifikacij Perlinovega šuma. (Vir: lasten)

Poglavje 3

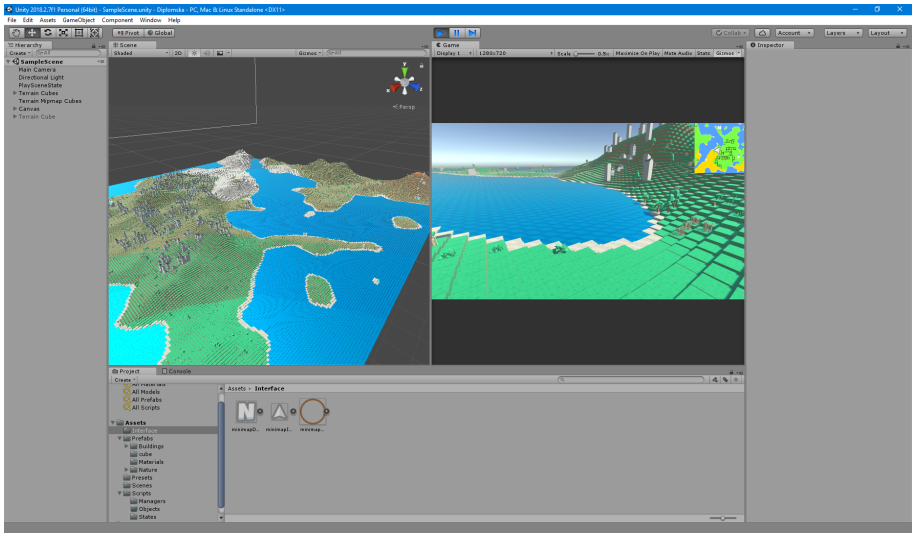
Predstavitev uporabljenih tehnologij

3.1 Unity

Generacija terena je bila implementirana v igralnem pogonu Unity, v programskem jeziku C#. Unity je eden izmed najbolj razširjenih trenutno dostopnih igralnih pogonov. Glavne funkcije igralnega pogona Unity so dostopnost urejevalnika na Windows in na Mac računalnikih, možnost ustvarjanja iger v 2D in v 3D, enostavna orodja za izdelavo uporabniških vmesnikov, vgrajene simulacije fizike v 2D in 3D, možnost ustvarjanja orodij po meri ter možnost razvoja igre za več platform hkrati. Trenutno je podprtih več kot 25 platform za izdelavo iger. Nekatere izmed najbolj razširjenih so: Android, iOS, Windows, MaxOS, Linux, WebGL, PS4, Xbox One, Nintendo 3DS, Oculus Rift,Steam, PlayStation VR, Android TV, tvOS ter Nintendo Switch [5]. Slika 3.1 prikazuje grafični vmesnik igranlega pogona Unity, s prikazom primera generacije terena.

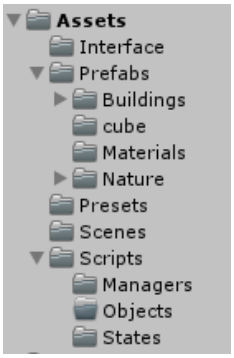
3.1.1 Organizacija Unity projekta

Vse datoteke generacije terena se nahajajo v mapi 'Assets' ter so razdeljene v podmape. Podmapa 'Interface' vsebuje slike vseh 2D grafičnih elementov



Slika 3.1: Uporabniški vmesnik igralnega pogona Unity. (Vir: lasten)

uporabniškega vmesnika. Podmapa 'Prefabs' vsebuje vse 3D modele ter predloge objektov, podmapa 'Presets' vsebuje predloge konfiguracij generacije terena, 'Scenes' vsebuje glavno sceno ter 'Scripts' vsebuje vso kodo generacije terena. Delovanje ter uporaba posameznih komponent generacije terena je razložena v poglavju 'Navodila za uporabo'. Slika 3.2 prikazuje drevesni pogled organizacije map Unity projekta.



Slika 3.2: Drevesni pogled organizacije map Unity projekta. (Vir: lasten)

Poglavje 4

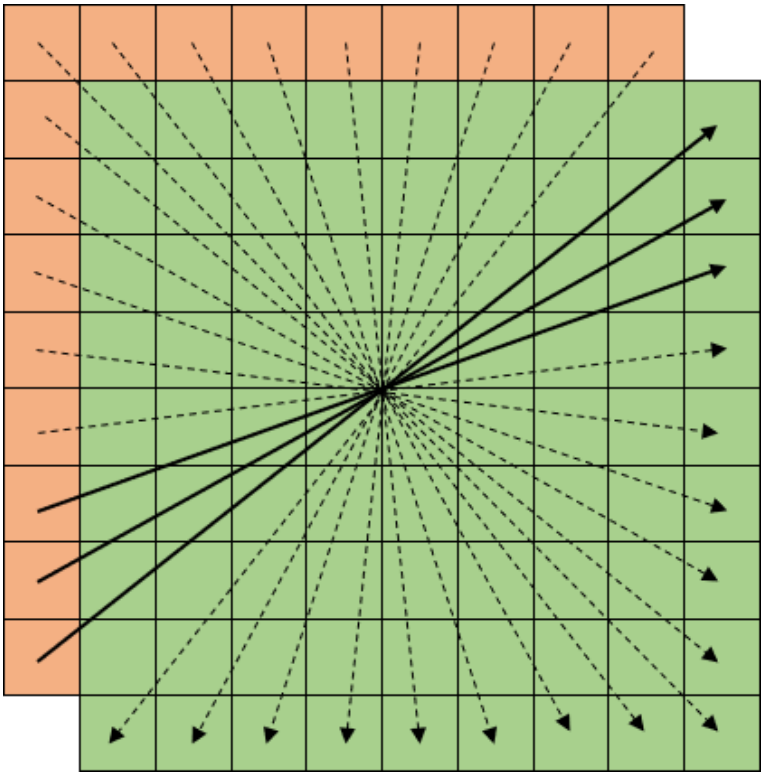
Optimizacija projekta

4.1 Ponovna uporaba delov terena

Celoten teren je po navadi prevelik, zato igra vsak trenutek izrisuje le del terena - to je vidno polje igralca. Z vsakim premikom kamere, je tako potrebno zamenjati vidne elemente, saj se vidno polje s premikom kamere spremeni. To je lahko zelo potratno, zato se zamenjajo le elementi, ki so v skrajnostih vidnega polja. Zamenjava elementov poteka tako, da se vsak del terena, ki je izven vidnega polja, premakne na novo lokacijo, kot prikazuje slika 4.1. Na sliki je novo vidno polje označeno z zeleno barvo, z oranžno pa so obarvani elementi, ki jih je potrebno premakniti in so bili del predhodnega vidnega polja. Po premiku pa se element ponovno inicializira. S ponovno inicializacijo odstranimo potrebo po brisanju in ponovnem kreiranju elementov. Prikaz terena tako deluje zelo tekoče.

4.2 Zmanjševanje količine detajlov v daljavi

Poleg ponovne uporabe delov terena, smo v igro vključili tudi zmanjševanje količine detajlov v daljavi. Zmanjševanje količine detajlov deluje tako, da se ustvari več vidnih polj za igralca. V notranjem (najmanjšem) vidnem polju se izrisujejo delci terena velikosti 1 krat 1, z vegetacijo ter mesti. V prvem

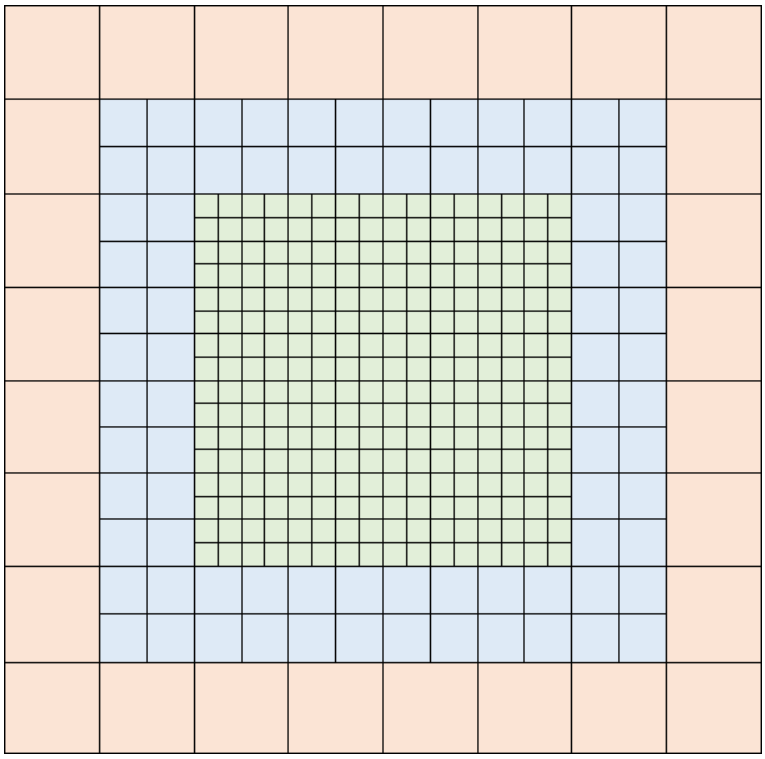


Slika 4.1: Prikaz premikov lokacij delov terena izven vidnega polja. (Vir: lasten)

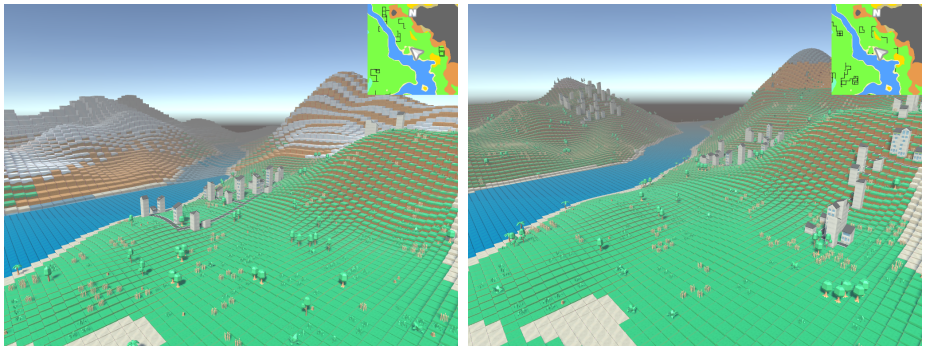
zunanjem (srednje velikem) vidnem polju se izrisujejo delci terena velikosti 2 krat 2, brez vegetacije ter brez mest. V drugem zunanjem (največjem) vidnem polju pa se izrisujejo delci terena velikosti 4 krat 4. V zunanjih vidnih poljih tako potrebujemo izris le enega večjega delca, namesto izrisa več manjših delcev. Poleg tega izrisujemo le najbolj zunanje elemente, saj je teren v notranjosti že izrisan z uporabo manjših delcev. Z uporabo zmanjševanja količine detajlov v daljavi, lahko tako zmanjšamo količino delcev terena, ki jih je potrebno prikazovati vsak trenutek igranja.

Kot primer lahko uporabimo vidno polje velikosti 32 krat 32, ki vsebuje 1024 delcev terena. Z uporabo zmanjševanja količine detajlov, dobimo tri vidna polja - najmanjše (16 krat 16 delcev terena), prvo zunanje (24 krat 24 delcev terena) ter drugo zunanje (32 krat 32 delcev terena). Seštevek

delcev terena, ki jih moramo izrisati, je 364, kar je okoli 2,81 krat manj kot pri implementaciji brez zmanjševanja količine detajlov v daljavi. Primer je prikazan na sliki 4.2, kjer je z zeleno označeno najmanjše vidno polje, z modro prvo zunanje ter z oranžno drugo zunanje vidno polje.
Slika 4.3 prikazuje pogled na teren z in brez uporabe zmanjševanja količine detajlov v daljavi. Sliki se razlikujeta po velikosti kock v daljavi, kar je najbolj vidno na vzpetinah. Razlika med slikama pa je tudi zaradi dveh različnih generacij, zaradi česar sta toplota ter vlaga na vsaki sliki drugačni.



Slika 4.2: Prikaz primera porazdelitve vidnih polj pri uporabi zmanjševanja količine detajlov. (Vir: lasten)



Slika 4.3: Prikaz terena z in brez zmanjševanja količine detajlov v daljavi. (Vir: lasten)

Poglavje 5

Proceduralno generiranje

Proceduralno generiranje sveta je razdeljeno na več delov. Najprej se generira teren, nato se na podlagi višine, vlage ter toplote vsakemu koščku terena določi ekosistem. Sledita generiranje zelenja in mest, ki pri postavitvi upoštevata ekosisteme. Vsak izmed delov se izvaja povsem ločeno in z drugo tehniko proceduralne generacije.

5.1 Generiranje terena

Za izdelavo terena se upoštevajo podatki iz treh različnih manipuliranih Perlinovih šumov: višine, vlažnosti ter toplote. V nadaljevanju bomo predstavili vse tri postopke generiranja manipulacij Perlinovega šuma.

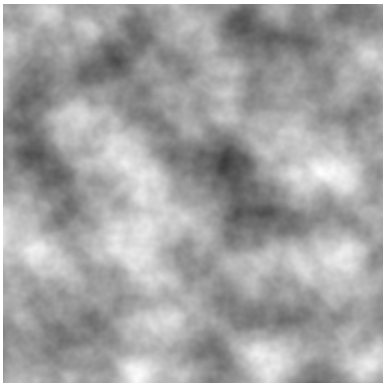
5.1.1 Generiranje višine

Izvirni Perlinov šum se najprej skalira na velikost sveta, ki je v našem primeru 1024 krat 1024, nato pa se nad njim izvedejo sledeče manipulacije:

1. Generacija izvirnika s frekvenco 4.
   Šum frekvence 4 je imel najboljše rezultate za našo velikost sveta, saj je generirani svet deloval realistično z visokimi gorami, planotami ter rekami.

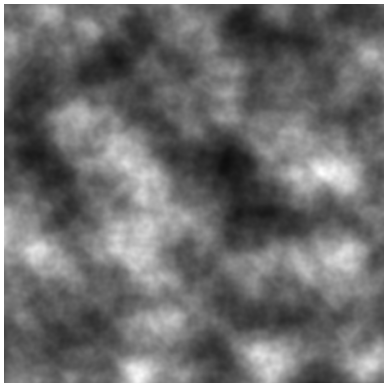
Slika 5.1: Izvirnik generacije višine. Svetlejša barva prikazuje večje vrednosti. (Vir: lasten)

2. Dodajanje 5 oktav.
   Terenu z dodajanjem oktav dodamo definicijo ter "grobost". V našem primeru se je z dodajanjem oktav povečala razgibanost terena.



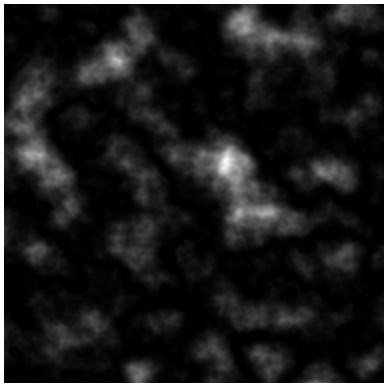
Slika 5.2: Izvirnik generacije višine po dodajanju 5 oktav. (Vir: lasten)

3. Potenciranje s potenco 2.
   Potenciranje šuma spremeni teren v še bolj razgibanega, saj poveča razlike mad najnižjimi ter najvišjimi predeli terena. Tako smo dobili visoke gore ter nizke doline.



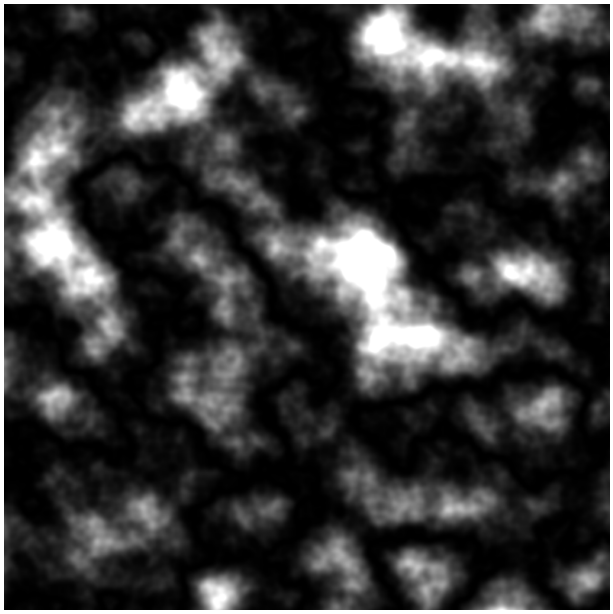
Slika 5.3: Izvirnik generacije višine po dodajanju 5 oktav ter potenciranjem z 2. (Vir: lasten)

4. Absolutna vrednost.
   Z računanjem absolutne vrednosti na terenu smo dobili ostre doline, ki so zelo dober približek rekam.



Slika 5.4: Izvirnik generacije višine po dodajanju 5 oktav, potenciranju z 2 in opravljeni operaciji absolutna vrednost. (Vir: lasten)

5. Množenje rezultata z 1,2.
   Z množenjem preprosto višamo teren - zmanjšali smo količino jezer, morij ter rek na terenu.



Slika 5.5: Izvirnik generacije višine po dodajanju 5 oktav, potenciranju z 2, opravljeni operaciji absolutna vrednost in množenjem z 1,2. To je končna višinska slika terena. (Vir: lasten)

5.1.2 Generiranje vlažnosti

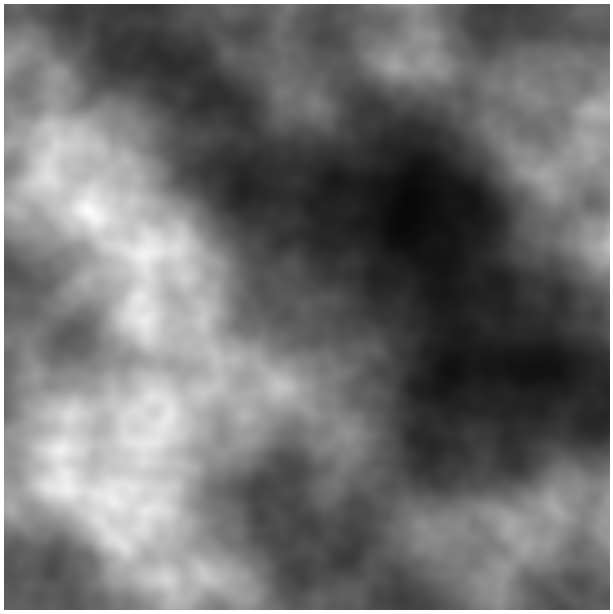
Sledi generiranje vlažnosti za vsak košček terena. Vlažnost se generira po podobnem postopku kot višina:

1. Generacija šuma s frekvenco 2.
   Izbrali smo manjšo frekvecnco kot pri generaciji višine, saj smo želeli dobiti večje površine vlažnih predelov. Želeli smo, da naša generacija sveta izgleda realistično, kot je v naravi sami, in da so blizuležeči višinski deli povezani med seboj s podobnimi ekosistemi.

2. Dodajanje 3 oktav.
   Dodali smo manj oktav kot pri generaciji višin, saj smo pri dodajanju več oktav, dobili nenaravne prehode iz vlažnih v suhe predele sveta.

3. Potenciranje s potenco 2,5.
   S potenciranjem smo bolj definirali prehode med vlažnimi in suhimi predeli sveta.

4. Množenje rezultata z 0,75.
   Zaradi prevelike količine vlažnega terena smo vlažnost zmanjšali za faktor, ki je prinesel najbolj naraven izgled generiranega sveta.



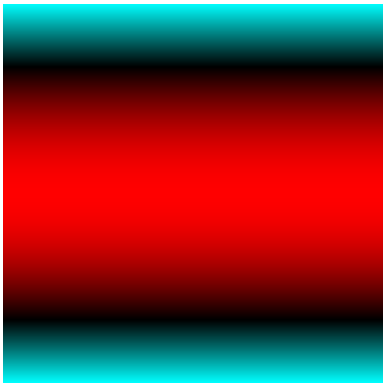
Slika 5.6: Končna slika vlažnosti terena. Svetlejša barva prikazuje višje vrednosti. (Vir: lasten)

5.1.3 Generiranje toplote

Nazadnje generiramo toploto. Za generiranje toplote se uporablja nekoliko drugačen postopek. Generacije ne začnemo s Perlinovim šumom, temveč s sinusoido. Z uporabo sinusoide lahko simuliramo severni in južni pol, ki sta mrzla, ter ekvator, ki je toplejši.

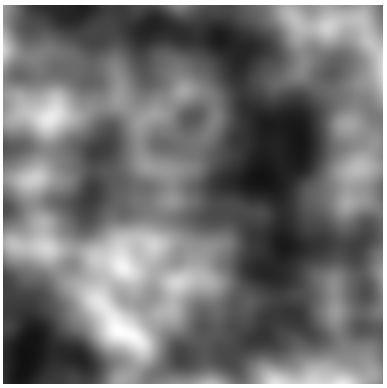
1. Generacijo začnemo s sinusoido, pri kateri se vrednosti nahajajo na intervalu [-2, 2]. Na sliki rdeča barva predstavlja pozitivne vrednosti,

modra barva pa negativne vrednosti.



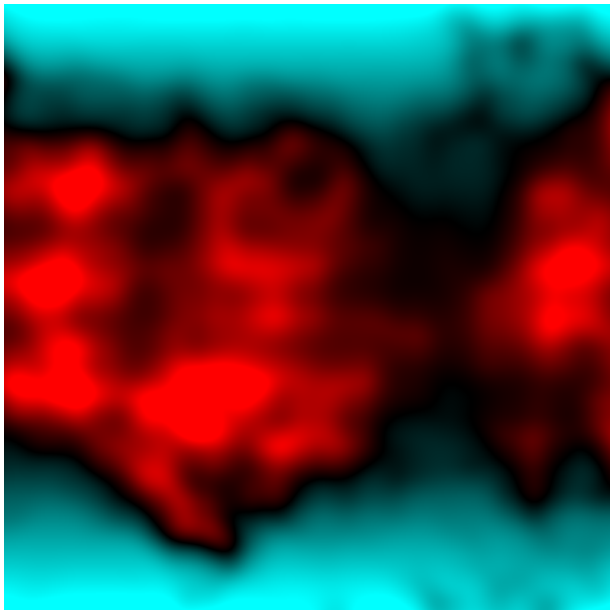
Slika 5.7: Sinusoida kot izvirnik generiranja toplote. Rdeča barva predstavlja pozitivne vrednosti, modra barva predstavlja negativne vrednosti. (Vir: lasten)

2. Generira se Perlinov šum, ki se uporabi za maskiranje prelivanja visokih ter nizkih temperatur. Uporaba samo sinusoide izgleda zelo nenaravno, saj se je v našem primeru led na severnem in južnem polu od morja ločil z ravno črto.



Slika 5.8: Perlinov šum za uporabo pri maskiranju izvirnika generiranja toplote. (Vir: lasten)

3. Sinusoido iz prve točke smo množili s Perlinovim šumom iz druge točke. Tako smo dobili naraven efekt prelivanja toplot.



Slika 5.9: Končna toplotna slika po maskiranju izvirnika s Perlinovim šumom. (Vir: lasten)

5.2 Ekosistemi

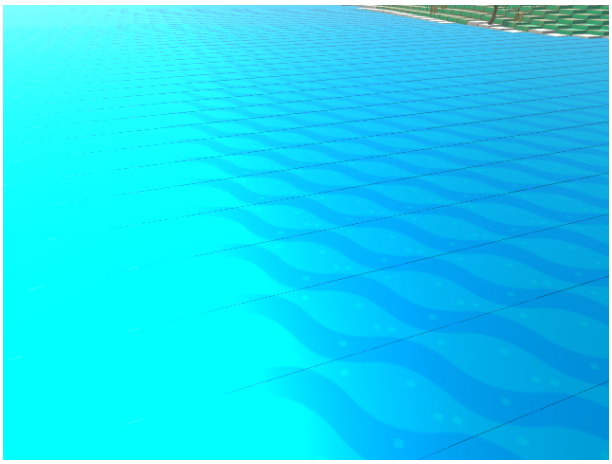
Glede na višino, vlažnost ter toploto se za vsak košček terena določi ekosistem. Vseh ekosistemov je 13. Višina vsebuje vrednosti na intervalu [0, 1], vlažnost na intervalu [0, 1] ter toplota na intervalu [-1, 1].
Vsak ekosistem ima svojo barvo tal ter barvo zelenja, ki se generira na njem. Ta barva je enotna za cel ekosistem. Vseh modelov zelenja je 20. En model se uporabi večkrat, pri čemer se glede na ekosistem zamenja le barva listja. S tem optimiziramo velikost in hitrost delovanja igre, saj lahko isti model uporabimo v več ekosistemih. Možne nadgradnje barv ekosistemov so ustvarjanje variance v barvah znotraj posameznega ekosistema in prelivanje barv iz enega v drug ekosistem. V diplomskem delu tega nismo implementi-

rali, saj smo želeli omejiti izbor barv, da se vidi meja med ekosistemi.
Meje višine, vlažnosti ter toplote smo za vsak ekosistem pridobili s poskušanjem. Izbrali smo tiste meje, ki nam najbolje generirajo vsak posamezen ekosistem. Naš cilj pa je bil imitirati planet Zemlja.

Morje

Morje se generira pri naslednji kombinaciji parametrov:

- toplota [-0˙3, 1], višina [0, 0˙03], vlažnost [0, 1]

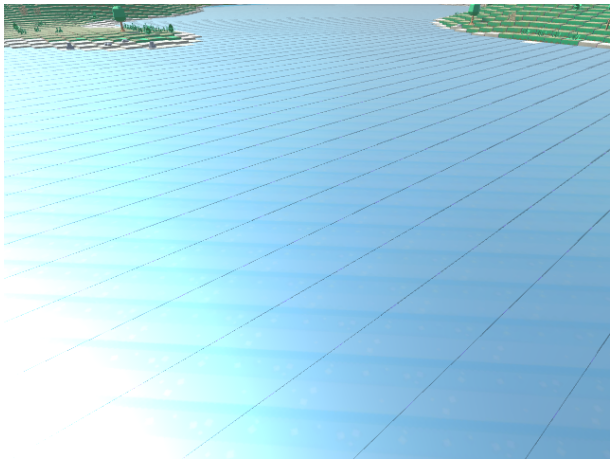


Slika 5.10: Prikaz generiranega morja. (Vir: lasten)

Led

Led se generira pri naslednji kombinaciji parametrov:

- toplota [-1, -0˙3), višina [0, 0˙03], vlažnost [0, 1]

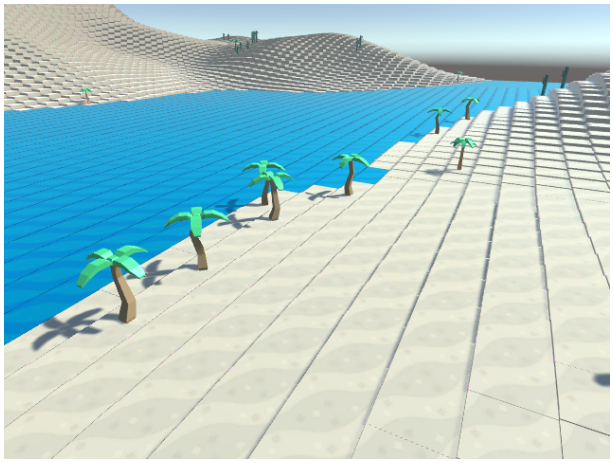


Slika 5.11: Prikaz generiranega ledu. (Vir: lasten)

Plaža

Led se generira pri naslednji kombinaciji parametrov:

- toplota [-0˙2, 1), višina (0˙03, 0˙05], vlažnost [0, 1]

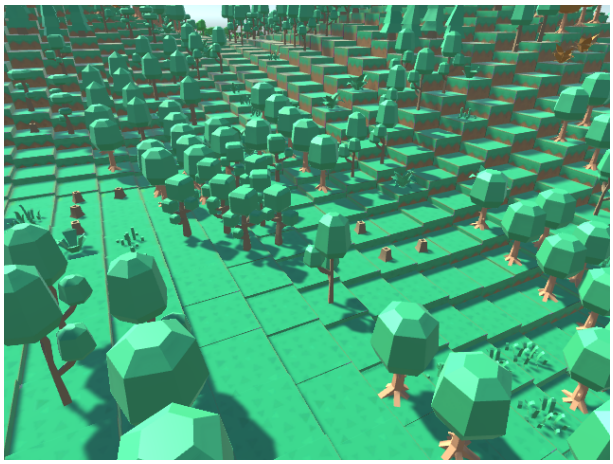


Slika 5.12: Prikaz generirane plaže. (Vir: lasten)

Listnati gozd

Listnati gozd se generira pri naslednjih kombinacijah parametrov:

- toplota (0˙9, 1], višina (0˙3, 0˙5], vlažnost [0˙4, 1]
- toplota [-0˙3, 0˙9], višina (0˙25, 0˙4], vlažnost [0˙5, 1]

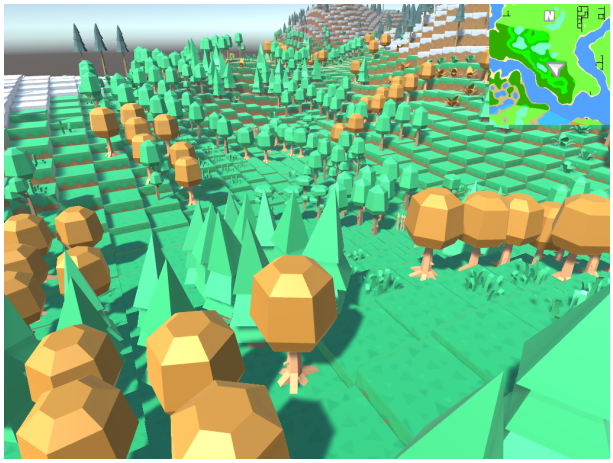


Slika 5.13: Prikaz generiranega listnatega gozda. (Vir: lasten)

Mešani gozd

Mešani gozd se generira pri naslednjih kombinacijah parametrov:

- toplota (0˙9, 1], višina (0˙5, 0˙7], vlažnost [0˙5, 1]
- toplota [-0˙3, 0˙9], višina (0˙4, 0˙5], vlažnost [0˙5, 1]

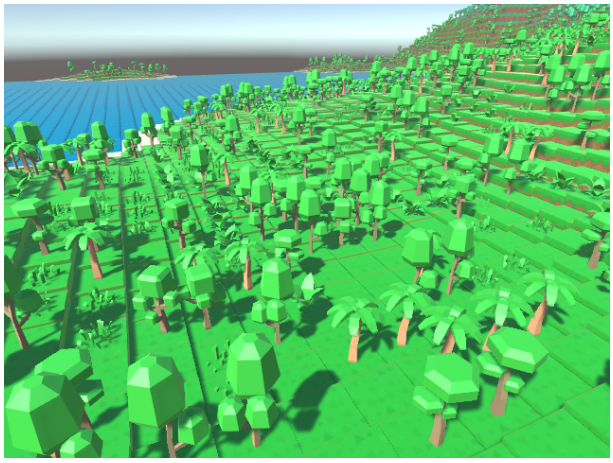


Slika 5.14: Prikaz generiranega mešanega gozda. (Vir: lasten)

Tropski deževni gozd

Tropski deževni gozd se generira pri naslednjih kombinacijah parametrov:

- toplota (0˙9, 1], višina (0˙05, 0˙3], vlažnost [0˙4, 1]
- toplota [-0˙3, 0˙9], višina (0˙05, 0˙25], vlažnost [0˙5, 1]

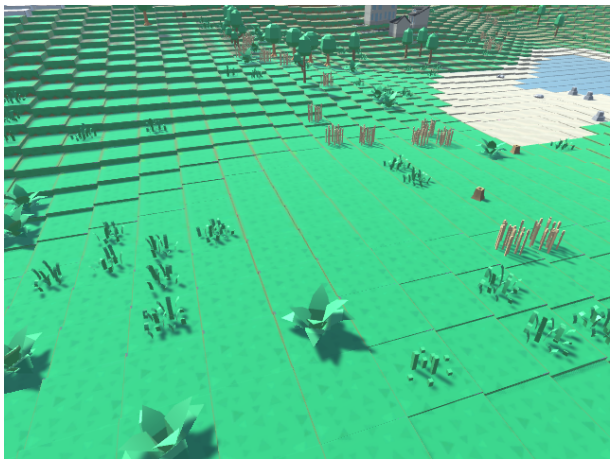


Slika 5.15: Prikaz generiranega tropskega deževnega gozda. (Vir: lasten)

Travnik

Travnik se generira pri naslednjih kombinacijah parametrov:

- toplota [-1, -0˙3), višina (0˙05, 0˙3], vlažnost [0˙5, 1]
- toplota (0˙9, 1], višina (0˙05, 0˙5], vlažnost [0˙3, 0˙4)
- toplota [-0˙3, 0˙9], višina (0˙25, 0˙5], vlažnost [0˙15, 0˙5)
- toplota [-0˙3, 0˙9], višina (0˙05, 0˙25], vlažnost [0˙2, 0˙5)

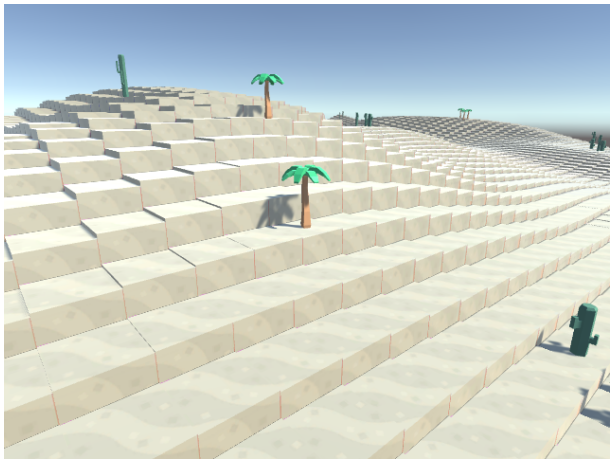


Slika 5.16: Prikaz generiranega travnika. (Vir: lasten)

Puščava

Puščava se generira pri naslednjih kombinacijah parametrov:

- toplota (0˙9, 1], višina (0˙05, 0˙3], vlažnost [0, 0˙3)
- toplota [-0˙3, 0˙9], višina (0˙25, 0˙5], vlažnost [0, 0˙15)
- toplota [-0˙3, 0˙9], višina (0˙05, 0˙25], vlažnost [0, 0˙2)

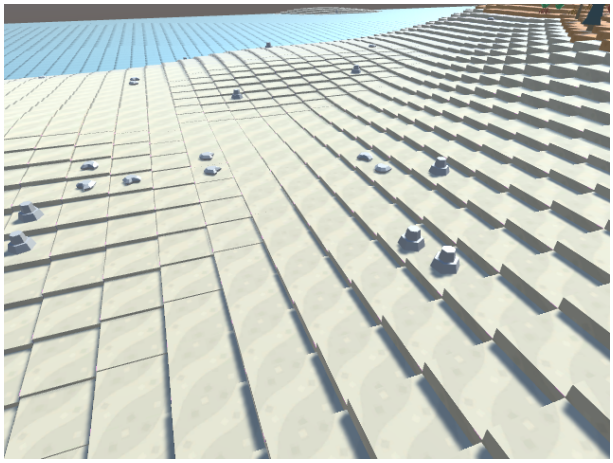


Slika 5.17: Prikaz generirane puščave. (Vir: lasten)

Hladna puščava

Hladna puščava se generira pri naslednjih kombinacijah parametrov:

- toplota [-1, 0˙2], višina (0˙03, 0˙05], vlažnost [0, 1]
- toplota [-1, -0˙3), višina (0˙3, 0˙7], vlažnost [0, 0˙15)
- toplota [-1, -0˙3), višina (0˙05, 0˙3], vlažnost [0, 0˙2)
- toplota (0˙9, 1], višina (0˙5, 0˙7], vlažnost [0, 0˙3)
- toplota [-0˙3, 0˙9], višina (0˙5, 0˙7], vlažnost [0, 0˙15)

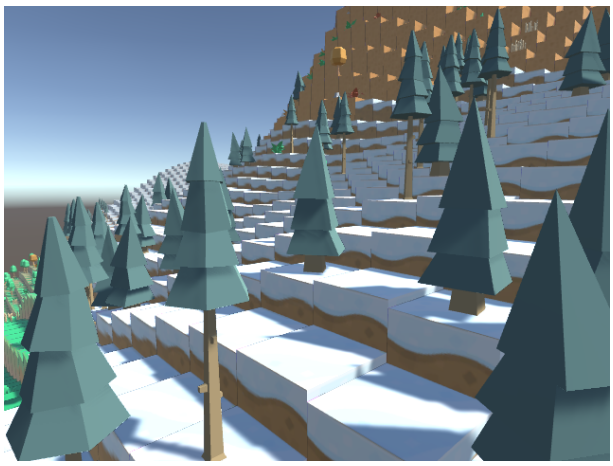


Slika 5.18: Prikaz generirane hladne puščave. (Vir: lasten)

Tajga

Tajga se generira pri naslednjih kombinacijah parametrov:

- toplota [-1, -0˙3), višina (0˙5, 0˙7], vlažnost [0˙3, 1]
- toplota [-1, -0˙3), višina (0˙3, 0˙5], vlažnost [0˙5, 1]
- toplota [-0˙3, 0˙9], višina (0˙4, 0˙5], vlažnost [0˙3, 1]

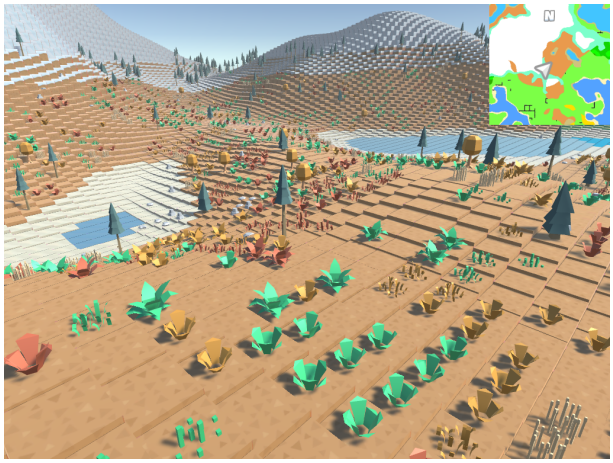


Slika 5.19: Prikaz generirane tajge. (Vir: lasten)

Tundra

Tundra se generira pri naslednjih kombinacijah parametrov:

- toplota [-1, -0˙3), višina (0˙5, 0˙7], vlažnost [0˙15, 0˙3)
- toplota [-1, -0˙3), višina (0˙3, 0˙5], vlažnost [0˙15, 0˙5)
- toplota [-1, -0˙3), višina (0˙05, 0˙3], vlažnost [0˙2, 0˙5)
- toplota (-0˙9, 1], višina (0˙5, 0˙7], vlažnost [0˙2, 0˙3)
- toplota [-0˙3, 0˙9], višina (0˙7, 1], vlažnost [0˙2, 0˙3)
- toplota [-0˙3, 0˙9], višina (0˙5, 0˙7], vlažnost [0˙15, 0˙3)

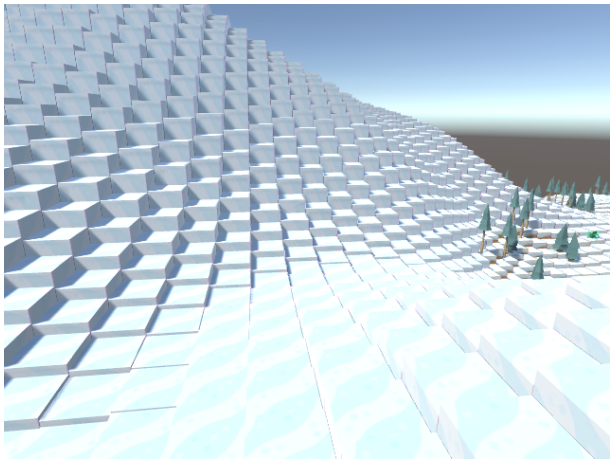


Slika 5.20: Prikaz generirane tundre. (Vir: lasten)

Sneg

Sneg se generira pri naslednjih kombinacijah parametrov:

- toplota [-1, -0˙3), višina (0˙7, 1], vlažnost [0˙2, 1]
- toplota (-0˙9, 1], višina (0˙7, 1], vlažnost [0˙4, 1]
- toplota [-0˙3, 0˙9], višina (0˙7, 1], vlažnost [0˙4, 1]

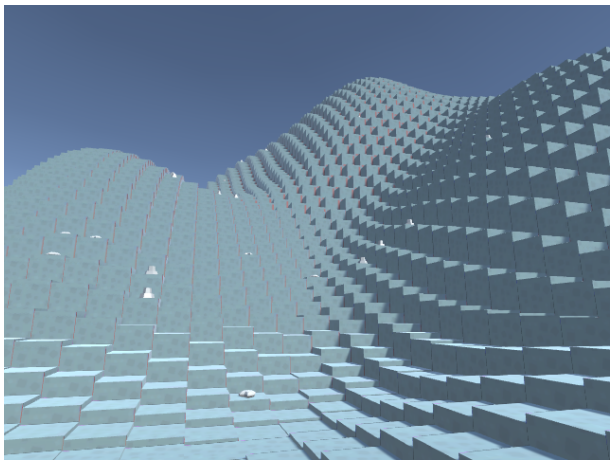


Slika 5.21: Prikaz generiranega snega. (Vir: lasten)

Gorski vrh

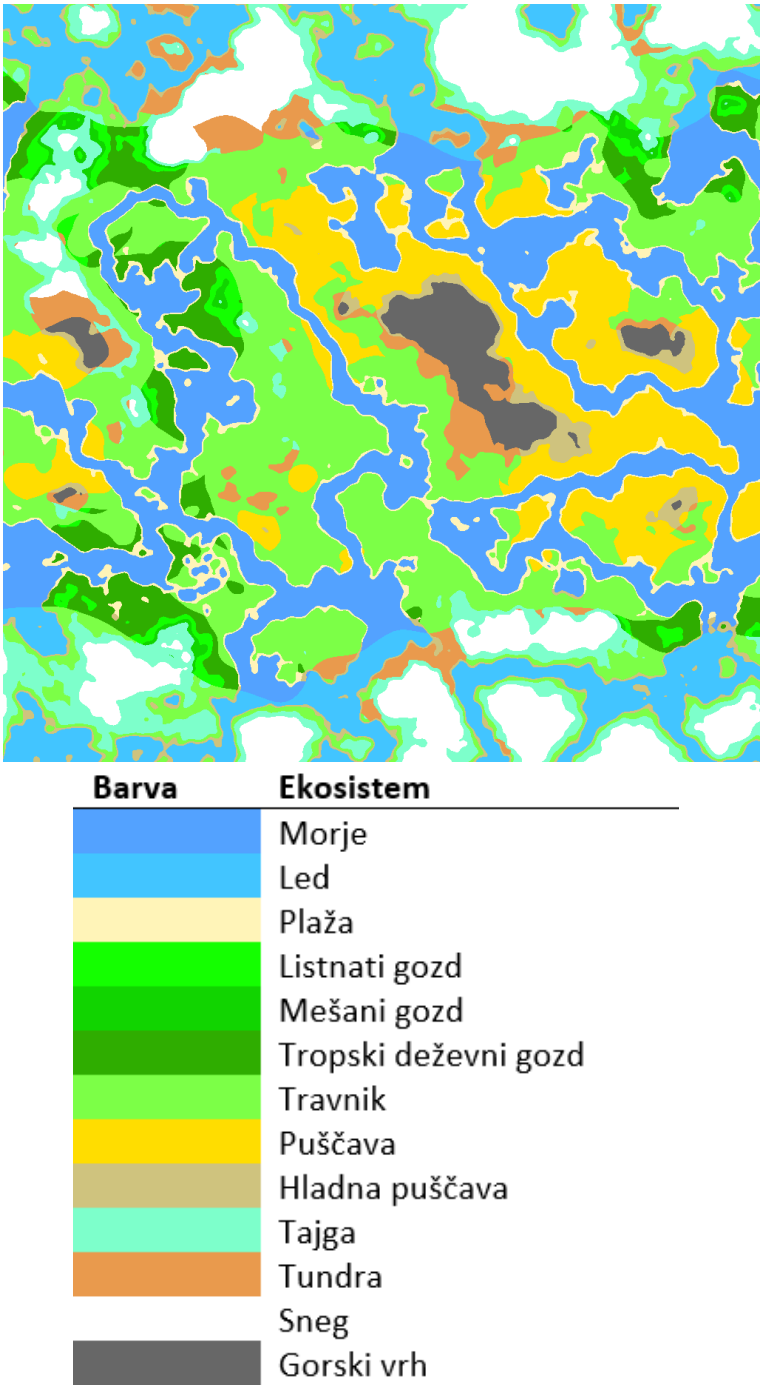
Gorski vrh se generira pri naslednjih kombinacijah parametrov:

- toplota [-1, 0˙9), višina (0˙7, 1], vlažnost [0, 0˙2)
- toplota (-0˙9, 1], višina (0˙7, 1], vlažnost [0, 0˙4)



Slika 5.22: Prikaz generiranih gorskih vrhov. (Vir: lasten)

Končni zemljevid sveta, z legendo barv ekosistemov:



Slika 5.23: Prikaz generiranega zemljevida sveta z legendo. (Vir: lasten)

5.3 Generiranje zelenja

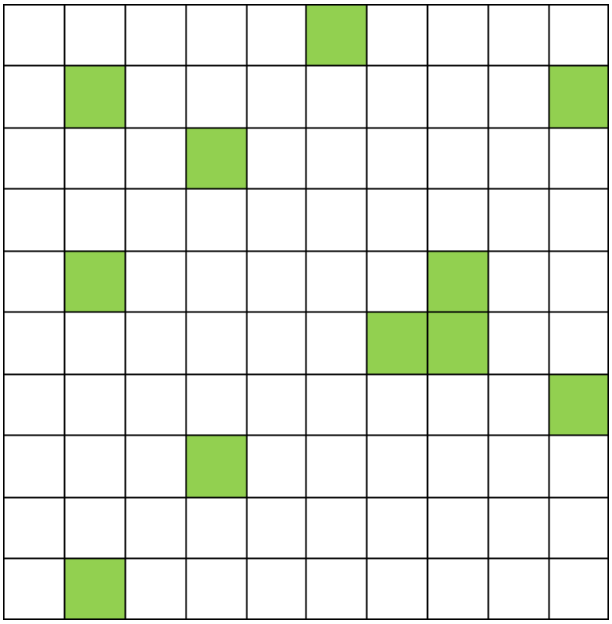
Generiranje zelenja poteka v več iteracijah. V prvi generiramo tako imenovana 'semena', ki v naslednjih generacijah služijo kot začetna točka gruč istovrstnega zelenja. Zelenje pri generaciji upošteva ekosistem, kar pomeni, da se na primer v puščavi generirajo kaktusi ter palme, v tajgi pa iglasta drevesa. Vsak ekosistem ima drugačna pravila za postavitev ter množenje dreves. V nekaterih ekosistemih prevladujejo posamezna drevesa, v drugih pa gruče dreves - gozdovi.
Algoritem za generiranje zelenja se uporablja tudi za generiranje raznih drugih predmetov v ekosistemih, kot so kamenje, podrta drevesa ter trava.

1. V prvi iteraciji generiranja zelenja ima vsaka celica enako verjetnost, da se na njej generira zelenje. Če je na voljo več različnih vrst zelenja, se postopek ponovi za vsako vrsto. Če pa je na voljo več primerkov znotraj ene vrste, se izmed njih izbere naključni primerek. V spodnjem primeru se uporablja ena vrsta ter en primerek zelenja.

0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1

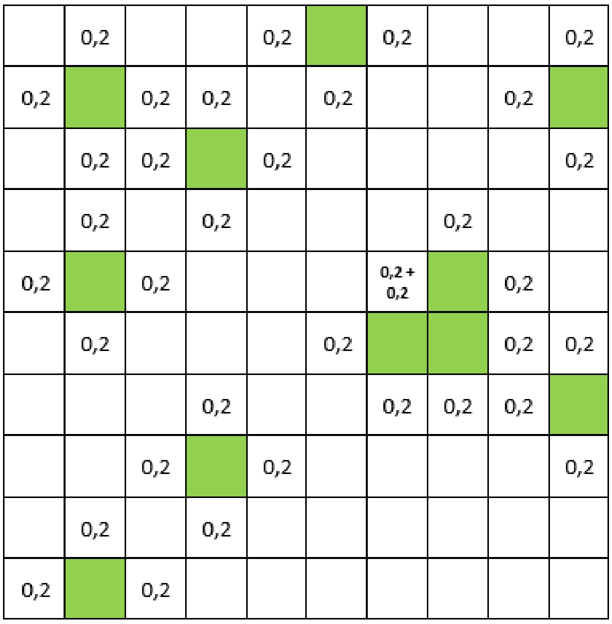
Slika 5.24: Prikaz verjetnosti za generacijo zelenja v prvi iteraciji. (Vir: lasten)

2. Naključna generacija je v spodnjem primeru določila 11 celic, na katerih se bo nahajalo zelenje.



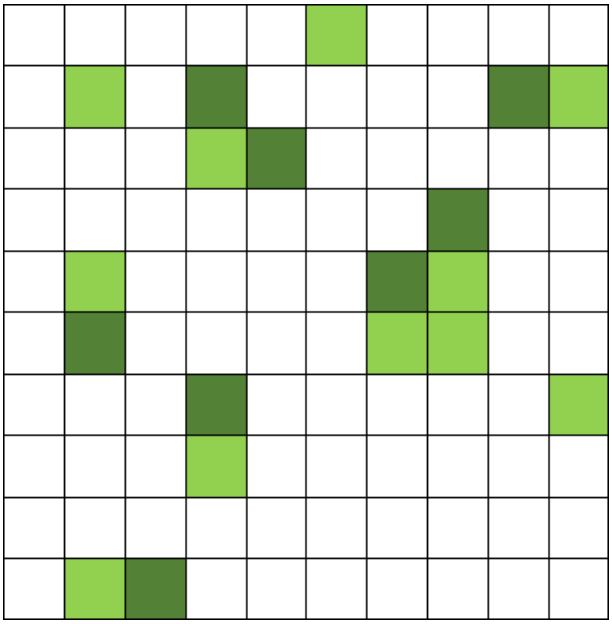
Slika 5.25: Prikaz primera terena po opravljeni prvi iteraciji generacije zelenja. (Vir: lasten)

3. Celice iz prejšnjega koraka delujejo kot 'semena' za naslednjo iteracijo. Vsaka vrsta zelenja ima vnaprej določeno verjetnost, da se množi na sosednje celice, vendar se pri tem lahko 'seme' množi le v isto vrsto.



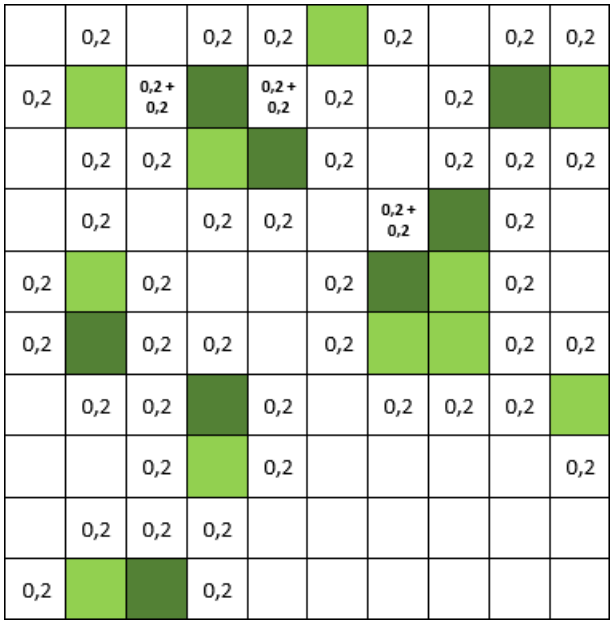
Slika 5.26: Prikaz verjetnosti za generacijo zelenja v drugi iteraciji. (Vir: lasten)

4. Rezultat množenja zelenja iz prejšnjega koraka. Novo zelenje je označeno s temno zeleno barvo.



Slika 5.27: Prikaz primera terena po opravljeni drugi iteraciji generacije zelenja. (Vir: lasten)

5. Postopek množenja zelenja se ponavlja za vnaprej določeno število iteracij, pri čemer vse že generirano zelenje deluje kot 'seme' za naslednjo iteracijo.



Slika 5.28: Prikaz verjetnosti za generacijo zelenja v tretji iteraciji generacije zelenja. (Vir: lasten)

Z večanjem števila iteracij množenja večamo posamezne gruče zelenja. Iz enega 'semena' se lahko v nekaj iteracijah ustvari gozd. V nasprotnem primeru, če želimo visoko raznolikost zelenja, zmanjšamo število iteracij ter zvečamo verjetnost generacije zelenja v prvi iteraciji.

5.4 Generiranje naselij

Generacija naselij poteka v dveh korakih. V prvem koraku se na teren generirajo ceste, v drugem koraku pa se ob cestah generirajo naključne stavbe.

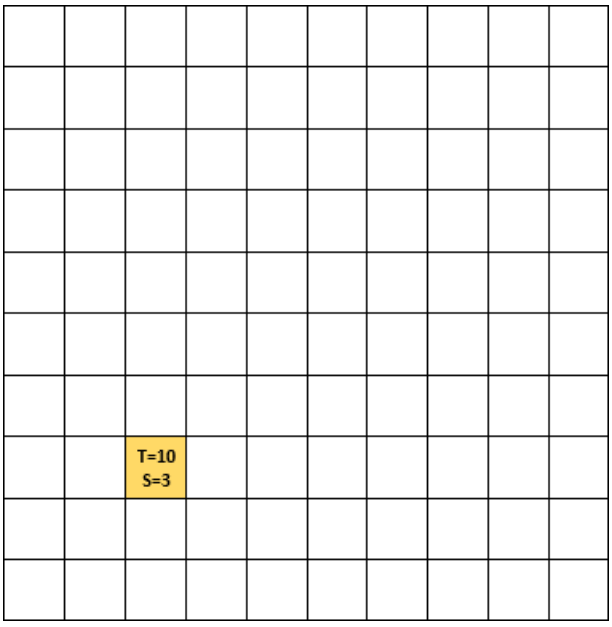
5.4.1 Generiranje cest

Generiranje cest poteka po podobnem postopku kot generiranje zelenja, le da je iteriranje kompleksnejše, saj želimo generirati podobo mesta. Pomembno

je omeniti, da imajo ceste pri postavitvi prednost pred zelenjem. Celicam, kjer zelenje že stoji in se nanje želi generirati cesta, zelenje izbrišemo.

1. Prva iteracija generiranja cest je zelo podobna prvi iteraciji generiranja zelenja, pri čemer so verjetnosti za postavitev ceste bistveno manjše od verjetnosti za postavitev zelenja, saj je v svetu število mest bistveno manjše od površine zelenja.

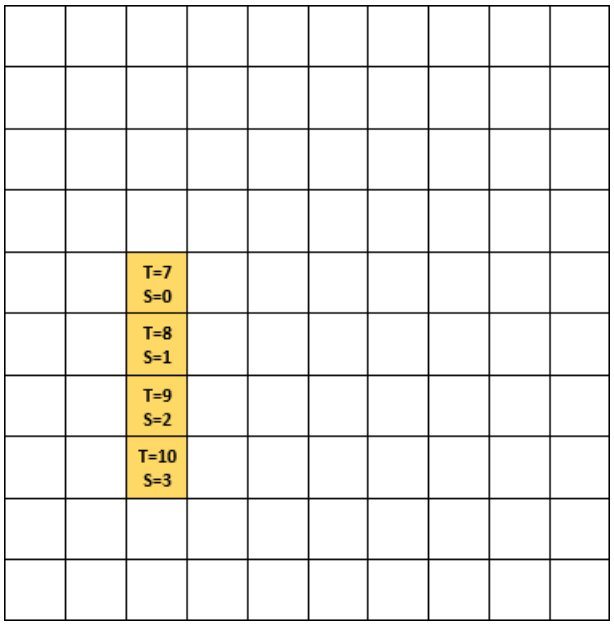
2. Po prvi iteraciji se je cesta generirala le na eni celici. Vsaka celica v tem koraku predstavlja 'začetek' ceste nekega naselja. Vsaki celici se določi maksimalno dolžino ceste (na sliki označeno s T), naključno smer širitve ter dolžino širitve (na sliki označeno s S).



Slika 5.29: Prikaz primera terena po opravljeni prvi iteraciji generacije cest. (Vir: lasten)

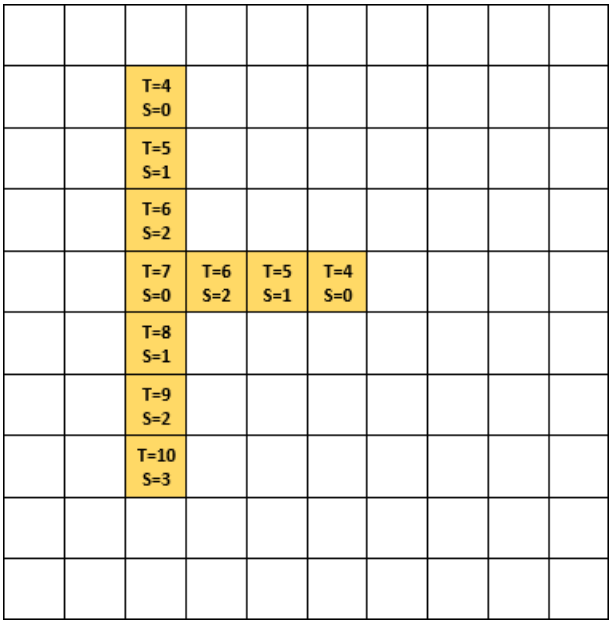
3. Ko cesta konča s prvo širitvijo, se določi njena nadaljnja pot, če to dopušča maksimalna dolžina ceste. Na voljo so štiri možnosti: nadalje-

vanje naravnost, sprememba smeri, trosmerno križišče ter štirismerno križišče.



Slika 5.30: Prikaz primera terena po opravljeni prvi širitvi cest. (Vir: lasten)

4. Rezultat po drugi iteraciji gradnje cest.
   V našem primeru se je algoritem z generiranjem naključnega števila odločil za trosmerno križišče.



Slika 5.31: Prikaz primera terena po opravljeni drugi iteraciji širjenja cest. (Vir: lasten)

5. Rezultat po tretji iteraciji gradnje cest.
   Gradnja cest se konča, ko vse celice dosežejo maksimalno dolžino.

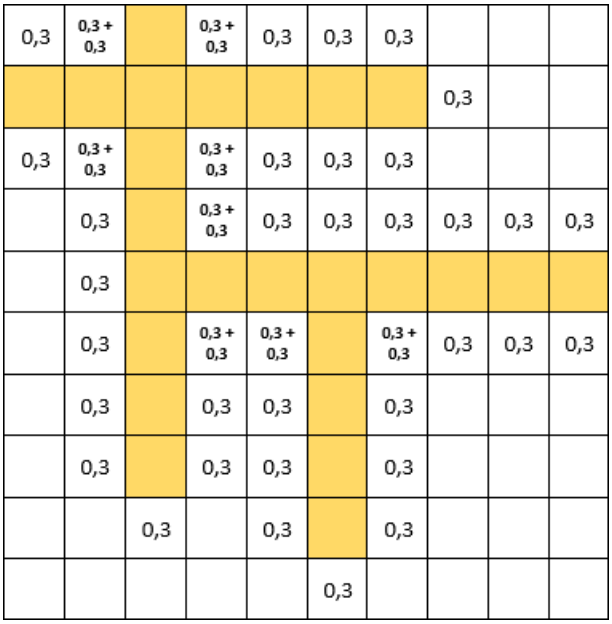
		T=3 S=3							
T=2 S=2	T=3 S=3	T=4 S=0	T=3 S=3	T=2 S=2	T=1 S=1	T=0 S=0			
		T=5 S=1							
		T=6 S=2							
		T=7 S=0	T=6 S=2	T=5 S=1	T=4 S=0	T=3 S=3	T=2 S=2	T=1 S=1	T=0 S=0
		T=8 S=1			T=3 S=3				
		T=9 S=2			T=2 S=2				
		T=10 S=3			T=1 S=1				
					T=0 S=0				

Slika 5.32: Prikaz primera terena po opravljeni tretji iteraciji širjenja cest. (Vir: lasten)

5.4.2 Postavitev zgradb

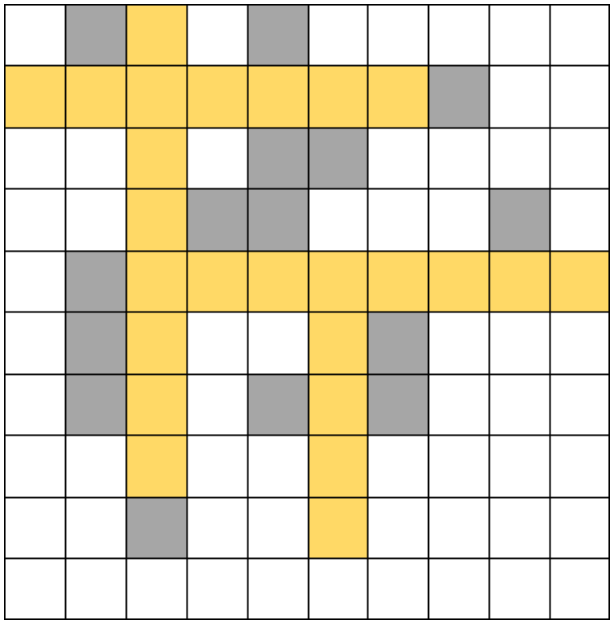
Po končanem generiranju cest je ob ceste potrebno postaviti ter generirati zgradbe. Postavljanje zgradb določa le lokacijo zgradbe in ne njenega izgleda. Postopek generiranja izgleda je opisan v naslednjem poglavju.

1. Generacija postavitve zgradb je podobna drugi iteraciji generiranja zelenja, le da se za 'semena' uporabijo postavljene ceste. Vsaka cesta ima verjetnost, da se ob njej postavi zgradba. Pomembno je omeniti še, da imajo zgradbe prioriteto nad zelenjem, kar pomeni, da celicam, ki že vsebujejo zelenje, le-to izbrišemo, če se nanje želi postaviti zgradba.



Slika 5.33: Prikaz verjetnosti za postavitev zgradb. (Vir: lasten)

2. Izvede se le ena iteracija postavitve zgradb. Na spodnji sliki so s sivo obarvane celice, ki vsebujejo zgradbo.



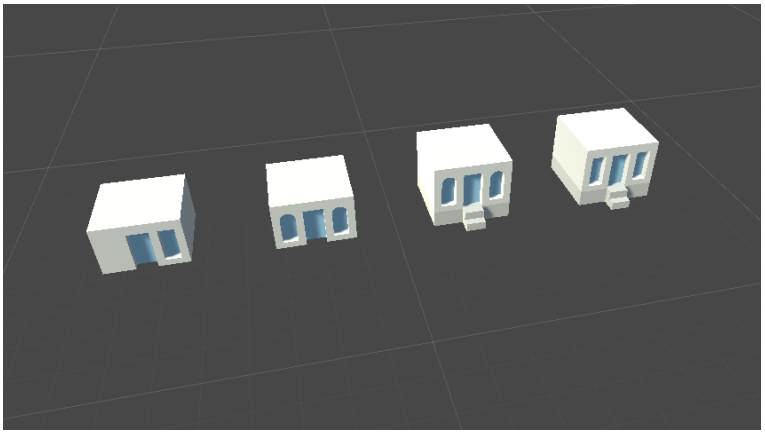
Slika 5.34: Prikaz primera postavitve zgradb. (Vir: lasten)

5.4.3 Gradnja zgradb

V prejšnjem poglavju smo opisali generacijo postavitve zgradb. Zgradbe moramo še naključno zgraditi. Vsak tip mesta vsebuje podatke o tem, kako naj zgradbe izgledajo. Vsaka zgradba je sestavljena iz enega ali več nadstropij, ki se naključno generirajo.

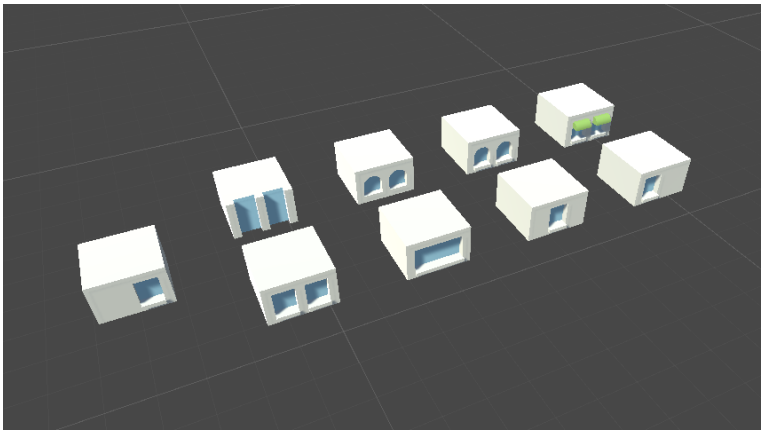
Vsaka zgradba vsebuje pritličje - v primeru stolpnic je to nadstropje z vhodom, v primeru šotorov pa je to kar cel šotor. Večnadstropne zgradbe so sestavljene še iz vmesnih nadstropij in strehe.

Generacija zgradbe se začne z definicijo maksimalne višine ter modelov nadstropij, ki jih lahko uporablja. V spodnjem primeru so predstavljeni modeli nadstropij za generiranje stolpnice. Za prvo nadstropje (vhod) so na voljo 4 različni modeli:



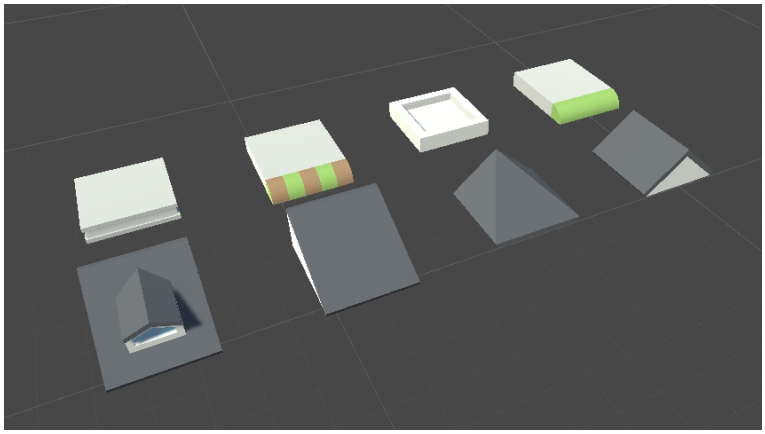
Slika 5.35: Prikaz modelov prvih nadstropij zgradb. (Vir: lasten)

Nadaljujemo z generacijo vmesnih nadstropij. Modeli se izbirajo naključno izmed naslednjih 9 modelov:



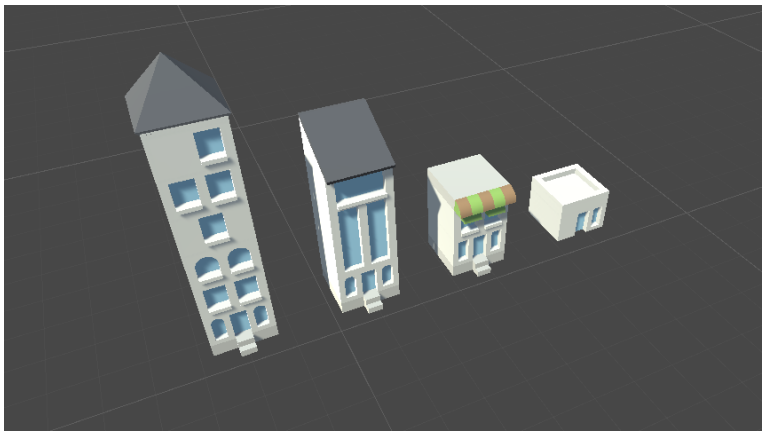
Slika 5.36: Prikaz modelov vmesnih nadstropij zgradb. (Vir: lasten)

Nazadnje na zgradbo postavimo še streho. Na voljo je 8 modelov:



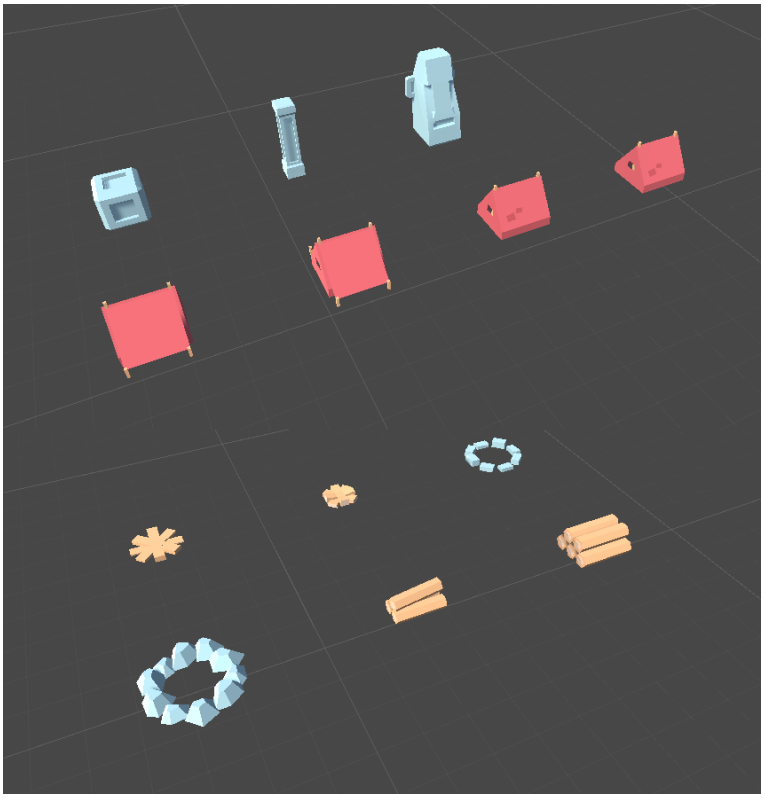
Slika 5.37: Prikaz modelov streh zgradb. (Vir: lasten)

Spodnja slika prikazuje (od desne proti levi) zgradbo višine 2, zgradbo višine 3, zgradbo višine 5 ter zgradbo višine 7. Višina je seštevek pritličja, vseh nadstropij ter strehe.



Slika 5.38: Prikaz primerov generiranih zgradb različnih višin. (Vir: lasten)

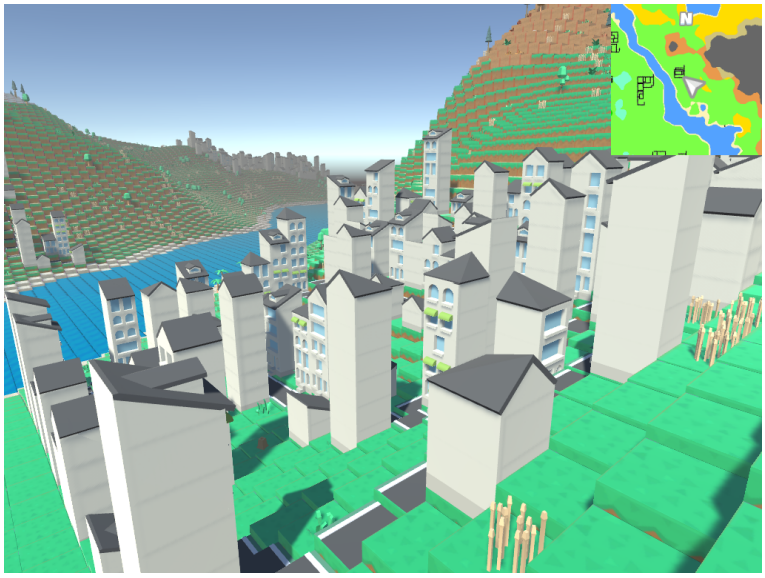
Posebna skupina zgradb so zgradbe višine 1. Pri teh zgradbah se generira samo pritličje. V naši generaciji sveta, kot primer zgradb višine 1, najdemo šotore različnih oblik, kamnite skulpture, ognjišča ter kupe debel.



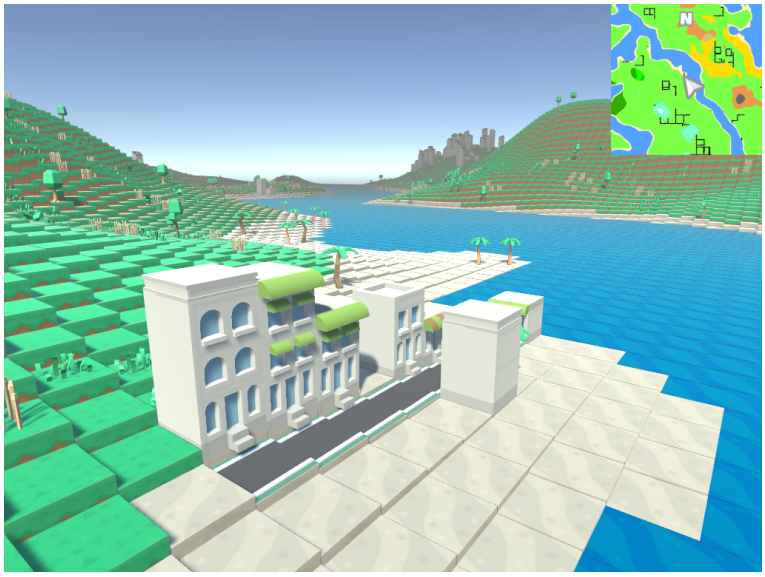
Slika 5.39: Prikaz modelov zgradb višine 1. (Vir: lasten)

5.4.4 Implementirani tipi naselij

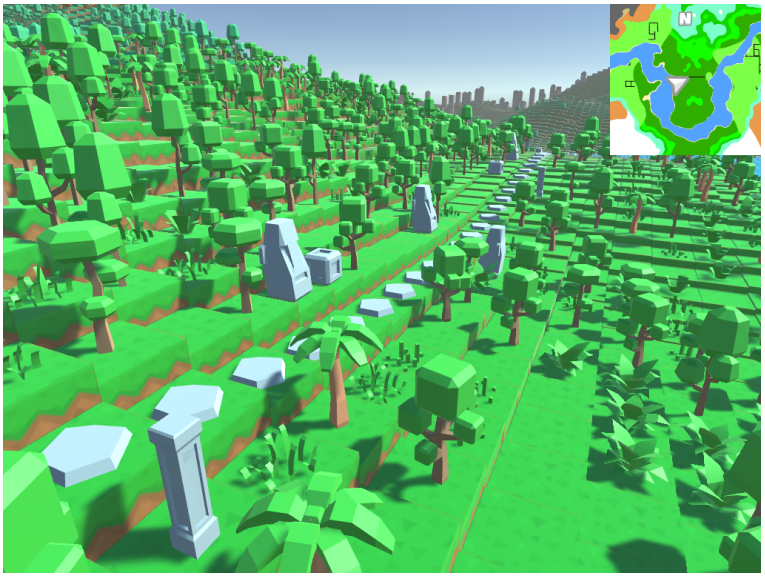
V generaciji sveta je vključenih več naselij, ki se generirajo v določenih ekosistemih. Večja mesta se generirajo na travniku, manjša obalna mesta se generirajo na plaži, kampi se generirajo v listnatem gozdu in gozdu, ostanki kamnitih mest pa se generirajo v tropskem deževnem gozdu.



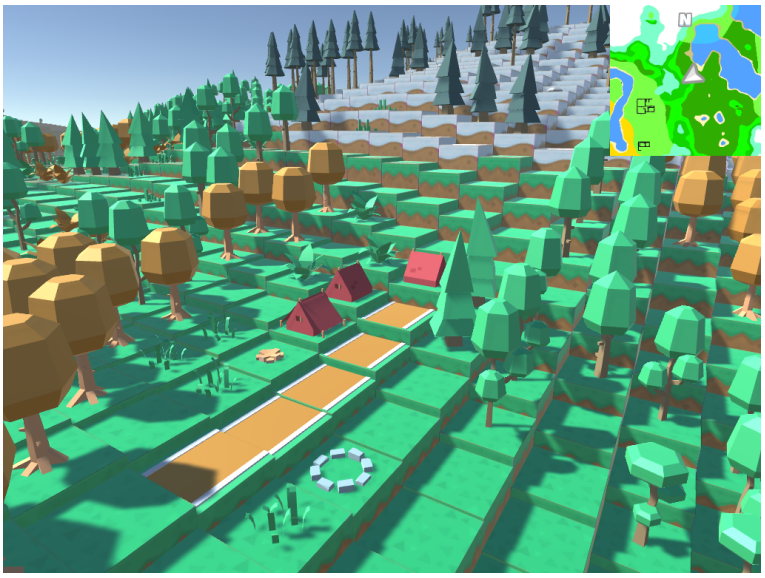
Slika 5.40: Primer večjega mesta. (Vir: lasten)



Slika 5.41: Primer obalnega naselja. (Vir: lasten)



Slika 5.42: Primer arheoloških ostankov kamnitega mesta v tropskem deževnem gozdu. (Vir: lasten)



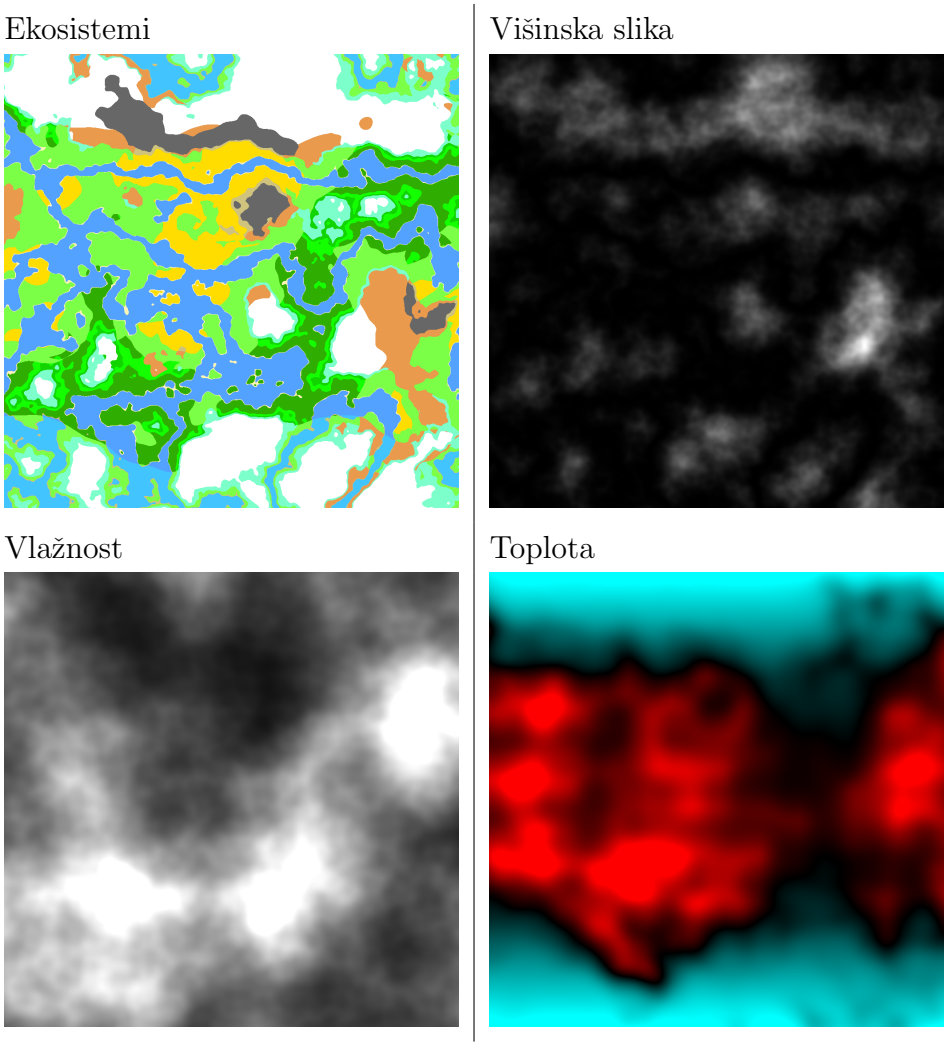
Slika 5.43: Primer kampa. (Vir: lasten)

5.5 Prikaz rezultatov

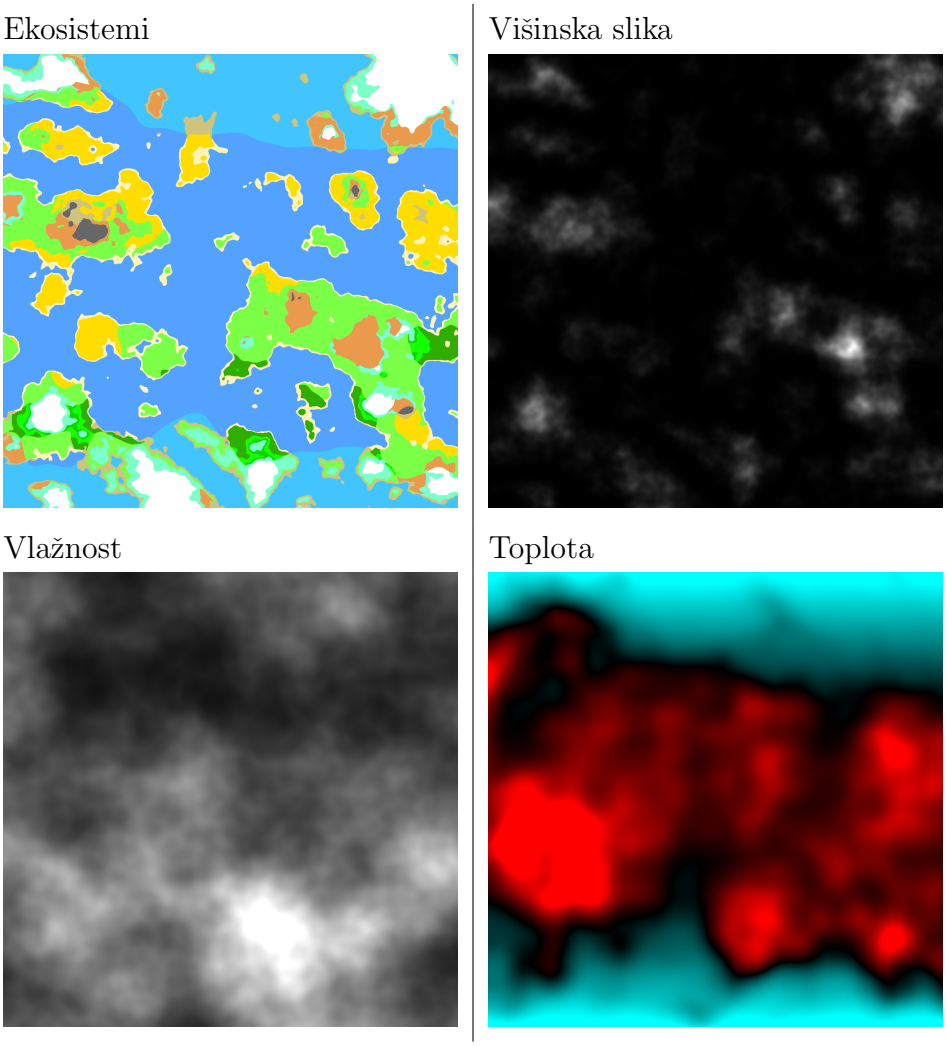
Ocena rezultatov diplomske naloge glede na kriterije iz poglavja 2, ki povzemajo zaželjene lastnosti proceduralne generacije:

- Hitrost generiranja - Menim, da smo osvojili dovolj dobro hitrost generiranja. Svet velikosti 2048 krat 2048 delcev se popolnoma zgenerira na povprečnem računalniku v manj kot sekundi.
- Zanesljivost generacije - Z večkratnim poskušanjem generiranja nam je vedno uspelo generirati uporaben teren. Vsi tereni so bili enake kvalitete.
- Nadzorljivost - Vse nastavitve naključne generacije terena je mogoče spreminjati skozi kodo, ali pa preko uporabniškega vmesnika igralnega pogona Unity.
- Izraznost in raznolikost - Pri vsaki generaciji terena je rezultat podoben, saj so pravila za generacijo terena vedno ista, kar s pomočjo prilagojenih nastavitev za posamezne ekosisteme zagotavlja njihovo izrazno prepoznavnost ter raznolikost okolja v igri. Kljub temu pa je vsaka nova generacija sveta tudi različna od prejšnje, saj se vedno uporablja drugačen Perlinov šum za izračun višine, vlažnosti ter toplote.
- Kreativnost in resničnost - Generiran teren v veliki večini izgleda, kot da smo ga izdelali ročno, saj prehodi med ekosistemi izgledajo zelo naravno, za kar smo se potrudili pri nastavitvah prek več poskusov. Zelenje je sicer naključno porazdeljeno, kljub temu pa smo dosegli naraven izgled porazdelitve.

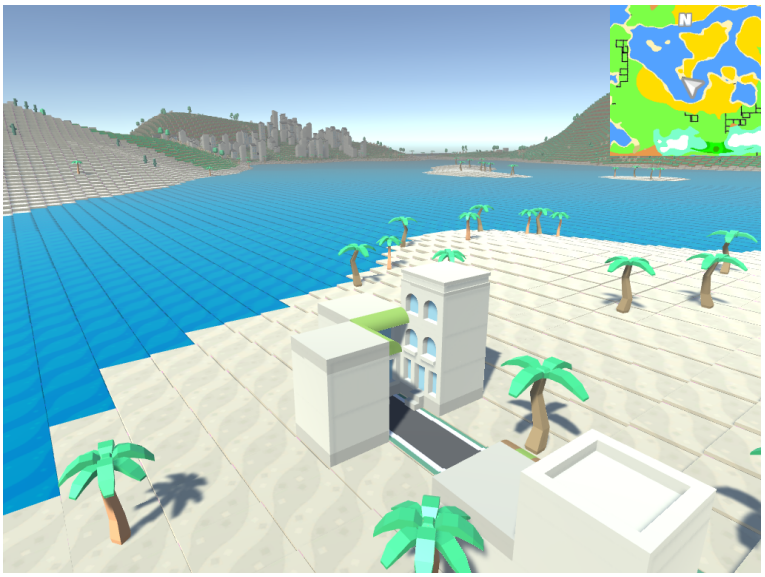
Prikazali bomo rezultate naključnega generatorja terena (slike generiranih terenov). Sledijo slike najprej nekaj primerov zemljevidov generiranega terena, nato slike iz 3D simulacije.



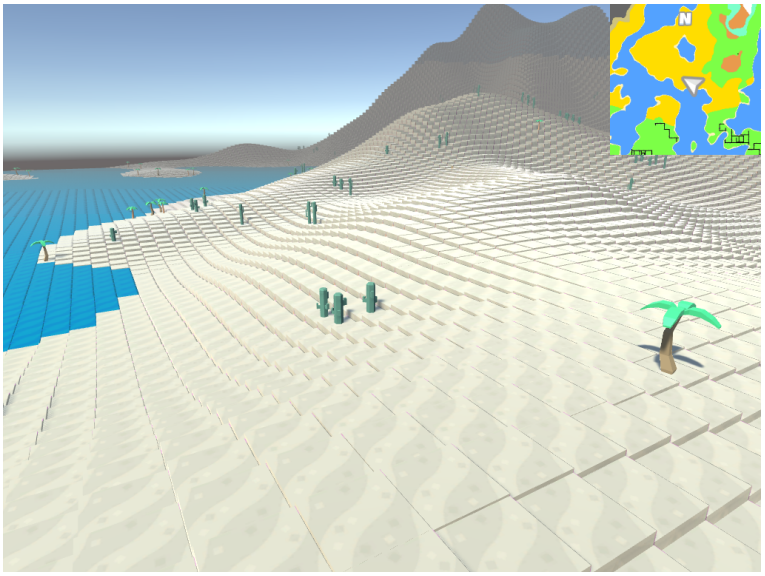
Slika 5.44: Slika prikazuje primer generiranega sveta, pri katerem se je simuliralo kopno. (Vir: lasten)



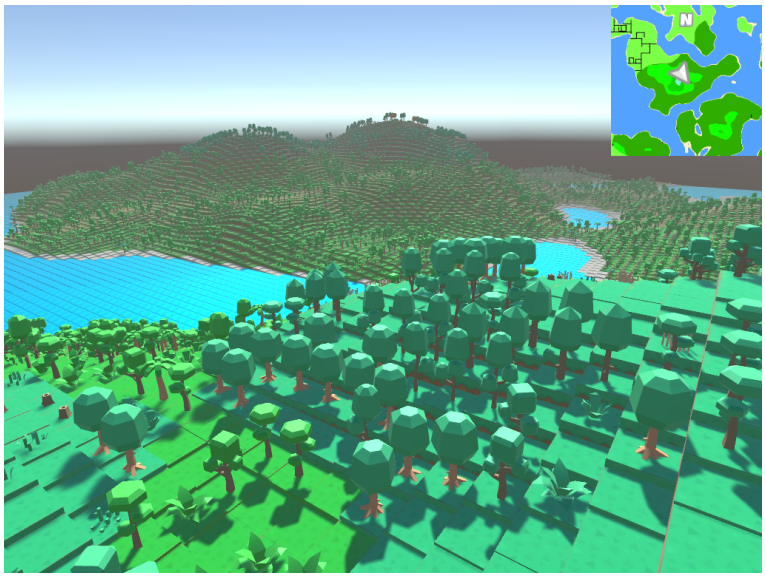
Slika 5.45: Slika prikazuje primer generiranega sveta, pri katerem se je simuliralo otočje. (Vir: lasten)



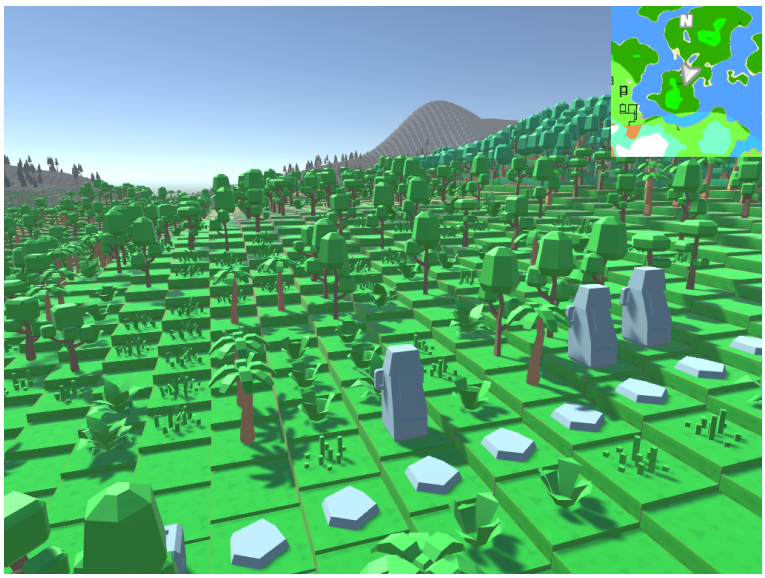
Slika 5.46: Prikaz morja, otokov ter majhnega obalnega naselja. (Vir: lasten)



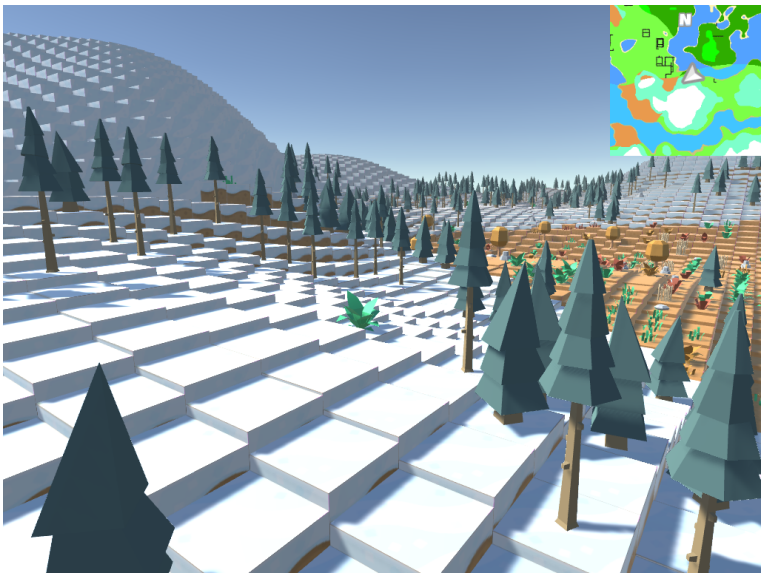
Slika 5.47: Prikaz puščave ob reki. (Vir: lasten)



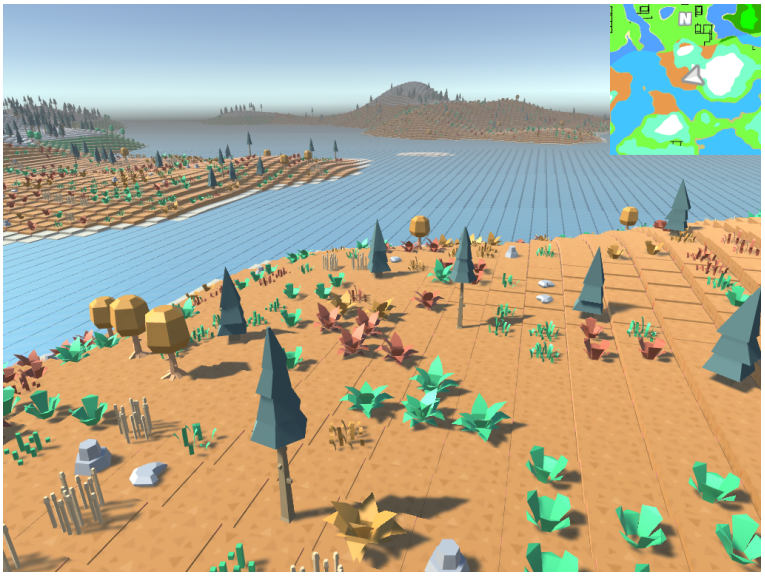
Slika 5.48: Prikaz tropskega deževnega gozda in listnatega gozda. (Vir: lasten)



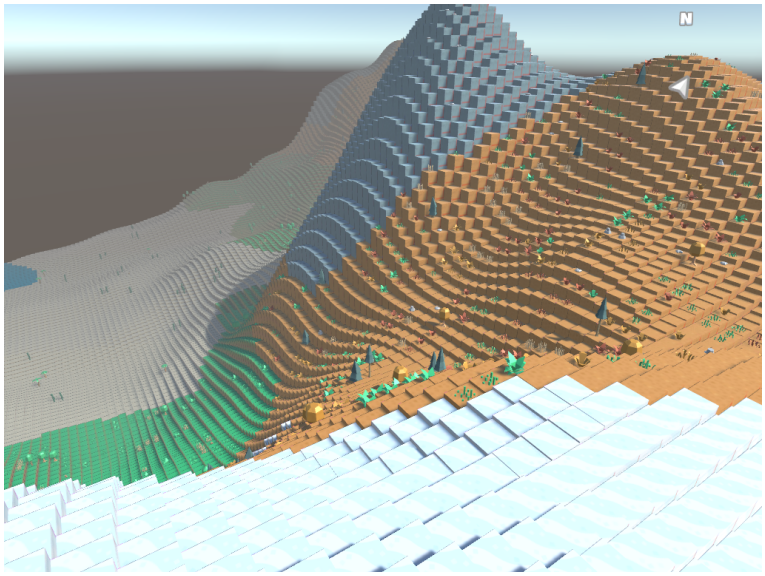
Slika 5.49: Prikaz tropskega deževnega gozda z ostanki kamnitega mesta. (Vir: lasten)



Slika 5.50: Prikaz mrzlega, vlažnega podnebja - tajge. (Vir: lasten)



Slika 5.51: Prikaz tundre ter zaledenelega morja. (Vir: lasten)



Slika 5.52: Prikaz gore ter puščave v dolini. (Vir: lasten)

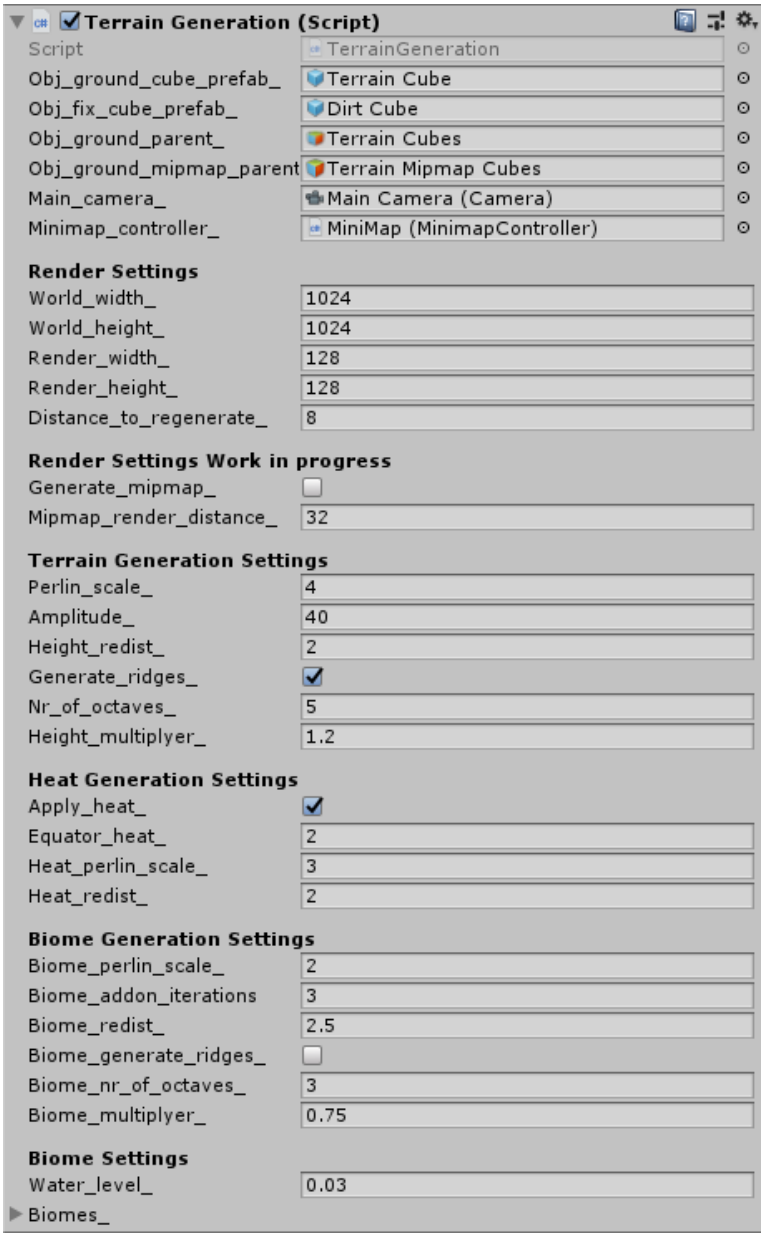
Poglavje 6

Navodila za uporabo

Nastavitve za izrisovanje sveta, generacijo terena in generacijo ekosistemov se nahajajo v skripti poimenovani 'TerrainGeneration'. Skripta vsebuje vse funkcije, ki so potrebne za izrisovanje sveta. V grafičnem vmesniku je možno spremeniti večino nastavitev delovanja izrisovanja terena.

6.1 Nastavitve izrisovanja sveta

Nastavitev	**Opis**
World_width_	Širina sveta
World_height_	Višina sveta
Render_width_	Širina izrisanega sveta
Render_height_	Višina izrisanega sveta
Distance_to_regenerate_	Potrebna sprememba pozicije za regeneracijo
Generate_mipmap_	Vklop ali izklop generiranja večjih kosov sveta na robovih
Mipmap_render_distance_	Razdalja večjih koščkov sveta na robovih



Slika 6.1: Prikaz nastavitev za generacijo višin, toplote, vlage in zelenja. (Vir: lasten)

6.2 Nastavitve generacije terena

Nastavitve generacije višin:

Nastavitev	**Opis**
Perlin_scale_	Frekvenca Perlinovega šuma
Amplitude_	Amplituda nastavi maksimalno višino terena
Height_redist_	Faktor potenciranja Perlinovega šuma
Generate_ridges_	Vklop ali izklop uporabe absolutne vrednosti pri generaciji terena
Nr_of_octaves_	Število oktav
Height_multiplier_	Faktor za množenje višin

Nastavitve generacije toplote:

Nastavitev	**Opis**
Apply_heat_	Vklop ali izklop uporabe toplote pri generaciji terena
Equator_heat_	Faktor toplote ekvatorja
Heat_perlin_scale_	Frekvenca Perlinovega šuma
Heat_redist_	Faktor potenciranja Perlinovega šuma

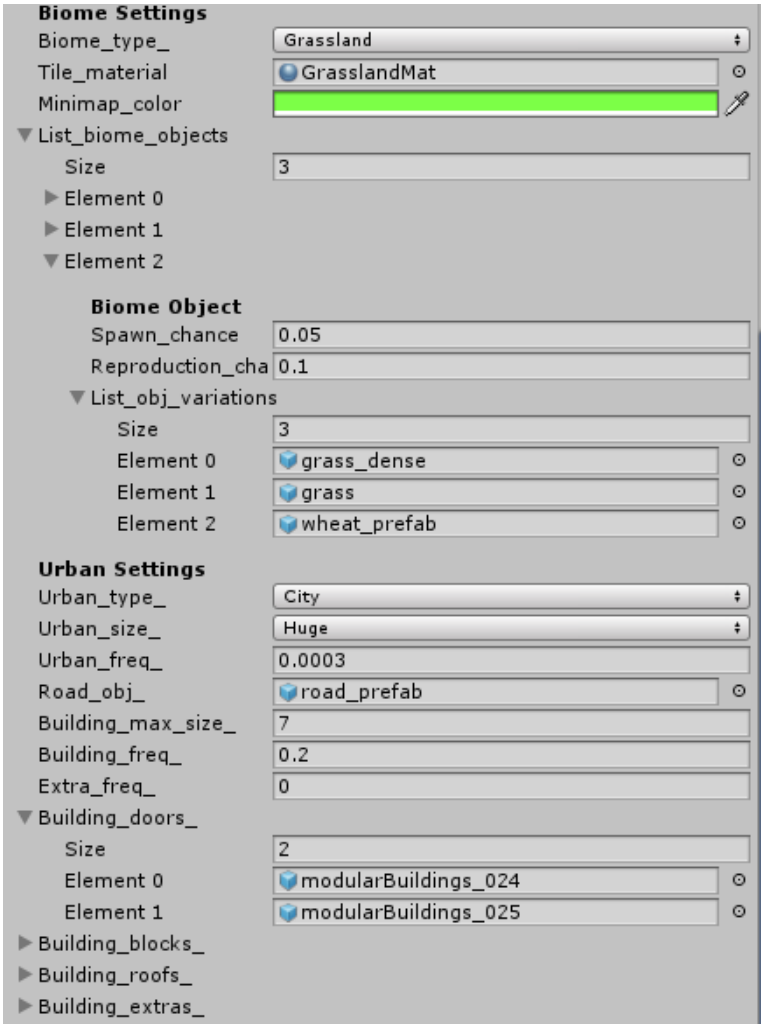
6.3 Nastavitve generacije ekosistemov

Nastavitve generiranja vlažnosti ter generacije zelenja:

Nastavitev	**Opis**
Biome_perlin_scale_	Frekvenca Perlinovega šuma
Biome_addon_iterations_	Število iteracij množenja generiranja zelenja
Biome_redist_	Faktor potenciranja Perlinovega šuma
Biome_generate_ridges_	Vklop ali izklop uporabe absolutne vrednosti pri generaciji ekosistemov
Biome_nr_of_octaves_	Število oktav vlažnosti
Biome_multiplier_	Faktor za množenje vlažnosti
Water_level_	Višina do katere sega voda

Nastavitve posameznih ekosistemov:

Nastavitev	**Opis**
Biome_type_	Tip ekosistema, možnost izbire izmed vseh ekosistemov
Tile_material_	Material koščka tal
Minimap_color	Barva ekosistema, ki se izriše na zemljevidu v zgornjem desnem kotu
List_biome_objects_	Seznam, ki vsebuje vse objekte, ki se generiranjo z uporabo generacije zelenja
Spawn_chance	Verjetnost generacije objekta v prvi iteraciji generacije zelenja
Reproduction_chance	Verjetnost množenja objekta v vseh naslednjih iteracijah generacije zelenja
List_obj_variations	Variacije modelov objekta
Urban_type_	Tip urbanih naselij, ki se generirajo
Urban_size_	Velikost urbanih naselij
Urban_freq_	Verjetnost generacije naselja v prvi iteraciji generacije cest
Road_obj_	Model ceste
Building_max_size_	Najvišja možna velikost stavb
Building_freq_	Verjetnost generacije stavbe ob cesti
Extra_ferq_	Verjetnost generacije drugih objektov ob cesti, ki niso stavbe
Building_doors_	Seznam vseh možnih modelov vrat
Building_blocks_	Seznam modelov vmesnih nadstropij
Building_roofs_	Seznam modelov streh
Building_extras_	Seznam modelov drugih objektov



Slika 6.2: Prikaz nastavitev za ekosistem travnika. (Vir: lasten)

Poglavje 7

Sklepne ugotovitve in zaključek

Končna implementacija proceduralne generacije terena zanesljivo generira raznolik teren. Poleg tega je proceduralno generacijo terena mogoče upravljati na različne načine, odvisno od potreb uporabnika. Spremeniti je možno skoraj vse nastavitve, od višine gora do izgleda posameznih ekosistemov. Zato je naša implementacija proceduralne generacije terena uporabna v zelo velikem številu raznovrstnih iger.

Menimo, da bi proceduralno generacijo terena lahko izboljšali še s povečanjem raznolikosti ter številom ekosistemov, z dodajanjem simulacije vetra za izboljšavo implementacije generiranja vlažnosti in na mnogo drugih načinov. Našo implementacijo pa je že mogoče uporabiti v raznih računalniških igrah, kot so strateške igre, kjer lahko igralec ustvarja svoja naselja ter raziskuje svet, igre tipa 'peskovnik', kjer lahko igralec manipulira svet po svojih željah, strelskih in pustolovskih igrah.

Literatura

[1] D. Green. *Procedural Content Generation for C++ Game Development.* Packt Publishing, 2016.

[2] Perlin noise. Perlin noise. `https://en.wikipedia.org/wiki/Perlin_noise/`, 2018. [Online; accessed 20.9.2018].

[3] Noor Shaker, Julian Togelius, and Mark J. Nelson. *Procedural Content Generation in Games: A Textbook and an Overview of Current Research.* Springer, 2016.

[4] Julian Togelius, Emil Kastbjerg, David Schedl, and Georgios Yannakakis. What is procedural content generation? mario on the borderline. 01 2011.

[5] Unity. Unity. `https://unity3d.com/unity`, 2018. [Online; accessed 21.9.2018].