Osnovni koncepti i alati

1 Osnovni Unity koncepti i alati

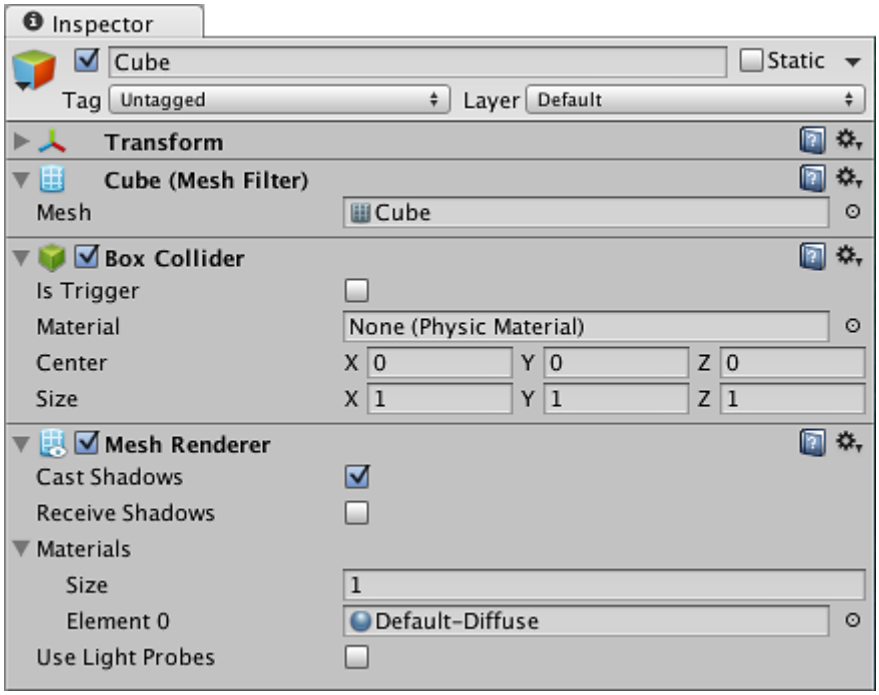
U ovom poglavlju prolazi se kroz osnovne koncepte i alate koje sadrži Unity. Tijekom svakodnevnog rada na igri, obično se sa mnogim alatima susrećemo većinu potrošenog vremena. Zbog toga su alati u nastavku teorijski obrazloženi i pokriveni.

1.1 Često korišteni pojmovi

U ovom podpoglavlju razjašnjeni su pojmovi koji se koriste u radu, te su nužni za razumijevanje.

1.1.1 GameObject

GameObjects, u prijevodu objekti igre, temeljni su objekti koje Unity koristi za prezentaciju likova, rekvizita te scenografije. Objekti igre, kao sami nemaju neku ulogu, već služe kao kontejneri (spremnici) za komponente u kojima je prava funkcionalnost. Uvijek sadrži komponentu transformacije koja predstavlja poziciju i orijentaciju, i nije ju moguće ukloniti (Unity Technologies, 2014b).



Slika 1.1. Primjer standardnog objekta igre (Unity Technologies, 2014b)

Objekt svjetlosti je kreiran tako da se objektu igre dodjeli komponenta svjetlosti.

1.1.2 Lightmapping

Lightmapping, u doslovnom prijevodu znači svjetlosno mapiranje, ili mapiranje svjetlosti. Odnosi se na svjetlost i sjene koje nastaju s obzirom na osvjetljenje. Lightmap je zapravo struktura podataka koja sadrži razinu svjetlosti trodimenzionalnih grafičkih modela i tijela. Obično se koriste samo za statične objekte, te se moraju prethodno izračunavati. U Unity alatu se koristi engleski naziv "Bake" za izračunavanje.

1.1.3 Scena

Scene sadrže objekte igre i koriste se za izradu glavnog izbornika, individualnih nivoa, te bilo čega sličnog. Razumijevanje scena je najjednostavnije tako da se svaka scena zamisli kao jedinstveni nivo. U svakoj sceni izrađuje se okruženje, postavljaju prepreke i osnovni dijelovi igre koja se izrađuje (Unity Technologies, 2014).

1.1.4 Shader

U doslovnom prijevodu znači sjenčatelj (eng. shader), a odnosi se na male skripte kojima se konfigurira hardver koji procesira grafiku. Omogućava definiranje svakog izvršnog piksela ili poligona, te ih ima više vrsta, ovisno o korištenoj tehnologiji. Mogu se programirati vlastiti sjenčatelji, većina igara na tržištu koristi kompleksne sjenčatelje kako bi prikazali napredne efekte (Unity Technologies, 2014).

1.1.5 Pathfinding

Izračunavanje putanje (eng. pathfinding) je računalnom izračunata najkraća ruta između dvije točke. U video igrama se koristi za izračun putanje koji se brine o izbjegavanju prepreka na mapi. Sastavni dio umjetne inteligencije igre koja ga koristi (Unity Technologies, 2014).

1.1.6 Renderiranje

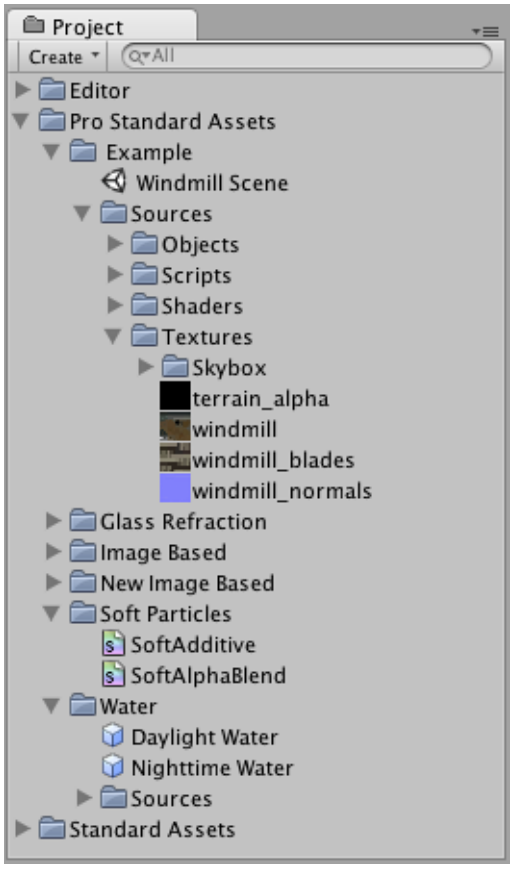
Renderiranje (eng. Rendering) je proces generiranja slike od 2D ili 3D modela koji se nalaze unutar scene pomoću posebnog programa. Unity je program koji omogućava renderiranje u kojem podržava namještanje dodatnih parametara za kvalitetu slike.

1.2 Resursi

Kontroliranje i upravljanje resursima igre je jedna od osnovnih radnji koje mora omogućiti sustav za izradu igara.

1.2.1 Ubacivanje resursa

Datoteke koje se koriste u igri su resursi igre (eng. assets) te čine igraće objekte (eng. GameObjects) u Unity alatu. Pomoću pogleda projekta (eng. project view) vrlo brzo možemo pristupiti svim resursima i objektima koji čine igru.



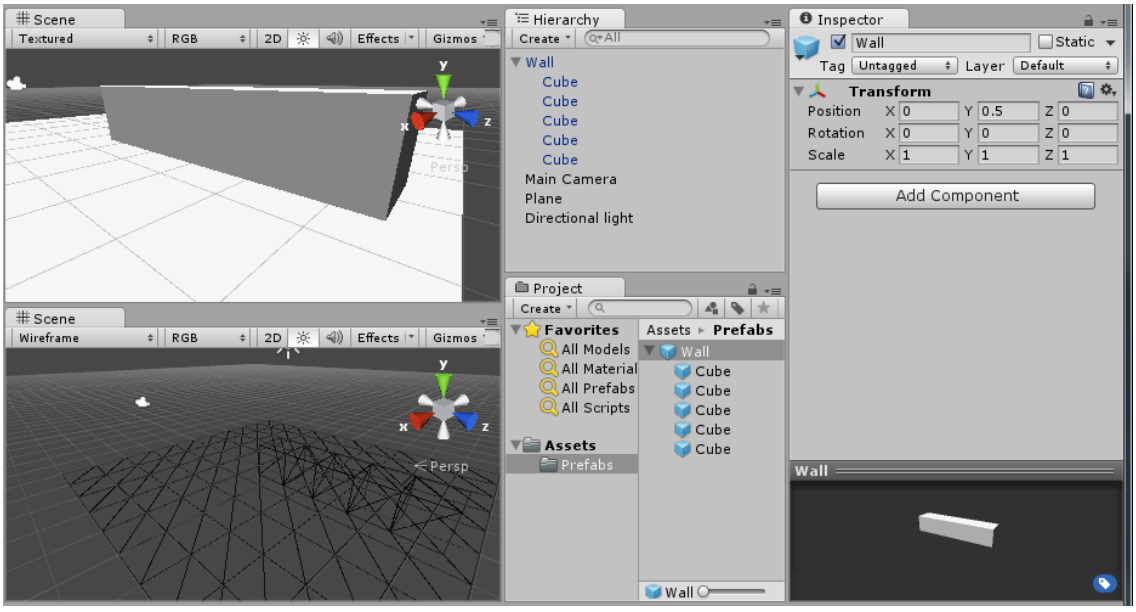
Slika 1.2. Struktura resursa projekta (Unity Technologies, 2014b)

Pogled omogućava organizaciju datoteka korištenih u projektu i igri, te svaka promjena na datotekama se automatski reflektira u igri. Kako bi dodali datoteke u naš projekt jednostavno ih prekopiramo u mapu "Assets" koja se nalazi unutar mape našeg projekta na disku. Svaka promjena se automatski ažurira u programu te je prebačena datoteka spremna za korištenje. Datoteku u projekt možemo prebaciti i na drugi, jednostavniji način, tako da je odvučemo na

panel pogleda projekta, čime se automatski doda. Dodavanje određenog resursa u igru vrši se tako da se objekt odvuče unutar hijerarhijskog ili scenskog pogleda. Ukoliko se radi o datoteci koja se primjenjuje na neki objekt igre (eng. GameObject), primjerice skripta, zvuk ili tekstura, datoteku odvučemo na traženi objekt.

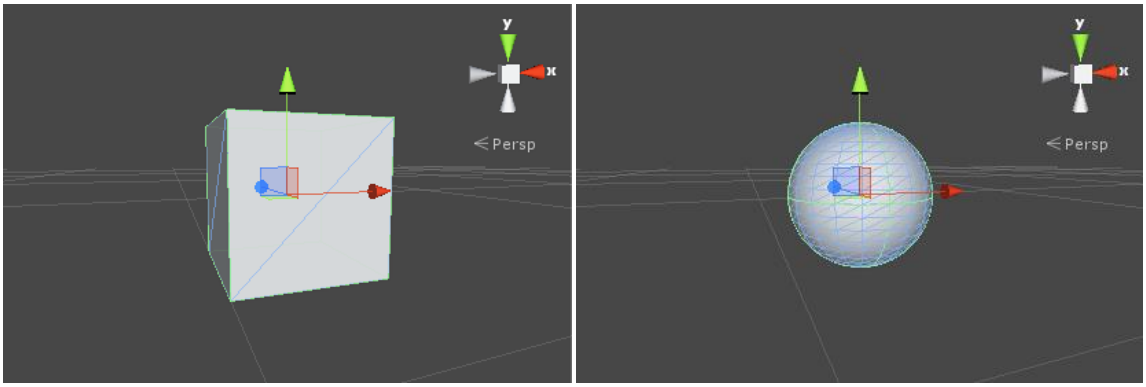
1.2.2 Kreiranje resursa

Unity podržava više vrsta objekata i resursa. Kreiranje većine resursa poput zvuka ili grafički složenijih modela, vrši se u programima koji nisu sastavni dio Unity-a. Osim tih resursa, postoje i objekti koji se mogu kreirati u Unity sustavu za izradu igara. Tako je moguće kreirati osnovne geometrijske modele, svijetlost, kameru, efekte, teren ili pak prazan objekt. Objekti kreirani unutar scene nalaze se u hijerarhijskom pogledu. Tim objektima igre moguće je pridružiti različite vrste komponenti i modificirati ih. Kako bi se taj modificirani objekt mogao koristiti u cijelom projektu, a ne samo u trenutnoj sceni, potrebno ga je na neki način spremiti. Odvlačenjem objekata iz hijerarhijskog pogleda u pogled projekta, objekt igre se sprema kao montažni element, tj. resurs tipa Prefab. Na slici ispod, objekt igre "Wall" koji se sastoji od 4 objekta u obliku kocke, spremljen je u projekt kao novi resurs sa svim svojim atributima.



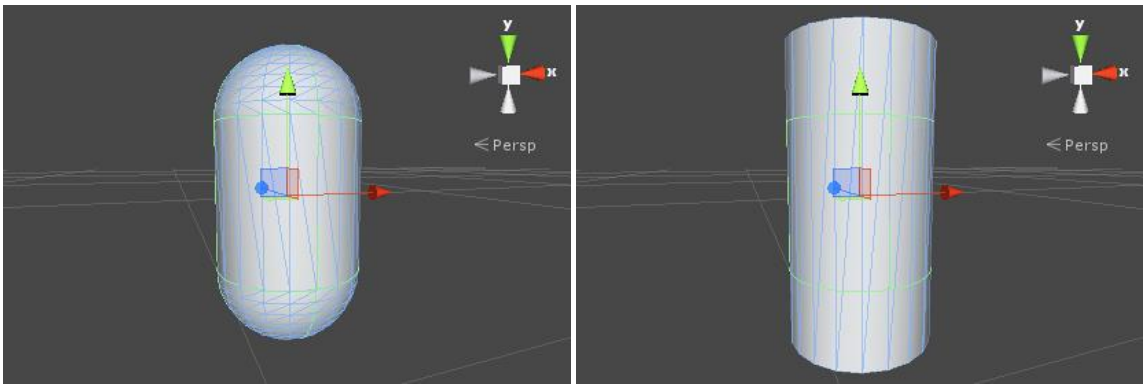
Slika 3.Resurs kreiran iz objekta igre "Wall"

Osnovni 3D modeli koji se mogu kreirati, su geometrijskih tijela poput plohe, kocke i sfere. Ti objekti imaju svoju svrhu i često se koriste kao zamjena za buduće modele u svrhe testiranja. Navedene objekte možemo dodati tako da unutar izbornika GameObject > Create Other odaberemo željenu vrstu objekta objekt.



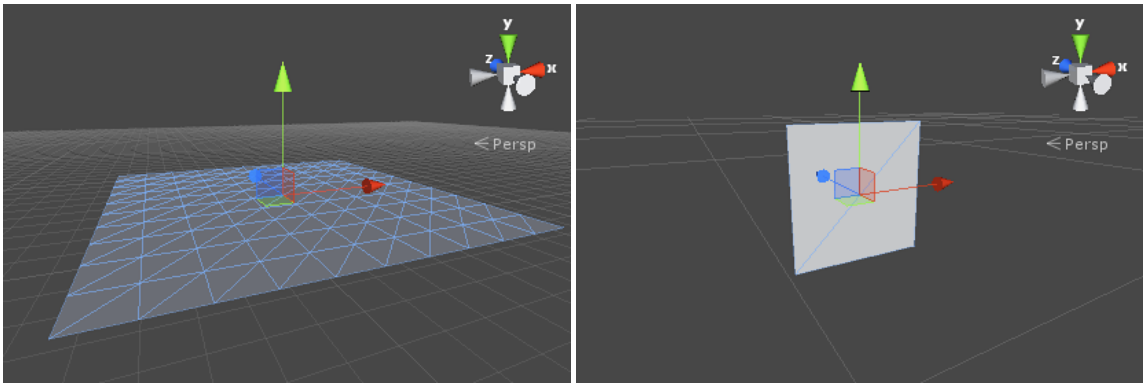
Slika 4. Prikaz kocke i sfere u Unity-u (Unity Technologies, 2014b)

Kocka (eng. Cube) je jednostavan objekt, tekstura takvih da se slika ponavlja na svih 6 ploha. Često se koristi za zidove, stepenice, kutije i slične objekte. Sfera (eng. Sphere) sadrži teksturu takvu da omotaje oblik sfere jednom sa "stegnutim" polovima. Dobro prezentiraju lopte, planete, projektile, ali i vizualne efekte gdje sfera predstavlja radijus efekta (Unity Technologies, 2014).



Slika 5. Prikaz kapsule i cilindra u Unity-u (Unity Technologies, 2014b)

Kapsula (eng. Capsule) je cilindar sa polukružnim krajevima. Visine je dvije jedinice, a širine jedne jedinice, te ima teksturu koja omotava kapsulu točno jednom. Svrha ovog objekta je najčešća u izradi prototipa, jer za mnoge uloge koje uključuju fiziku, kapsula je puno bolji odabir od primjerice kocke. Cilindar (eng. Cylinder) se obično koristi za izradu raznih štapova, šipki ili pak kotača. Kod kolizije koja se koristi za međudjelovanje i interakciju između dva ili više objekata, ne postoji oblik cilindra, već ovaj objekt za oblik kolizije koristi kapsulu (Unity Technologies, 2014).



Slika 6. Prikaz plohe i četverokuta u Unity-u (Unity Technologies, 2014b)

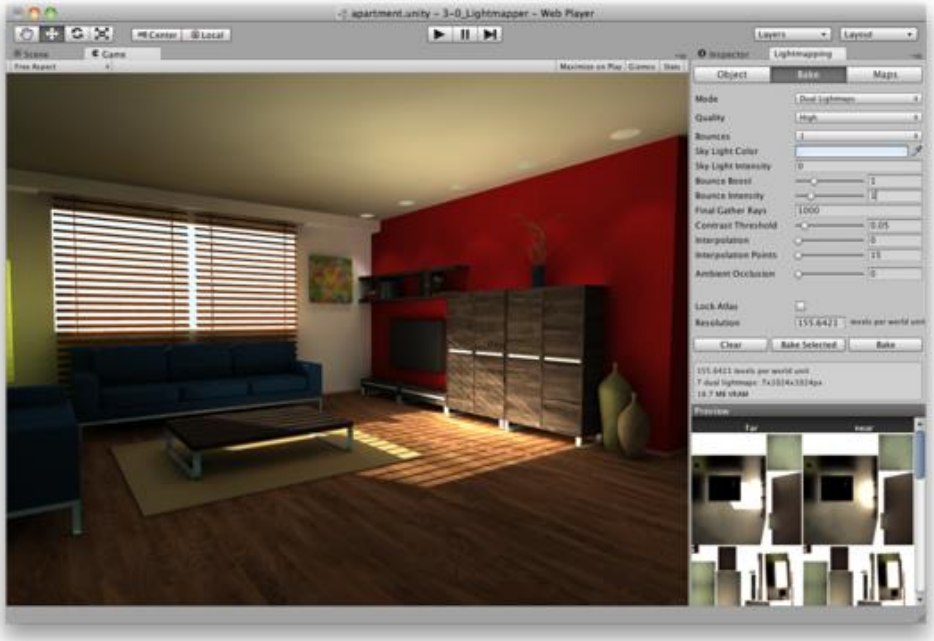
Ploha (eng. Plane) je oblika plosnate kocke veličine 10 jedinica, postavljena na XZ os, te sadrži takvu teksturu koja se jednom prikazuje na cijeloj plohi. Koristi se za različite vrste poda, zidove, također se koristi za prikaz slika ili videa u grafičkom sučelju (GUI). Četverokut (eng. Quad) na prvi pogled podsjeća na plohu, ali je njegova veličina samo jedne jedinice, postavljen je na XY os te je podijeljen samo na 2 trokuta. Koristan je u slučajevima gdje se mora koristit objekt scene kao prikaz slike za video ili sliku. Može se još koristiti za prikaz jednostavnih informacija na grafičkom sučelju, za razne vrste efekata često korištenih u 2D igrama (Unity Technologies, 2014).

1.3 Grafika

Jedan od ključnih stvari za razumijevanje ovog sustava za izradu igara je razumijevanje grafike, grafičkih elemenata i način na koji funkcioniraju.

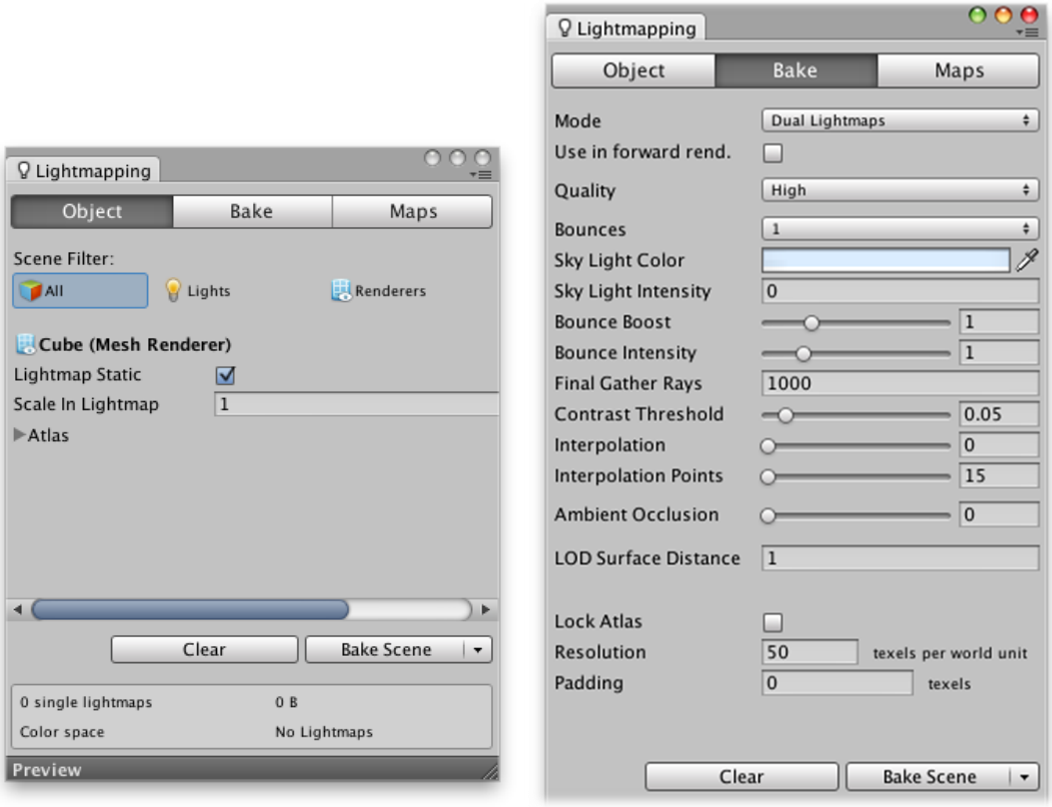
1.3.1 Osvjetljenja

Unity ima u potpunosti implementiranu tehnologiju lightmapping-a, odnosno mapiranja osvjetljenja. Svi statički modeli procesiraju s obzirom na osvjetljenje te je njihova osvijetljenost ili neosvijetljenost statička i zahvaljujući tom načinu rada igra puno manje troši resursa od dinamičkog osvjetljenja.



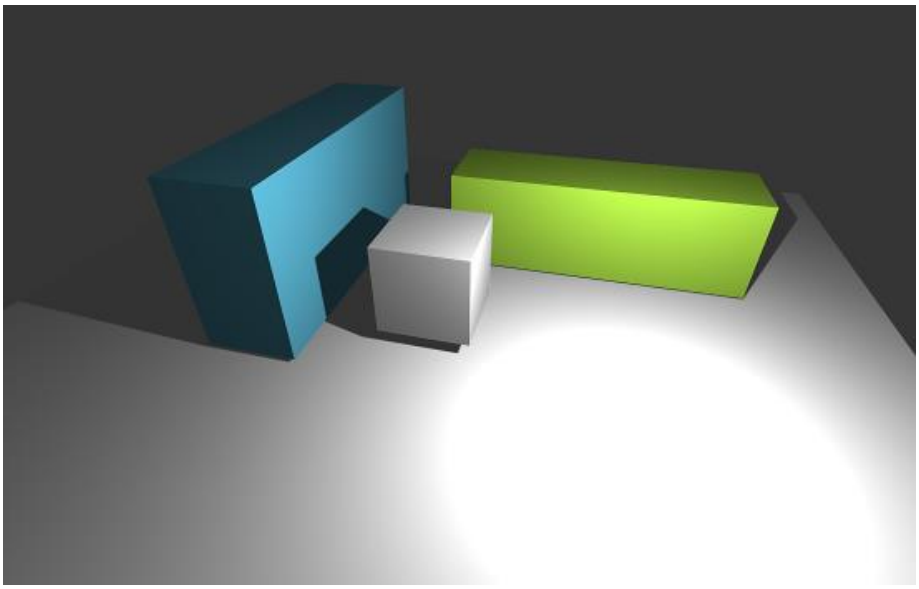
Slika 7. Osvijetljena scena u igri (Unity Technologies, 2014b)

Unutar prozora Lightmapping, pod opcijom Bake podešavamo parametre koje Unity nudi kako bi osvjetljenje u igri bilo što bolje ili što bliže željenom. Kada su svi parametri podešeni, na dnu prozora, klikom na Bake Scene, sva označena osvjetljenja u našoj sceni se izračunavaju i pripremaju nakon čega se scenski pogled, te pogled igre ažuriraju sa novom lightmapom.



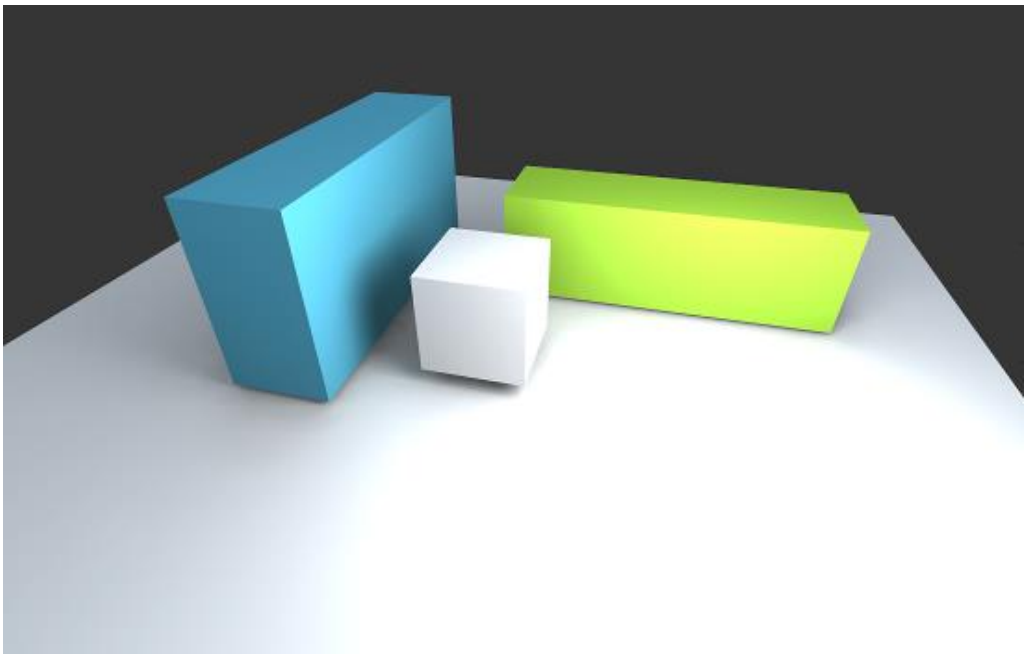
Slika 8. Postavke svijetlosti (Unity Technologies, 2014b)

Na slici ispod se nalazi primjer scene sa "oštrim", "tvrdim" sjenama.



Slika 9 Objekt sa "tvrdim" sjenama (Unity Technologies, 2014b)

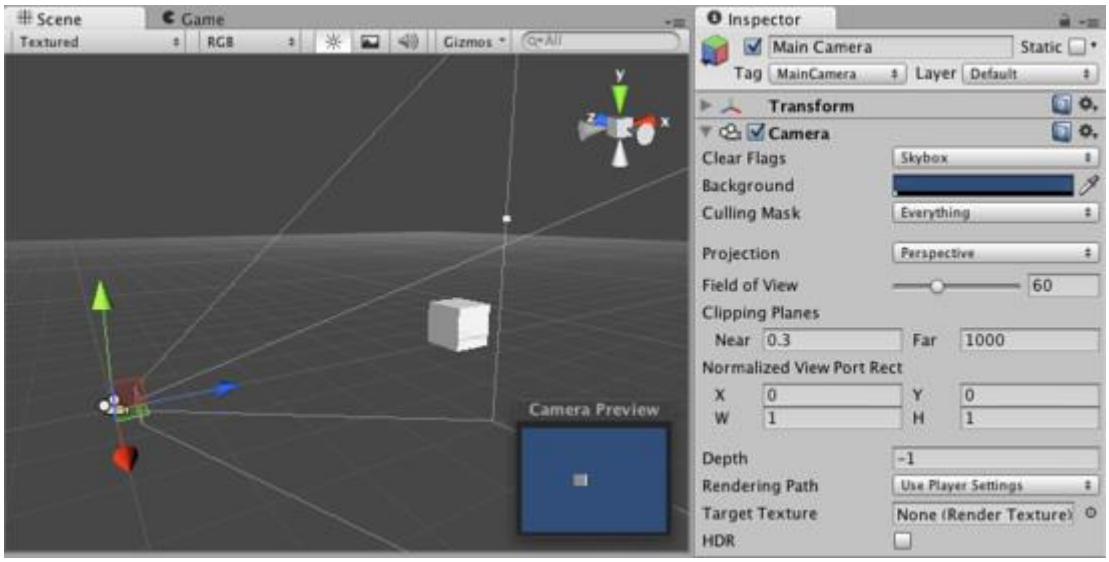
U idućem primjeru, namještanjem parametara, u ovom slučaju parametar Bounces na 1, te Sky Light Intensity na 0.5 dobiva se efekt sjena kao na slici ispod koji je sličniji realnom svijetu od prvog slučaja.



Slika 10. Objekt sa "mekim" sjenama (Unity Technologies, 2014b)

1.3.2 Kamere

Poput pravih filmskih kamera, kamere se koriste za prikaz priče, svijeta iz igre publici, tj. igraču. U sceni se uvijek mora nalaziti barem jedna kamera, a moguće ih je imati i više. Pomoću 2 ili više kamera, moguće je dobiti podijeljen ekran za 2 ili više igrača ili pak za vlastite efekte. Kamere se mogu animirati, kontrolirati, te na njih može utjecati i fizika.



Slika 11. Kamera u scenskom pogledu (Unity Technologies, 2014b)

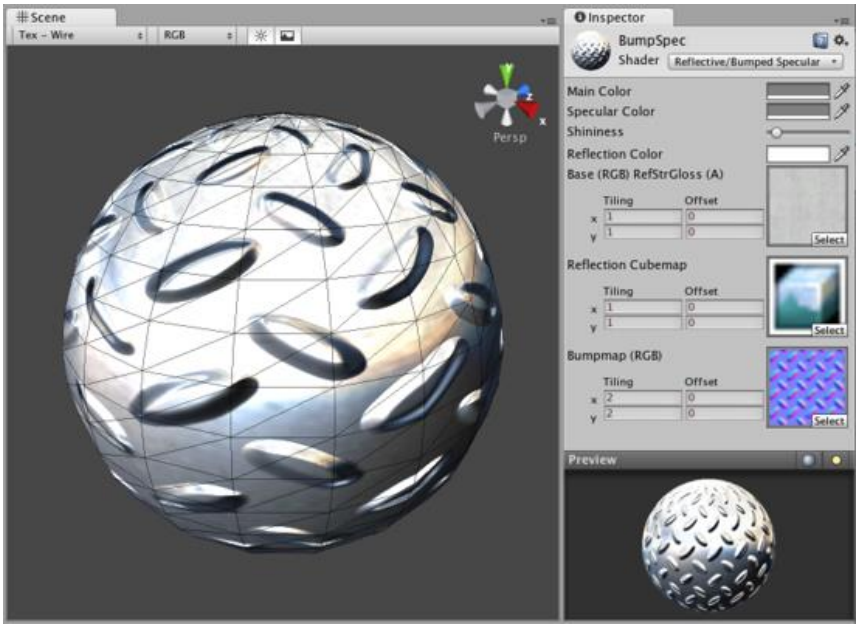
Pozicija kamere bitno utječe i na vrstu i tip igre. Ukoliko se radi o avanturističkoj igri koja je većinom u trećem licu, kamera će biti iznad igrača. Ako se radi o strategiji ili igri na poteze, kamera će se nalaziti iznad svih ostalih objekata prikazujući veći dio tla. Kod pucačine u prvom licu, pozicija kamere je unutar samog objekta igrača ili nadomak ispred.

1.3.3 Shaderi

Procesiranje grafike se izvodi uz pomoć sjenčatelja (eng. shader), malih skripti pomoću kojih se konfigurira hardver za procesiranje. U Unitiju se sjenčatelji izvode na jedan od tri različita načina.

Površinski sjenčatelji(eng. surface shaders) su najbolji ukoliko trebaju biti pod utjecajem svjetlosti i sjenama. Vertex i Fragment sjenčatelji se koriste kada nema potrebe za interakcijama sa svjetlošću ili kada postoji potreba za posebnim efektima koje površinski sjenčatelji ne podržavaju. Nedostatak im je što je potrebno puno programirati i teško je napraviti interakciju sa svijetlosti. Popravljeni funkcijski sjenčatelji (eng, fixed function shaders) se koriste kod slabijeg i starijeg hardvera koji ne podržava programski kreirane shadere (Unity Technologies, 2014).

U Unitiju postoji veza između materijala i sjenčatelja. Sjenčatelji koriste kod koji definira svojstva i vrstu koju resursi koriste, dok materijali prilagođava svojstva i dodjeljuje resurs.

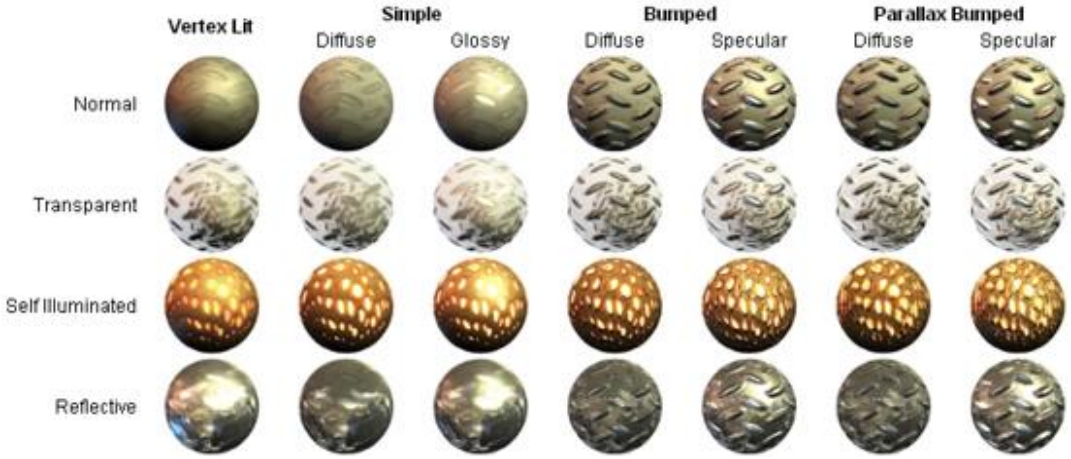


Slika 12. Sjenčatelj implementiran kroz materijal (Unity Technologies, 2014b)

Sjenčatelji koji dolaze uz Unity, dovoljni su za većinu projekata. Glavne sjenčatelje koji dolaze moguće je podijeliti u iduće kategorije:

- Normalne
- Transparentne
- Transparenti izrez (eng. TransparentCutOut)
- Self-Illimunated, za objekte koji na nekim dijelovima emitiraju svijetlost
- Reflektivni

U svakoj grupi postoje sjenčatelji koji se dijele na kompleksnost, od jednostavnih "VertexLit" do kompleksnih "Parallax Bumped" shadera.

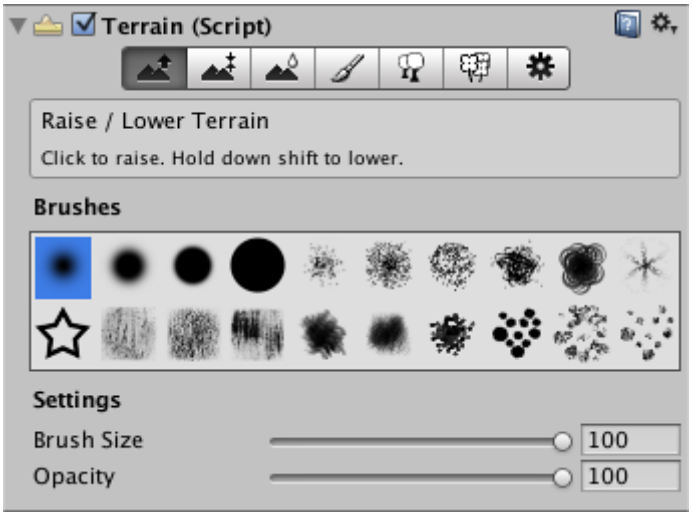


Slika 13. Shaderi te poredani po kompleksnosti (Unity Technologies, 2014b)

Uz navedene sjenčatelje, postoje i druge kategorije sa svojim svrhama. FX sjenčatelji se koriste za svjetlosne i vodene efekte. GUI za korisničko sučelje. Nature za stabla i okolinu. Particles za efekt čestica, te Toon za procesiranje grafike u stilu crtanih filmova.

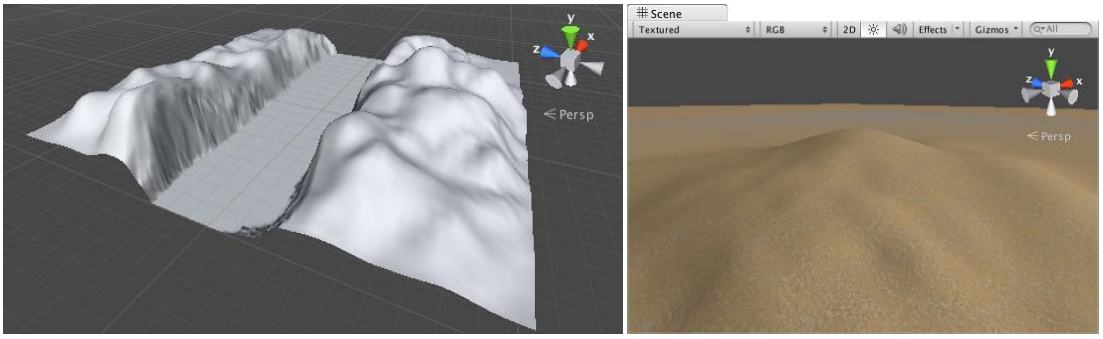
1.3.4 Engine za izradu terena

Za potrebe izrade terena ili zemljišta, imamo poseban engine, pod engleskim nazivom Terrain engine. Izradom objekta Terrain, unutar inspektora imamo dodatke jako korisne za pretvorbu našeg terena u brda i doline, pustinjski prostor, ili pak tropsku šumu.



Slika 14. Engine za izradu terena

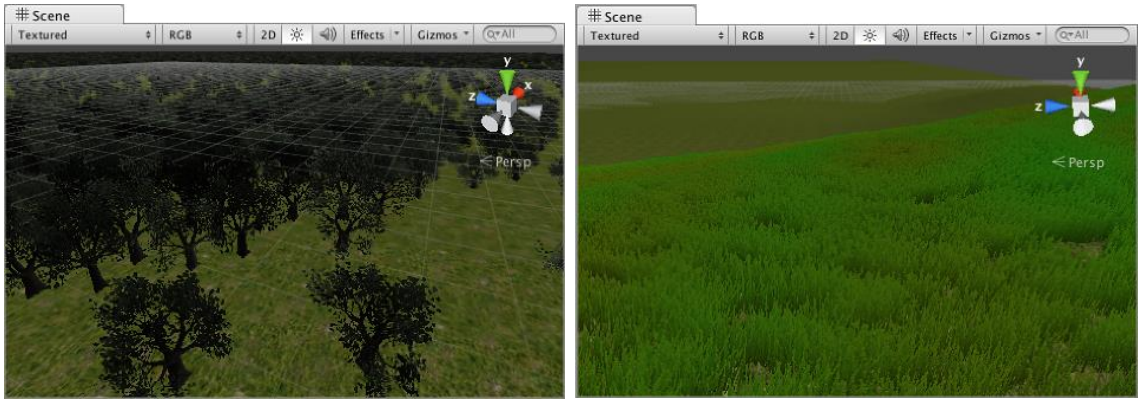
Na raspolaganju su nam nekoliko alata. Tako imamo alat za podizanje ili spuštanje razine zemljišta pomoću kojeg je jednostavno izraditi razna brda i doline. Zatim moguće je teren postaviti na točno određenu visinu, te pomoću ovog alata je lako podesiti mjestimičnu različitost visine terena.



Slika 15. Prikaz dobivenog terena sa i bez teksture (Unity Technologies, 2014b)

Zatim, imamo alat "Smooth" za uglađivanje terena koji jako korisno dođe kao obrada nakon podizanja i spuštanja razine terena zbog toga što mogu nastati oštri nerealni prijelazi koji se ovim alatom rješavaju.

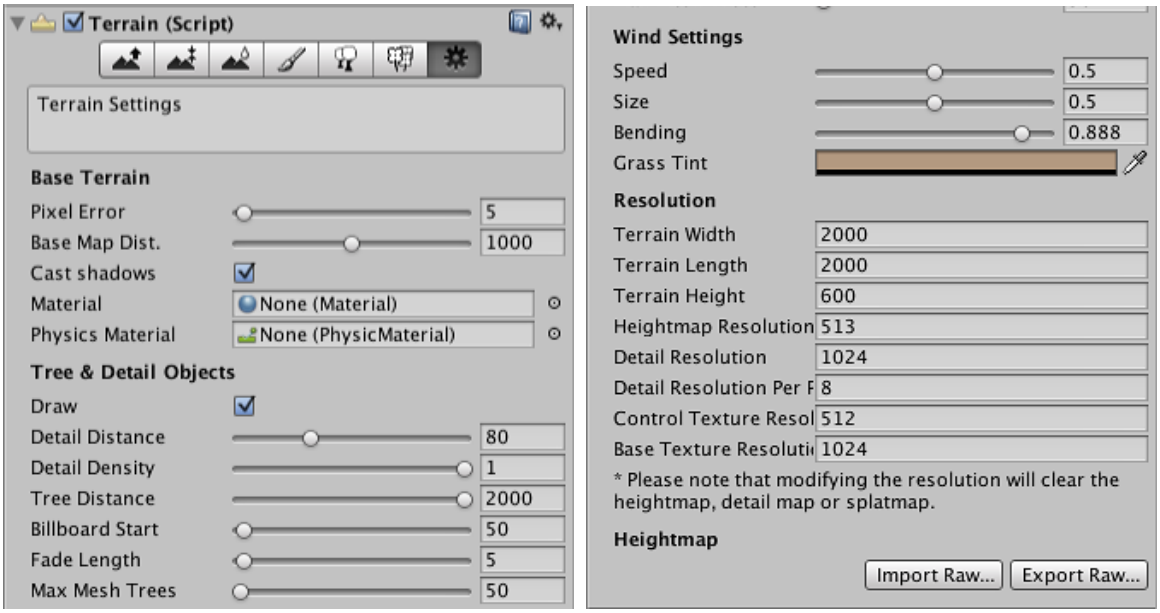
Idući alat je postavljanje tekstura i "bojanje" terena. Moguće je birati više tekstura, pretvarajući teren u što realniju okolinu uz kombinaciju većeg broja tekstura u isto vrijeme.



Slika 16. Postavljanje stabala i trave (Unity Technologies, 2014b)

Postoji i alat postavljanja stabala, gdje za odabrani model stabla u scenskom pogledu postavljamo jedno ili više stabala razbacanih unutar odabranog opsega kruga.

Drugi alat koji možemo odabrati je vezan za travu i ostale detalje. Pomoću tog alata možemo kreirati travu s obzirom na odabranu teksturu te ju postaviti na naš teren.



Slika 17. Postavke terena a) Prvi dio postavki b) Preostali drugi dio postavki

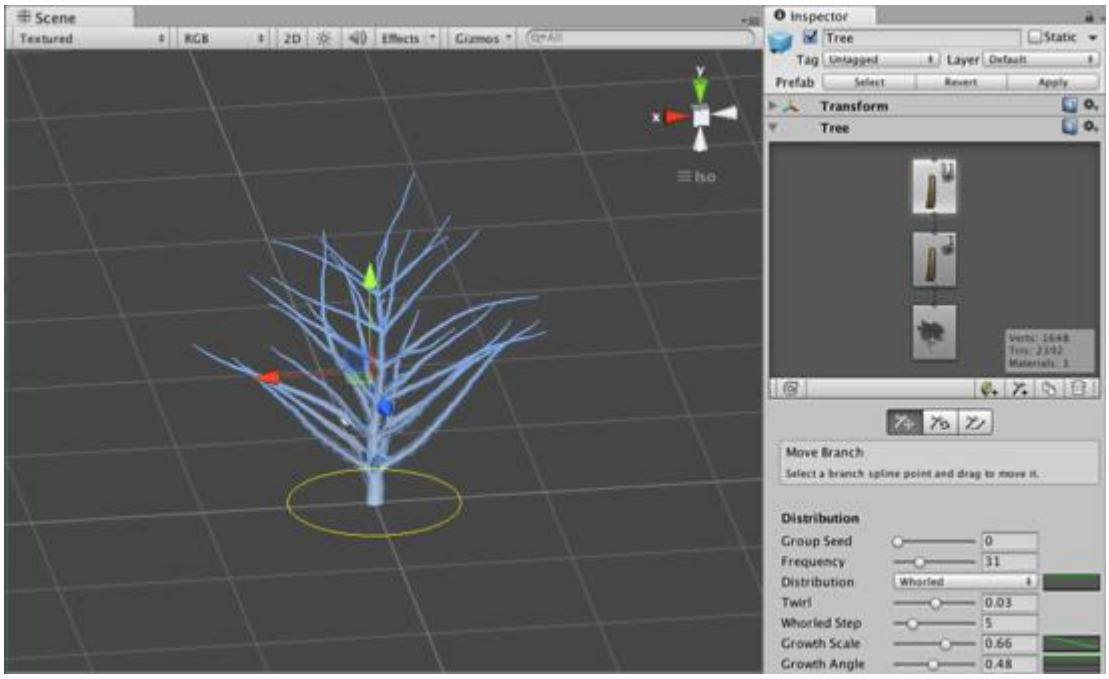
Na kraju imamo i općenite postavke terena gdje postoji više opcija. Postoje parametri vezani za vjetar koji utječe na stabla i efekte poput vatre koji se mogu povezati preko modula External Forces, odnosno vanjskih sila.

U postavkama terena postoje još mnogi parametri, neki od najvažnijih su veličina terena koja se namješta s obzirom na širinu i dužinu terena.

1.3.5 Kreator stabla

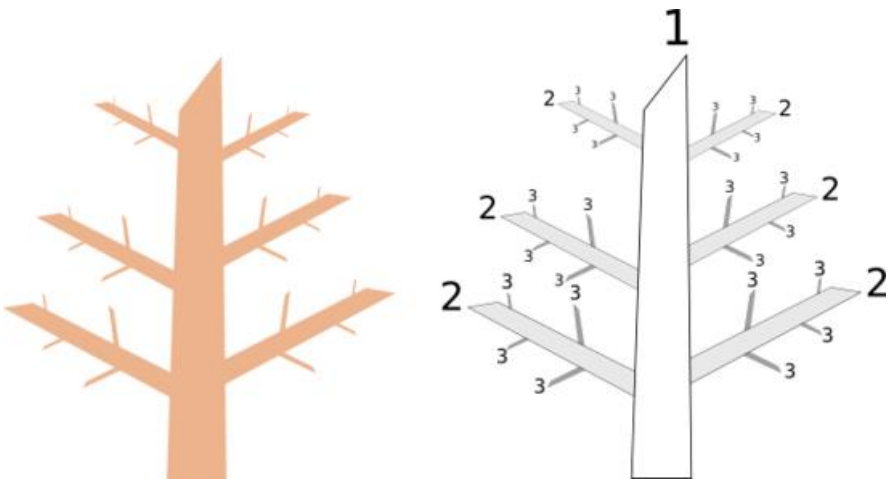
Kreator stabla, kao što i sam naziv kaže je alat za izradu stabala. Nudi dobre mogućnosti te nije kompliciran za upotrebu. Dodavanjem novog objekta Tree, kreiramo stablo koje možemo raditi po vlastitoj volji. Tako stablu dodajemo deblo te ga usmjeravamo i pozicioniramo prema želji, zatim glavnom deblu dodajemo grane i listove koje također možemo pozicionirati po želji.

Kako bi učitali ovaj alat potrebno je otići u Assets, Import te odabrati Tree Creator



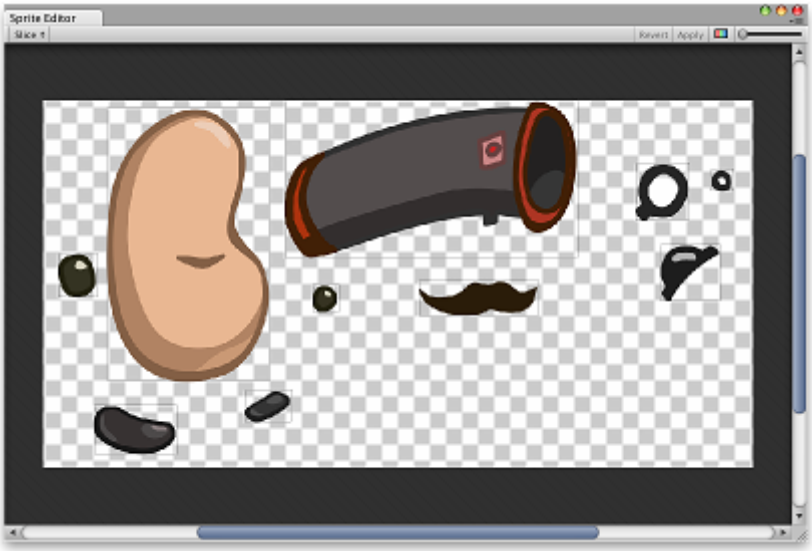
Slika 18. Kreator stabla (Unity Technologies, 2014b)

Princip izrade stabla možemo vidjeti na slici ispod gdje je vidljiv redoslijed izrade pojedinih dijelova.



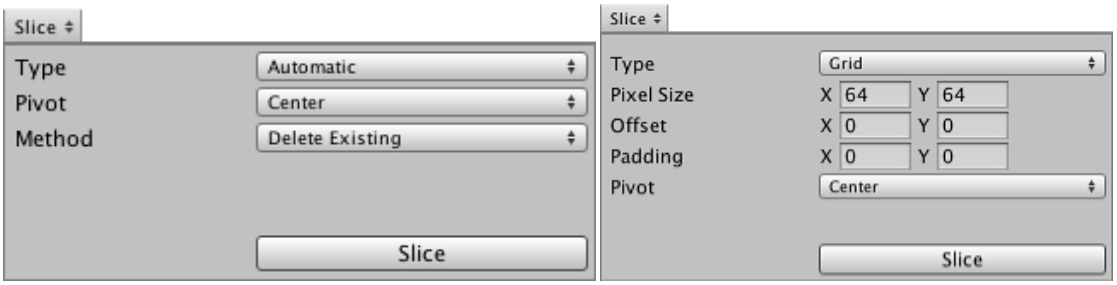
Slika 19. Redoslijed izrade stabla (Unity Technologies, 2014b)

1.3.6 Sprite Editor (2D)



Slika 20. Sprite Editor (Unity Technologies, 2014b)

Kod izrade 2D igre, za grafiku se koriste slike kojih ukoliko se radi o animaciji može biti više. U slučaju animacije svaki pokret se nalazi unutar jedne slike zvane sprite sheet kojeg uz pomoć ovog alata možemo razdvojiti na pojedinačne dijelove. Ukoliko se radi o razdvojenim dijelovima neke cjeline unutar iste slike, poput primjera na slici iznad, govorimo o atlas-u kojeg poput animacija možemo razdvojiti na individualne dijelove uz pomoć sprite editora.

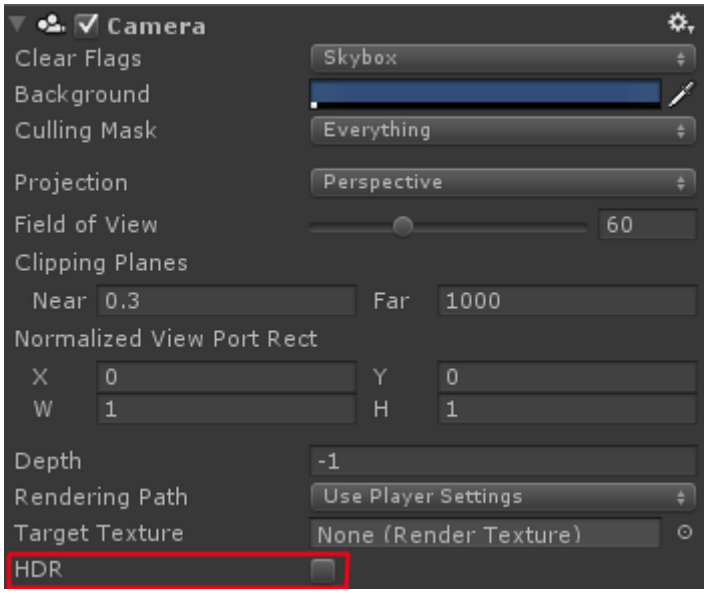


Slika 21. Parametri rezanja (Unity Technologies, 2014b)

Zahvaljujući mogućnostima Unity-a i Slice alatu, na jednostavan način možemo podesiti i razdvojiti svaki pokret animacije iz sprite sheeta, ili pak razdvojiti sve dijelove nekog lika. Ukoliko se radi o jednostavnijim igrama, razdvojeni dijelovi lika se mogu jednostavno animirati pomicanjem samo pokretnih dijelova čime se štede performanse korištenja tog načina rada.

1.3.7 Napredne metode renderiranja

Prilikom procesiranja grafike i igre, moguće je podesiti neke napredne parametre na osnovu kojih je moguće dobiti bolje rezultate izgleda igre. Jedan od tih naprednih parametara je raspon visokog dinamičkog renderiranja (eng. high dynamic range).



Slika 22. HDR (Unity Technologies, 2014b)

Sa uključenim HDR parametrom omogućeno je korištenje različitih efekata koji dodjeljujemo kameri. Efekti su dostupni samo u profesionalnoj verziji alata. Primjer efekta pod nazivom bloom je prikazan na slici ispod.

Slika 23. Efekt bloom (Unity Technologies, 2014b)

Također u profesionalnoj verziji alata, moguće je namještati razinu vidljivosti detalja s obzirom na udaljenost. Na taj način je moguće povećati performanse igre. Obično se za naziv te tehnike koristi kratica LOD (Level Of Detail).

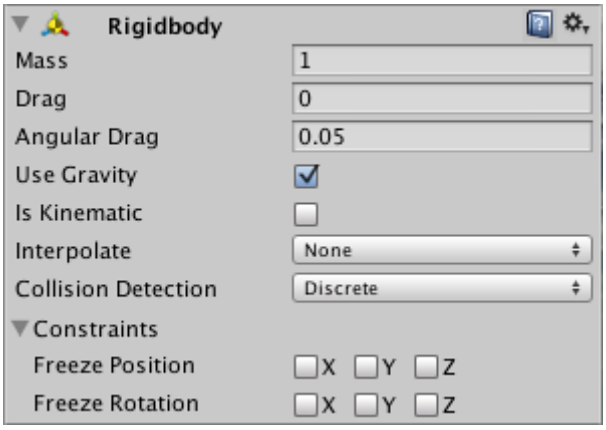
Uz navedene parametre, Unity podržava korištenje DirectX 11, zatim korištenje velikih tekstura poput teksture veličine 16384x16384 koje se dijele na manje dijelove kako bi ih bilo moguće očitati u memoriju.

1.4 Fizika

Za izradu igre sa uvjerljivim fizičkim ponašanjem, objekti u igri se moraju ispravno ponašati u kolizijama sa drugim objektima, te izravno djelovati pod utjecajem gravitacije i ostalih sila. Unity ima čak 2 fizička sustava koja se brinu za to. Jedan se koristi kod 3D fizike drugi kod 2D. Komponenta za 3D fiziku nalazi se pod nazivom Rigidbody, dok za 2D fiziku je Rigidbody 2D.

1.4.1 Rigidbody

Rigidbody je glavna komponenta koja omogućava fizička svojstva i ponašanje za objekt. Objekt kojem dodamo komponentu Rigidbody automatski će biti pod utjecajem gravitacije.



Slika 24. Rigidbody komponenta

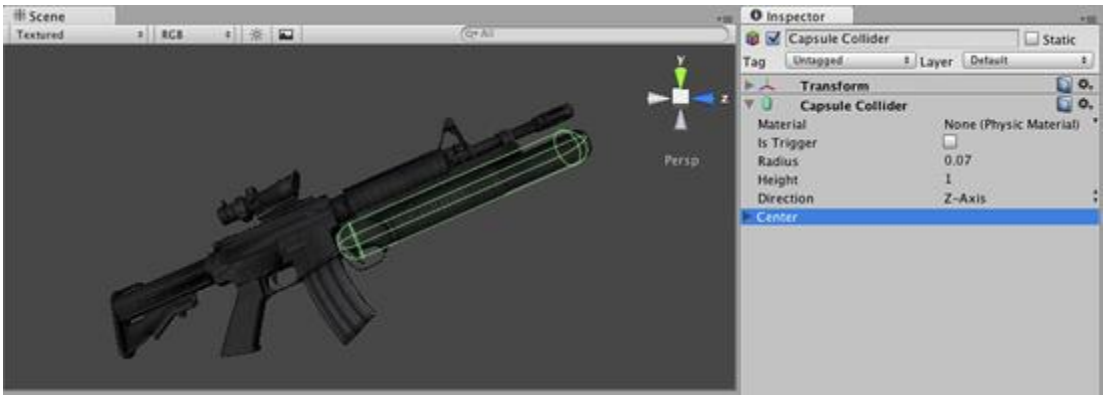
Prilikom programiranja kretnji, objekt koji sadrži ovu komponentu ne bi se smio pomicati pomoću Transform svojstava gdje mijenjamo poziciju i rotaciju. Umjesto toga postoje sile pomoću kojih se objekt kreće.

Postoje slučajevi u kojima želimo Rigidbody komponentu bez fizike. Tako je moguće podesiti objekt kao kinematski.

1.4.2 Colliders

Ova komponenta definira objekt u svrhe fizičkih kolizija, te bez nje ne bi bilo fizičkih interakcija između objekata u sceni, prolazili bi kroz drveće i objekte kao da ne postoje. Collider je nevidljiv i ne treba imati jednak oblik kao objekt. Zbog toga su jednostavni oblici većinom puno bolji i jednostavniji za procesiranje. Za 3D kolizije postoje: Box Collider, Sphere Collider, Capsule Collider, dok kod 2D kolizija imamo Box Collider 2D te Circle Kollider 2D. Objektu možemo dodati više Collidera kako bi kreirali složeniji oblik kolizije.

Ukoliko nisu dovoljni navedeni oblici, postoje još Mesh Collider kod 3D kolizija i Polygon Collider kod 2D kolizija.



Slika 25. Postavljanje Collider-a (Unity Technologies, 2014b)

Kolizije je moguće dodavati bez Rigidbody komponente, kao npr. za tlo, zid i ostale nepomične scene. Potrebno ih je postaviti kao statičnima, i ne bi ih smjeli pomicati unutar skripti jer će utjecati jako negativno na performanse. Statične kolizije mogu biti u interakciji sa dinamičkim, samo zbog nedostatka Rigidbody komponente neće se pomicati zbog interakcije.

Prilikom interakcije objekata, objekt se simulira s obzirom na materijal. Npr. led će biti klizav, dok će se gumena loptica značajno jače odbijati. Trenje i odskok objekta može se podesiti koristeći Physics Materials, odnosno materijal.

1.4.3 Character Controllers

Character, odnosno lik u igri kojeg kontroliramo, obično treba fiziku baziranu na kolizijama kako ne bi propao kroz pod i prolazio kroz zidove. Iako lik može imati nerealna ponašanja koja nisu sukladna sa fizikom, zbog kojih će moći ubrzati, stati i promijeniti smjer kretanja skoro u istom trenutku. Takvo ponašanje je moguće kreirati pomoću komponente "Character

Controller". Nudi jednostavni oblik kolizije u obliku kapsule, te nudi vlastite funkcije za promjenu brzine te nije potrebno koristiti komponentu rigidbody.

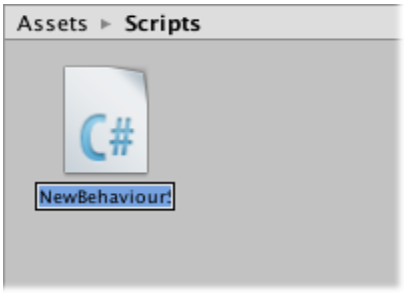
1.5 Skriptiranje i programiranje

Prilikom izrade igre, skoro je neizbježno programiranje. Kreiranim objektima unutar Unity editora se može pristupiti preko skripte te manipulirati sa njima. Moguće je kreirati vlastite komponente, ponašanja, kreiranje dinamičkih objekata ili bilo što, što nam je potrebno u igri. Kao zadani program pomoću kojega otvaramo i pišemo skripte je Mnono Develop, no ukoliko želimo moguće je koristiti i Visual Studio.

1.5.1 Kreiranje i korištenje skripti

Kreiranjem vlastite komponente za objekt igre vrši se pomoću skripti. Unity nativno podržava tri programska jezika:

- C#
- UnityScript, jezik dizajniran specifično za Unity, modeliran po JavaScript
- Boo



Slika 26. Kreiranje skripte

Kreiranje skripte unutar Unity-a može se izvršiti na više načina. Dodavanjem nove komponente gdje odaberemo opciju nova skripta. Zatim unutar projektnog pogleda se može kreirati nova skripta koja se automatski stvori unutar otvorene mape.

Klikom na skriptu, skripta se otvara unutar odabranog editora. Glavna struktura skripte sastoji se od jedne klase koja sadrži 2 funkcije, kao što je na primjeru ispod.

- Klasičan oblik skripte

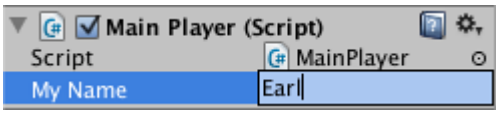
```
using UnityEngine;
```

```
using System.Collections;


public class MainPlayer : MonoBehaviour {

    // Use this for initialization
    void Start () {
    }

    // Update is called once per frame
    void Update () {
    }
}
```

Kako bi skripta bila aktivna naziv klase i naziv datoteke skripte mora biti jednak. Funkcija Start će se pozvati prva, prije izvođenja scene, dok funkcija "Update" se izvršava u svakom frameu, tj. svakoj pojedinoj slici koja se izvršava i preko 30 puta u sekundi.

1.5.2 Kontroliranje GameObjekata pomoću komponenti

Javnim varijablama u skripti moguće je pristupiti i preko inspektora u Unity Editoru. Zahvaljujući tome, jednostavno je promijeniti vrijednosti javnih varijabli, također prilikom testiranja igre, promjene vrijednosti varijabli se u realnom vremenu ažuriraju u igri. Prekidom testiranja igre, vrijednosti se vraćaju na stanje prije pokretanja igre kako bi se spriječila mogućnost nastajanja štete što omogućava eksperimentiranje različitih vrijednosti.



Slika 27. Pristupanje varijablama preko Editora (Unity Technologies, 2014b)

1.6 Zvuk

Zvukove u stvarnom životu proizvode neki objekti, a slušaju ih osluškivači. Takav princip je implementiran i unutar unitija. Zvuk koji nastaje unutar špilje različit je od onog na otvorenom, te udaljavanjem objekta koji proizvodi zvuk od osluškivača, zvuk je sve tiši i tiši. Unutar unity-a možemo dodavati različite vrste zvukova, te snimati zvukove unutar samog alata .

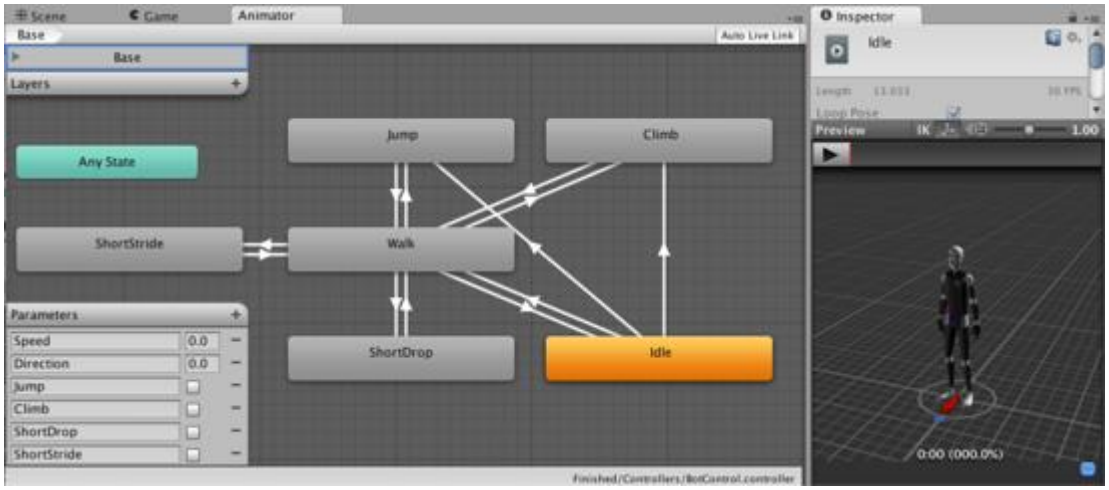
Slika 5.28. Osluškivač zvuka (Unity Technologies, 2014b)

Kako bi simulirao efekt pozicije, unity zahtjeva postavljanje komponente Audio Source na objekte, te komponentu Audio Listener postavljenu na drugi objekt, često na kameru.

Unity na isti način uvozi zvučne datoteke kao i ostale. Podržani formati audio datoteka su AIFF, WAV, MP3 i Ogg. Unity podržava i tzv. tracker module, koji koriste kratke zvukove kao "instrumente" koji predstavljaju neku melodiju. Podržani formati takvih vrsta zvučnih datoteka su .xm, .mod, .it, i .s3m ali se koriste jednako kao i normalne zvučne datoteke.

1.7 Animacije

Unity ima bogat i kvalitetno napravljen sustav animacija pod nazivom Mecanim. Pomoću Mecanima na jednostavan se način podešavaju animacije modela gdje odmah sa desne strane u Inspectoru imamo pregled istih.



Slika 29. Mecanim (Unity Technologies, 2014b)

Radni tok u Mecanimu se može podijeliti na 3 glavne faze. Prva faza je priprema i uvoz modela. Ovaj dio obično rade animatori i modelatori, uz pomoć programa za 3d modeliranje poput 3Ds Max, Maya ili Blender. Korak je neovisan o samom Unityu.

Drugi korak je postavljanje modela za Mecanim, izvodi se na 2 različita načina. Prvi način je postavljanje čovjekolikog modela (eng. humanoid character setup). Mecanim ima posebno okruženja za čovjekolike modele koje znatno olakšava rad te omogućuje jednostavno prekopiranje animacija za ostale modele istog tipa. Drugi način je generičko postavljanje modela (eng. generic character setup). Ovaj način se koristi obično za modele u obliku stvorenja, četveronoge životinje, i sličnih.

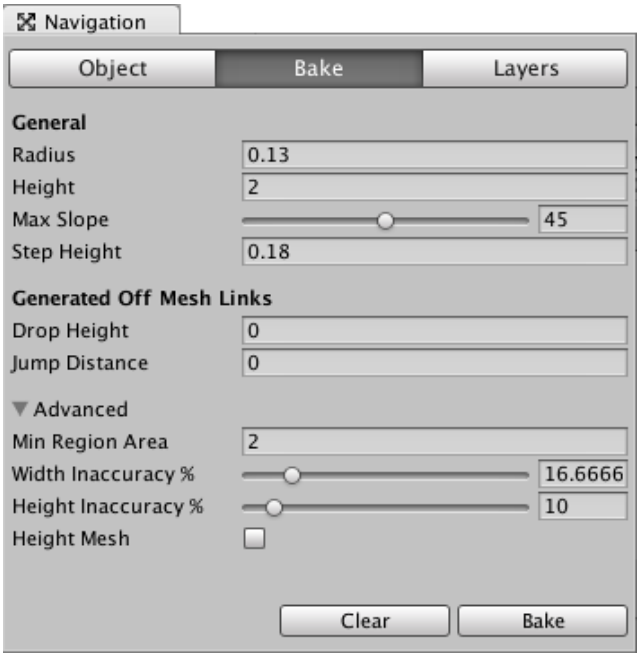
Zatim zadnji korak je u kojem modele dovodimo u život. Korak uključuje postavljanje animacijskih klipova, interakcije između njih, postavljanje parametara i kontroliranje animacija iz koda.

Iako je Mecanim sustav preporučen za rad, ponekad se koriste i standardni načini sustava za animacije, obično za starije modele kreirane prije Unity 4.0. U tom načinu se jednostavno definira određena animacija koja se u programskom kodu pokreće, dok u Mecanim sustavu se animacije automatski vrše pomoću parametara za brzinu, skok, kut okretanja i ostalih parametara koji su definirani.

1.8 Navigacija i putanje

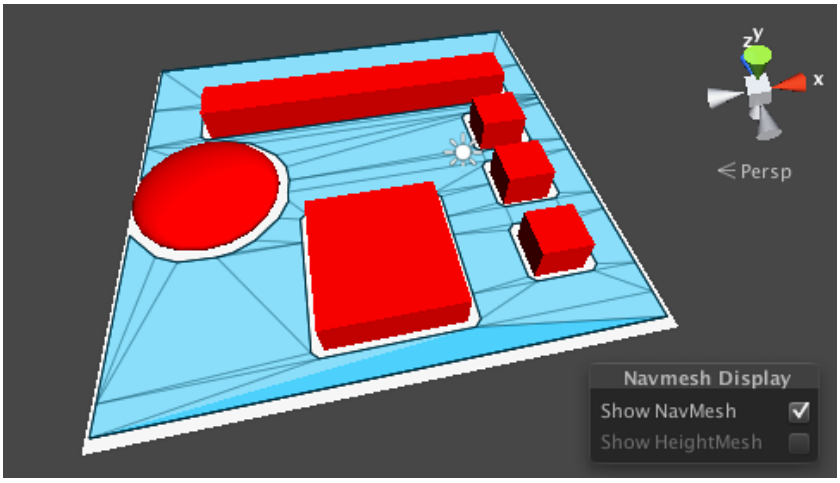
U mnogim igrama, razni likovi i modeli poput neprijatelja moraju automatski putovati od trenutne lokacije do zadane. Čest je slučaj kada igrač odabere određeni objekt u igri, automatski se lik igrača približi tom objektu. Postoje puno složenije kretnje od navedenih, gdje likovi mogu ići različitim putanjama u kojima je potrebna određena logika. Neke naprednije stvari zahtijevaju i udio umjetne inteligencije(AI) kako bi likovi u igri djelovali što realnijima. Taj posao je poznat pod nazivom pathfinding, te postoji već ugrađen u Unity-u.

Na kreiranoj mapi se računaju moguće putanje kojima se može pristupiti. Podešavanje parametara je vidljivo na slici 5.30. Moguće je podesiti više parametara među kojima je radijus koji predstavlja preciznost izračuna putanja, visina te udaljenost sa koje objekt može pasti ili skočiti.



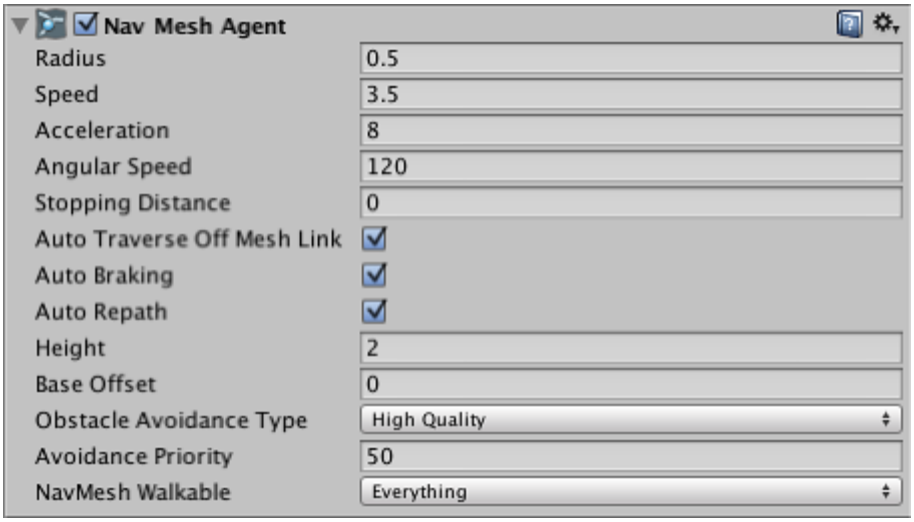
Slika 30. Podešavanje parametara navigacije

Izračunata navigacijska mapa, odnosno putanje vidljive su na slici 5.31. Plavom bojom označava se prostor po kojem se objekt sa izračunatom putanjom može kretati.



Slika 31. Izračunata navigacijska mapa (Unity Technologies, 2014b)

Za kontrolu objekta koji koristi navigaciju i putanje, služi komponenta pod nazivom Nav Mesh Agent prikazan na slici 5.31. Moguće je podesiti brzinu kretanja, okretanja, visinu i ostale parametre vezane za kretanje i objekt. Zadavanje lokacije prema kojoj je objekt usmjeren se vrši pomoću skripte u kojoj naredba `agent.SetDestination(lokacija)` omogućuje postavljanje lokacije.



Slika 32. Komponenta Nav Mesh Agent (Unity Technologies, 2014b)

Izvor

Sadržaj preuzet iz završnog rada „*Proces razvoja mobilnih igara na primjeru korištenja Unity alata*“, 2014, Marko Alerić.