**Grob u prašumi – izrada 3D igre**

**Tehnička dokumentacija**

**Verzija 1.0**

**Studentski tim:**  Dana Dodigović

Marin Hrkec

Helena Hrženjak

Antonio Junaković

Adrian Komadina

Lovro Katalinić

Luka Radivoj

Nastavnik: **prof. dr. sc. Željka Mihajlović**

**Sadržaj**

Sadržaj

[1. Opis razvijenog proizvoda 3](#_Toc535879471)

[2. Tehničke značajke 4](#_Toc535879472)

[2.1. Level Editor 4](#_Toc535879473)

[2.2. Izrada okruženja 5](#_Toc535879474)

[2.2.1. Alat krajolika 5](#_Toc535879475)

[2.2.2. Alat za raslinje 8](#_Toc535879476)

[2.2.3. Materijali 10](#_Toc535879477)

[2.3. Logika i programiranje 12](#_Toc535879478)

[2.3.1. Općenito o logici 12](#_Toc535879479)

[2.3.2. Kretnje igrača 12](#_Toc535879480)

[2.3.3. Mehanizam penjanja 15](#_Toc535879481)

[2.3.4. Inventar i sučelje 16](#_Toc535879482)

[2.4. Modeli 17](#_Toc535879483)

[2.4.1. Blender 17](#_Toc535879484)

[2.4.2. Proces modeliranja 17](#_Toc535879485)

[2.4.3. Primjeri izrađenih modela 21](#_Toc535879486)

[2.5. Osvjetljenje 22](#_Toc535879487)

[2.5.1. Globalno osvjetljenje 22](#_Toc535879488)

[2.5.2. Točkasto osvjetljenje 23](#_Toc535879489)

[2.6. Vizualni učinci 24](#_Toc535879490)

[2.6.1. Efekt magle 24](#_Toc535879491)

[2.6.2. Sustavi čestica 24](#_Toc535879492)

[2.6.3. Odsjaj leće 25](#_Toc535879493)

[2.6.4. Zamućenje pokreta 26](#_Toc535879494)

**Tehnička dokumentacija**

# Opis razvijenog proizvoda

U sklopu predmeta *Projekt iz programske potpore* izrađena je zabavna računalna igra pod nazivom *Grob u prašumi - Braća Seljan*. Igra je prvenstveno namijenjena mlađoj publici s ciljem upoznavanja i popularizacije hrvatskih istraživača iz 19. stoljeća. Naime, osim što su braća Seljan zbog svojih neobičnih pustolovina bila inspiracija za stripove i romane, oni su pridonijeli znanosti i kulturi provodeći klimatološka, geomorfološka i etnografska istraživanja te pišući brojne putopisnice i nekoliko knjiga na različitim jezicima. Također, zbirku predmeta sa svojih putovanja darovali su Etnografskom muzeju u Zagrebu s kojim je napravljena suradnja u sklopu ovog projekta. Računalna igra je medij prikladan za prikaz ove teme jer na zanimljiv i interaktivan način prikazuje isječak stvarnosti iz života ovih povijesnih ličnosti nakon kojeg se publika može lakše zainteresirati za više detalja o njima.

Prilikom pokretanja igre, otvara se glavni izbornik putem kojeg korisnik započinje novu igru. Igrač se nalazi u ulozi brata Steve Seljana koji je smješten u Južnoj Americi u indijanskom selu u prašumi. On je u potrazi za izgubljenim bratom Mirkom koji se nije vratio iz velike ekspedicije na ovom području. Kroz igru igrač nailazi na prepreke poput potrganog mosta kojeg treba popraviti, zaključane škrinje čiji se ključ pojavljuje tek nakon uspješnog skupljanja predmeta s izložbe, te raslinja koje onemogućuje daljnje kretanje. Cilj igre je proći kroz sve zagonetke i doći do uzvišenog područja na kojem se nalazi Mirkov kostur nakon čega se pojavljuje epilog o njihovim zajedničkim pustolovinama i iskrenom prijateljstvu.

Pri razvoju igre korišteni su različiti alati koji su navedeni u Tablici 1. Osim vlastito izrađenih modela unutar alata *Blender* korišteni su i gotovi besplatni modeli iz razvojne okoline *Unity*.

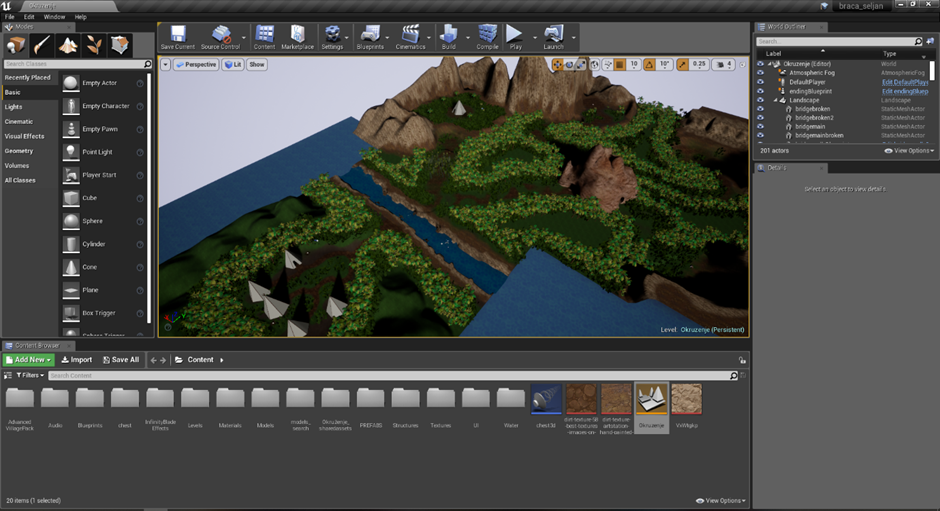
*Tablica 1.1: Alati korišteni pri izradi projekta*

|  |  |
| --- | --- |
| **Naziv alata** | **Namjena alata** |
| Unreal Engine 4 | izrada okruženja i logike |
| Blender | izrada modela |
| Photoshop, Krita, Paint.NET | obrada slika i izrada loga |
| GitLab | stvaranje inačica i kontrola toka |

# Tehničke značajke

## Level Editor

Kada se kreira projekt, Unreal Engine 4 automatski prelazi u uređivač razine (*Level Editor*). U njemu se odvija većina kreacije projekta. Na slici 2.1.1. prikazano je jedno moguće stanje uređivača. Najbitniji dio uređivača je prozor za pregled (*Viewport*) to jest prozor u kojem se promatra kako izgleda razina odnosno okruženje koje se trenutno uređuje. On se nalazi u samom središtu ekrana. U njemu se također provode vizualne promjene poput oblikovanja krajolika, slikanja krajolika i sličnog. Ispod njega se nalazi preglednik datoteka i direktorija (*Content Browser*) koji služi za pristup i organizaciju datoteka, koje čine projekt. Kroz njega se obavlja unos objekata u projekt, kreiranje novih objekata unutar projekta, interakcija s unesenim i kreiranim objektima i slično. S lijeve strane prozora za pregled nalazi se ploča načina rada (*Modes Panel*). Ona mijenja način u kojem uređivač razine radi. Naime, uređivač razine se ponaša nešto drugačije ovisno o tome koji je način rada odabran. Tako na primjer, uređivač u načinu rada za oblikovanje krajolika u fokus stavlja glumca za krajolik te pomoću prozora za pregled daje opciju da se kistom oblikuje taj glumac. S druge strane, u načinu rada za upravljanje i postavljanje glumaca pokretač nudi mogućnosti za postavljanje i promjenu pozicije glumaca, za mijenjanje njihovih svojstava i slično. Nadalje, s desne strane prozora za pregled nalazi se popis svih glumaca (*World Outliner*) u okruženju kojega se trenutno uređuje. Popis hijerarhijski organiziran. Odabirom nekog glumca ili unutar prozora za pregled ili unutar popisa glumaca, otvaraju se detalji tog glumca unutar kojih je moguće mijenjati određena svojstva odabranog glumca. Detalji prikazuju neposredno ispod popisa glumaca unutar popisa svojstava (*Details Panel*).



Slika 2.1.1. Primjer stanja uređivača

## Izrada okruženja

### Alat krajolika

Oblikovanje prostranih krajolika jedan je od glavnih zadataka u današnjem svijetu razvoja video igara. S napretkom tehnologije veličina video igara samo raste. Teži se sve većim i prostranijim razinama i okruženjima. Većina žanrova danas dolazi u kontakt s takozvanim „Open World“ dizajnom. Za takav dizajn i oblikovanje igre potrebno je oblikovati velike krajolike koji se moraju prikazivati u realnom vremenu. To je zahtjevno iz perspektive tehničke zahtjevnosti to jest zahtjevnosti za procesore i grafičke kartice. Kako bi se smanjila tehnička zahtjevnost, pokretač uvodi optimizacije pod zajedničkim nazivom razina detaljnosti (*LOD – Level Of Detail*). Drugim riječima, pokretač okolinu prikazuje s različitom razinom detalja ovisno o tome koliko je određeni dio krajolika udaljen od igrača. To bi od dizajnera okoline zahtjevalo da za osnovu krajolika izradi različite iteracije statične mreže (*Static Mesh)* različitog broja poligona, koje bi pokretač koristio pri skaliranju. To je izrazito zahtjevan posao kojeg olakšava alat za izradu krajolika, koji nudi Unreal Enginea 4. Alat omogućava izradu i oblikovanje krajolika unutar samog pokretača koji će pokretač automatski optimizirati.

Početni krajolik možemo kreirati na nekoliko načina:

1. Kreirati prazan krajolik uz određene parametre
2. Kreirati krajolik pomoću mape visine
3. Kreirati prazan krajolik koji ispunjava svijet

Za potrebe projekta, krajolik će se izraditi tako da se koristi prva metoda odnosno tako da se zadaju parametre veličine i kreira prazan krajolik te potrebne veličine koji će se oblikovati ručno. Nakon što je kreiran prazan krajolik, potrebno ga je oblikovati.

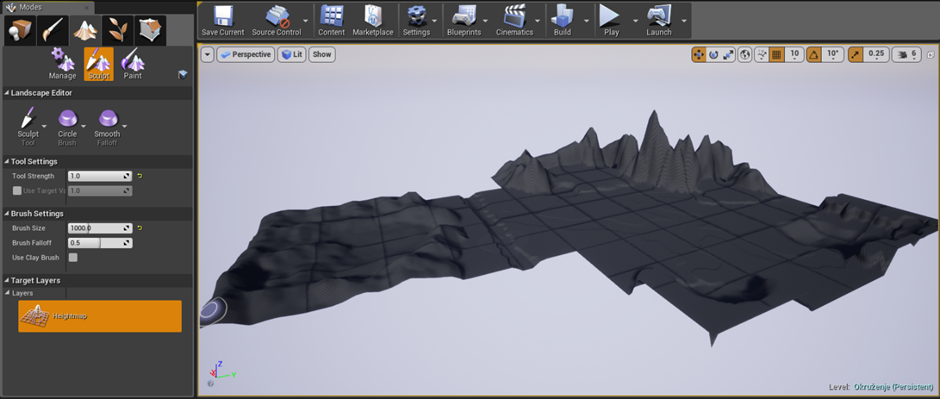
#### Oblikovanje krajolika

Oblikovanje krajolika obavlja se raznim alatima.

Alati koji će se koristiti su:

1. *Sculpt Tool* – alat koji omogućava izdizanje ili udubljivanje određenog dijela krajolika
2. *Thermal Erosion Tool* – alat koji simulira toplinsku eroziju krajolika
3. *Water Erosion Tool* – alat koji simulira utjecaj erozije kiše i vode na krajolik
4. *Noise Tool* – uvodi nasumičnost u krajolik
5. *Smooth Tool* – alat koji omogućava zaglađivanje neprirodnih strmina i slično
6. *Retopologize Tool* – raspoređuje gradivne blokove krajolika kako ne bi došlo do rastezanja tekstura

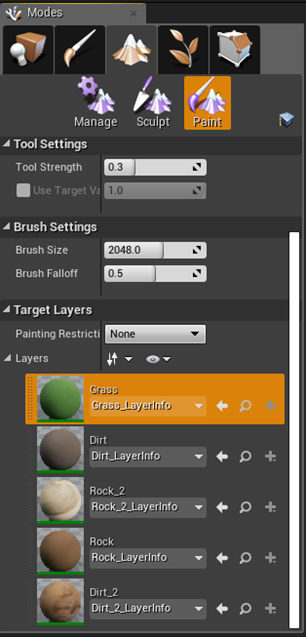
Proces započinje generalnim oblikovanjem pomoću alata za oblikovanje. Krajolik se ili izdiže ili udubljuje na odabranima mjestima kako bi nastali objekti poput planina, dolina, riječnih korita i slično. Nakon toga se primjenjuju alati koji simuliraju različite vrste prirodnih erozija kako bi krajolik izgledao što realističnije. Sljedeće, na ostale veoma ravne površine pomoću alata za šum dodaju se nasumičnosti kako bi krajolik izgledao prirodnije. Na kraju svega, pomoću alata za zaglađivanje, izglađuju se eventualne grubosti, koje se uoče u krajoliku nastale prethodnim koracima. Rezultat ovog postupka, naravno, veoma ovisi o osobi koja oblikuje krajolik. Vremenski interval za izradu krajolika uvelike ovisi o dimenzijama praznog krajolika te o željenoj dozi realističnosti i razini detalja u samom krajoliku. Za veoma velike krajolike metoda koja koristi mapu visina zasigurno je prikladnija, no treba napomenuti kako je i takav krajolik potrebno ručno doraditi zbog toga što automatska kreacija nije svršena. Rezultat ovog postupka je ili statična mreža koja predstavlja krajolik i ima pretpostavljeni materijal ili materijal koji mu je ručno pripisan ovisno o tome je li pri izradi dodan krajoliku materijal ili nije. Slika 2.2.1.1.1. prikazuje primjer oblikovanja krajolika.



Slika 2.2.1.1.1. Oblikovanje krajolika

#### Slikanje krajolika

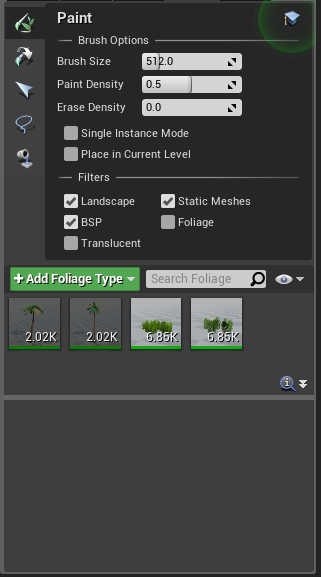
Prilikom izrade krajolika nije dovoljno samo oblikovati krajolik. Kada se završi s oblikovanjem krajolika, potrebno je dati teksturu oblikovanom krajoliku (slika 2.2.1.2.1.). Ako pri izradi krajolika krajoliku nismo pridružili materijal, krajolik će na sebi imati općeniti materijal koji se pridodaje bilo kojem objektu u pokretaču koji nema svoj materijal. Ako mu je pridružen materijal koji predstavlja jednostavnu teksturu, krajolik će cijeli biti prekriven njome. Međutim, takav pristup teksturiranju krajolika je loš zato što se njegovom primjenom dobiva krajolik koji sadrži samo jednu, jedinu teksturu što izgleda veoma neprirodno. Generalno se za krajolike kreira materijal koji sadrži više različitih tekstura odnosno podmaterijala. Ideja iza takvog materijala je da će razni dijelovi krajolika imati razne kombinacije tekstura koje su sadržane u materijalu pridruženom krajoliku. Kako bismo proveli teksturiranje takvim materijalom, Unreal Engine 4 sadrži unutar alata za kreiranje krajolika alat za slikanje (*Paint Tool*). Prije početka slikanja, potrebno je za svaki prepoznati podmaterijal unutar materijala pridruženog krajoliku kreirati sloj koji će sadržavati podatke o tome gdje se sve u krajoliku nalazi taj podmaterijal, odnosno potrebno je pored svakog podmaterijala navedenog na popisu unutar alata pritisnuti znak '+' i dodati zadani sloj. Unreal Engine 4 zapravo generira datoteke koje će čuvati informacije o određenom sloju. Odabir postavki alata ovisi o vizualnim učincima. Nakon pripreme alata, alat se koristi analogno alatu za oblikovanje. Treba uočiti da je u jednom trenutku moguće slikati samo jednim od navedenih podmaterijala.



Slika 2.2.1.2.1. Materijali za slikanje krajolika

### Alat za raslinje

Kako bi krajolik izgledao što realističnije potrebno ga je ispuniti velikim količinama objekata koji se prirodno nalaze u tom krajoliku. Za naš krajolik prašume, karakteristično je da se u njemu nalazi mnoštvo trave i palmi. Ručno unošenje i pozicioniranje raznoraznih, tako mnogobrojnih objekata zahtjeva izrazito mnogo vremena i truda. Takav način ima smisla kada se kreiraju igre u kojima su prostori manji i ograničeni te je potrebno dati posebnu karakteristiku iz dizajnerskog aspekta okruženju. Međutim, pri izradi velikih otvorenih prostora ručno unošenje je neefikasno te čak narušava izgled okruženja zbog toga što čovjek nije u mogućnosti proizvesti razinu nasumičnosti koju može proizvesti računalo, a koja ima smisla za objekte u prirodi poput trave, drveća i kamenja. Iz tog razloga unutar Unreal Enginea 4 postoji alat koji pomaže pri ispunjavanju okruženja takvim objektima (slika 2.2.2.1.). Alat za izradu raslinja omogućuje da se u okruženje postave velike količine kopija nekog objekta uz zadane parametre. Alat ima slične postavke onima koje posjeduje alat za izradu krajolika. Za svaki je objekt moguće odrediti raznorazne postavke koje utječu na to kako alat za izradu raslinja postavlja objekte unutar krajolika.



Slika 2.2.2.1. Alat za raslinje

#### Postavljanje trave

Kod postavljanja trave je bitno da pozicioniranje bude nasumično. Nasumična je samo rotacija oko vlastite osi i pozicija unutar krajolika, no nije poželjna nasumičnost koja se odnosi na odmak po Z-osi. Također je poželjno da trava od instance do instance objekta bude nasumične veličine odnosno nasumično skalirana. Također nije poželjno ni da trava raste u smjeru normale u točki krajolika zbog toga što bi trava na primjer brdovitom krajoliku izgledala neprirodno. Također ne smeta da se instance trave međusobno sijeku i da su gušće zbijene.

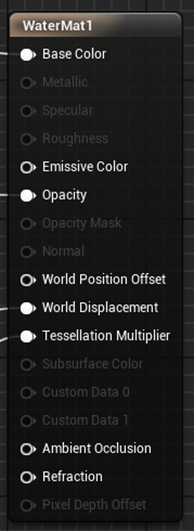
#### Postavljanje palmi

Prilikom postavljanja drveća poželjna je nasumičnost u pozicioniranju i veličini. Također drveće, kao i trava, ne smije rasti u smjeru normale iz istog razloga kao i trava, no prilikom postavljanja drveća treba paziti na postojanje razmaka između instanci drveća. Također je potrebno uključiti opciju da se može postavljati preko raslinja ako je trava postavljena prije.

Postavke postavljanja objekta pomoću alata za stvaranje raslinja variraju od objekta do objekta i ovise o prirodi stvari koji objekt predstavlja. Alat, nakon postavljanja, pokazuje broj instanci svakog od objekata koji je postavljan pomoću alata. Također pokretač ovisno o željenim postavkama performansi skalira broj objekata postavljenih ovim alatom.

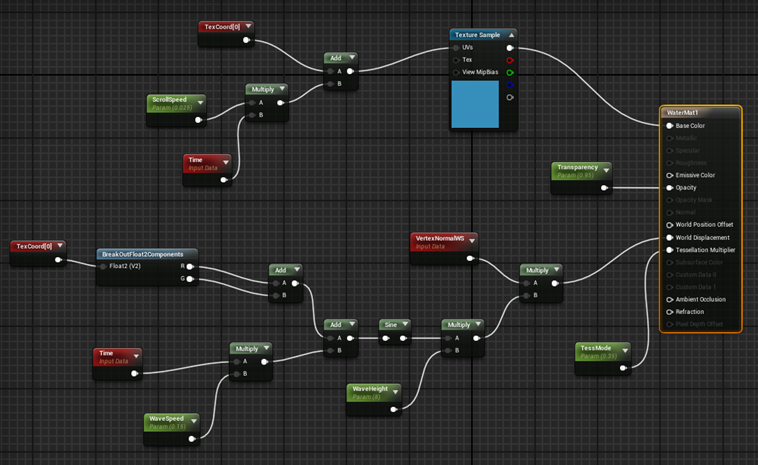
### Materijali

Materijali su vizualna sredstva koja predstavljaju način na koji će se prikazivati ploha trodimenzionalnog modela u Unreal Engine-u. U suštini, materijali su ekvivalentni programu za sjenčanje (engl. *shader*) u API-jima niže razine. Materijali se izrađuju povezivanjem čvorova koji predstavljaju različite matematičke operacije, konstante, parametre i sl. Krajnji dio izračuna se povezuje sa izlaznim čvorom materijala čiji parametri utječu na kakav će se način prikazivati objekt unutar igre.



*Slika 2.2.3.1. Izlazni čvor materijala*

Na slici 2.2.3.1. je prikazan izlazni čvor materijala, koji na temelju vlastitih svojstava odlučuje koji su parametri aktivni (prikazani bijelom bojom), a koji nisu (prikazani sivom bojom).



Slika 2.2.3.2. Čvorovi materijala za vodu

Na slici 2.2.3.2. je prikazan materijal korišten za prikazivanje vode. Voda je predstavljena običnom plohom podijeljenom na mali broj trokuta. Na materijalu vode je postavljen prozračan (engl. *translucent*) princip miješanja boja, a također je aktivirana teselacija (koja zahtijeva *Direct3D* verziju 11 ili *OpenGL* verziju 4.0). Teselacija nam omogućuje podpodjelu plohe na više manjih elemenata, a gustoća tih manjih elemenata je određena njihovom udaljenosti od kamere prikaza (bliže kameri se stvara više manjih elemenata). U materijalu možemo pristupiti svakome vrhu od tih novonastalih elemenata, te ga dispozicionirati (u našem slučaju, to radimo samo po Z osi). Za izračun razmještaja u visini svakog vrha koristimo UV teksturnu koordinatu svake točke te vrijeme od početka igre kao parametar. Vrijednosti ubacujemo u sinusoidu koja će nam omogućiti da na relativno primitivan način dobijemo „valovito“ ponašanje, koje istovremeno neće biti previše zahtjevno za crtanje na grafičkoj kartici. Osim toga, također dodajemo odmak na uzorkovanje (engl. *sampling*) teksture koji se množi sa vremenom od početka igre, što nam daje dodatan privid pomicanja valova.

Konstante brzine kretanja valova, amplitude vala, pomicanja odmaka teksture i veličina svakog polja teksture su pretvorene u uniformne parametre, tako da se stvaranjem nove instance istog materijala mogu mijenjati navedene vrijednosti bez potrebe za ponovnim prevođenjem.

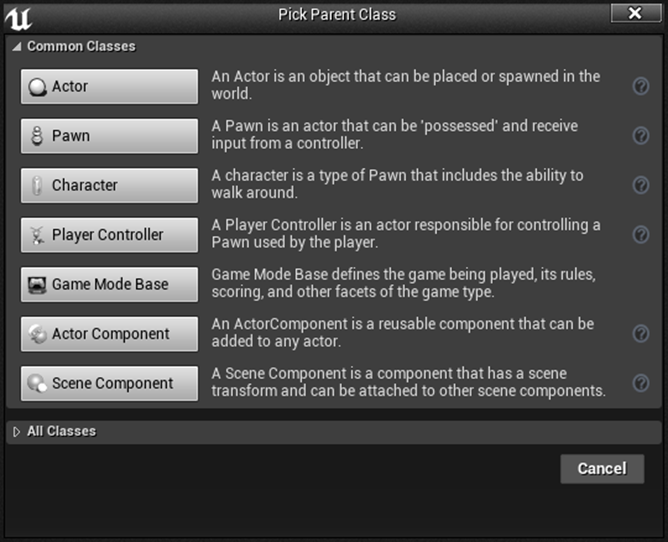
Materijali ostalih modela su mnogo jednostavniji, te uglavnom samo koriste glavnu izlaznu boju te mijenjaju parametre fizičko baziranog prikazivanja da bi materijal odgovarao njegovim svojstvima u stvarnom životu.

## Logika i programiranje

### Općenito o logici

Unreal Engine 4 podržava dva načina stvaranja logike igre, a to su programiranjem u jeziku C++ ili vizualnim povezivanjem čvorova u Blueprintu. Prevođenje C++ koda zahtjeva instaliran C++ prevoditelj (primjerice MSVC, GCC, Clang i sl.). Nazivi funkcija i parametara iz C++ API-ja su ekvivalentni onima iz Blueprinta, te kombiniranje C++ i Blueprint logike je dozvoljeno. Zbog jednostavnosti i razumljivosti svim članovima tima, odlučeno je da svu logiku igre napišemo u Blueprintu.

### Kretnje igrača

Postoji nekoliko tipova baznih razreda koji se mogu koristiti za stvaranje Blueprint skripte. Svi ti razredi su prikazani na slici 2.3.2.1.

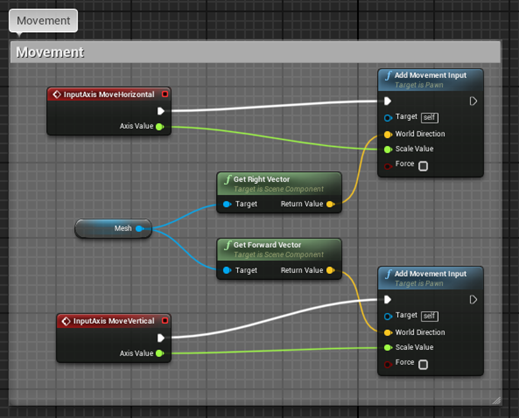
Slika 2.3.2.1. Stvaranje nove Blueprint skripte

Razred *Actor* je najosnovniji tip objekta i koristi se kao baza svakog objekta u trodimenzionalnom okruženju. Njega nasljeđuje razred *Pawn*, koji se koristi kao minimalna osnova upravljača igrača u Unreal Engine-u. Glavno svojstvo koje *Pawn* posjeduje jest obuzimanje (engl. *possess*) od strane igrača, što mu omogućuje da preusmjeri ulaze tipkovnice, miša i kontrolera igrača na svoje rutine za obradu događaja te također da koristi svoju komponentu kamere kao vizualni prikaz na igračevom ekranu (u slučaju da nema tu komponentu, koristit će se zamišljena kamera sa koordinatama lokacije *Pawn* objekta i usmjerenjem prednjeg vektora). Razred *Character* nasljeđuje razred *Pawn* i tako posjeduje sva svojstva koja i on. Štoviše, razred *Character* posjeduje dodatne funkcije koje omogućuju da se lakše realizira igrač koji se kreće po XY ravnini, a na njega djeluje gravitacija po Z osi.

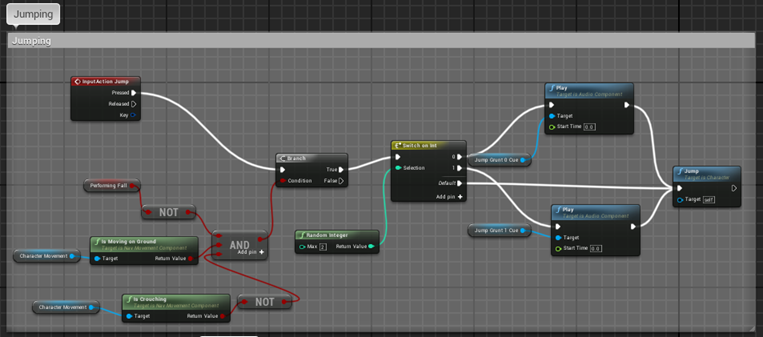
**Za izradu skripte igrača korišten je razred *Character* kao baza. Kontrole igrača su mapirane putem sustava preferenci, kao što prikazano na slici 2.3.2.2.

Slika 2.3.2.2. Popis kontrola igrača

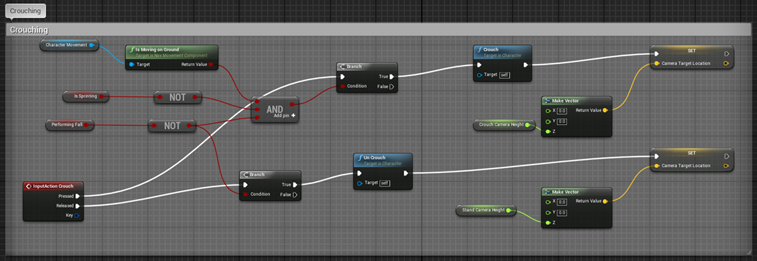
Postoje dvije glavne skupine mapiranih kontrola igrača, akcijska (engl. *action*) i osna (engl. *axis*). Razlika između njih jest u tome što su stanja akcija prezentirana putem binarnih stanja (gumb je pritisnut ili gumb nije pritisnut), a stanja osi su prezentirana putem zapisa sa pomičnim zarezom (engl. *floating point*).



Slika 2.3.2.3. Blueprint čvorovi zaduženi za kretnje igrača

Na slici 2.3.2.3. su prikazani čvorovi koji upravljaju kretnjama igrača. Dva lijeva crvena pravokutnika predstavljaju događaje koji se aktiviraju svaki otkucaj (engl. *tick*) sa parametrom vrijednosti osi za koje su zadužene u intervalu [-1,1]. Čvor *Mesh* predstavlja komponentu igrača iz čije rotacije dobavljamo prednji i desni vektor usmjerenja. Te vektore koristimo kao smjer u svijetu u kojem će se igrač kretati na dva desna čvora (dio *Pawn* razreda).

Slika 2.3.2.4. Blueprint čvorovi zaduženi za skakanje igrača

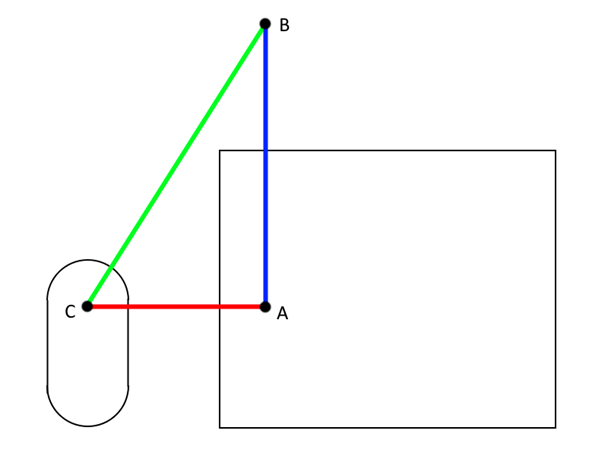
Na slici 2.3.2.4. su prikazani čvorovi koji omogućuju skakanje. Ovi čvorovi se aktiviraju u trenutko kada se stanje akcije *Jump* promijeni iz neaktivnog u aktivno (tj. gumb je pritisnut a prethodni otkucaj nije bio pritisnut). Grananje nam omogućuje da ograničimo izvođenje skoka samo za trenutke kada je on omogućen (igrač ne pada, nalazi se na tlu i ne čuči). Postoje dva zvučna efekta koja se reproduciraju kada igrač skoči, a onaj koji se reproducira je nasumično odabran. Desni čvor *Jump* je funkcija razreda *Character* i interno dodaje brzinu prema +Z osi igrača da bi se ostvario skok.

Slika 2.3.2.5. Blueprint čvorovi zaduženi za čučanje igrača

Na slici 2.3.2.5. su prikazani čvorovi koji omogućuju čučanje igrača. Za prelazak u stanje čučenja, treba se osigurati da se igrač nalazi na tlu, da ne trči i da ne pada. Ako su ti uvjeti zadovoljeni, poziva se metoda razreda *Character* koja aktivira tu funkcionalnost. Također se visina kamere reducira na visinu čučenja tako da se postavi varijabla koja se kasnije u događaju otkucavanja koristi za glatki prijelaz lokacije kamere. Analogno tomu je realiziran i način prijelaza iz stanja čučanja.

### Mehanizam penjanja

Mehanizam penjanja je dodan da bi igrač mogao pristupiti visokim platformama inače nepristupačnih običnim skokom. Za realizaciju takvog mehanizma je korišten sustav emitiranja zraka (engl. *raycast*) fizičkog sustava Unreal Engine-a i slijedno linearno animiranje lokacije i rotacije igračevog objekta i kamere.



Slika 2.3.3.1. Detekcija mogućnosti penjanja nakon skoka

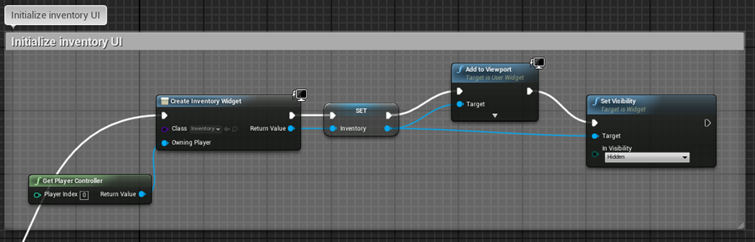
Na slici 2.3.3.1. je prikazana shema koja se koristi za detektiranje jeli je skok moguć u određenom trenutku. Kapsula prikazuje igrača, a pravokutnik prikazuje čvrsti objekt. Točka C je definirana u odnosu na poziciju igrača i njegov prednji vektor. Točke A i B su definirane u odnosu na točku C. Udaljenosti među tim točkama su zapisane kao varijable u Blueprint skripti.

Prvi uvjet da bi se ispitivanje penjanja nakon skoka uopće provelo jest da igrač se nalazu u zraku (to jest, igrač je skočio). Ako je taj uvjet zadovoljen, emitira se zraka od točke C prema točki A koja provjerava dali je na njihovom putu detektiran čvrsti objekt. Ako je, onda se emitira zraka iz točke B prema točki A koja na svom putu mora također detektirati isti čvrsti objekt. Osim toga, kut između normale objekta u toj točki sudara i +Z osi mora biti jako mali (da bi se osiguralo da objekt na koji se penje nije previše strmovit). Na kraju se također emitira zraka između točaka C i B koja ne smije detektirati niti jedan objekt (da bi se osiguralo da je put skoka oslobođen). Nadalje, radi jednostavnosti, s obzirom da ovaj mehanizam se koristi samo na par mjesta u igrici, omogućen je samo da se provodi na točno određenim čvrstim objektima.

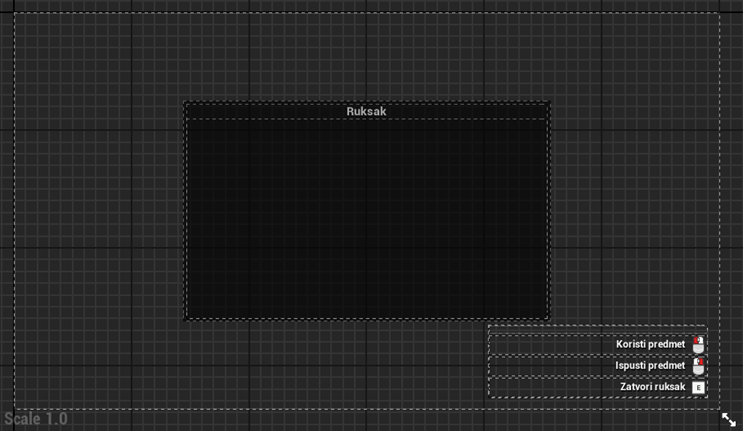
Ako su uvjeti za penjanje nakon skoka zadovoljeni, onda igrač prelazi u stanje „animiranog“ penjanja, koje u jako kratkom vremenu animira igračevu lokaciju i rotaciju kamere po putu do vrha objekta na kojeg se igrač želi popeti, U trenutku kada stigne do te lokacije, izlazi se iz stanja „animiranog“ penjanja i vraća se kontrola igraču te se reaktivira fizički sustav.

### Inventar i sučelje

U skripti koja je tipa *GameMode* je definiran sustav za rad sa inventarom igrača. Objekti koje igrač može sakupljati ili koristiti nasljeđuju razred *ItemBase*, a popis svih objekata koje igrač trenutno nosi u inventaru se nalazi u varijabli niza referenci tog razreda. Na slici 2.3.4.1. je prikazano stvaranje korisničkog sučelja za inventar igrača.



Slika 2.3.4.1. Stvaranje korisničkog sučelja za inventar

Korisničko sučelje inventara je stvoreno u UMG UI dizajneru, a njegov izgled je prikazan na slici 2.3.4.2.

Slika 2.3.4.2. Korisničko sučelje inventara

Objekti koje je igrač sakupio u svijetu će biti prikazani u prozoru, a koristeći tipke miša će biti u mogućnosti da nosi predmet (primjerice sjekira ili baklja) ili da izbaci predmet iz inventara.

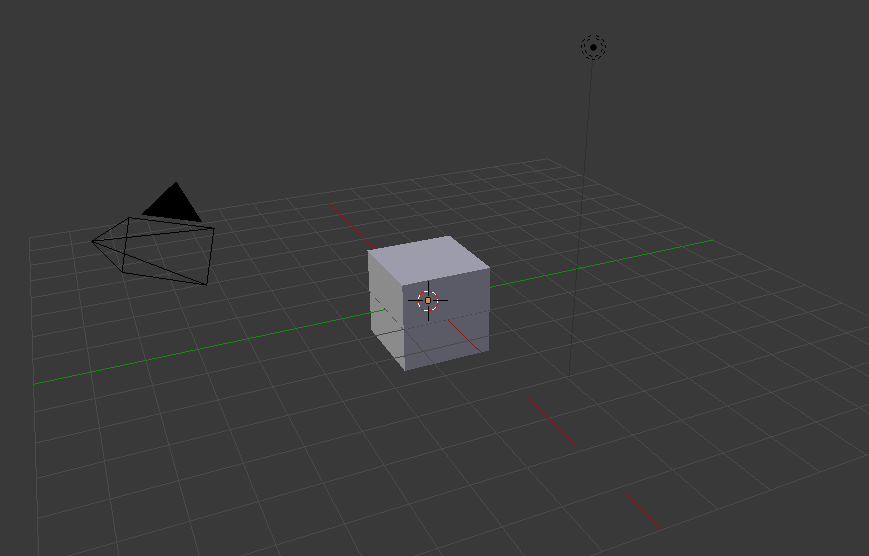
## Modeli

### Blender

Blender je besplatan softver otvorenog koda koji omogućuje modeliranje 3D objekata, izradu tekstura, simulacija, animacija, iscrtavanje prikaza na ekranu (eng. *rendering*) te uređivanje video zapisa. U ovom projektu korišten je za modeliranje trodimenzionalnih predmeta koji su uvezeni u Unreal Engine i tamo postavljeni kao objekti unutar igre.

### Proces modeliranja

Postupak modeliranja unutar Blendera započinje stvaranjem nove scene. Otvara se zadana scena koja sadrži koordinatnu mrežu sa oznakom ishodišta, jednostavnim objektom kocke smještenim u središte scene te objektom kamere i rasvjete koji se koriste za vizualizaciju scene iz određene perspektive (slika 2.4.2.1.).

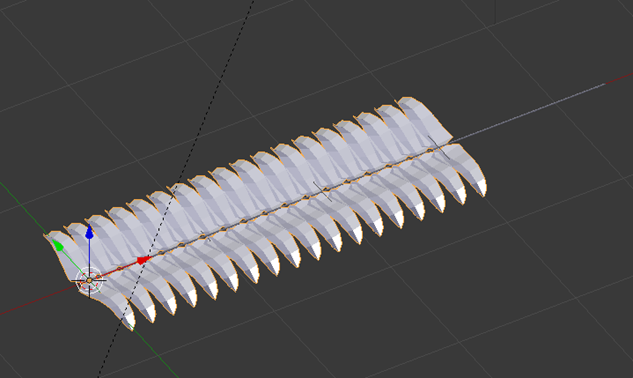


Slika 2.4.2.1. Zadana scena u Blenderu

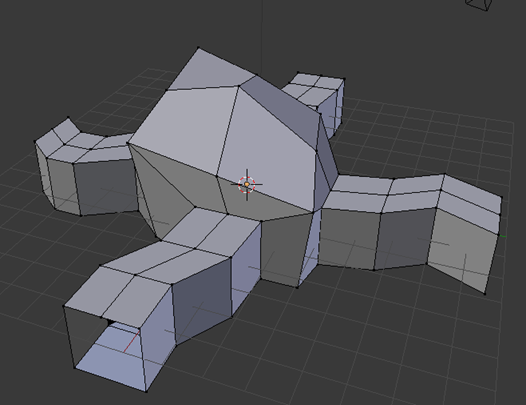
Tijekom modeliranja koristi se nekoliko načina rada (eng. *mode*) koje softver nudi. Način rada *Object mode* koristi se za stvaranje novih objekata unutar scene, promjenu njihove veličine, pozicije i rotacije. Neki osnovni objekti koji se koriste kao temelj za izradu modela su ravnina, kocka, krug, sfera, valjak itd. U slučaju da želimo modelirati objekt daske, stvorit ćemo kocku, objekt najsličniji traženom modelu i prilagoditi njene dimenzije.

U slučaju da je potrebno urediti postojeće ili dodati nove vrhove, bridove ili strane objekta, pristupa se načinu rada *Edit mode*. Pored navedenih funkcionalnosti, ovaj način rada nudi brojne druge opcije za deformaciju objekta. Jedna od njih je opcija izvlačenja (eng. *extrude),* koja omogućuje “izvlačenje” nove strane iz postojeće strane objekta (eng. *face)*. Korištenjem opcija izvlačenja i promjene pozicija vrhova objekta, iz zadanog objekta kocke moguće je u kratkom roku doći do modela nalik unutrašnjosti špilje, koji se koristi u samoj igri (slika 2.4.2.2.).

Pored ručnog oblikovanja, Blender nudi mogućnost primjene modifikatora nad objektima. Modifikatori su automatske operacije nad objektom koje bi bile presložene za ostvariti ručno. Nad jednim objektom mogu se primijeniti višestruke modifikacije. Neki od korištenih modifikatora tijekom stvaranja modela za projekt su generativni modifikatori *Array* (slika 2.4.2.3)*,* koji stvara niz primjeraka nekog objekta sljedeći oblik zadane krivulju te *Subdivision Surface* koji izgladi površinu objekta dijeljenjem jedne strane na više dijelova.

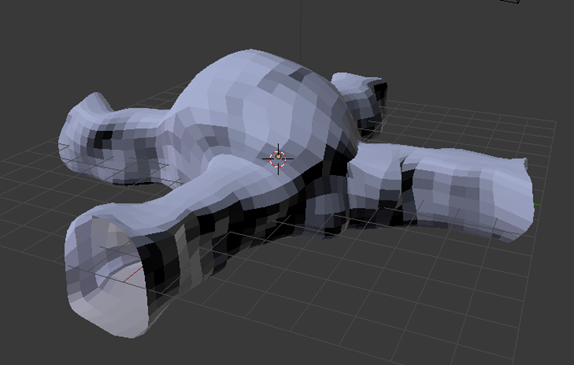


Slika 2.4.2.3. Modifikator Array



Slika 2.4.2.2. Model napravljen pomoću opcije Extrude

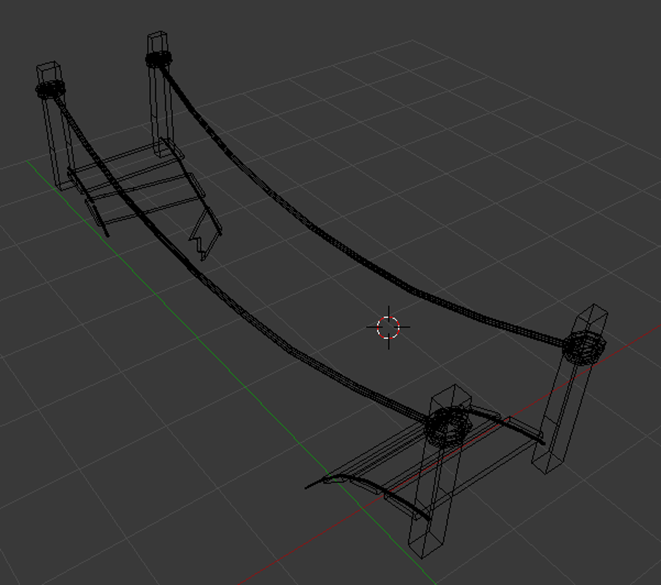
Za deformiranje objekta unutrašnjosti špilje, korišten je deformirajući modifikator *Displace Modifier,* koji je primijenjen nad objektom u smjeru X i Y koordinate (slika 2.4.2.4.).



Slika 2.4.2.4. Modifikator Displace

Pored navedenih načina rada koji se najčešće koriste, dostupni su i *Sculpt Mode* koji omogućuje uređivanje objekta korištenjem alata za skulpturiranje, *Texture Paint Mode* koji dopušta bojanje mreže objekta različitim teksturama, *Particle Edit Mode* koji se koristi za uređivanje sustava čestica koji se nalaze na nekom objektu te mnogi drugi.

Objekti unutar scene mogu biti prikazani na nekoliko načina (eng. *Shading Modes)*. Zadani način koji se koristi na prethodnim slikama naziva se *Solid*, koristi jednobojni prikaz objekta i jednostavno osvjetljenje. Za preciznu predodžbu scene koristi se način *Rendered* koji za prikaz koristi odabrani pogon za iscrtavanje (eng. *Render engine*) te u obzir uzima korištene materijale i izvore svjetlosti koji se nalaze na scenu. Način *Wireframe* prigodan za promjenu izgleda objekta, a prikazuje ga kao mrežu linija koje predstavljaju njegove bridove (slika 2.4.2.5.).



Slika 2.4.2.5. Način prikaza objekta Wireframe

Nakon izrade modela, on se izvozi iz Blendera kao datoteka formata *fbx* koju je moguće uvesti u Unreal Engine.

U slučaju da je model kompliciran i sadrži puno vrhova, bridova i strana, korisno je prije izvoza dodati jednostavne objekte (kvadar, kugla, ...)*,* koje Unreal interpretira kao pomoćne objekte iz kojih se može generirati mreža kolizije (eng. *Collision mesh*) koja je manje zahtjevnija za procesor od one koja se automatski stvara bez njih. Pomoćni objekti za izradu mreže kolizije mogu biti tipa UCX (bilo koji konveksni 3D oblici), UBX (samo objekti kocke), UCP(objekti kapsule) ili USP (sferni objekti).

### Primjeri izrađenih modela

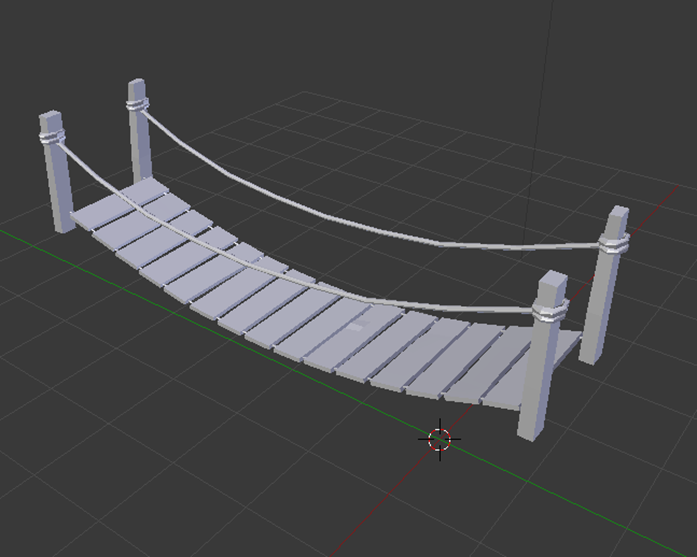
Pored nekoliko modela koji su preuzeti sa Unity-jevog store-a, ostale su kreirali članovi tima unutar Blendera. Neki od modeliranih objekata su ukosnica koja predstavlja autentični predmet sa izložbe o braći Seljan (slika 2.4.3.1.), model indijanskog šatora koji se koristi u selu unutar kojeg započinje igra (slika 2.4.3.2.) te model mosta koji omogućuje prijelaz s jedne na drugu obalu rijeke (slika 2.4.3.3.).



Slika 2.4.3.2. Model indijanskog šatora



Slika 2.4.3.1: Model ukosnice



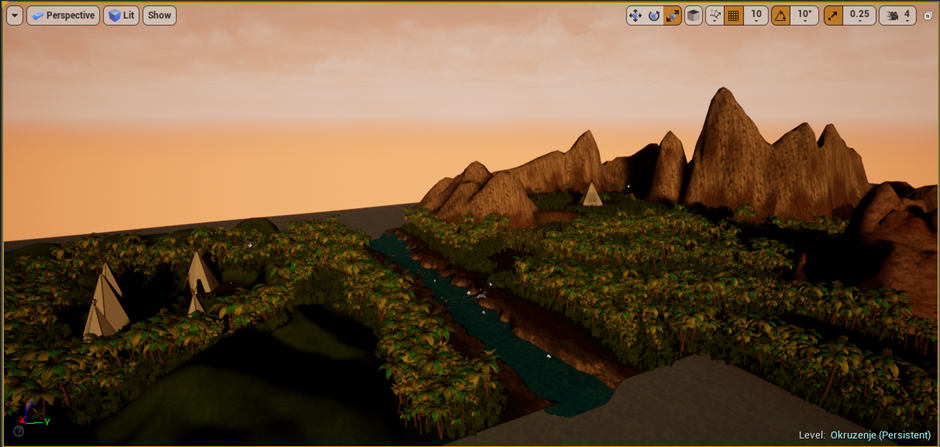
Slika 2.4.3.3: Model mosta

## Osvjetljenje

Osvjetljenje je jedna od ključnih stavki pri kreiranju uvjerljivog okruženja. Osvjetljenje čini toliku razliku da se dva okruženja koja se razlikuju samo u načinu kako se implementira osvjetljenje mogu činiti kao dva potpuno drugačija okruženja od kojih jedno izgleda vizualno manje atraktivno od drugog upravo zbog osvjetljenja. Osvjetljenje također pomaže pri stvaranju određenog ugođaja kojeg se želi postići unutar okruženja. Važan korak nakon dodavanja i promjene različitih osvjetljenja je izgradnja svjetlosnih mapa (*Build Lightning*). To je postupak kojim pokretač izrađuje mape svijetla koje su potrebne kako bi osvjetljenje moglo funkcionirati kako je namijenjeno. Postupak je zahtjevan iz perspektive vremena koje je potrebno kako bi ga pokretač obavio.

## Globalno osvjetljenje

Kada se kreira razina po predlošku koji sadrži nebo i početnu ploču, unutar razine se nalaze dva glumca koji simuliraju nebo. Jedan od tih glumaca je predstavljen ikonom sunca. Ta ikona predstavlja glavno, odnosno globalno osvjetljenje. Ono u suštini u pratećem projektu simulira svjetlost koja dolazi od sunca pošto je okruženje krajolik. Tom globalnom osvjetljenu moguće je mijenjati razna svojstva kako bi se stvorio željeni ugođaj. Za potrebe projekta, simulira se ugođaj zore. Slika 2.5.1.1. prikazuje promijenjeno osvjetljenje. Kako bi se postigao takav ugođaj biti će potrebno promijeniti nekoliko svojstava globalnog osvjetljenja. Jedna od drastičnijih promjena je promjena boje svjetla. Ona se može postaviti na realistične nijanse boje, no može se postaviti i na neprirodne boje ukaže li se za to potreba u dizajnu. Osvjetljenje unutar projekta je u nijansama narančaste boje. Sljedeća promjena je promjena pozicije osvjetljenja. Naime, potpuno bi neprirodno bilo da sunce bude visoko na horizontu, a željeni je ugođaj ugođaj zore. Zato se rotira izvor svijetla kako bi mu se odredila visinu na horizontu. Također može se mijenjati i smjer iz kojeg svijetlo dolazi. Promjena pozicije globalnog svjetla vrši se isto kao i za svakog glumca odnosno objekt. Ostale se promjene vrše pronalaskom odgovarajuće opcije unutar popisa svojstava (*Details Panel*) odabranog glumca glavnog osvjetljenja. Postoje mnoge druge postavke, koje je moguće mijenjati kako bi glavno svjetlo poprimilo različita svojstva. Jedna od tih postavki je mogućnost stvaranja takozvanih Božjih zraka (*God Rays*), odnosno efekta koji simulira senzaciju prolaska svjetla kroz raslinje. Efekt esencijalno ne dodaje ništa mehanički u igru, no pomaže pri realizaciji realističnog okruženja.



Slika 2.5.1.1. Osvjetljenje

## Točkasto osvjetljenje

Točkasto svjetlo je najjednostavniji oblik osvjetljenja. Ono je glumac kojeg je moguće postaviti na mjesto unutar okruženja koji bi trebao isijavati određenu vrstu svjetlosti. Unutar pratećeg projekta koristit će se u svrhu dodavanja svijetla koje je proizvod ognjišta pored kolibe. Iz te svrhe slijede svojstva poput ograničenog radijusa djelovanja te boje, koja je zbog toga u spektru crvene boje. Promjena pozicije i svojstava vrši se isto kao i kod globalnog osvjetljenja.

## Vizualni učinci

## Efekt magle

Kako bi se stvorio što bolji ugođaj ranog jutra u udolini, krajolik će biti upotpunjen efektom magle. Za dobivanje efekta magle koristit će se gotovi glumac Exponential Height Fog. Kao i većini drugih glumaca, i glumcu magle je moguće mijenjati razna svojstva poput gustoće magle, udaljenost od igrača na kojoj se pojavljuje, boju i slično. Slika 2.6.1.1. prikazuje okruženje s prisutnim efektom magle.

Slika 2.6.1.1. Prikaz efekta magle

## Sustavi čestica

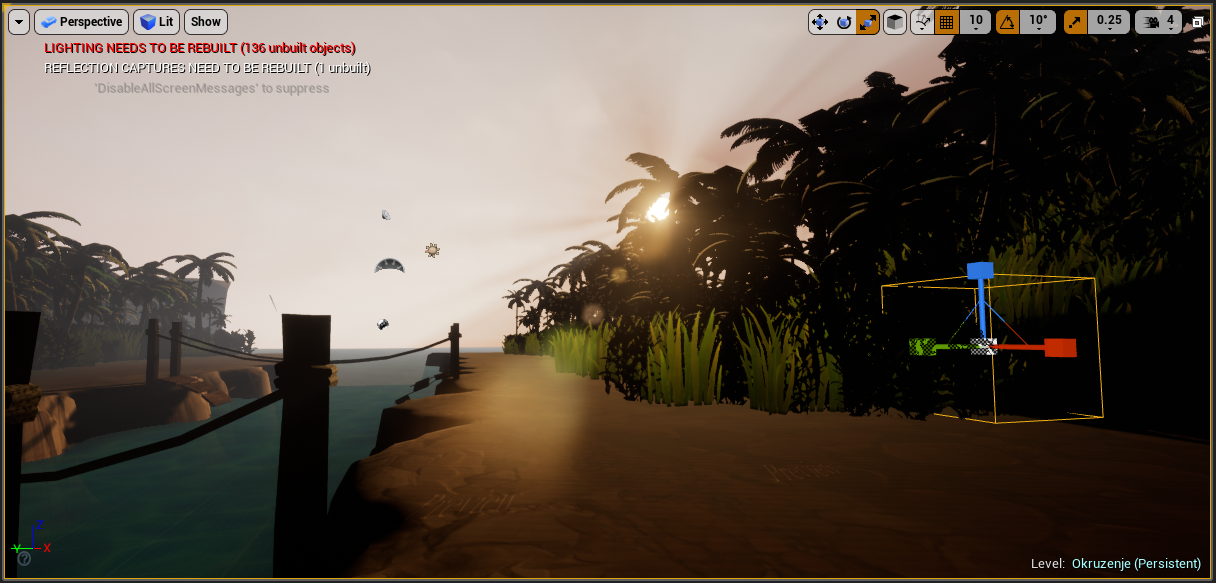
Sustavi čestica (*Particle Effects*) vizualni su učinci koji simuliraju prirodne pojave poput vatre ili dima. Generalno gledano ovi efekti su izrazito skupi gledajući performanse s njima i bez njih, no takvi efekti su jedni od efekata koji uvelike povećavaju dojam realističnosti kojeg okruženje ostavlja na igrača. Za potrebe projekta koristit će se gotovi sustavi čestica za vatru baklje. Inače, moguće je kreirati vlastite sustave čestica koristeći Unreal Engineov 4 uređivač efekata čestica (*Cascade Editor*). Na slici 2.6.2.1. prikazan je navedeni sustav postavljen unutar okruženja projekta.



Slika 2.6.2.1. Prikaz čestica za vatru

## Odsjaj leće

Odsjaj leće je efekt koji simulira senzaciju jarkog svjetla u obliku konusa koji nastaje kada se gleda direktno u veliki izvor svjetlosti poput sunca. Njemu je moguće mijenjati svojstva poput jačine efekta, finoće (hoće li efekt biti u obliku mnogokuta ili nešto zaobljeniji) i slično. Slika 2.6.3.1. prikazuje efekt odsjaja leće.



Slika 2.6.3.1. Primjer odsjaja leće

## Zamućenje pokreta

Zamućenje pokreta je efekt koji simulira zamućenost dijela vidnog polja prilikom kretanja. Također, kao i kod drugih efekata, moguće je mijenjati određena svojstva poput jačine efekta. No jačina efekta zapravo samo skalira efekt pošto sam efekt djeluje na način da ovisno o iznosu brzine pojačava odnosno smanjuje zamućenost. Zamućenje pokreta je jedan od češće korištenih efekata u industriji video igara današnjice.

Odsjaja leće i zamućenje pokreta spadaju u skupinu učinaka naknadne obrade. Učinci naknadne obrade su efekti koji se dodaju jednoj sličici (*frameu*) kao zadnji korak u njezinom generiranju. U današnjoj industriji video igara pri izradi igre koriste se mnogi takvi efekti, no većina njih je opcionalna, odnosno ostavlja se igraču na izbor želi li ih imati prisutne prilikom igranja ili ne. Neki od tih efekata su odsjaj leće (*Lens Flare*), zamućenje pokreta (*Motion Blur*), ambijentalno zasjenjenje (*Ambient Occulsion*) te mnogi drugi. U pratećem projektu bit će primijenjeni samo neki od mnoštva mogućih efekata. Efekte je moguće primijeniti na cijelu razinu ili određenim dijelovima razine pridodati određene navedene efekte.

## 