Izrada fotorealističnog 3D okruženja pomoću Unreal Enginea

Košuljandić, Toni

Master's thesis / Diplomski rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka / Sveučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:195:297437

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija

Repository / Repozitorij:

Repository of the University of Rijeka, Faculty of Informatics and Digital Technologies - INFORI Repository

zir.nsk.hr

DIGITALNA KNJIŽNICA

Image not found or type unknown

Sveučilište u Rijeci – Odjel za informatiku

Informacijski i komunikacijski sustavi

Toni Košuljandić

Izrada foto realističnog 3D okruženja pomoću Unreal Enginea

Diplomski rad

Mentor: izv. prof. dr. sc. Marina Ivašić-Kos

Rijeka, prosinac 2019.

ODJEL ZA INFORMATIKU
SVEUČILIŠTE U RIJECI

Rijeka, 3.6.2019.

Zadatak za diplomski rad

Pristupnik: Toni Košuljandić

Naziv diplomskog rada**: Izrada foto realističnog 3D okruženja pomoću Unreal Enginea**

Naziv diplomskog rada na eng. jeziku: **Creating photorealistic 3D environment with Unreal Engine**

Sadržaj zadatka:

Opisati osnovne karakteristike 3D modela te postupke njihove izrade. Istražiti tehnologiju i alate koji se koriste za izradu 3D modela i postavljanje okruženja te ukratko opisati ključne karakteristike alata koji će se koristiti u praktičnom dijelu rada. U praktičnom dijelu rada izradit će se 3D model Kampusa s prilaznim cestama koristeći alate Autodesk Maya 2019, Zbrush i Unreal Engine.

Mentor:	Voditeljica za diplomske radove:
Izv. prof. dr. sc. Marina Ivasic-Kos	Izv. prof. dr. sc. Ana Meštrović
Komentor:	Zadatak preuzet: 06.06.2019

Toni Košuljandić

(potpis pristupnika)

Zahvala:

Ovim putem zahvaljujem se izv. Prof. dr. sc. Marini Ivašić-Kos zbog strpljenja i pružene prilike da radim na ovoj zanimljivoj temi. Također, posebnu moram zahvaliti gospodinu Velimiru Vrziću, fotografu za Google Maps, koji je nesvjesno ispao junak i uštedio mi brojne sate prikupljanja podataka.

Dugujem zahvalu univ. bacc. Inf. Robertu Ružiću i Mihaeli Vrban za brojne rasprave i strpljenje koje su vodile do završetka ovog rada. Također hvala Aleksandri Pilepić na toleranciji i razumijevanju za vrijeme izrade rada.

najveću zahvalu bi posvetio Tari Pilepić, svojoj boljoj polovici, na podršci kroz cijelo vrijeme izrade rada i sprječavanju da se svijet ne uruši oko nas. Bez tebe, ovo ne bih uspio.

Sadržaj:

Zahvala: 2
Sažetak 4
Ključne riječi 4
Uvod 5
Alati za 3D modeliranje 7
Sučelje Autodesk Maye 8
Unreal Engine 11
Razvijanje Unreal Enginea 12
Sučelje Unreal Enginea 14
Collision 17
Materijali 18
Vrste tekstura 20
Izrada Terena 26
Vegetacija 27
Performanse 31
Praktični rad 32
Hardware i njegova uloga u obradi 3d prostora 36
DLSS 37
RTX 38
Radeon Image Sharpening 39
Ray Tracing – tehnologija budućnosti, danas 40
Zaključak 43
Popis literature 44
Popis Priloga 44
Popis slika 45

Sažetak

U ovom diplomskom radu izrađena je foto realistična scena Unreal Engineu. Scena uključuje područje od zgrade Filozofskog fakulteta do restorana Kampus po Sveučilišnoj Aveniji. U radu su objašnjena sučelja programa Maya i Unreal Engine, površni uvid u model praćenja zraka svijetlosti, izrada materijala i terena i optimizacija performansi u Unreal Engineu. U ovom radu svi materijali prikazani su u potpunosti besplatni zahvaljujući edukacijskim programima Autodeska i Epic Games studia. Teksture su preuzete sa besplatnih paketa iz Unreal trgovine i stranice sa besplatnim foto realističnim paketima, Quixel.

Ključne riječi

Autodesk Maya, Unreal Engine, kampus, 3D prostor, foto realistična scena, simulacija

Uvod

Računalna grafika je prošla dugi put od prvih svijetlih piksela na crnim pozadinama do danas kada su virtualni svjetovi popunjeni i najsitnijim detaljima. Računalna grafika je grana informatike koja se bavi stvaranjem slika uz pomoć računala [11].

Danas je računalna grafika osnovna tehnologija u digitalnoj fotografiji, filmu, video igrama, mobilnim i računalnim aplikacijama, te mnogim specijaliziranim aplikacijama. Razvijeno je mnogo specijaliziranog hardvera i softvera, koji pogone većinu uređaja koje imaju zaslon. Računalna grafika je odgovorna za prikaz slika na razumljiv i prihvatljiv način korisniku, kao i za značajni utjecaj na animaciju, filmove, reklamiranje i video igre [11]. Samu računalnu grafiku bi mogli podijeliti na dvije kategorije: 2D dizajn i 3D modeliranje. 3D modeliranje je proces razvijanja matematičkog prikaza bilo koje površine objekta (bilo nežive ili žive) u tri dimenzije putem specijaliziranog softvera [11]. Kako bi prikazali 3D prostor na 2D zaslonu monitora, potrebno je provesti proces renderiranja ili prikazivanje. Renderiranje je metoda preslikavanja objekata iz 3D prostora na 2D ravninu s obzirom na svojstva objekata (vrsti materijala, oblika, s obzirom na sjene i drugim)[10]. 3D modeliranje je doživjelo procvat početkom 1990-tih. Pojavom osobnih računala i širenjem tržišta, 3D grafika se također proširila. Više nije trebala velika radna stanica koja je zauzimala pola sobe za proizvodnju 3D modela. Jedni od prvih projekata koji su donijeli veliku popularnost 3D grafici kakvu danas znamo su animirani film *Toy Story* od studija *Pixar* i *Virtua Racing*, igra koja je predstavila široj publici 3D poligonsku grafiku [11].

U praktičnom djelu rada možemo vidjeti scenu područja Kampusa, točnije Sveučilišnu aveniju sa pripadajućim zgradama fakulteta. Razlog odabira ovog područja ne leži u uzbudljivosti scene (i nije uzbudljiva, Kampus je relativno siromašan detaljima i bojama), već u potencijalu i radoznalosti kako bi područje moglo izgledati kroz nekoliko godina. Teren je prilagođen kako bi scena dobila potrebnu vegetaciju i paletu boja da suzbijemo sivilo betona i metala.

Sveučilište u Rijeci postoji od 1973. godine, a danas broji 11 fakulteta i 4 odjela. Sama gradnja Kampusa je započela 2003. godine sa izgradnjom zgrade Odjela, Građevinskog fakulteta i Filozofskog fakulteta. Godine 2006. počinje projekt rekonstrukcije vojarne Trsat kako bi danas udomila Akademiju primijenjenih umjetnosti. Sam prostor Kampusa je i dalje u izgradnji, a zadnji dovršeni projekt je Studentski dom i samostalna zgrada Studentskog centra. Na slikama od 1 do 4 prikazana je razlika u izgradnji Kampusa u razdoblju od 8 godina, između 2011. godine i danas.



Slika 1 Sveučilišna avenija, kolovoz 2011



Slika 2 Sveučilišna avenija, listopad 2019



Slika 3 Kampus, kolovoz 2011



Slika 4 Kampus, listopad 2019

Prije početka rada, bitno je bilo odrediti koji elementi se mogu napraviti, a koji će se zbog limitiranih resursa morati odbaciti. Unatoč tome što je Kampus ograđen cestama, područje je relativno veliko za jednu osobu koja izrađuje modele, teksture i materijale, sastavljanje terena i vegetacije, naljepnica i vrši optimizaciju prostora. Značajni problem bio je kako ograničiti teren. Teren se može ograničiti na nekoliko različitih načina, ovisno o reljefu, ali u konačnici većina rješenja ima sliku ili poligone niske razlučivosti u daljini koji su ograničeni

nekom barijerom. Primijenjeno je rješenje izrade velikih stranica sa jako sjajnim materijalom i glatkom površinom koja reflektira veliku količinu svijetlosti.

Alati za 3D modeliranje

Paralelno sa razvijanjem naprednijih grafičkih procesorskih jedinica, razvijale su se i aplikacije za 3D modeliranje. Danas na raspolaganju imamo mnoge složene aplikacije za 3D modeliranje koje su prezasićene opcijama i alatima. Neke od tih aplikacija su:

Autodesk Maya – Sa velikom bazom korisnika i podrške, Maya već godinama se postavlja kao industrijski standard, koji sadrži alate za izradu čestica, kose, tkanina, simulacije fluida i animacije likova.

Houdini – Pruža proceduralni pristup 3D modeliranju koji omogućuje velike količine automatizacije u izradi. Houdini koristi sustav povezivanja čvornih točaka kako bi stvorili potrebne modele sa maksimalnom kontrolom. Proceduralno programiranje 3D modela nam omogućuje da napravimo prepravke u kratkom vremenom koje bi inače uzele jako puno resursa da smo koristili klasične metode.

Cinema 4D – Program za 3D modeliranje s jednom od najlakših krivulja učenja na tržištu. Program ima podršku za modularnu nadogradnju i veliku galeriju priručnika za poteškoće u učenju.

Autodesk 3DS Max – 3DS Max se razvija u istoj kompaniji kao i Maya, ali pruža veći fokus na samo modeliranje za razliku od Maye koja je primarno fokusirana prema animacijama. Nedostatak programa je što je specijaliziran za uporabu na operativnom sustavu Microsoft Windows, ali može se pohvaliti opcijama za klasično i proceduralno modeliranje.

Blender – Blender je besplatni program otvorenog koda sa velikom podrškom korisnika. Nudi mnoge alate i opcije kao i komercijalna konkurencija i zato je interesantan korisnicima koji rade pod malim budžetom i ne mogu si priuštiti komercijalne alternative. Sama zajednica korisnika redovno objavljuje sadržaje za alat kojemu se jedino zamjera što sučelje nije intuitivno.

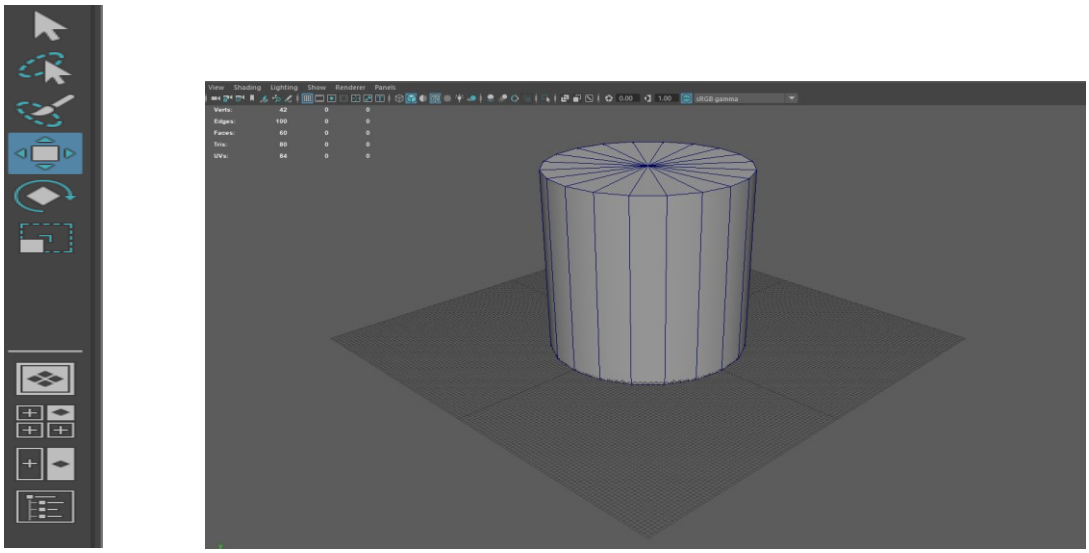
ZBrush – Za razliku od konkurencije, ZBrush nudi bogatu paletu alata za izradu skulptura. Kako klasični alati nude 3D modeliranje kao sastavljanje složenih objekta od jednostavnih geometrijskih likova, ZBrush ima pristup 3D modeliranju kao prema kiparstvu. Umjesto da sastavljamo naš složeni objekt, možemo ga izmodelirati iz postojećeg jednostavnog geometrijskog lika. Zanimljivo je da ZBrush može obrađivati nekoliko milijuna poligona na istom objektu bez gubitka u performansama. Radi promjene u metodi rada, treba spomenuti da je preporučljivo koristiti grafički tablet za rad u ZBrushu.

U ovom radu se koristio program Autodesk Maya zbog njezine besplatne licence za studente. Neke od mnogih drugih prednosti Maye je prilagodljivo sučelje, simulacija više različitih materijala nego bilo koji konkurentni program, kompatibilnost i mogućnosti kao što su eksportiranje u Unity ili Unreal engine [4].

Autodesk Maya je programski paket namijenjen izradi 3D modela, filmskih scena, vizualnih 3D efekata i animacija. Maya je nastala kombiniranjem koda iz „The Advanced Visualizer" od tvrtke Wavefront Technologies i „PowerAnimator" od tvrtke Alias Research te je verzija 1.0. službeno izašla u veljači 1998. godine [1][2]. Uz prepoznatu kvalitetu od strane Disney-a i Academy of Motion Picture Arts and Sciences, Maya dobiva na popularnosti i cijeli projekt kupuje Autodesk i preimenuje je u Autodesk Maya[2][4]. Mnogi autori daju prednost Mayi pred konkurentnim alatima radi vrhunski osmišljene navigacije i lakšeg snalaženja prilikom izrade animacija. Teško je uspoređivati Mayu sa glavnim konkurentom, 3DS Maxom, kada iza oba alata stoji ista kompanija, Autodesk. Kroz godine, oba programa su se postavila kao standardi u industriji, Maya za izradu animacija, a 3DS Max u izradi specijalnih efekata.

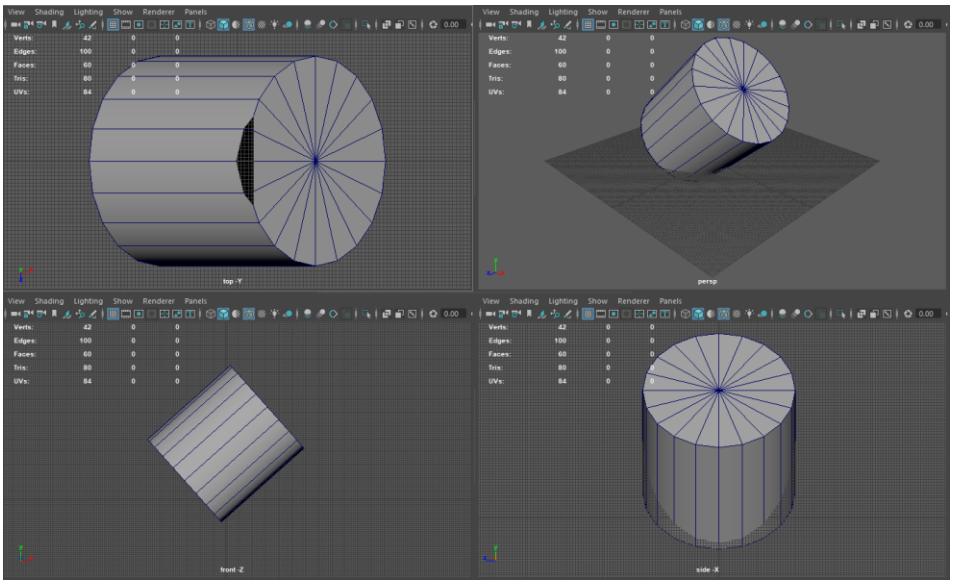
Sučelje Autodesk Maye

Prilikom paljenja Maye korisniku se prikaže veliki prozor sa 3D prostorom. Unutar 3D prostora se nalazi mreža koja predstavlja ravninu, a središte mreže definira ishodište koordinata (0,0,0). Neposredno iznad 3D prostora možemo naći alatnu traku sa postavkama kamere kao što su zaključavanje kuta, odabir kamere i generalnih postavki kamere (Slika 5). Nakon kamere, u istoj alatnoj traci možemo naći i funkcije prikaza objekata u prostoru i osvjetljenje scene ovisno koji renderer koristimo. Prilikom instalacije Autodesk Maye, program će doći sa 2 renderera, a predefinirani je ViewPort 2.0 koji prikazuje sve elemente modela i kutove gledanja osvjetljenima, kako korisnik ne bi imao prestrmu krivulju učenja. Mogućnosti prikaza modela su tu da pomognu korisniku kako bi lakše prepoznao rubove modela, uklonio sjene iz prikaza ili sakrio materijale kako bi lakše locirao tražene modele. Na samom kraju imamo korekciju gama vrijednosti kamere i postavke za Xray prikaz objekta. Xray je postavka unutar Maye koja objektima povećava transparentnost, omogućavajući korisnicima da označe objekte koje ne mogu vidjeti u određenim kutovima.



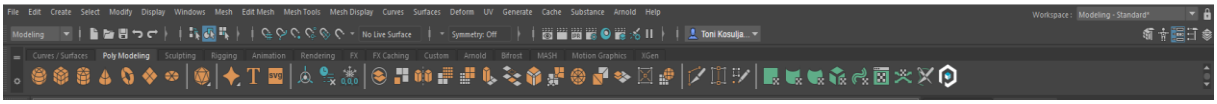
Slika 5 Elementi sučelja Maya, lijeva navigacija i 3d prostor sa mrežom

Lijevo od 3d prostora nalaze se alati za označavanje i manipulaciju objektima u prostoru. Alati omogućuju pomicanje, rotiranje i skaliranje objekata (Slika 5). Označavanjem objekta njegovi bridovi promijene boju, a odabirom pojedinog alata, na modelu se prikazuje i pivot točka sa osima (vrhovi osi su strelice ili kockice ovisno radi li se o pomicanju ili skaliranju). Alat za rotaciju objekta prikazuje kružnice u drugim bojama oko ishodišta rotacije, gdje nam svaka kružnica omogućava rotaciju po definiranoj koordinati. Rotaciju možemo i vršiti pomoću četvrte kružnice, koja uklanja barijere pojedine osi, ali predstavlja ograničenu uporabu i rijetko korištenje zbog problema manipulacije 3 koordinate u dvodimenzionalnom prikazu (na ekranu možemo vidjeti samo dvije osi, i miš možemo pomicati po dvije osi). Ispod alata manipulacije, nalaze se gumbi za različite prikaze sadržaja u 3d prostoru. Mogućnost da paralelno na ekranu vidimo nekoliko kutova uveliko olakšava određivanje veličina i izbacuje mnoge pretpostavke iz modeliranja (Slika 6). Sama naredba za paralelni prikaz tlocrta, nacrta, bokocrta i slobodne kamere je tipka space na tipkovnici. Svaku od tri druga prikaza možemo sa spaceom i uvećati, a unutar tih prikaza je onemogućeno rotirati kameru.

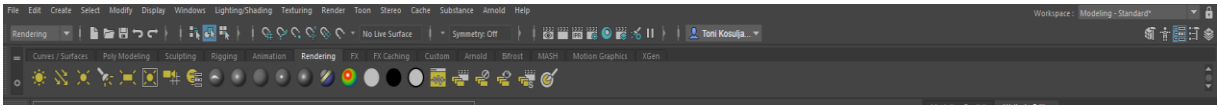


Slika 6 Maya, Paralelni prikaz više kamera

Na vrhu ekrana se nalazi nekoliko različitih alatnih traka. Na samom vrhu su izbornici za kontrolu projektima, uređivanje, selekciju, modificiranje, prikaz i pomoć. Ostali izbornici će se izmjenjivati ovisno u kojem okruženju radimo (Slika 7 i 8). Maya ima nekoliko okruženja koje biramo iz padajućeg izbornika i svako okruženje izmjenjuje glavnu alatne traku. Potpune opcije okruženja postižemo promjenom u padajućem izborniku i odabirom kartice sa imenom okruženja. Alatna traka sa karticama okruženjima je po zadanim postavkama treća od vrha. Kao primjer, u okruženju za modeliranje imamo alatnu traku sa dodavanjem različitih oblika u prostor i njihovu obradu kao što su dupliciranje, dijeljenje stranice, zaobljivanje rubova i druge, dok u okruženju za renderiranje imamo naredbe za dodavanje i manipulaciju svijetlom. Druga alatna traka ima gumbe za set različitih korisnih naredbi koje objedinjuju sva okruženja.

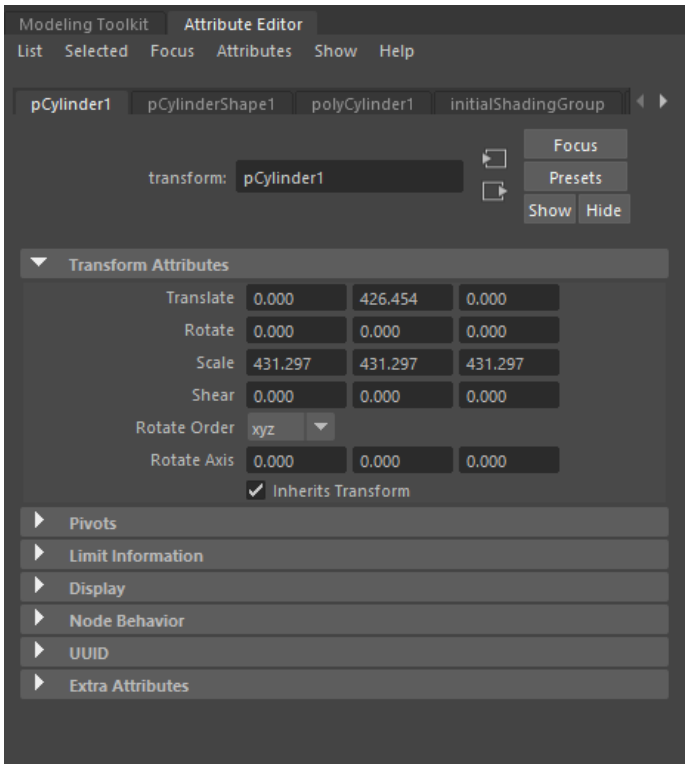


Slika 7 Maya, Glavna navigacija za modeliranje



Slika 8 Maya, Glavna navigacija za osvjetljenje

Desna strana nam služi za prikaz informacija o odabranom objektu (Slika 9). Klikom na objekt dobivamo informacije o njegovim dimenzijama, primijenjenom materijalu, broju stranica i mnoge druge podatke.



Slika 9 Maya, Desna navigacija i detalji o objektu

Unreal Engine

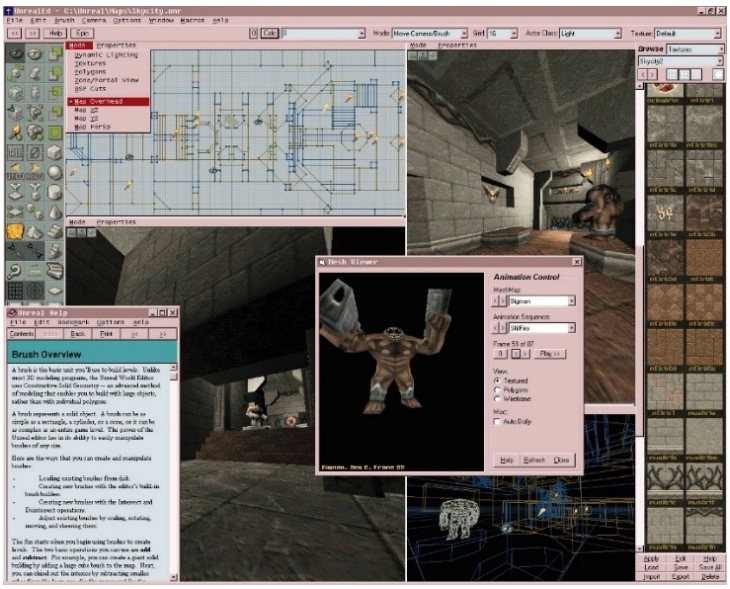
Unreal Engine (skraćeno UE) je razvojno okruženje za računalne igre od kompanije Epic Games [9]. Unreal Engine je prvi puta demonstriran javnosti kroz igre Unreal i nedugo zatim Unreal Tournamentom 1998. godine. Unatoč tome što je razvijan sa namjenom za računalne igre iz prvog lica, uspješno je našao primjenu i u drugim žanrovima. Veliku popularnost stekao je zbog originalno napisanog koda u C++ i velikom portabilnosti [9].



Slika 10 Tim Sweeney

Razvijanje Unreal Enginea

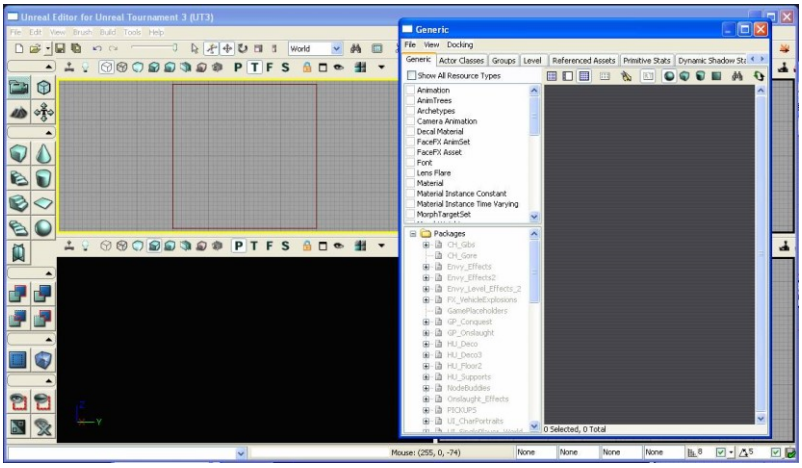
Prvu verziju razvio je Tim Sweeney (Slika 10), osnivač Epic Gamesa kroz 3 godine, a po nekim izvorima, sam je napisao 90% koda [9]. Glavna vodilja razvijanja novog enginea je bilo natjecati se sa ID Softwareom i njihovom dominacijom tržišta u igrama Doom i Quake. Glavne prednosti Unreala su bile detekcija sudara sa objektima, mogućnost višebojne rasvjete, grubu formu filtriranja tekstura i aplikaciju za izradu razina igre sa podrškom za konstruktivnu solidnu geometriju[9]. Aplikacija za izradu razina igre je omogućila mnogo brže provođenje promjena i time drastično smanjila vrijeme potrebno za razvijanje igra. Aplikacija je puštena u javnost i time izazvala stvaranje zajednice modelatora koji su izrađivali vlastite razine i objavljivali ih na internetu (Slika 11). Konstruktivno solidna geometrija je primjena pravila skupova nad geometrijskim tijelima. John Carmack, suosnivač ID Softwarea je priznao i nahvalio Unreal Engine za postavljanje novih standarda industrije, uvođenjem 16-bitne palete boja i volumetrijske magle [9]. Prva verzija Enginea je napisana sa fokusom da procesor računa većinu operacija, što se uvođenjem 3D akceleratora i razvijanjem tehnologije grafičkih kartica moralo promijeniti. Sweeney je nekoliko puta prepravljao osnovne algoritme za optimalni rezultat [9].



Slika 11 Unreal Level Editor, prva verzija

Sa velikim uspjehom originalne verzije i nudeći licenciranje drugim tvorcima igra za 350 000$, početkom 1999 počinje razvijanje druge verzije Enginea. Druga verzija je službeno izašla 2002 godine i sposobna prikazati i do 100 puta više detalja od originalne verzije [9]. Program je nadograđen značajkama kao što su kosturi za lakše izrade animacija, sustavom čestica i kompatibilnim formatom za Autodesk 3DS Max i Mayu.

Treća verzija programa je odličan primjer modularnosti (Slika 12). Programerski dio sa objektno orijentiranim dizajnom, modularnosti prema potrebama razvoja i sustav skripti je prisutan od originalne verzije, dok elementi koji vide korisnici je napisan od nule kako bi se zadovoljila potražnja za sve boljom grafikom. U nove dijelove su ulazili sustav fizike, sustav zvuka i sustav obrade tekstura. UE3 je predstavio i u potpunosti programibilan sustav sjena, tako što je računanje sjena preusmjerio po pikselu, a ne po vertikalama, tj. točkama poligona. Verzija je izašla u 2006, a uz pomoć tvrtke Mozilla, uspjeli su prenijeti UE3 na web. Kao veliki pokazatelj čistog koda koji je korišten za napisati Unreal Engine, govori i podatak da je suradnja Mozille i Epica kako bi prebacili UE na web trajala 4 dana!



Slika 12 Unreal Engine treća verzija

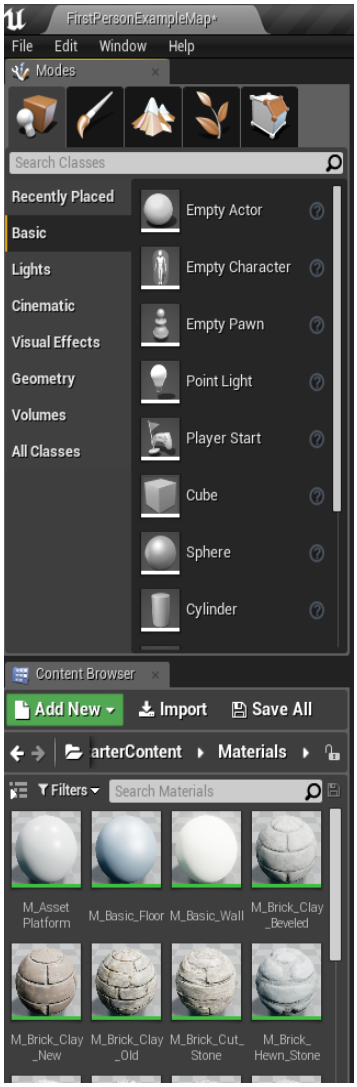
Mark Rein iz Epic Gamesa je 2005 objavio da četvrta verzija Unreal Enginea (skraćeno UE4) (više od godinu dana prije nego što su prve igre bazirane na 3. verziji uopće objavljene) u razvoju od 2003. Do 2008. sam predsjednik Epic Gamesa je programirao 4. generaciju enginea, a prvi prikazi mogućnosti nove verzije prikazani su 2012. pred zatvorenom publikom [9]. Glavna značajka koja je planirana za 4. generaciju je globalna iluminacija korištenjem konusnog traženja koje je trebalo zamijeniti predefinirano svijetljenje. Konusnog praćenja su derivacija zrakastog traženja koja dodaju debljinu zrakama kako bi ih računalo lakše obradilo. Radi nastalih komplikacija, algoritam osvjetljenja je zamijenjen jednostavnijim radi smanjenja opterećenja performansi. UE4 je uveo sustav baziran na skriptama („blueprints system“) koji je uveo rapidno izmjenjivanje i razvijanje logike u igrama bez uporabe C++. Ovim potezom razlika između dizajnera, programera i stručnjaka tehničkog dizajna se osjetno smanjila i kroz novije generacije bi mogla u potpunosti nestati. Epic Games je 2014 najavio uvođenje pretplate od 20$ na UE4, da bi početkom 2015. godine radi ogromne zainteresiranosti postao u potpunosti besplatan [9]. Poslovni model kojim se Unreal Engine uzdržava je da uzima 5% zarade od objavljenih proizvoda koji zarade više od 3000$ po kvartalu. Otvaranjem njihove online trgovine za igre, Unreal je ponudio i mjesto korisnicima da objave svoje radove, ukomponirajući 5% u 12% troškova što uzimaju za objavu igre u njihovoj trgovini. Uz trgovinu

igrama, Epic Games je napravio i trgovinu u kojoj autori mogu prodavati modele, teksture, zvukove, specijalne efekte i modele ponašanja po pristupačnim cijenama.

Sučelje Unreal Enginea

Prilikom otvaranja Unreal Enginea ponudi nam se odabir stvaranja novog projekta ili odabira postojećeg. Ovdje je izraz projekt upotrjebljen kako bi se definirala igra ili grupacija modela, dok unutar projekta možemo stvarati odvojene razine. UE4 nas pita kakav projekt radimo (radi li se o perspektivi iz 3. lica, bočnom pogledu i mnogim drugim) i ovisno o našem odabiru postavlja nam jednostavni primjer u projekt. Sam editor izgleda prepunjen informacijama, gdje u sredini vidimo trodimenzionalni prostor, a po rubovima ekrana vidimo razne postavke i mogućnosti.

Počevši od lijevog gornjeg ugla, vidimo klasičnu traku sa mogućnostima upravljanja datotekom, uređivačkim naredbama, kontrolom prikaza na ekranu i kraticom pomoći. Upravljanje datotekom se sastoji od naredbi za stvaranjem novih, otvaranjem postojećih razina i projekata, uvoza elemenata iz drugih programa i spremanja postojećeg projekta. Uređivačke naredbe su standardne i uključuju kopiranje i rezanje objekata i opcije koraka unazad i unaprijed. Na dnu izbornika možemo i vidjeti opcije za uređenje postavki editora i samog projekta. Izbornik kontrole prikaza ima opcije raznih statističkih prikaza kao i zasebnih uređivača koji nisu prikazani na zadanom prikazu. Osim navedenog, pruža nam i fokuse na naše definirane interesne točke radi bržeg kretanja preko velikih površina. Pomoć se sastoji od hiperveze prema detaljno razrađenoj dokumentaciji, forumima i mnogim drugim korisnim izvorima informacija. Ispod trake sa izbornicima, nalazi se traka sa radnim okruženjima (Slika 13). Prvo radno okruženje je posvećeno dodavanju objekata u prostor i možemo vidjeti dvije padajuće liste, jednu sa kategorijama elemenata, a drugu sa elementima iz pojedine kategorije. Uz kategoriju osnovnih elemenata kao što su geometrijska tijela i pozicioniranje likova u prostor, imamo kategorije sa svjetlosnim unosima, postavljanje više kamera za snimanje



Slika 13 Unreal lijeva navigacija

više uglova, kategoriju sa vizualnim efektima kao što je magla i razne varijacije odsjaja i kategorije geometrije i raznih zvukova.

Svijetla su podijeljena u direktna svijetla koja su namijenjena zatvorenim prostorima, ambijentalna svijetla za slabiju raspršenu svijetlost i globalna svijetla poput sunca. Zvukovi su podijeljeni ovisno o njihovoj namjeni, kao što su zvukovi raznih okidača, ambijentalni zvukovi prirode, tehnički zvukovi i zvukovi posljedica akcija likova. Podjela svijetla i zvukova nam omogućava veću kontrolu nad prostorom i brže snalaženje prostorom koji poprima veće količine kompleksnosti dodavanjem sadržaja.

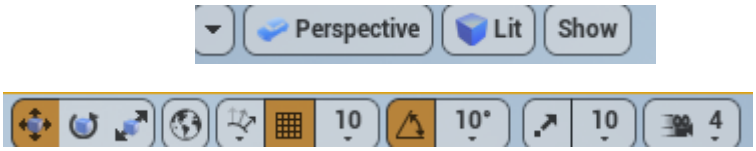
Na dnu ekrana možemo vidjeti preglednik naših mapa, sadržaja unutar mapa, sa opcijama da uvezemo novi sadržaj u projekt. Preglednik sadržaja nam nudi brzi pregled postojeći datoteka i modela i dvoklikom na stavku otvara se posebni prozor sa prilagođenim mogućnostima izmjene (u slučaju tekstura otvara se preglednik za slike, za modele se otvara izolirani preglednik sa 3d prostorom i za nacrte i materijale se otvara uređivač skripti). Kretanje kroz preglednik je pojednostavljeno i nedostaju funkcionalnosti navigacije na koje je korisnik naviknut u operativnom sustavu. Preglednik nudi filtriranje rezultata radi lakšeg snalaženja i postavke pregleda ovisno o korisničkim preferencijama.

Iznad 3d prostora možemo vidjeti traku funkcija koju predvodi gumb za spremanje projekta i kontrolu izvora datoteka (Slika 14). Niz se nastavlja sa gumbima za otvaranje zasebnog preglednika datoteka i veza do službene trgovine. U traci se nalaze i gumbi za kontrolu postavki projekta i kontrolu nacrta, gumb za izradu videa te opcije za izgradnju segmenata projekta, testiranje razine i gumb za kontrolu pokretanja na različitim platformama.



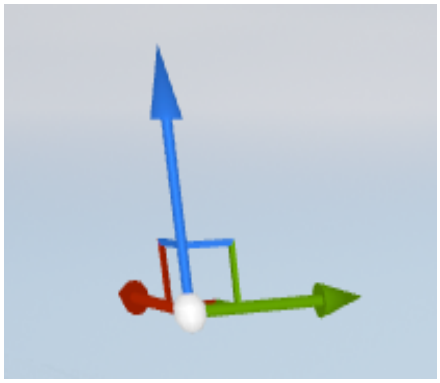
Slika 14 Unreal, traka funkcija

Uz prikaz 3D prostora u sredini ekrana nalaze se i 2 kategorije gumbi. Prva kategorija postavki nam služi za kontrolu prikaza 3d prostora i mjerenje performansi našeg prostora (Slika 15). Prikazi izoliranih komponenti pomažu korisniku da detektira razloge pada performansi i donese zaključke kako ih popraviti. Korisnik može i sakriti pojedine grupacije elemenata kako bi se mogao posvetiti na detalje i kontrolirati kameru sa navigacijskim funkcijama. Druga grupacija naredbi je pozicioniranja na desnoj strani i služi nam za pomicanje, rotaciju i skaliranje statičnih objekata. Možemo definirati i minimalne hodove pomaka, kutova i skaliranja kao i brzinu preleta kamere kroz prostor.

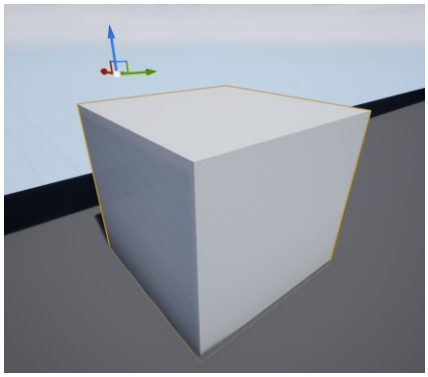


Slika 15 Unreal, naredbe unutar 3D prostora

Klikom na model pojavljuje nam se unutar 3D prostora pivot točka. Pivot točka nam pokazuje osi podijeljene u boji radi lakšeg snalaženja i omogućava nam kontrolu i transformacije objekta u prostoru (Slika 16). Objekt ima određenu poziciju pivot točke ovisno je li uvezen iz drugog programa ili je nastao unutar UE4 sučelja. Točka se može pomicati neovisno od njezinog objekta i može se pridružiti objektu za lakšu kontrolu (Slika 17). Klikom na osi možemo upravljati modelom, dok klikom na spojeve između dvije osi, može omogućiti pomak ili skaliranje objekta na razini odabranih osi. Treća os će biti zaključana i pomak neće biti primijenjen na nju.

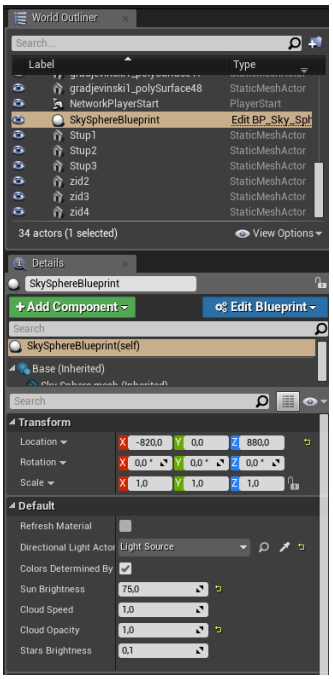


Slika 16 Pivot točka



Slika 17 Pivot točka nije vezana za objekt

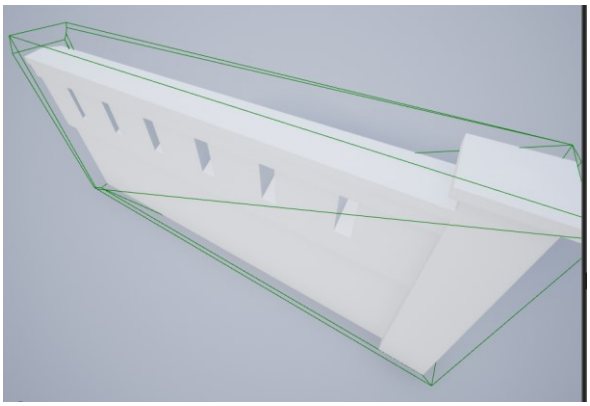
Sadržaj na desnoj strani ekrana nam pruža uvid u elemente koji se trenutno nalaze unutar prostora, sa mogućnosti da na njih kliknemo i pozicioniramo kameru u njihovu blizinu, fokusiranu na objekt. Ispod liste, nalaze se detalji i postavke koji variraju ovisno o odabranom objektu (slika 18). Vrh liste detalja nam pokazuje koordinate u prostoru, kutove rotacije i faktore skaliranja raspoređenih po osima. Možemo objektu mijenjati ponašanje u prostoru između statičnog, stacioniranog i pomičnog, ovisno o naravi objekta sa kojim radimo. Podno transformacijskih faktora, nalazi se informacija koji objekt koristimo i mogućnost da zadržimo faktore i da promijenimo sami objekt na ovoj lokaciji. Bitno je za napomenuti da se koordinate odnose na poziciju pivot točke, koja nije uvijek u središtu objekta. Slični izbornik nam nudi i promjenu materijala površine objekta. Na dnu padajuće liste nalaze se postavke fizike, postavke ponašanja, osvjetljenja i mnoge druge.



Slika 18 Unreal desna navigacija

Collision

Prilikom stvaranja ili uvoza modela u Unreal Engine, taj objekt će se ponašati kao hologram. Model će biti vidljiv, ali također ćemo moći proći kroz njega. Kako se to ne bi događalo sa fizičkim objektima koji predstavljaju prepreke (zidovi, stupovi, veće kamenje, vozila, ograde i sl.) moramo stvoriti *collision* objekta. Collision ili kontrola sudara je nevidljiva mreža točaka koja onemogućuje liku da prolazi kroz objekte. Unreal ne prepoznaje svojstva objekata i zato mu moramo reći kroz koje objekte korisnik može proći a kroz koje ne. Ljuska objekta kroz koju lik ne može proći se generira ovisno o složenosti objekta. Unreal može automatski stvoriti ljusku, ali takva ljuska neće uvijek biti savršena, tj. moguće su nedopustive pogreške koje će rezultirati time da lik propadne kroz tlo ili nevidljivim barijerama (Slika 19). Ljuska se može stvoriti i u drugim programima i uvesti zajedno sa modelom, što povećava kontrolu nam objektima i omogućuje korisniku da stvori vlastiti dizajn ljuske.



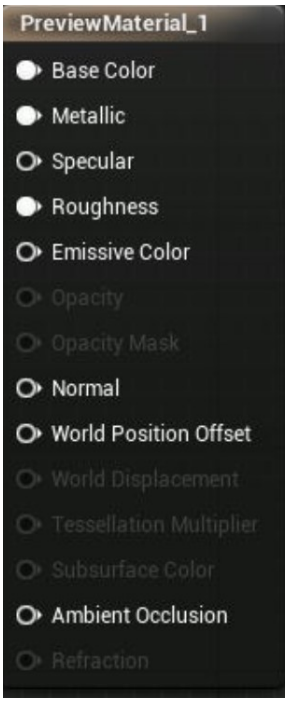
Slika 19 Primjer Collisiona

Materijali

Nakon što smo uveli modele u prostor i postavili ih na pozicije, vrijeme je da primijenimo materijale. U suštini, materijali su elementi koji opisuju kako će element izgledati u prostoru. Korisnik dodjeljuje materijalima slike i teksture modela i matematičke izraze kako bi definirao odbijanje svijetlosti jednom kada se materijal primjeni na model. U suštini, materijalom se može određivati boja modela, refleksija svijetlosti, prozirnost i brojne druge opcije.

Korisnik prilikom izrade mora razumjeti kakav materijal želi napraviti i neke njegove osnovne vizualne karakteristike. Primjer bi bio izrada betona kao materijala koji nema visoki sjaj već dosta neravnu i grubu površinu i recimo zlata koji ima visoki sjaj i glatku površinu. Materijali uz boju i refleksiju mogu pomoći sa dodavanjem raznih detalja na model, bez da dodaju količinu poligona i bez da dižu kompleksnost modela. Neki od osnovnih primjera bi bili dodavanje fuga na pločice, pukotina i nepravilnosti na materijale kao što su asfalt i drvo. Osim grafičkih mogućnosti, materijali unutar *Unreal enginea* imaju veliki arsenal korisnih funkcija koje pomažu oblikovati materijal za naše potrebe. Materijali izrađeni u ovom radu imaju mnoge takve funkcije kako bi kontrolirali veličinu i poziciju tekstura koji se primjenjuju na model.

Izrada materijala započinje otvaranjem posebnog *editora* unutar *Unreala.* Materijali se ne programiraju na klasični način, već slaganjem mreže vizualnih čvornih točaka koje sadrže naredbe („*Material expressions*“). Svaka čvorna točka sadrži ciljani dio HLSL koda („*High-Level Shading Language*“) [7]. U konačnici slažemo materijale tako što slažemo čvorne točke HLSL koda pomoću vizualnog skriptiranja. Učitavanjem tekstura u *editor* materijala, *Unreal* prepoznaje kanale boja i sastavlja vektor sa vrijednostima za svaki piksel. Vrijednosti su izraženi kao decimalni brojevi sa vrijednostima od 0 do 1.0 umjesto od 0 do 255 [7]. U slučaju prekoračenja određenih vrijednosti možemo postići zanimljive efekte i pojačati željene rezultate. Većinu vremena kada izrađujemo materijale, zapravo korigiramo brojevne vrijednosti piksela. U kontekstu materijala, teksture su slike koje pružaju podatke o pikselima. Ovi podaci mogu biti u nekoliko tipova, kao što su boja, količina sjaja, prozirnost i mnoge druge informacije. U starijim metodama, smatralo se da je teksturiranje metoda kako nanijeti boju na model, dok se danas na to gleda kao komponentu materijala. Svaki materijal započinje sa osnovnom zadanom čvornom točkom na koju nadovezujemo i usmjeravamo ostale čvorne točke sa našim teksturama i matematičkim izrazima (Slika 20).



Slika 20 Osnovna čvorna točka materijala

Unosom tekstura i parametara u baznu čvornu točku možete definirati gotovo pa svaku vrstu fizičke površine. Prilikom izrade materijala, treba predvidjeti koje od ulaza ćemo koristiti, a koje nećemo. Kada recimo stvaramo materijal da prikažemo svijetlost, nemamo potrebe za dodavanjem tekstura koje definiraju koliko je površina gruba ili metalik, također ako ih i dodamo, *Unreal* će detektirati da su nepotrebne i neće ih primijeniti. Kako bi lakše odredili što nam od ulaznih podataka treba, imamo tri glavna kontrolna svojstva:

Blend Mode – Kontrolira kako će se materijal stopiti sa pikselima u pozadini

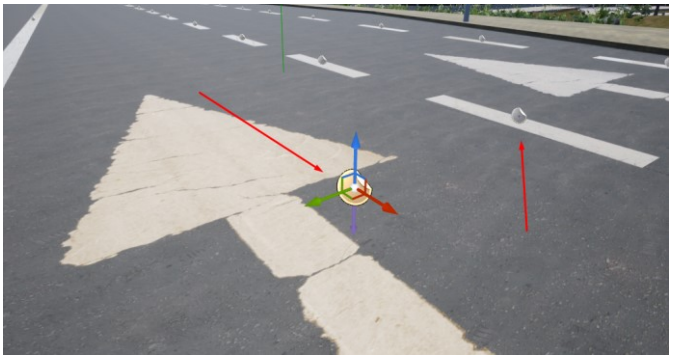
Shading Model – Definira kalkulacije svijetla na površini materijala

Material Domain – Kontrolira kako će se materijal koristiti, tj. hoće li biti dio površine, funkcija svijetla ili poslije procesni materijal

Prilikom korigiranja *Blend* mogućnosti, možemo odrediti koliko proziran nam materijal treba biti, hoćemo li mu dodijeliti masku ili zbrojiti (ili množiti) vrijednosti tekstura. Zbroj tekstura može stvoriti zanimljive efekte kao što su para, vatra ili hologrami, zbog toga što se u ovom slučaju crna boja prikazuje kao razina transparencije.

Mogućnosti *Shading* modela nam nudi nekoliko opcija kako će se svjetlo ponašati sa našim materijalom. Osim što možemo odrediti materijalu da neće biti osvijetljen, imamo i nekoliko opcija koje koristimo kako bi prikazali materijale kože i prozirne prekrivače materijala kao što je lak na autima.

Prilikom izrade novog materijala, domena materijala je postavljena na opciju fizičke površine. Pod nazivom fizičke površine smatraju se sve površine metala, plastike, kože, drva i drugih. Među opcijama možemo naći i svojstvo za materijale svijetla, koji može sadržavati čestice po zraku i drugačije utjecati na prikaz slike, kao i opcija da pretvorimo naš materijal u naljepnicu. Naljepnica ne prekriva cijelu površinu (ili model), već odabranu teksturu zalijepi na površinu sa odvojenom pivot točkom, što omogućuje da je pomičemo neovisno od objekta (Slika 21). Primjena naljepnica u sceni praktičnog rada su prometne linije po cesti, kao i šahte.



Slika 21 Naljepnica prometne signalizacije

Kako rotiramo kontrolna svojstva, *Unreal* će nam zaključati svojstva koja nam ne trebaju i time onemogućiti da si stvaramo dodatni posao.

Vrste tekstura

Prolazeći kroz ulaze osnovne čvorne točke, na vrhu primjećujemo *Base Color*. Bazna boja uzima RGB vektor i svaku od 3 vrijednosti konvertira iz vrijednosti od 0-255 u skalu od 0-1. U slučaju da se radi o fotografiji stvarnog objekta, onda se radi o boji koja se dobije polarizirajućim filterom koji uklanja reflektirajuća svojstva nemetala kada dođe do poravnavanja. U baznu boju najčešće se nadovezuju čvorne točke, točke teksture ili čvorna točka vektora gdje možemo pomoću vrijednosti odrediti o kojoj se boji radi. U pojedinim situacija imamo određenu fotografiju nekog uzorka (recimo pločica) i želimo joj promijeniti boju. U takvim slučajevima trebamo zbrojiti vrijednosti fotografije sa vektorom boje kako bi ostvarili željeni rezultat.

Metallic svojstvo kontrolira koliko će površina izgledati metalno. Unatoč jednostavnoj definiciji gdje čisti materijali imaju *metallic* vrijednost ili 0 ili 1, gdje 0 označava ne metale a 1 metale, postoje mnogi materijali koji zahtijevaju neku vrijednost u ovom intervalu [7]. Čisti materijali su sve rjeđi u industrijskom standardu jer razvijanjem tehnologije, nastajala je kreativna atmosfera u kojoj se mogu prikazati složeniji hibridni materijali koji nam omogućavaju prikazati nepravilnosti materijala kao što su prašina, korozija i pogreške u proizvodnji. *Metallic* unatoč jednostavnosti i mogućnostima nije standard u drugim alternativnim programima. Gotovo identični rezultati se mogu postići i sa *Specular* opcijom

koja ima veću prisutnost i zbog toga tvrtke koje su posvećene izradi teksturama i izradama materijala nude različite pakete za metale ovisno u kakvom okruženju radite.

Specular ili refleksija je svojstvo materijala koje određuje dozu odbijanja svijetlosti od površine. Refleksija se u ovom rasponu kreće od kompletnog upijanja svijetlosti (vrijednost 0) do efekta zrcala. Jedna od ključnih razlika između *metallic* i *specular* materijala je zapravo u temeljnoj boji. Metallic okruženje koristi svijetlije boje kako bi postigli rezultat, dok specular tamnije, gotovo crne. Unatoč tome što oba ulaza rade sličnu stvar i u mnogim slučajevima dovoljno im je proslijediti skalarnu vrijednost, većina tvrtki za izradu tekstura posvećuju cijelu teksturu kako bi povećali kontrolu finalnog materijala. U praktičnom djelu ovog rada, u većini materijala korištene su cijele teksture za što vjerodostojni prikaz. Na slici 22 možemo vidjeti 3 materijala koja imaju različite specular vrijednosti.

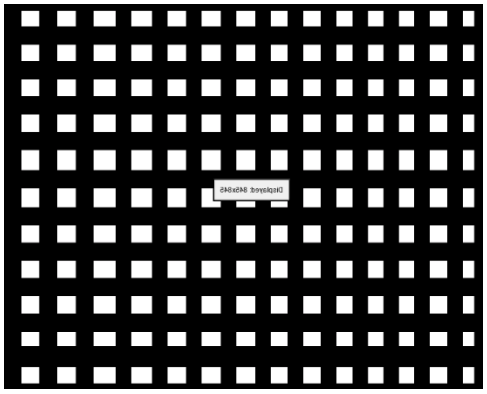


Slika 22 Razlika u intenzitetu specular teksturi

Materijali u stvarnosti nisu uvijek idealno ravni i za to nam se pobrinulo sljedeće svojstvo na našoj listi, *Roughness.* Glatkoća definira koliko je površina materijala glatka i kako će se svijetlost raspršiti kada se odbije od nje. Postavljanjem glatkoće na 0, dobiti ćemo savršeno gladak materijal koji će zrcaliti snopove svijetlosti, dok povećanjem vrijednosti glatkoće ćemo dobiti grublje površine koje neće odbijati svijetlost. Prilikom metala, ovo svojstvo će stvoriti zamućeni sjaj ovisno kako vrijednost raste. Kako nam u većini slučajeva nije idealno da nam je glatkoća ravnopravno raspoređena, ovdje je idealno usmjeriti cijelu teksturu koja će dodati fizičku raznolikost u materijal. Dobar primjer nepravilnosti bi bio brušenost metala ili masnoće na površinama.

Emissive Color je svojstvo zračenja svijetlosti. Ovo svojstvo sjaja se ispunjava kombinacijom boje i maskom koja govori koji elementi na površini će sjajiti. Boja je određena vektorom, dok se maska sastoji od bijelih i crnih površina u kojima bijele površine označavaju mjesta gdje će svijetlost zračiti, a crne govore materijalu da će taj dio biti nepropustan. Ovakav efekt je super za prikazivati razne svjetlosne reklame nepravilnog oblika bez da se povećava kompleksnost poligona na modelu. Ovo naravno, nije jedina primjena maski koje se često primjenjuju u svrhu prozirnosti materijala. Sama prozirnost (*Opacity*) i maska prozirnosti

(*Opacity mask*) su svojstva materijala da vidimo kroz njega. Prozirnost nam može pomoći kako bi smanjili vidljivost pojedinih objekata, kao što su magla ili staklo, dok maska prozirnosti nudi veću kontrolu zbog mogućnosti da kombinacijom crnih i bijelih polja stvorimo razne otvore na materijalu bez pretjeranog kompliciranja sa brojem poligona modela. U praktičnom djelu rada, primjeri maske prozirnosti (Slika 23) su korištene kako bi se napravio metalno stepenište kroz koji se može vidjeti, kao i sjenila na prozorima Građevinskog fakulteta (Slika 24. Unatoč prozirnosti, stakla na prozorima se i dalje rade od reflektirajućeg, glatkog materijala na svim mjestima gdje je to moguće, jer u protivnom bi autori trebali uložiti mnogo više vremena u dodatnu izradu interijera, koji će tratiti dragocjene resurse. Također bitno je napomenuti da područja materijala koja ostanu vidljiva nakon uporabe maske će i dalje reflektirati svijetlost, primijeniti definiranu glatkoću i zadanu boju.

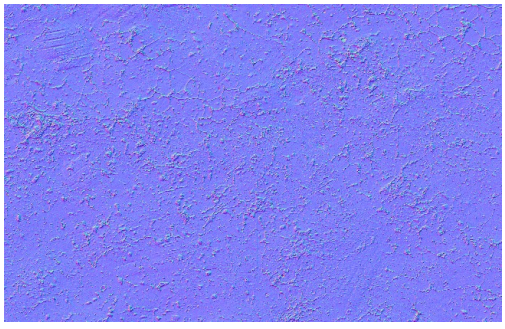


Slika 23 Primjer maske



Slika 24 Primjer maske u primjeni

Normal svojstvo materijala je tekstura koja služi dodavanju broja fizičkih detalja na površinu bez da dodaje broj poligona na model (Slika 25). Normal zapravo određuje kut gledanja na površinu svakog piksela površine. Mogućnosti kuta gledanja na površinu sežu od jednostavnih pukotina do kompleksnih uzoraka kao što su kora drveća ili istrošenost oštrica metala.



Slika 25 primjer normal teksture

Normal teksture nastaju tako da se napravi model visokog broja poligona i tekstura izvuče iz njega, te naknadno primjeni na poligon nižeg broja poligona. Bitno je napomenuti kako prilikom učitavanja normal teksture treba invertirati vrijednost kanala zelene boje. Svaki kanal

boje (RGB) u normal teksturi definira pomak, te je ručno potrebno korigirati zeleni kanal kako bi se stvorile izbočine na modelu (Slika 26). U slučaju da korisnik zaboravi na ovaj detalj, model će nakon primjene materijala ostati „ravan“ i bez iskorištenog potencijala.



Slika 26 Razlika koju normal tekstura stvara na modelima

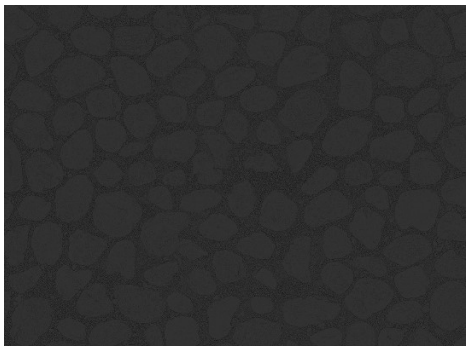
World Position Offset omogućuje materijalu da pomiče točke modela u virtualnom prostoru i njegovu interakciju. Svojstvo omogućuje pomake, promjene oblika, rotacije i mnoge druge efekte koji se koriste prilikom animacije. Primjer takvog pomaka bi bilo njihanje trave i lišća, bez da korisnik mora kodirati kompleksne animacije vjetra. Uz pomake, imamo i *World Displacement* te *Tessellation Multiplier* razlika između pomaka između navedenih je u točkama koje odabiru.

Kako bi se stvorili kompleksniji materijali koji imaju prekrivene boje ispod bazne boje, koristimo *Subsurface Color.* Navedeno svojstvo nam omogućava da se boja izmjeni ovisno o količini svjetlosti koja prođe kroz površinu. Primjer bi bila crvena boja ispod obraza na licu, koja bi pod dovoljnom količinom svjetlosti prikazala crvenilo u licu. Ovo svojstvo je primjenjivije u animacijama i kompleksnijim statičnim modelima nego u arhitekturi.

Ambient Occlusion ili skraćeno *AO* je tekstura koja govori materijalu gdje se nalaze sjene koje stvara sam model. Sa povećanjem detalja na materijalima, nastaju i udubljenja i rezovi na modelu koji bacaju sjenu na ostatak modela, te se te informacije pohranjuju u AO teksturu. Nedostatak ove teksture relativno neće naštetiti finalnom materijalu, ali ako je dostupan, preporučljivo ga je pridodati. Refrakcija bi bio zadnji ulaz na zadanu čvornu točku, koji definira index refrakcije materijala površine. Najčešće korišteno svojstvo za izradu stakla, vode i drugih sličnih materijala koji djelomično upijaju svijetlost.

Teksture koje koristimo u materijalu koje ne koristimo kako bi izrazili boju je potrebno konvertirati u sRGB standard kako bi se uštedjele dragocjene performanse. sRGB ima najmanji pojas pokrivenosti boja i nalazi primjenu na web stranicama [12]. U izradi materijala, teksture koje nemaju utjecaj na boju objekta su najčešće u tonovima sive, sa ponavljajućim uzorkom. Primjeri takvih tekstura određuju glatkoću površine, refleksiju i maske. Prilikom izrade složenih

scena, ne možemo si dopustiti trošenje prostora i drugih resursa. U praktičnom radu koriste se teksture sivih tonova kako bi naznačili neravnine na materijalima i dodali modelima više detalja. Vidljiva razlika u teksturama sRGB formata i klasičnog RGB formata u ovoj primjeni ne postoji (Slika 27 i 28).

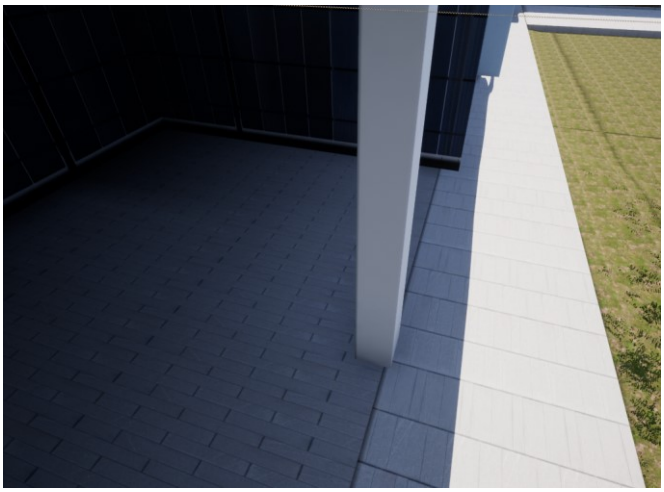


Slika 27 Specular tekstura u sRGB formatu



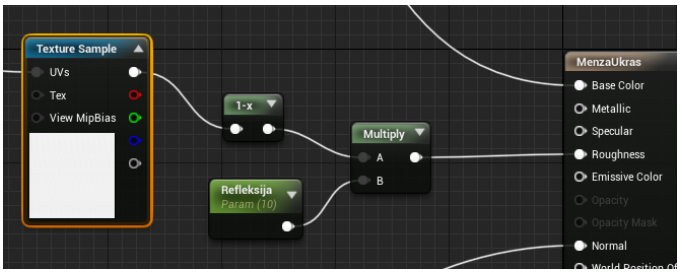
Slika 28 AO tekstura u sRGB formatu

Navedeni ulazi zadane ključne točke (popunjeni ovisno o zahtjevima materijala kojeg izrađujemo) daju dobar uvid u izgled materijala i u konačnici površine na koji će materijal biti primijenjen. Problem se pojavljuje u tome što nemamo kontrolu nad veličinom materijala, rotacijom, intenzitetom pojedinih tekstura i mnogih drugih parametara. Učestalo se događa da naš model nije istih dimenzija kao i teksture unutar materijala. Problem će rezultirati rastezanjem materijala kako bi prekrio cijelu površinu ili u slučaju manje površine od predviđene materijalom, dobiti ćemo nakupljanje i gubljenje detalja. Modeli mogu biti i nepravilnog oblika što će također rezultirati nerealnom primjenom materijala. Metoda koja je primijenjena u praktičnom djelu rada je dodavanje čvorne točke koja kontrolira i množi teksture ovisno o osi prostora i stvara efekt pločica. Stvaranjem efekta pločica („*Tiling textures*“) možemo povećati površinu koju će materijal prekriti bez da gubimo na kvaliteti pojedine teksture. Nisu sve teksture pogodne za korištenje ove metode, ali je zato nužna za teksture zidova, ploča, i drugih ponavljajućih uzoraka. U slučaju da imamo model koji je nepravilnog oblika (recimo oblik slova „L“) moramo uzeti u obzir da će broj ponavljanja teksture u osi biti primijenjen na cijelom modelu. U praksi to bi značilo da pokušavate staviti isti broj pločica na razmaku od 2m kao i na razmaku od 10m. Jedno od rješenja ovog problema bi bilo rastavljanje ovog modela na dva odvojena koji bi dobili ispunu pravilne veličine. Na slici vidimo detalj južnog prilaza Filozofskog fakulteta koji se proteže cijelom dužinom zgrade. Lijevo od prilaza, u sjeni vidimo polu zatvorenu, popločanu površinu koja je grupirana sa prilazom u jednu cjelinu. Kako smo prilagodili materijal dužini prilaza, sve ploče koje smo upotrebili za izradu prilaza, UE4 je iskoristio na površini koja je 3 puta manja (Slika 29).



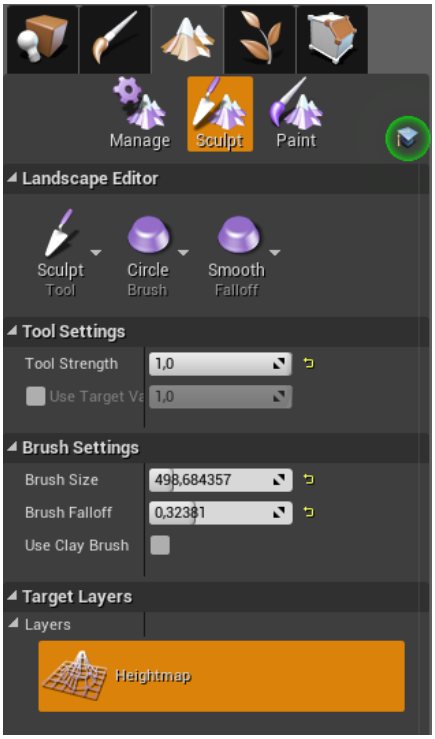
Slika 29 skaliranje tekstura na elementima različite veličine

Srodan problem se pojavljuje kada želimo materijal primijenjen na površini zarotirati za određeni kut. Materijal će se rotirati zajedno sa modelom, a rotacijom tekstura nećemo dobiti rotirani materijal. Kako bi riješili problem, dodajemo čvornu točku koja nam pomaže da rotiramo teksture unutar *editora.* Na čvornu točku moramo nadovezati skalar kojemu ćemo definirati vrijednost od 0 do 1 ovisno o tome koliku rotaciju želimo postići (0,25 predstavlja 90° stupnjeva). Određivanje kuta neće biti dovoljno, već trebamo informaciju o rotaciji prenijeti i svim teksturama koje koristimo u materijalu. U čvornu točku rotacije trebalo bi dopeljati informacije o broju ponavljanja tekstura kako ne bi došlo do pogreške. Uz svega 3 čvorne točke (ako računamo skalar rotacije) uspostavili smo kontrolu nad našim materijalom. Prilikom izrade materijala, možemo se naći u situaciji da nam pojedina tekstura ne pruža željeni intenzitet. Primjer bi bio nedovoljna količina emitiranja sjaja na površini. Umjesto da korigiramo teksturu, trebamo dodati čvornu točku koja množi ulaze, od kojih će jedan biti tekstura, a drugi će biti skalar koji će množiti RGB kanale teksture. Kao primjer iz rada je množenje teksture glatkoće, kako bi joj pojačali efekt. Konkretno u ovom primjeru je materijal kamenčića u vazama kojemu je intenzitet povećan 10 puta (Slika 30). Prilikom dodavanja skalarnih parametara koji utječu na materijal putem rotacije ili množenjem pojedine teksture bitno im je dodijeliti odgovarajuća imena, jer će se ta imena prikazati kao polja za unos veličine kada budemo radili daljnje istance materijala. Korina čvorna točka u izradi materijala je i *OneMinus* koji nam invertira vrijednosti. Maske prozirnosti su binarne (0 za crno i 1 za bijelo) i primjena ove čvorne točke će nam ubrzati rad i dodati zanimljive efekte na materijal.



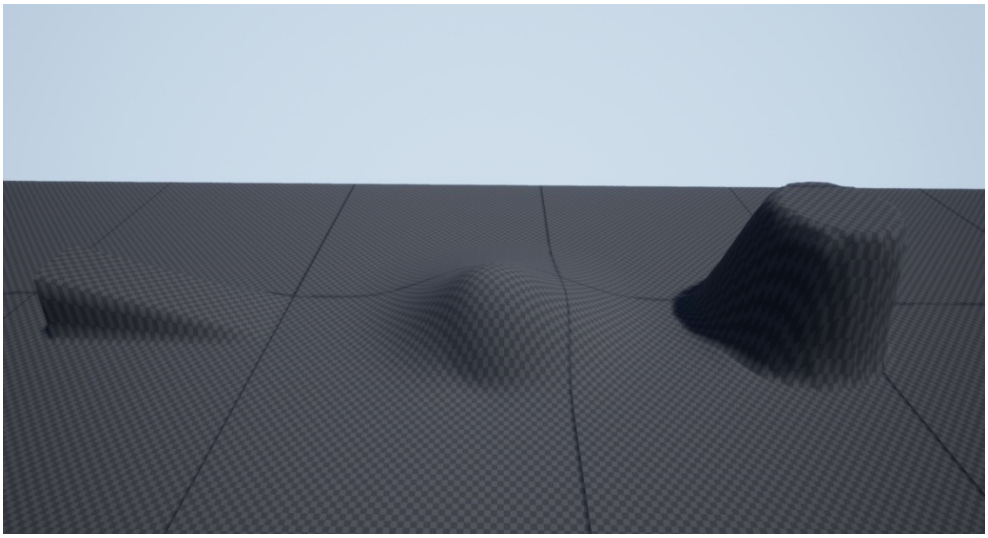
Slika 30 primjer množenja teksture sa skalarom

Izrada Terena



Slika 31 Kartica oblikovanja terena

U zadnjih desetak godina i rastom računalnih performansi, rasla je i potreba za izradom većih, otvorenijih i realističnijih područja. Jedan od logičnih koraka s obzirom na računalne igre sa početka stoljeća je bio i ostaviti zatvorene i klaustrofobične sustave hodnika iza nas i okrenuti se velikim otvorenim prostorima kakve danas poznajemo u računalnim igrama. Jedna od mogućnosti *Unreal Enginea* je generiranje velikog ravnog terena koji nudi površinu za pozicioniranje naših modela. Kako bi generirali novi teren ili krajolik (eng. „*Lanscape*“) potrebno je u glavnom prozoru odabrati kraticu sa nacrtanom planinom i otvoriti će nam se niz opcija i alata (slika 31). Među opcijama možemo uočiti opciju pridruživanja materijala terenu kojeg planiramo napraviti, opcije pomaka, rotacije i veličine, te broj kvadrata. Broj kvadrata ima predefinirane vrijednosti ovisno koliko nam terena treba, ali moramo uzeti u obzir da će veći teren i uzimati više performansi. Za praktični dio ovog rada korišten je teren od 63x63 kvadrata. Nije potrebno unaprijed dodijeliti materijal terenu, jer teren prilikom generiranja će biti ravna ploča koju moramo alatima za oblikovanje prilagoditi našim potrebama. Alati za oblikovanje nam nude mogućnosti izdizanja i spuštanja dijelova terena (ovisno o veličini koju nam alat pokriva), zaglađivanje terena kako bi sakrili oštre padove gdje nisu potrebni, zaravnavanje površine ovisno treba li izdizati ili spuštati teren do ciljane razine, izrada rampi i nekoliko vrsta erozije terena (Slika 32). Alatima možemo korigirati veličinu djelovanja, intenzitet, fokus i oblik područja djelovanja ovisno o našim potrebama. Mogućnosti kontroliranja intenziteta i veličine alata oblikovanja možemo lokalizirati željeni učinak i primijeniti manje korake oblikovanja kako bi dobili što precizniji rezultat.



Slika 32 Primjeri manipulacije terenom

Izrada materijala koji bi jednoznačno definirao teren na njegovom cijelom području ima drugačiju metodu izrade. Prilikom izrade materijala moramo staviti kontrolnu točku koja će nam definirati visinu našeg terena. Visinu možemo podijeliti na nekoliko kontrolnih točaka koji će služiti kao pragovi promjene teksture. Ovim potezom ćemo imati jedinstvenu visinu terena u kojoj će se primjenjivati određena tekstura. Ova mogućnost nam omogućava da stvorimo zelene livade za nizinska područja, stjenovite obronke i snježne planine bez da moramo trošiti resurse kako bi ručno bojali područja. Primijenjeni materijal će također automatski primijeniti efekt pločica, ali kontrolne čvorne točke će nam pomoći da korigiramo veličinu originalne teksture.

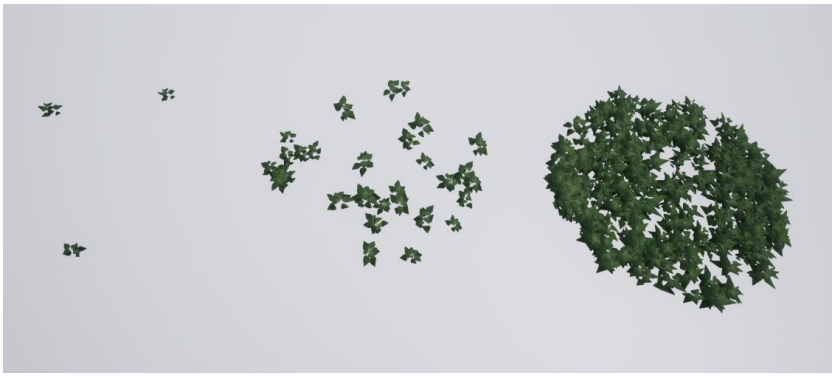
Vegetacija

Na teren možemo postavljati naše prethodno napravljene statične modele, ali oni neće biti ograničeni postojanjem terena, što znači da modeli mogu prolaziti kroz teren. Ova karakteristika nije poželjna u izradi scena osim ako se ne radi o specijalnoj primjeni. Kako nam je u interesu da se objekti vide na terenu, trebamo urediti teren tako da nema preklapanja. Za primjer možemo uzeti zgrade, koje zauzimaju veliku površinu i nećemo često imati savršen slučaj da je nadmorska visina terena ista u svim točkama terena koje okružuju zgradu. Teren koji je u skriven unutar zgrade se neće vidjeti u simulaciji, te ne moramo posvetiti vrijeme njegovom uređivanju. Dovoljno je urediti rubove, kako bi zgrada izgledala kao da ima uređeno temelje. Na slici 33 je prikazan kadar iz zgrade Odjela, gdje se može vidjeti teren koji se skriva iza objekta.



Slika 33 Teren iza objekta koji korisnik ne vidi

Sada kada imamo teren sa pripadajućim materijalom i oblika koji nam pogoduje sceni, trebamo na teren dodati vegetaciju, kamenje i udahnuti mu život. Kako bi bilo izrazito dugotrajno dodavati svaku biljku, stablo i kamen pojedinačno, pobrinuo se *Unreal* sa alatom *Foliage.* Kako bi koristili alat, potrebno je u glavnom prozoru, u gornjem lijevom uglu pored alata za oblikovanje terena odabrati karticu sa nacrtanom biljkom. *Foliage* alat nam omogućuje da odaberemo prije napravljene modele biljka, kamenja i drveća (nismo ograničeni samo logičnim odabirima, ako korisnik želi teren pun kolača ili prometnih čunjeva, to je također moguće) i on će nam nasumičnim odabirom poslagati po odabranom djelu terena. Kako bi kontrolirali alat, imamo opcije veličine djelovanja, gustoće odabira nasumičnih lokacija i gustoću brisanja u slučaju pogreške (Slika 34). Na sljedećoj slici vidimo primjenu alata s obzirom na gustoću. Vidimo nasumično odabrane pozicije za objekte. Prilikom brisanja, alat razrjeđuje objekte sa označene pozicije.



Slika 34 Primjer različite gustoće alata za vegetaciju

Kako bi preciznije odabrali podlogu po kojoj želimo naše objekte da se generiraju, možemo ograničiti djelovanje alata da radi samo prelaskom preko drugih modela, recimo zgrada

(primjer bi bilo postavljanje mahovine na rustikalne krovove ili probijanje trave na pukotinama asfalta). Ispod opcija kontrole djelovanja, imamo prostor u koji ubacujemo modele koje želimo primijeniti. Jednom kada ih pridodamo alatu, možemo ih označavati kako bi selektivno dodavali samo označene modele (ranije smo na terenu uređivali primorski krajolik pun kamenja, a potom nogometno igralište gdje ne želimo kamenje na travnjaku). U slučaju brisanja viška vegetacije sa terena, selekcija će nam pomoći da samo odabrane elemente uklonimo sa površine. Modeli imaju svoj set opcija unutar alata koji nam omogućuju da određujemo njihovu minimalnu i maksimalnu veličinu, stvaranje sjene, ograničenje performansa i mnoge druge. Primjenom minimalne i maksimalne veličine možemo dodati faktor na nasumičnu poziciju tako što će i modeli biti nasumične veličine na nasumičnoj poziciji što stvara dodatni sloj raznolikosti.

Prilikom izrade materijala za naše modele koji će popunjavati teren, napredne tehnike omogućuju i izradu „prekrivača“ koji će uvijek prekriti gornju stranu objekta bez obzira na rotaciju. Ova opcija nam u nekim slučajevima ograničava oko višestruke namjene objekta, ali sa druge strane nam pruža mogućnost automatskog prekrivanja modela drugim materijalom, kao što je snijegom i mahovinom (naravno i drugim materijalima kao što je recimo šlag). Sama ideja o namjeni materijala ne mora strogo definirati njegovu namjenu. Materijal koji predstavlja željezo, može uz svega nekoliko manjih promjena poslužiti kao zamjena za zlato, bakar, olovo i druge metale (Slika 35).



Slika 35 Primjeri sitnih razlika u materijalu

Intenzivno korištenje *Foliage* alata može ozbiljno narušiti performanse finalnog proizvoda i zato primjenjujemo nekoliko različitih metoda optimizacije. Jedna od najlakših metoda je označiti statične modele koje smo koristili u alatu i korigirati vrijednost *Cull Distance* parametra. Parametar će utjecati na udaljenost koja je potrebna da bi se model prikazao. Ovom metodom će elementi koji su izvan određene vrijednosti „nestati“ i biti će prikazani samo kada im se dovoljno približimo. Ovo je standardna metoda kako bi se sakrila količina detalja na velikim prostorima. Druga metoda je izrada različitih LOD-ova („*Level Of Detail*“). Smanjenjem razine detalja na objektima koji su udaljeniji od nas, možemo uštediti

performanse. Vrijedi napomenuti kako se razina detalja korigira ovisno o tome kako se približavamo objektu, tako da nam se neće dogoditi da imamo nisko detaljne modele u našoj neposrednoj blizini. Konkretno, na slikama ispod odlomka možemo vidjeti stablo u sceni sa razmakom od svega nekoliko metara, što je rezultiralo promjenom razine detalja u prikazu. Na slici 36 vidimo jednostavnu paletu boja i osnovne linije stabla, ali ne vidimo detalje, dok na slici 37 vidimo veću razinu detalja, i raznovrsnije boje, kao i veće količine lišća.



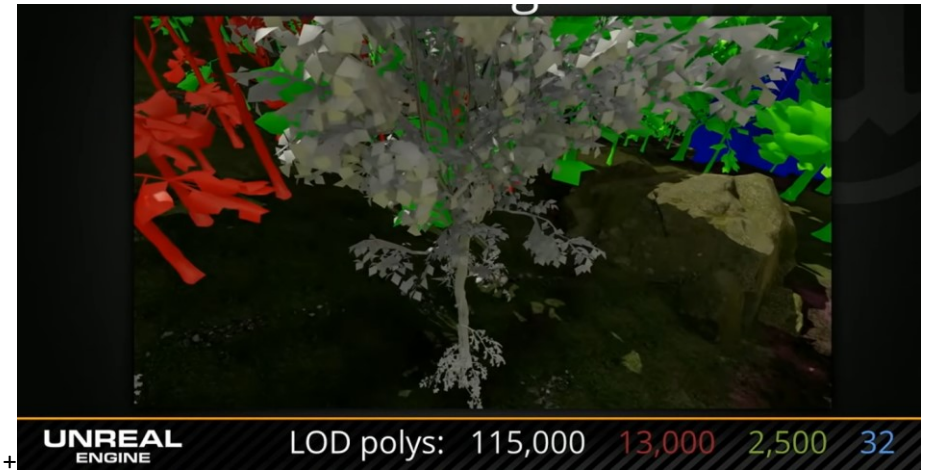
Slika 36 Stablo sa niskom razinom detalja iz daljine



Slika 37 Stablo sa visokom razinom detalja izbliza

Ovom metodom se mogu stvoriti virtualni prostori od nekoliko desetaka kilometara kvadratnih, bez da imamo dugotrajna učitavanja sadržaja. Naprednija metoda, koja također se uvelike primjenjuje u profesionalnoj industriji je i definiranje kuta kamere koju korisnik vidi u prostoru i skrivanja svih detalja koje ne ulaze u taj kut. Ova metoda aktivno prati igračev kut gledanja i instantno stvara i uklanja okoliš ovisno o njegovom smjeru gledanja. Kako se ne bi dogodilo da su objekti izmijenili poziciju između dva pogleda, pozicije objekata se pohranjuju. Zanimljivost za spomenuti je da jednom od posljednjih prikazivanja mogućnosti *Unreal Enginea* njihov interni tim je demonstrirao prostor od veličine 100 kilometara kvadratnih metodom dijeljenja udaljenosti u slojeve. Dizajnom nekoliko slojeva diktirali su razinu detalja koja se može vidjeti na objektima, na isti način kao što ljudsko oko ne vidi sve veličine slova sa iste udaljenosti. U ishodištu slojeva ili neposredno ispred kamere, nalazio se nulti sloj (naziv koji su primijenili je „*Hero layer*“) ili gdje bi najbliži objekti u smjeru kamere sadržavali maksimalnu razinu detalja (Slika 38). Na slici ispod teksta možemo vidjeti primjer gdje su stabla u šumi podijeljena bojom ovisno koliko poligona je potrebno da bi bila prikazana na ekranu.

Broj poligona stabla prema kojem je kamera fokusirana ima najviše poligona dok broj poligona opada kako udaljenost raste. Trošenje resursa na objekte koji nisu u kadru je jedan od najčešćih problema ne optimiziranih scena.



Slika 38 Kadar iz videa o predstavljanju performansi

Performanse

Optimizacija performansi mora svakom tvorcu igara biti visoko na listi prioriteta. Ako usporedimo igru koja ima kvalitetnu grafiku i visoki standard optimizacije performansi sa igrom koja ima siromašniju grafiku i nema kontrolu nad optimizacijom performansi, uočiti ćemo da naše mogućnosti računala nisu krive za loš performans i ukupni dojam igre. Osim gore navedenih optimizacija, posebno možemo kontrolirati performanse procesora i grafičke kartice.

Najčešći problem opterećenja procesora je u iscrtavanju previše modela na ekran. Kod igri, procesor određuje glavnu procesorsku jezgru koja će biti glavna za izračune i potom manje poslove dodjeljuje drugim jezgrama. Zbog toga su i Intelovi procesori bili na glasu kao bolji odabir u kupnji za igranje zbog bržih pojedinačnih jezgri. Ovo je česti problem i rješava se kombiniranjem više objekata u veće objekte [8]. Primjer bi bio spajanje više zidova u jednu cjelinu. Kako procesor obrađuje svaki objekt, također treba obraditi i njegove postavke o materijalu, svijetlosti, fizičkoj prisutnosti i mnoge druge. Složeniji materijali imaju i veću razinu složenosti. Procesor mora pripremiti naredbe za grafičku karticu za svaki objekt koji se treba prikazati, kao što su konstantne pojave (magla, dim, ponavljajući uzorci i sl.), teksture, sjene i postavke instanci objekta. Kontrolom statistike možemo vidjeti pozive za iscrtavanjem. Statističke brojeve možemo smanjiti reduciranjem broja objekata (modela i čestica), reduciranjem vidljive udaljenosti, postavljanjem novog kuta gledanja (pomicanjem objekata da smanjimo opterećenje scene), izbjegavanje korištenja dijeljenje ekrana („Split screen“), kombiniranje materijala u složenije materijale i smanjenje broja materijala[8]. Uklanjanje

bespotrebnih sjena će također olakšati posao procesoru kao i ograničenje na pojedinim izvorima svijetla da ne bacaju sjenu (ako imamo veliki izvor svijetla u sobi, nema potrebe da manja ukrasna svijetla bacaju dodatnu sjenu i uništavaju performanse). Možemo i podijeliti zadatke u skupine kako bi preciznije odredili izvor problema.

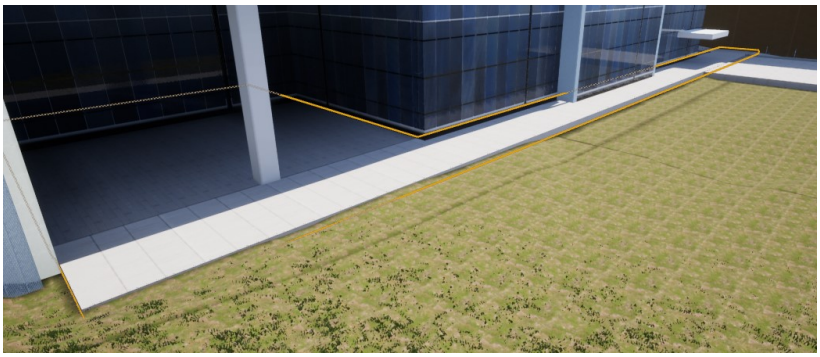
Praktični rad

U praktičnom radu modeli su modelirani sa posebnom namjenom za rad u Unreal Engineu. Maya i Unreal Engine prikazuju virtualni prostor u različitim veličinama, tako da modeli koji se naprave u Mayi moraju biti uvećani i do 1000 puta kako bi bili prikazani u stvarnoj veličini unutar Unreal Enginea. Modeli prikazani u ovom radu su od faze dizajna imali prilagođena mjerila. Nakon završenog modeliranja, bitno je razdijeliti složene modele na manje cjeline, tako da u Unreal Engineu možemo svakom od tih segmenata dodijeliti svoj materijal. Dodavanje više materijala na isti složeni model će završiti lošim performansama i nije preporučljivo. Takvi segmenti nisu vezani veličinom, tako da na slici ispod možemo vidjeti primjer sa kantom za smeće koja se sastoji od 2 segmenta (slika 39).



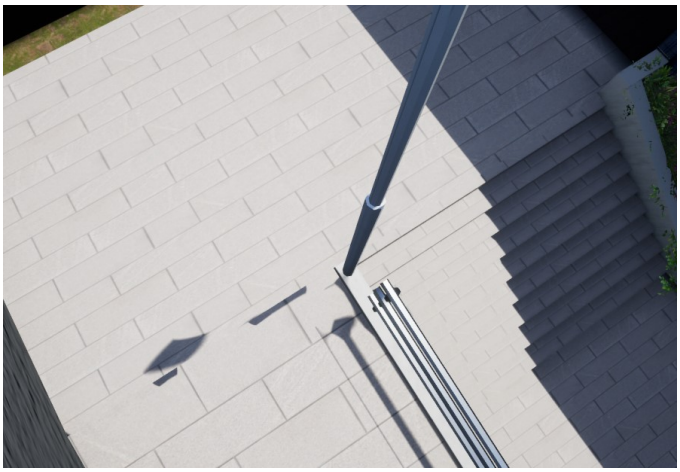
Slika 39 Primjer razdjeljivanja segmenata modela

Razlikujemo nekoliko različitih spremanja modela. Možemo spremiti jedan segment u jednu datoteku, ali na velike količine segmenata to će uzeti jako puno vremena i rezultirati stotinama datoteka koje se moraju učitati u Unreal Engine. Kako bi ubrzali spremanje segmenata i smanjili broj datoteka, možemo grupirati segmente po materijalu, približnoj veličini i poželjno da se pružaju po jednoj osi. Dobar primjer ovog slučaja je šetnica od nekoliko segmenata. Ako dođe do potrebe za skaliranjem objekata u Unreal Engineu nakon što smo završili izradu segmenata, lakše ćemo ih skalirati bez preklapanja ako se pružaju po jednoj osi. Grupacija u ovom slučaju će pohraniti i poziciju segmenata i smanjiti nam posao izgradnje u Unreal Engineu. U trećem slučaju, ako spremimo segmente bez prethodne grupacije, UE4 neće učitati njihovu poziciju iz Maye. Posljedica toga je grupa segmenata učitana sa jednim nazivom datoteke koje ćemo morati preimenovati i složiti ručno cijeli model u prostoru.



Slika 40 Primjer nepravilnog oblika modela

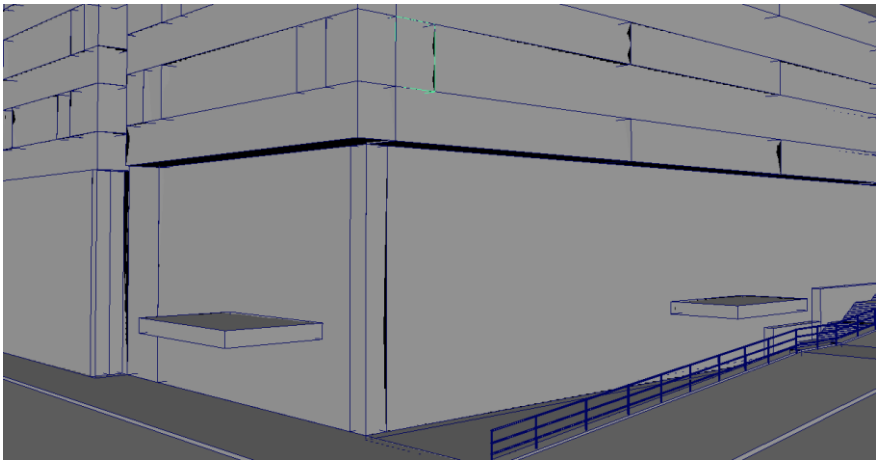
Modeli prilikom grupacije moraju biti i podjednake veličine(Slika 40). Grupacije segmenata koje nisu približne veličine, zahtijevaju izradu zasebnih materijala kako nebi došlo do rastezanja tekstura. Kako grupacije koje smo napravili u Mayi nije moguće rastaviti unutar UE, prisiljeni smo vratiti model u Mayu i prilagoditi grupacije. U tekstu se spominju grupacije po približnoj veličini jer ovisno o materijalu, imamo i različite granice odstupanja. Za zidove koji imaju jednostavne uzorke, granica tolerancije će biti puno veća nego kod ponavljajućih manjih uzoraka kao što su recimo pločice (Slika 41).



Slika 41 Loša grupacija objekata i primjena istog materijala na sve elemente grupacije

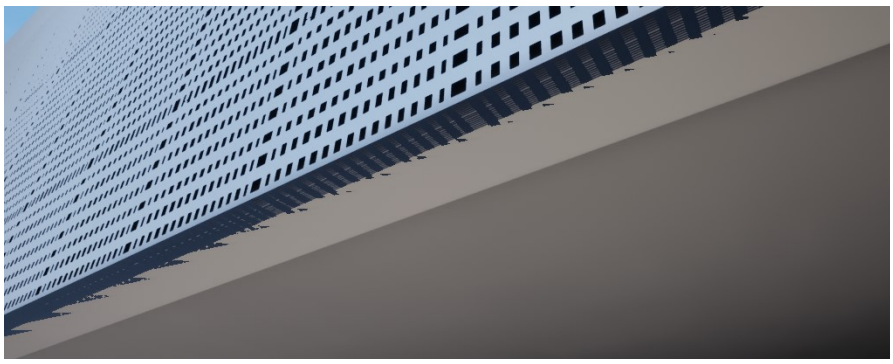
Prilikom izrade modela radimo sa geometrijskim tijelima i kod slaganja kompleksnijeg modela, dogoditi će nam se da se dvije stranice međusobno dodiruju. Vodeći računa da imamo uredne modele sa umjerenim brojem poligona, trebali bi ukloniti stranice i bridove koje korisnik neće moći vidjeti. Samo uklanjanje stranica poligona je intenzivan posao i bitno je da korisnik bude temeljit u uklanjanju višaka. Primjer sa konkretnim brojevima iz praktičnog rada se dogodio sa modelom zgrade Filozofskog fakulteta gdje je zgrada bez uklonjenih višaka imala gotovo 8000 bridova. Nakon čišćenja, taj broj je spušten ispod 3000. Razlika u broju poligona

drastično može narušiti performanse. Na slici možemo vidjeti detalj sa modela sa zgrade, gdje su prozori i zidovi rađeni od kvadara, ali kada se poredaju u složeniji oblik, samo jedna od 4 stranice je vidljiva korisniku. Ostale stranice treba ukloniti (Slika 42).



Slika 42 Složeni objekt kojemu treba ukloniti višak stranica

Moramo voditi računa da nemamo stranice koje se preklapaju. Preklapanje stranica se dogodi kada se preko istog prostora pruža više stranica (Slika 43). U fazi modeliranja, nećemo vidjeti veliki utjecaj preklapanja stranica, ali kada dodijelimo materijale modelima, doći će do konflikta. Obje teksture se nalaze u istoj točki prostora, ali obje pokušavaju odbijati svijetlost kako im materijal diktira.

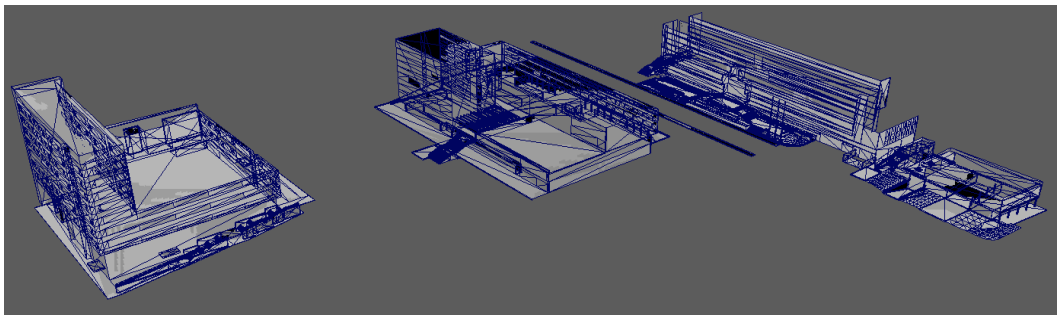


Slika 43 Primjer preklapanja tekstura

Scena najčešće sadrži veliki broj modela i pred korisnike se postavlja pitanje, je li bolje razdijeliti scenu u veće cjeline ili raditi sve u istom projektu unutar Maye? U slučaju Kampusa, posao je podijeljen u nekoliko datoteka, ovisno na kojoj zgradi se radilo. Prednost takvog rada je manja opasnost od *fatal errora* koji su u Mayi uobičajni pri velikom broju poligona. Također, lakše je ograničiti područje rada i fokus. Sa druge strane, rad u podijeljenim datotekama, može izazvati probleme sa veličinom, tj. može se dogoditi da visina istog objekta nije jednaka u 2

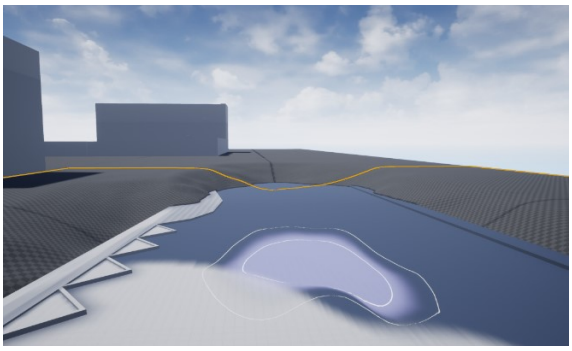
datoteke. Dok se radi o približnim veličinama koje ljudsko oko ne registrira (ako su objekti dovoljno daleko udaljeni da korisnik ne obraća pozornost ili je primijenjen drugačiji materijal da prekrije pogrešku) nema potrebe za prepravkama, već prilagodba će biti dovoljna. Kao zadnja linija korekcije bi nam trebale biti mogućnosti skaliranja objekta unutar Unreal Enginea, ali nije preporučljivo da se oslanjamo na njih.

Nakon pozicioniranja značajnih cjelina unutar UE4 (u konkretnom slučaju se radi o zgradama) scenu smo vratili u Mayu kako bi konstruirali okolinu (Slika 44). Scena ima osjetnu razliku u nadmorskoj visini i ovaj korak je bio nužan kako bi se definirala pozicija ceste i šetnica u prostoru. Prilikom modeliranja ceste, ne treba voditi računa o detaljima kao što su prometne linije, već ćemo ih naknadno napraviti kao naljepnice. Modeliranje okoline će nam uveliko pomoći u dodavanju terena u scenu. Postavljanje terena prije nego što smo modelirali okolinu može dovesti do krivih nagiba terena i u konačnici izazvati trošenje vremena na prepravke.

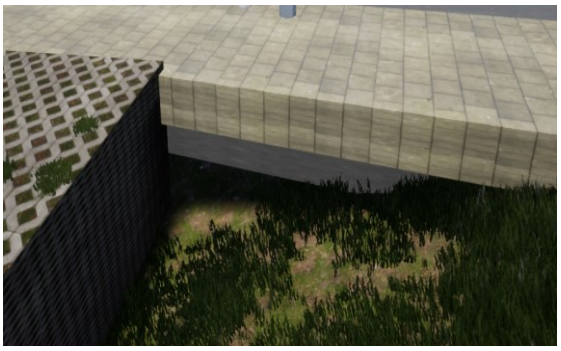


Slika 44 Scena vraćena u Mayu sa pozicijama objekata kako bi se konstruirala okolina

Dodavanje terena na scenu je drastično olakšano kada imamo pripremljene pozicije zgrada i okolinu. Moramo voditi računa da teren bude skriven na svim mjestima koje su prekrivene objektima. Posebno su nam zanimljiva mjesta gdje je pad terena presječen zidom i granice terena sa objektima na sceni. U izradi terena za Kampus, teren je pozicioniran na gornju granicu nadmorske visine te spuštan do donje granice. Ovim putem smo spuštanjem terena „otkopavali“ ceste i zgrade (slika 45). Ovim metodom imamo jasno definirane granice do kuda trebamo teren spustiti da izgleda realno. Kako bi osigurali da teren izgleda realno, poželjno je da teren minimalno presijeca objekte. Presijeci ne moraju biti veliki, ali ćemo izbjeći linearnost terena i veću razinu realizma. Da smo išli suprotnim putem i izdizali teren naišli bi na problem oko određivanja granica potrebnog. Izdizanjem terena bi došli približno do granice, ali bi učestalost pogreška bila veća, točnije pojavila bi se mjesta gdje bi objekti visili u zraku (Slika 46).

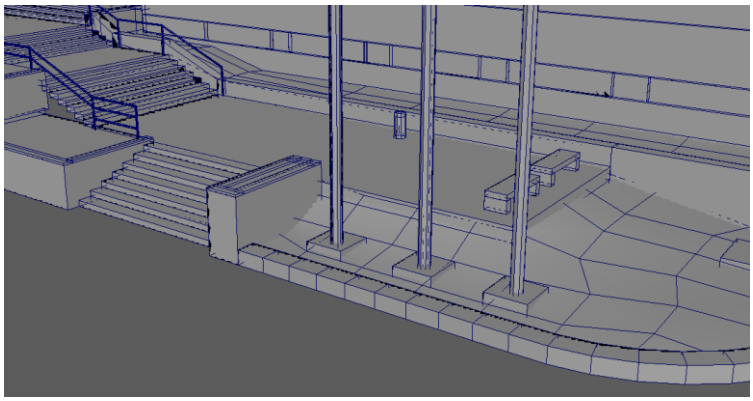


Slika 45 Iskopavanje objekata okoline



Slika 46 Objekt visi u zraku

Poseban slučaj su uređene površine koje su ograđene objektima koje smo napravili u Mayi. To su male površine koje bi napravile puno posla da se modeliraju od terena. Za takve slučajeve, lakše je napraviti u Mayi ravninu i prilagoditi je potrebama (Slika 47). Ravnina se ponaša kao objekt i može mu se dodijeliti materijal. Primjer takvih uređenih površina mogu se vidjeti sa južne strane zgrade Građevinskog fakulteta.



Slika 47 Ravnina u Mayi za prikaz ograđenog terena

Hardware i njegova uloga u obradi 3D prostora

Grafički procesor ima puno jedinica koje rade paralelno i učestalo je da se različite jedinice posvećuju različitim dijelovima prikaza. Zbog ovoga treba provjeriti gdje nam se performanse grafičke kartice koriste kada pričamo o grafičkoj kartici kao uskom grlu. Prva linija obrane od zagušenja grafičkog procesora je smanjenje broja piksela na ekranu, tj. promjenom rezolucije. U koliko nismo primijetili promjenu u performansama, moramo dublje tražiti. Za početak treba provjeriti brojeve točaka na objektima (razine detalja, LOD). Prevelika uporaba normal mapa može također oštetiti performanse kao i previše atributa pojedine točke i prevelika složenost objekata [8]. Manje učestali problemi su prevelika uporaba čestica na

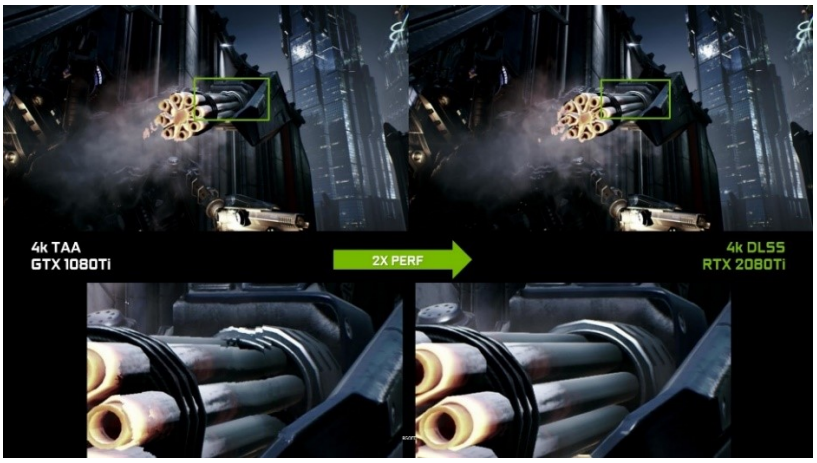
sceni i složene rasvjete scene. Različiti materijali na objektima imaju i različitu složenost, tako da klasični materijali su najjednostavniji, maske su srednji po složenosti, a najzahtjevniji su prozirni materijali zbog dodatnog prikaza.

Performanse računala prilikom izrade 3D grafike su i više nego potrebne. Proizvođači grafičkih procesorskih jedinica, Nvidia i AMD, također nude svojim kupcima zanimljive tehnologije karakteristične za njihov brend. Kako se ove tehnologije stavljaju u drugi plan pored oznake GPU-a, broja jezgri i količine grafičkog RAMa, malo ljudi priča o njima i zna da ih se može upotrijebiti u razvijanju specijalnih efekata ili u razvoju računalnih igara.

DLSS

Zelena ekipe Nvidia razvila je tehnologiju nazvanu DLSS (Deep Learning Super Sampling), koji se koristi u RTX seriji Nvidia grafičkih kartica [5]. U suštini, koristi se umjetna inteligencija kako bi se povećao broj sličica u sekundi pod grafički zahtjevnim scenama. DLSS najprije izvlači mnoge nasumične slike iz ciljane igre, te za svaku stvori odgovarajući "savršen verziju slike" koristeći ili super-uzorkovanje ili akumulacijsko prikazivanje. Parovi slika se prenose u superračunalo NVIDIA-e. Superračunalo trenira DLSS model kako bi prepoznao izvučene slike i generirao visokokvalitetne slike uglađenih rubova koje odgovaraju "savršenim slikama" što je moguće bliže [5]. Zatim ponavljamo postupak, ali ovaj put treniramo model da generira dodatne piksele umjesto da primjenjuje zaglađivanje rubova. To ima za posljedicu povećanje razlučivosti ulaza. Kombinacija obje tehnike omogućava GPU-u pružanje pune razlučivosti monitora pri većem brojem sličica po sekundi.

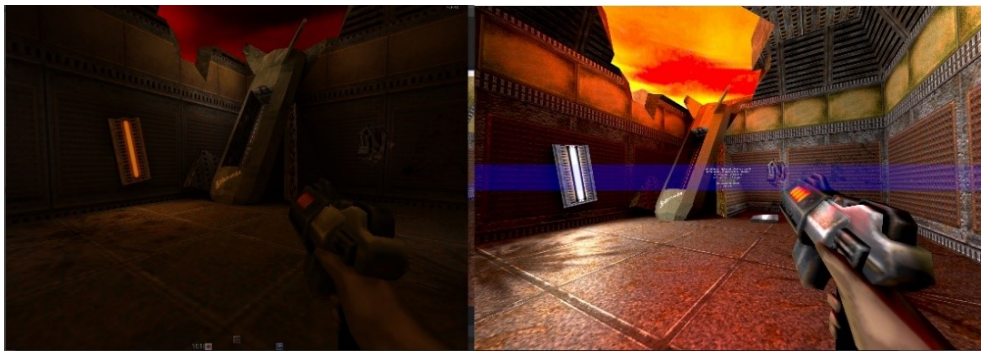
Rezultati DLSS-a variraju jer svaka igra ima različite karakteristike temeljene na engineu igre, složenosti sadržaja i vremenu provedenom na treningu modela. Poboljšavanje neuronske mreže dubokog učenja se provodi čak i nakon što igra krene sa prodajom. Kada dođe do poboljšanja u pogledu performansi ili kvalitete slike, Nvidia objavi nadogradnju za njihovu aplikaciju koja dolazi sa grafičkim karticama (Slika 48). Za pokretanje duboke neuronske mreže DLSS zahtijeva određeno vrijeme GPU-a po sličici prikaza. Tako igre koje rade pri nižem broju sličica po sekundi (proporcionalno manjim fiksnim radnim opterećenjem) ili višim razlučivostima (veće uštede u pikselu) imaju veću koristi od DLSS-a [5]. Za igre koje se izvode na visokom broju sličica po sekundi ili niskim razlučivostima, DLSS neće ostvariti toliki utisak na performanse. Ako je vrijeme potrebno za renderirati sličicu scene kraće od vremena potrebno za izvršavanje DLSS modela, DLSS neće biti prisutan kako bi štedio resurse. DLSS će se omogućiti samo u slučajevima kada možemo dobiti povećanje performansi. Dostupnost DLSS-a ovisi o igri i modelu grafičke kartice i rezoluciji zaslona [5].



Slika 48 Primjer DLSS-a

RTX

Nvidia je svojom Turing arhitekturom napravila osjetni skok u realizaciji praćenja zraka svijetlosti. Za razliku od prijašnje Pascal arhitekture, Turing arhitektura ima izdvojenu jezgru koja služi za izračunavanje Ray tracinga i time otvorila vrata nečega što se jako dugo smatralo neostvarivim. Problem je nastao što obrada ray tracinga nije dovoljno brza da pruža više od 40 do 50 sličica u sekundi čak i sa najjačim modelima, što je s obzirom na tržište, ispod standarda (standard je 60 sličica po sekundi, a za grafičku karticu koja na hrvatskom tržištu doseže i preko 12 tisuća kuna je ozbiljni problem). Bez uključivanja ray tracinga, isti model grafičke, RTX 2080ti isporučuje dovoljno sličica po sekundi i da zadovolji najbolje monitore na tržištu (preko 200 sličica po sekundi). Problem u trenutku demonstracije je nastao što ciljana publika neće moći uživati u punom potencijalu njihovih monitora za osjetno visoku cijenu. Nvidia nije odustala od svoje nove arhitekture, već je prikazala zavidne rezultate implementacijom ray tracinga na igrama koje nisu stvarale opterećenje na moderne grafičke kartice. Primjeri ovoga su Quake 2 koji je predstavljen 1997. godine (Slika 49) i Minecraft sa izlaskom 2011. godine. Kao pioniri u proizvodnji grafičkih kartica za velike mase koje podržavaju Ray tracing, daljnjim razvijanjem, mogli bi stvoriti novi standard grafike kako za programere, tako i za konkurente na tržištu.



Slika 49 Primjer Ray tracinga

Radeon Image Sharpening

Kao odgovor DLSSu, AMD je razvio tehnologiju Radeon Image Sharpening. U osnovi, RIS tehnologija stvara oštriji prikaz nakon završne obrade. AMD se izjasnio kako ovaj proces obrade slike u naknadnoj obradi, ne šteti na performanse, a statistika koju su proveli pokazuje pad od manje od 1% u laboratorijskim uvjetima.

Pretpostaviti ćemo da postoje dva ključna slučaja upotrebe ove tehnologije izoštravanja slike: prvi je za igre koje su 'meke' za početak. Mnogi se naslovi ovih dana koriste privremenim uglađenim rubovima ili TAA (*„temporal anti-aliasing“*), a to često može dovesti do zamagljenih prikaza. Radeon Image Sharpening je način da se pooštre te igre i dobiju jasnije slike [3]. Drugi je slučaj upotrebe za smanjenje rezolucije. Na primjer, ako imate 4K zaslon, ali želite bolje performanse, možete pokretati igre pri 1800p ili skalirati veličinu slike na 80% rezolucije. Ovo će neizbježno učiniti sliku mekšom jer se ne prikazuje u matičnoj razlučivosti. Radeonovo izoštravanje slike moglo bi korigirati sliku, dodati oštrine i približiti se nativnoj slici za gotovo bez gubitka performansi. Radeon izoštravanje slike nije samo filter, već koristi kontrastno adaptirajuće izoštravanje ili CAS algoritam (*„contrast adaptive sharpening“*) koji je AMD nedavno predstavio u svom FidelityFX paketu (Slika 50) [3]. Iako programeri igara mogu uzeti FidelityFX i implementirati ih u svoje igre kako smatraju prikladnim, RIS primjena je širokog spektra te je učinak koji ne zahtijeva implementaciju po igri.



Slika 50 Primjer Radeon Image Sharpeninga

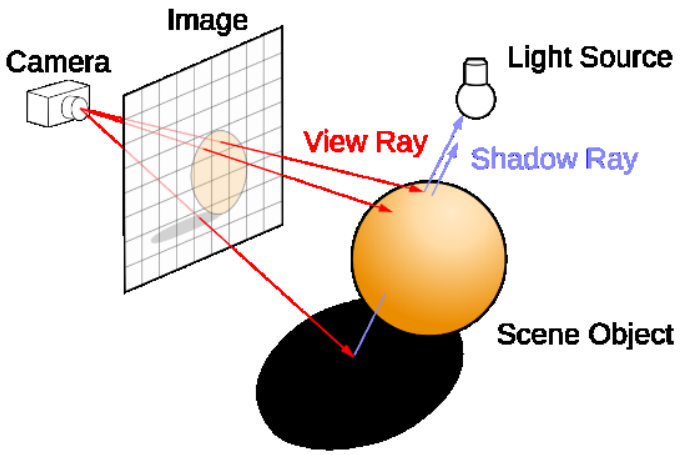
Ray Tracing – tehnologija budućnosti, danas

Ray Tracing ili praćenje zraka svijetlosti je tehnika renderiranja metodom praćenja svijetla kao piksele iz perspektive promatrača i simulirati efekte nastale pri kontaktu sa drugim virtualnim objektima [6]. Tehnika je sposobna proizvesti visoko realistične virtualne prostore i tehnološki je korak naprijed od renderiranja red po red, ali i ozbiljne potrebe za većim računalnim performansama. Zbog troška performansi, praćenje zraka svijetlosti je našlo veću primjenu u statičnim slikama i filmu gdje vrijeme za renderirati pojedinu sličicu nije presudno kao što je slučaj sa računalnim igrama, gdje je brzina presudna. Metoda praćenja zraka svijetlosti je sposobna prikazati širok spektar optičkih efekata, kao što su refleksija, refrakcija, raspršivanje svijetlosti i drugi.

Nalaženje najbližeg objekta kojeg zraka svijetlosti pogađa, zahtijeva presijecanje zrake s primitivnim geometrijskim oblicima koji sačinjavaju scenu [6]. Očito je da bi naivan pristup ovom problemu bilo presijecanje svake odaslane zrake sa svakim objektom u sceni bio neprihvatljivo zahtjevan za računalo i drastično bi narušilo brzinu izvođenja. U cilju ubrzanja ovog postupka grade se optimizacijske prostorne strukture koje velikim dijelom eliminiraju nepotrebna presijecanja.

Prvi algoritmi ray tracinga korišteni za renderiranje su predstavljeni 1968. godine od strane Arthura Appela [6]. U današnjem razumijevanju praćenja svjetlosnih zraka, Appelov algoritam se nazvao Ray casting ili ispaljivanje zraka svijetlosti. Algoritam je baziran na ideji da „ispaljujemo“ zrake iz oka za svaki piksel i lociramo najbliže objekte koji blokiraju putanju zrake. Koristeći svojstva materijala i utjecaja svijetla na scenu, algoritam može odrediti pripadajuću sjenu za taj objekt. Pojednostavljeni zaključak bi bio da ako površina okrenuta prema svijetlosti, svijetlost će doći do površine i neće biti blokirana. Bitna prednost ispaljivanja zraka svijetlosti nad metodom linijskog algoritma je mogućnost obrade neravnih površina i obrade stožaca i sfera [6]. Ako matematička površina može biti presječena zrakom, može biti i renderirana metodom ispaljivanja zraka svijetlosti, čak i kod složenih objekata.

Sljedeće važno otkriće došlo je od Turnera Whitteda 1979 godine. Prethodni algoritmi pratili su zrake iz oka na scenu sve dok nisu pogodili predmet, ali određivali su boju zrake bez rekurzivnog traženja više zraka [6]. Whitted je nastavio postupak. Kad zraka pogodi površinu, ona može stvoriti do tri nove vrste zraka: refleksiju, refrakciju i sjenu (Slika 51). Zraka refleksije prati se zrcalila. Najbliži objekt koji presijeca je ono što će se vidjeti u odrazu. Refrakcijske zrake koje prolaze kroz prozirni materijal djeluju slično, uz dodatak da odlomljena zraka može ući u materijal ili izaći iz njega. Zrake sjena se prati prema svakoj svjetlosti. Ako se između površine i svjetlosti nađe neki neproziran predmet, površina je u sjeni i svjetlost je ne osvjetljava. Rekurzivno traženje zraka unijelo je više realizma u slike bazirane na ovoj tehnici.



Slika 51 Primjer ray tracinga

Popularnost prikazivanja temeljenog na tehnici praćenja zraka svijetla proizlazi iz njene osnove u realističnoj simulaciji prenošenja svjetlosti, u usporedbi s drugim metodama prikazivanja, poput rasterizacije, koja se više fokusira na realnu simulaciju geometrije (Slika 52). Učinci poput refleksija i sjena, koje je teško simulirati pomoću drugih algoritama, prirodni su rezultat algoritma ray tracinga. Računalna neovisnost svake zrake omogućuje praćenje zraka podložnom osnovnom stupnju paralelizacije, ali divergencija putova zraka stvara visoku upotrebu paralelizma koju je prilično teško postići u praksi.

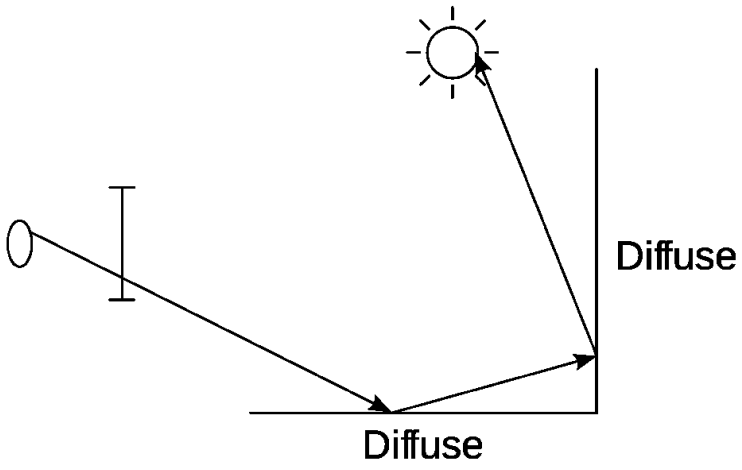


Slika 52 Mogućnosti ray tracinga

Ozbiljan nedostatak praćenja zraka svijetlosti su performanse (iako u teoriji mogu biti brže od tradicionalnog renderiranja linije po liniju, ovisno o složenosti scene u odnosu na broj piksela na zaslonu). Algoritmi renderiranja liniju po liniju i drugi algoritmi koriste koherencija podataka za dijeljenje izračuna između piksela, dok praćenje zraka svijetlosti normalno započinje postupak iznova, tretirajući svaku zraku oka odvojeno. Međutim, ovo odvajanje nudi

i druge prednosti, kao što je mogućnost ispaljivanja više zraka po potrebi za provođenje prostornog uklanjanja i stvaranje poboljšanja kvalitete slike tamo gdje je to potrebno

Iako se ručno obrađuju optički efekti poput refrakcije, tradicionalno praćenje zraka svijetlosti također nije nužno foto realistično. Pravi foto realizam događa se kada se jednadžba iscrtavanja precizno aproksimira ili u potpunosti implementira. Primjena renderske jednadžbe daje istinski foto realizam, jer jednadžba opisuje svaki fizički učinak strujanja svjetlosti (Slika 53). Međutim, to je obično neizvedivo s obzirom na računalne performanse.



Slika 53 Primjer rekurzije ray tracinga

Realnost svih metoda renderiranja može se procijeniti kao aproksimacija jednadžbe. Praćenje zraka svijetlosti, ako je ograničeno na Whittedov algoritam, nije nužno najrealnije. Metode koje prate zrake, ali uključuju dodatne tehnike (mapiranje fotona, praćenje putanja) daju daleko precizniju simulaciju rasvjete u stvarnom svijetu [6].

Zaključak

Realni prikaz svijeta u igrama doseže svoje limite. Imamo situaciju da smo dopeljali detalje modela i površina na maksimalne razine, ali ostaje nam nekoliko smjerova u kojima možemo kvalitetnije rezultate postići. Ako uzmemo u obzir optimizaciju performansi, razvijanjem novog hardwarea možemo usmjeriti optimizaciju prema drugim izazovima. Veličina prostora koja se proceduralno generira dostiže velike razmjere, ali takvi prostori gube osobnost. Novim tehnologijama omogućiti ćemo generiranje velikih prostora bez da imamo učitavanja i prekide u iskustvu. Sama računala su već dugi niz godina bazirana na istom modelu, samo se izmjenjuju arhitekture procesora, veličine izrade i brojevi u frekvencijama i količini. Osjetni skok u mogućnostima ćemo osjetiti kada reduciramo komponente u računalu. Kako su nam potrebni i procesor i grafički procesor za uspješno simuliranje 3d prostora, u budućnosti bi mogli vidjeti jedinstvene proizvode koji bi objedinili oba procesora. Također, mogućnosti enginea nas ograničavaju po broju radnji koje se mogu događati u prostoru odjednom. Veći Brojem radnji, možemo povećati fluidnost i masivnost scena što će dovesti do novih scenarija. Primjer scenarija od prije nekoliko godina je „masivna“ bitka koja je se sastojala od manje od 15 osoba na svakoj strani i svega 4 sa svake strne su se borila, zbog ograničenja enginea da prikazuje 8 radnji odjednom. Budućnost Unreala sa druge strane obećavajuća. Bez obzira na dosadašnja postignuća, besplatni engine za projekte, sa poslovnim modelom da naplaćuje samo ako smo prešli određeni prag zarade je ohrabrujući prema početnicima. Uz poslovni plan, UE nam nudi i besplatne elemente svaki mjesec u vrijednosti nekoliko stotina eura i sa zadnjim poslovnim potezom ukazuju da nemaju namjeru prestati sa ovom politikom. U 11. mjesecu 2019. kupili su kompaniju Quixel i sve njihove proizvode su besplatno dali na korištenje sa UE. Quixel je u tom trenutku imao nekoliko tisuća tekstura i modela visoke rezolucije i 3 programa za asistenciju izrade novih materijala koji se koriste na profesionalnoj sceni. U budućnosti vidim probijanje nove, mlade scene grafičkih programera koji će svoje znanje bazirati na besplatnim alatima koji su im dostupni danas.

Popis literature

[1] *Autodesk Maya Support*. (26. 11 2019). Dohvaćeno iz https://knowledge.autodesk.com/support/maya

[2] Autodesk Maya Wikipedia. (25. 11 2019). Dohvaćeno iz https://en.wikipedia.org/wiki/Autodesk_Maya

[3] DLSS vs RIS. (25. 11 2019). Dohvaćeno iz https://www.techspot.com/article/1873-radeon-image-sharpening-vs-nvidia-dlss/

[4] Košuljandić, T. (7 2017). 3D model objekta kulturne baštine. Rijeka.

[5] Nvidia DLSS. (25. 11 2019). Dohvaćeno iz https://www.nvidia.com/en-us/geforce/news/nvidia-dlss-your-questions-answered/

[6] Ray tracing Wikipedia. (25. 11 2019). Dohvaćeno iz https://en.wikipedia.org/wiki/Ray_tracing_(graphics)

[7] Unreal Engine Documentation. (25. 11 2019). Dohvaćeno iz https://docs.unrealengine.com/en-US/index.html

[8] Unreal Engine Performance and profiling documentation. (26. 11 2019). Dohvaćeno iz https://docs.unrealengine.com/en-US/Engine/Performance/index.html

[9] Unreal Engine Wikipedia. (25. 11 2019). Dohvaćeno iz https://en.wikipedia.org/wiki/Unreal_Engine

[10] Wikipedia - 3D rendering. (11. 12 2019). Dohvaćeno iz https://en.wikipedia.org/wiki/3D_rendering

[11] Wikipedia - Computer Graphics. (11. 12 2019). Dohvaćeno iz https://en.wikipedia.org/wiki/Computer_graphics#1980s

[12] Wikipedia - sRGB. (11. 12 2019). Dohvaćeno iz https://en.wikipedia.org/wiki/SRGB

Popis Priloga

3D modeli izrađeni u Mayi i video prezentacija prostora izrađena u Unreal Engineu se nalaze na priloženom CD-u.

Popis slika

Slika 1 Sveučilišna avenija, kolovoz 2011 6
Slika 2 Sveučilišna avenija, listopad 2019 6
Slika 3 Kampus, kolovoz 2011 6
Slika 4 Kampus, listopad 2019 6
Slika 5 Elementi sučelja Maya, lijeva navigacija i 3d prostor sa mrežom 9
Slika 6 Maya, Paralelni prikaz više kamera 10
Slika 7 Maya, Glavna navigacija za modeliranje 10
Slika 8 Maya, Glavna navigacija za osvjetljenje 10
Slika 9 Maya, Desna navigacija i detalji o objektu 11
Slika 10 Tim Sweeney 11
Slika 11 Unreal Level Editor, prva verzija 12
Slika 12 Unreal Engine treća verzija 13
Slika 13 Unreal lijeva navigacija 14
Slika 14 Unreal, traka funkcija 15
Slika 15 Unreal, naredbe unutar 3d prostora 16
Slika 16 Pivot točka 16
Slika 17 Pivot točka nije vezana za objekt 16
Slika 18¸Unreal desna navigacija 17
Slika 19 Primjer Collisiona 17
Slika 20 Osnovna čvorna točka materijala 19
Slika 21 Naljepnica prometne signalizacije 20
Slika 22 Razlika u intezitetu specular teksturi 21
Slika 23 Primjer maske 22
Slika 24 Primjer maske u primjeni 22
Slika 25 primjer normal teksture 22
Slika 26 Razlika koju normal tekstura stvara na modelima 23
Slika 27 Specular tekstura u sRGB formatu 24
Slika 28 AO tekstura u sRGB formatu 24
Slika 29 skaliranje tekstura na elementima različite veličine 25
Slika 30 primjer množenja teksture sa skalarom 26
Slika 31 Kartica oblikovanja terena 26
Slika 32 Primjeri manipulacije terenom 27
Slika 33 Teren iza objekta koji korisnik ne vidi 28
Slika 34 Primjer različite gustoće alata za vegetaciju 28
Slika 35 Primjeri sitnih razlika u materijalu 29
Slika 36 Stablo sa niskom razinom detalja izdaljine 30
Slika 37 Stablo sa visokom razinom detalja izbliza 30
Slika 38 Kadar iz videa o predstavljanju performansi 31
Slika 39 Primjer razdjeljivanja segmenata modela 32
Slika 40 Primjer nepravilnog oblika modela 33

Slika 41 Loša grupacija objekata i primjena istog materijala na sve elemente grupacije 33
Slika 42 Složeni objekt kojemu treba ukloniti višak stranica 34
Slika 43 Primjer preklapanja tekstura 34
Slika 44 Scena vraćena u Mayu sa pozicijama objekata kako bi se konstruirala okolina 35
Slika 45 Iskopavanje objekata okoline 36
Slika 46 Objekt visi u zraku 36
Slika 47 Ravnina u Mayi za prikaz ograđenog terena 36
Slika 48 Primjer DLSS-a 38
Slika 49 Primjer Ray tracinga 39
Slika 50 Primjer Radeon Image Sharpeninga 39
Slika 51 Primjer ray tracinga 41
Slika 52 Mogućnosti ray tracinga 41
Slika 53 Primjer rekurzije ray tracinga 42