SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4009

IZRADA SLOŽENIH MODELA INTERIJERA

Bruno Pregun

Zagreb, srpanj 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 11. ožujka 2015.

ZAVRŠNI ZADATAK br. 4009

Pristupnik: **Bruno Pregun (0036459671)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Izrada složenih modela interijera**

Opis zadatka:

Proučiti programske alate koji omogućuju izradu složenih modela unutrašnjih prostora. Potrebno je omogućiti postavljanje dinamičkih interaktivnih objekata, dinamičko upravljanje osvjetljenje te zvukovima u sceni. Realizirati model interijera te omogućiti virtualno kretanje prostorom uz interaktivno aktiviranje pojedinih komponenti. Načiniti ocjenu ostvarenih rezultata.
Izraditi odgovarajući programski proizvod. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 13. ožujka 2015.
Rok za predaju rada: 12. lipnja 2015.

Mentor:

Prof. dr. sc. Željka Mihajlović

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:

Prof. dr. sc. Siniša Srbljić

SADRŽAJ

1. Uvod

Od početka vektorskih zaslona, preko modernog rasterskog prikaza, do budućih tehnologija poput prividne stvarnosti, računalna grafika je bila, i biti će, jedno od mnogih područja računarstva koje se sve više unaprjeđuje i razvija. Iz godine u godinu, grafički procesori su sve moćniji, algoritmi razvijeniji, programeri vještiji i alati pristupačniji. Sve to zajedno čini računalnu grafiku raširenijom nego ikad dosad. Osim što se osnovna znanja iz računalne grafika svakodnevno koriste za svaki prikaz slike na računalnim zaslonima, uz to se puno naprednije tehnike iskorištavaju za prikaz trodimenzionalnih scena. Prikaz kompleksnih scena je od velike važnosti u aplikacijama koje pokušavaju simulirati sliku stvarnog svijeta. Takve aplikacije se najčešće koriste u polju arhitekture, marketinga, animacije, vojne industrije, i naravno računalnih igara. Upravo to su područja koja su najviše doprinijela razvoju računalne grafike kakvu poznajemo danas.

Kroz ovaj rad, biti će opisan razvoj jedne takve aplikacije. Cilj je modelirati i prikazati unutrašnjost građevine. Slične aplikacije koriste se kao moderna zamjena za arhitektonske makete, koje često prikazuju postojeći prostor, ili prostor koji se planira izgraditi. Također je potrebno omogućiti kretanje kroz virtualni model i interakciju s elementima u sceni.

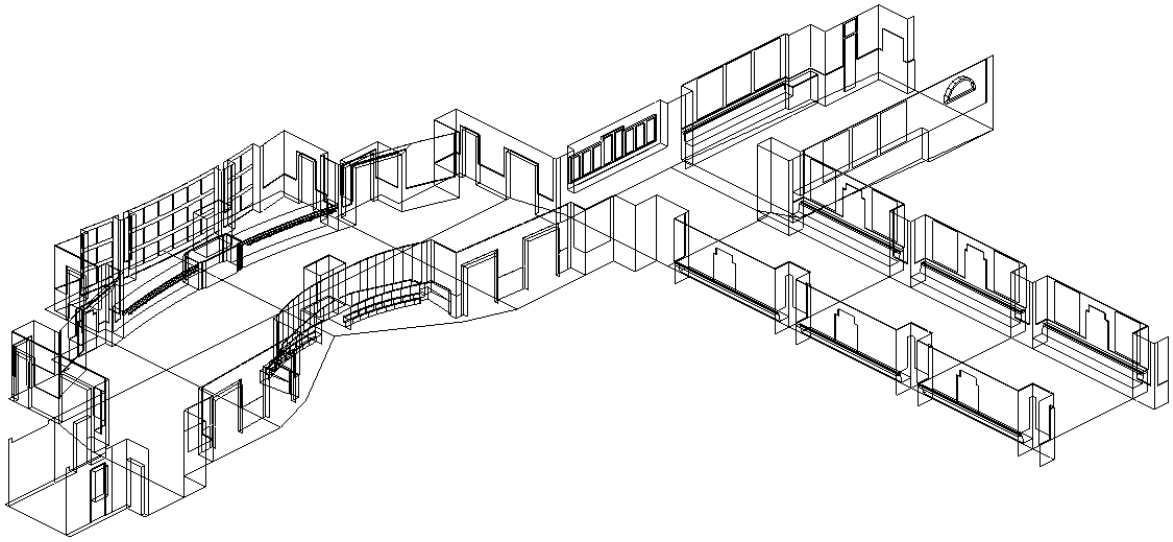
Za realističan prikaz scene potrebno je izraditi geometrijske modele, definirati materijale u sceni i konfigurirati osvjetljenje. Iako postoje razni načini prikazivanja geometrije, za prikaz u stvarnom vremenu najčešće se koriste modeli sastavljeni od poligona. Taj način prikaza je trenutno najraširenij, pa postoji mnoštvo alata koji pomažu kod izrade takvih modela. Materijali se definiraju pomoću skupine fizičkih svojstava, poput boje, faktora refleksivnosti, transparentnosti itd. Osvjetljenje scene se postiže raznim algoritmima koji nastoje simulirati ponašanje svijetla u stvarnom svijetu. Osvjetljenje ima vrlo važnu ulogu jer kompleksnost algoritama koji se koriste direktno utječe na realističan izgled scene. Iako je moguće scenu prikazati samo pomoću modela i materijala, simuliranje svjetlosti daje bolji osjećaj dubine i trodimenzionalnosti na dvodimenzionalnom zaslonu.

Zbog dualne prirode svjetlosti, izračun svakog vala i čestice u realnom mjerilu trenutno predstavlja nepremostiv problem, pa se kod osvjetljenja virtualnih scena često koriste razni trikovi koji na brojne načine pojednostavljuju prikaz.

Kroz nekoliko posljednjih godina, razvijeni su alati koji olakšavaju izgradnju i prikaz virtualnih scena. U ovom radu će se razmotriti dva moćna alata koja su u vrijeme pisanja ovoga teksta dostupni na internetu za besplatno korištenje. Za modeliranje će se koristiti alat Blender 2.74, a prikaz i interakcija biti će pogonjena pomoću Unreal 4.7 engine-a.

2. Modeliranje interijera

Kao predmet modeliranja u ovom radu odabran je dio prizemlja D zgrade, Fakulteta elektrotehnike i računarstva u Zagrebu. Model je podijeljen u tri hodnika. Glavni hodnik sadrži pult studentske službe, na njega se nadovezuju druga dva sporedna hodnika. Jedan vodi prema A zgradi na sjeveru, a drugi povezuje istočni dio D zgrade (tzv. aleja laptopa). Izgled prostora može se vidjeti na slici 1. Svaki hodnik dodatno je podijeljen na manje modele.



Slika 1. Izometrijski prikaz modeliranog prostora

2.1. Izrada modela

Za dostojan prikaz odabranog prostora potrebna je velika količina modela koji se mogu razvrstati u tri glavne skupine.

Prva skupina su modeli koji čine prostorije interijera. To su zidovi, podovi i stropovi. Modeli podova i stropova su u najčešćem slučaju vrlo jednostavni. Podovi su uglavnom ravne plohe, a stropovi imaju nekoliko razina udubljenja. Modeli zidova su najkompleksniji modeli u ovoj skupini jer sadrže veliki broj ploha i detalja. Neke od tih detalja je moguće zanemariti prilikom modeliranja te ih kasnije naknadno dodati pomoću tekstura, no neke je potrebno i modelirati.

Kod modeliranja zidova posebnu pažnju je potrebno posvetiti rubovima modela, jer će se ti modeli u konačnici sastavljati u cjelinu. Za svaki zid je potrebno odrediti granice gdje model počinje i gdje završava. Najpogodnije je te granice postaviti na mjesta gdje postoji granica u stvarnosti, npr. prijelaz iz jednog materijala u drugi između dva zida. Takvim postupkom se smanjuje vidljivost šavova nastalih pri spajanju modela.

Svi modeli prostorija su otvoreni, odnosno nemaju stražnju stranu. To znači da ako se gledaju straga neće biti vidljivi jer se stražnje strane poligona ne iscrtavaju. No to neće biti problem jer su zidovi interijera vidljivi samo s jedne strane. Time se dobiva na većem prostoru za teksture za vidljivu prednju stranu. Takvi jednostrani modeli znaju stvarati probleme kod dinamičnog osvjetljenja, no o tome kasnije.

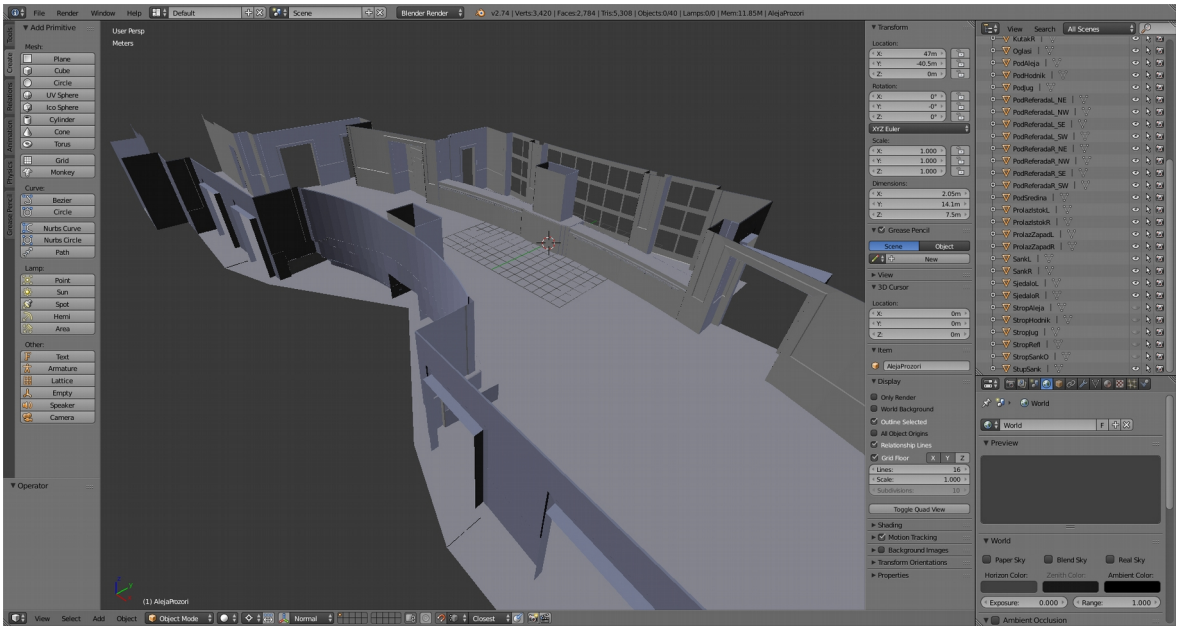
Druga skupina modela su "objekti", odnosno predmeti koji nisu dio prostorije ali se nalaze u njoj. Tu pripadaju i predmeti koji se ponavljaju ili zahtijevaju neku interaktivnost (stropne svjetiljke, stolovi, vrata, ...). Kompleksnost geometrije ovih modela ovisi o veličini i složenosti samog predmeta. Neki od predmeta su razdvojeni na više modela, npr. model vrata se sastoji od modela kvake, plohe staklenog prozora i modela samih vrata. Takvo razdvajanje omogućuje lakše definiranje različitih materijala i modelima daje svojstvo modularnosti.

Modularnost modela je vrlo poželjna, jer omogućuje korištenje istih modela u različite svrhe. Kod modeliranja simulacija stvarnih prostora, modularnost nije uvijek moguća zbog nastojanja da se vjerodostojno prikaže taj prostor kakav je u stvarnosti. S druge strane, modeliranje izmišljenih prostora omogućuje projektiranje modela i samih prostora tako da se umanji broj različitih modela koje je potrebno izraditi. To je često jedan od načina kojim se smanjuje količina posla, a samim time i troškova prilikom izgradnje prostora za video igre. U ovom slučaju modularnost je moguće ostvariti tek kod nekih predmeta i kod simetričnih dijelova prostora.

Treća skupina modela su okolni prostori i pomoćni modeli. To su modeli koji se nalaze izvan modeliranog prostora te se nije moguće kretati kroz njih, ali ih je moguće vidjeti iz glavnog modeliranog prostora. Ovi prostori služe samo kao pozadina za glavni model interijera pa ih nije potrebno detaljno modelirati.

Za izradu i manipulaciju modela korišten je program Blender 2.74. Sučelje programa prikazano je na slici 2. Iako Blender nije u potpunosti prilagođen CAD (*computer-aided design*) principima, pokazao se kao adekvatan alat za ovaj zadatak. Iako je prilikom modeliranja posebna pažnja bila posvećena mjerilu, ukazali su se problemi s malim (zanemarivim) pomacima zbog niske preciznosti zapisa položaja vrhova poligona. Uzrok problema je ostao nepoznat, no simptomi su umanjeni izbjegavanjem korištenja naprednijih funkcija.

Kao pomoć pri izradi modela, korištene su fotografije stvarnog prostora snimljene iz nekoliko različitih kuteva. Kasnije su iste fotografije bile korisne kod definiranja materijala i izrade tekstura.



Slika 2. Sučelje programa Blender

2.2. Razmatanje UV koordinata

Kako bi se teksture mogle pravilno preslikati na modele potrebno je odrediti odnosno razmotati UV-koordinate vrhova. Prilikom tog postupka treba imati na umu da Unreal 4 engine koji će se kasnije koristiti za prikaz, zahtjeva da se područja određena UV-koordinatama ne preklapaju. Razlog tome je što se te iste koordinate koriste za spremanje mape osvjetljenja (lightmap) prilikom gradnje scene. Ako se to pokaže problematičnim, moguće je modelu dati dvije skupine UV-koordinata. Jedne za teksturu, a druge za mapu osvjetljenja. To bi omogućilo da se skup UV-koordinata koji se koristi za teksture preklapa, a skup koji se koristi za mapu osvjetljenja ne preklapa. Preklapanjem koordinata bi uštedjeli na prostoru teksture i time bili u mogućnosti dobiti nešto veću rezoluciju detalja. U ovom slučaju to nije bilo potrebno, pa se za obje namjene koriste iste koordinate.

Za razmatanje korištena je *Unwrap* funkcija unutar Blendera. Ta funkcija omogućava automatsko razmatanje, no da bi dobili dobar rezultat potrebno je prethodno označiti bridove na kojima su šavovi. Pomoću tih bridova funkcija razmata UV-koordinate i kompaktno ih postavlja na kvadratni prostor. Nakon toga dobiveni rezultat je moguće spremiti kao sliku u *.png* formatu. Ta se slika dalje koristi kod izrade teksture.

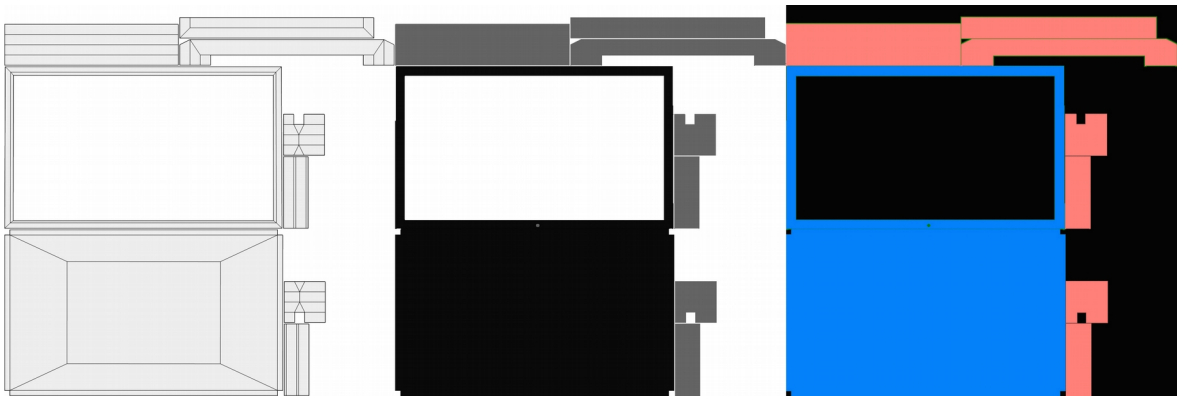
2.3. Izrada tekstura

Teksture je moguće izraditi u svakom programu za uređivanje slika. Za potrebe ovog rada korišten je program Gimp 2.8.14. Ovisno o materijalu koji se želi prikazati na teksturi potrebna je određena količina detalja koje je moguće reproducirati iz fotografija.

Kod uzimanja uzoraka s fotografija treba obratiti pažnju na razne nesavršenosti na fotografijama. Površina bi trebala biti uslikana pod okomitim kutem, no u tom slučaju zna doći do refleksije, pogotovo ako se koristi bljeskalica na fotoaparatu. Iz tog razloga je potrebno malo nakositi kut snimanja kako bi se izbjegla refleksija.

Uostalom, uporaba bljeskalice za snimanje tekstura se ne preporučuje jer rezultira neprirodnim bojama. Drugi utjecaj na fotografiju ima osvjetljenje. Ukoliko je prostor osvijetljen umjetnim svijetlom slike su sklone narančastoj nijansi. Dnevno svijetlo također utječe na boju fotografije. Kako bi se taj problem ublažio korištena je mogućnost kalibriranja bijele boje na fotoaparatu, uporabom bijelog lista papira kao referentne boje u prostoru. Druga mogućnost je snimanje fotografija u *.raw* formatu i dodatno uređivanje unutar alata za razvoj fotografija.

Materijali mogu biti definirani s više tekstura. Najčešća kombinacija je tekstura koja sadrži difuznu boju površine predmeta i tekstura koja sadrži informaciju o manjim izbočinama i udubljenjima na površini materijala. Za to se koristi mapa normala (*normal map*). Za definiranje mape normala svakog modela bilo bi potrebno izraditi puno detaljnije (*high-poly*) verzije tih modela ili ručno crtati detalje na mapi. Iz tog razloga mape normala se neće koristiti u ovoj aplikaciji.



Slika 3. UV-koordinate i teksture za model televizora sa zidnim nosačem

Unreal Engine 4 koristi tehniku fizikalno temeljenog prikaza (*Phisically Based Rendering*, ili *PBR*). Za materijale je potrebno definirati nekoliko parametara koji utječu na izgled materijala, pa je osim teksture s informacijama o boji, korištena i tekstura u čije su RGB kanale kodirane informacije o parametrima materijala. Na slici 3 mogu se vidjeti razvijene UV-koordinate (lijevo), difuzna tekstura (sredina) i tekstura s parametrima materijala (desno). Korištenjem te dvije vrste tekstura za svaki model omogućava da jedan model sadrži više različitih vrsta materijala. Detaljnije o materijalima u kasnijem poglavlju.

2.4. Izvoz u FBX format

Nakon što je model završen potrebno ga je prenijeti u Unreal. To se postiže korištenjem *.fbx* formata. FBX (**F**ilm**b**o**x**) format omogućuje pohranu raznih struktura podataka vezanih uz 3d modeliranje i animaciju u binarnom ili ASCII obliku. Materijali koji se mogu definirati unutar Blendera nisu u potpunosti kompatibilni s materijalima u Unreal-u, pa se prilikom izvoza modela prenose samo koordinate vrhova, njihove normale i UV-koordinate.

Prije prijenosa je potrebno dodatno prilagoditi model za Unreal. Najvažniji korak je prilagodba mjerila. Blender nema unaprijed definiran odnos između svojih jedinica mjerila i metričkog sistema, pa ako se želi koristiti konkretno mjerilo potrebno je ručno unijeti vrijednosti. Za modele je korišteno mjerilo od pola metra za svaku jedinicu unutar Blendera (0.5 m = 1 BU, Blender unit). Unreal kao svoju mjernu jedinicu koristi centimetar (0.01 m = 1 UU, Unreal unit). Iz tog razloga su modeli prije prijenosa skalirani za 0.005. Također je potrebno model postaviti u izvorište scene, na koordinate (0, 0, 0). U Blenderu je moguće definirati i modele koji bi Unreal mogao koristiti kao kolizijsku mreže (collision mesh) za model, kao i modele za razine kompleksnosti (Level of Detail, LOD), no te mogućnosti nisu korištene, već je kolizijska mreža automatski generirana u engine-u. Nakon ovog postupka preostaje samo unos modela u Unreal, a to se svodi na prijenos *.fbx* datoteke.

3. Izrada simulacije

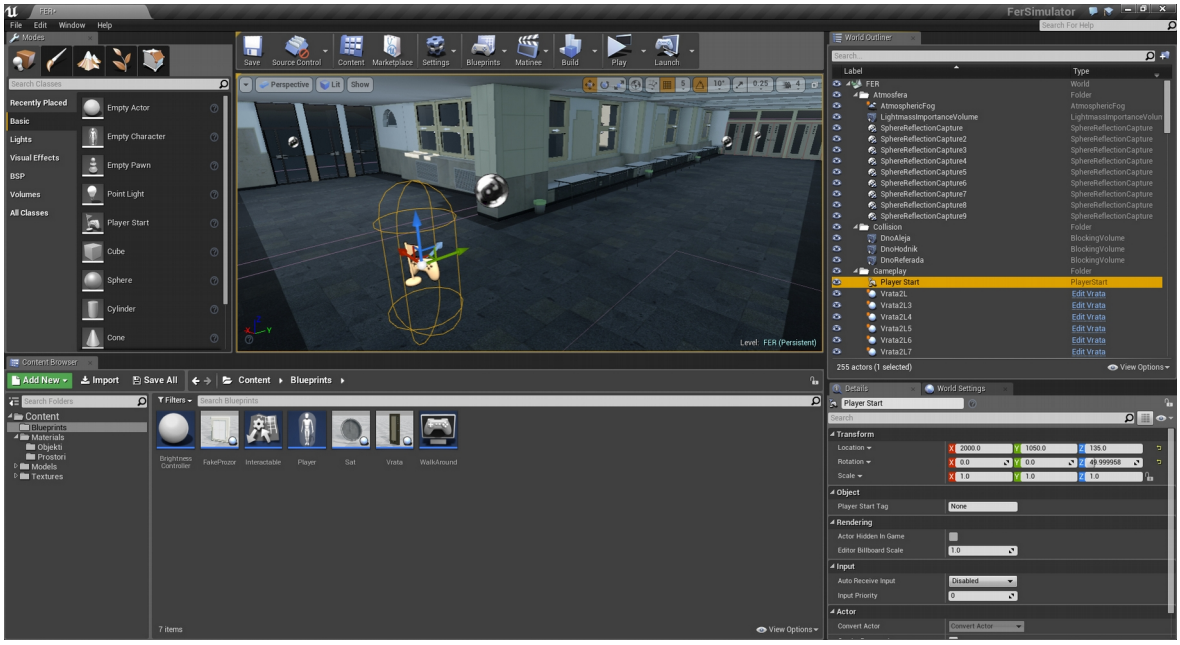
Svrha aplikacije je prikazati modelirani prostor u stvarnom vremenu, omogućiti kretanje kroz prostor i interakciju s okolinom. Također će se istražiti ostale mogućnosti koje pruža Unreal Engine 4.

Prva verzija Unreal engine-a je izrađen 1998, za potrebe izrade i izvršavanja video igre *Unreal* i *Unreal Tournament*. Već tada je to bio moćan alat koji je sadržavao sustave za iscrtavanje, animaciju, detekciju sudara, umjetnu inteligenciju, mrežnu povezanost, skriptiranje i upravljanje datotekama. Sve to zapakirano u cjelokupni paket. I baš je zbog te modularne arhitekture i podrška za skriptiranje, Unreal Engine postao iznimno popularan za izradu modifikacija za igre.

Kroz godine je razvijeno nekoliko verzija i iteracija, uključujući i *Unreal Development Kit*. Ta verzija je bila prvi puta da je Unreal Engine bio besplatno dostupan javnosti za nekomercijalne svrhe. Time je Unreal Engine postao jedan on najpopularnijih alata za izradu igara, no s novijim verzijama se počinje koristiti i u druge svrhe, poput animacije, filmskih specijalnih efekata, simulacija i sl.

U ožujku 2015. Unreal Engine 4 je ponovno postao besplatan za nekomercijalnu upotrebu. To je trenutno najnovija inačica i ona će se koristiti u praktičnom dijelu ovoga rada.

Unreal Engine 4 sadrži nekoliko sučelja. Glavno sučelje je prikazano na slici 4.



Slika 4. Glavno sučelje Unreal Engine 4 editora

3.1. Sastavljanje scene

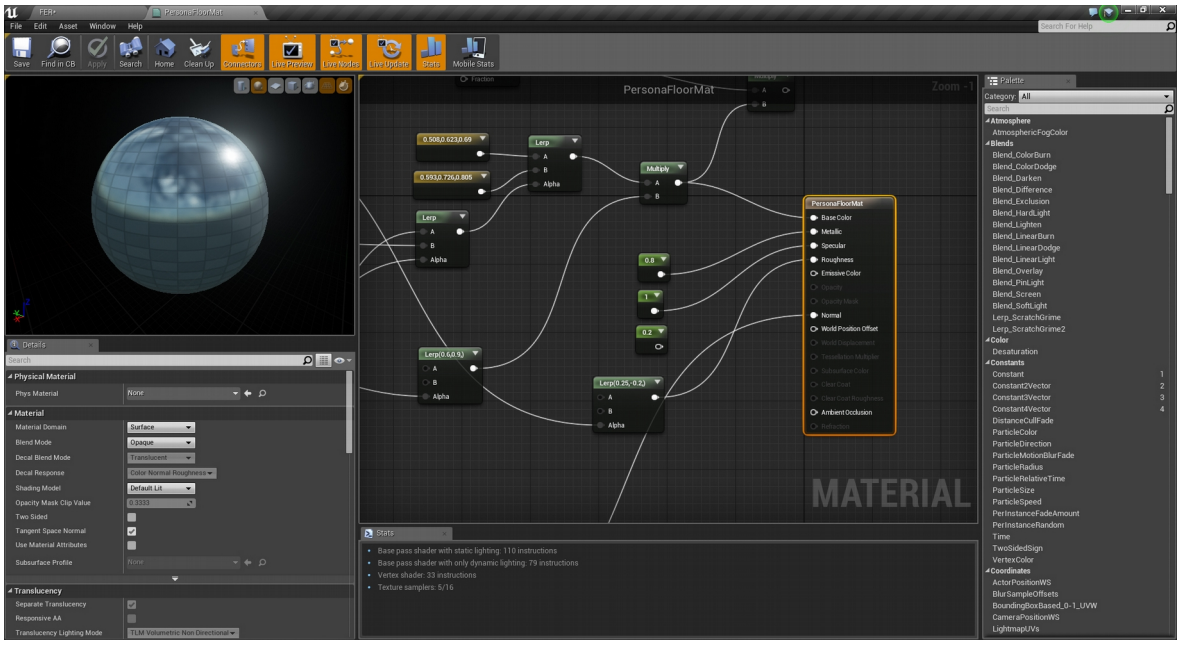
Prvi korak je importirati modele prostora opisane u prethodnim poglavljima. Uz pravilno pripremljene modele nije bilo nikakvih problema s importom. Postavljena mjerila prilikom pohrane rezultirala su odgovarajućim dimenzijama, te pozicioniranje modela u izvorište olakšalo je naknadno pozicioniranje u prostoru.

Objekti u prostoru Unreal-a nazivaju se akterima (*actor*), te mogu biti raznih vrsta. Svaki akter se sastoji od komponenata (*components*) koje mu daju funkcionalnost. Neke od tih komponenata su statički model (*static mesh*), svjetlost (light), model s kosturom za animaciju (skeletal mesh), umjetna inteligencija (AI), izvor zvuka (audio), razne fizičke komponente i sl. Pomoću mnogobrojnih kombinacija tih komponenata moguće je složiti mnoštvo različitih objekata s raznim svojstvima. Ti objekti se zatim mogu pohraniti u klase, koje se nazivaju nacrtom (*Blueprint*). Pomoću nacrta je olakšano stvaranje primjeraka objekata te njihova manipulacija u prostoru. Također je podržano nasljeđivanje svojstava iz roditeljskih nacrta. Nacrte je dodatno moguće definirati kao klasu pomoću C++ koda, to rezultira velikom fleksibilnosti samog engine-a.

Osim direktnog pisanja C++ koda, dodavanje funkcionalnosti akterima omogućeno je i pomoću vizualnog skriptnog jezika. Taj jezik se u starijim verzijama nazivao *Kismet*, no u najnovijoj verziji je skriptiranje u potpunosti ukomponirano u *Blueprint* sustav. Svaki akter osim svojih komponenata sadrži i funkcije koje je moguće skriptirati.

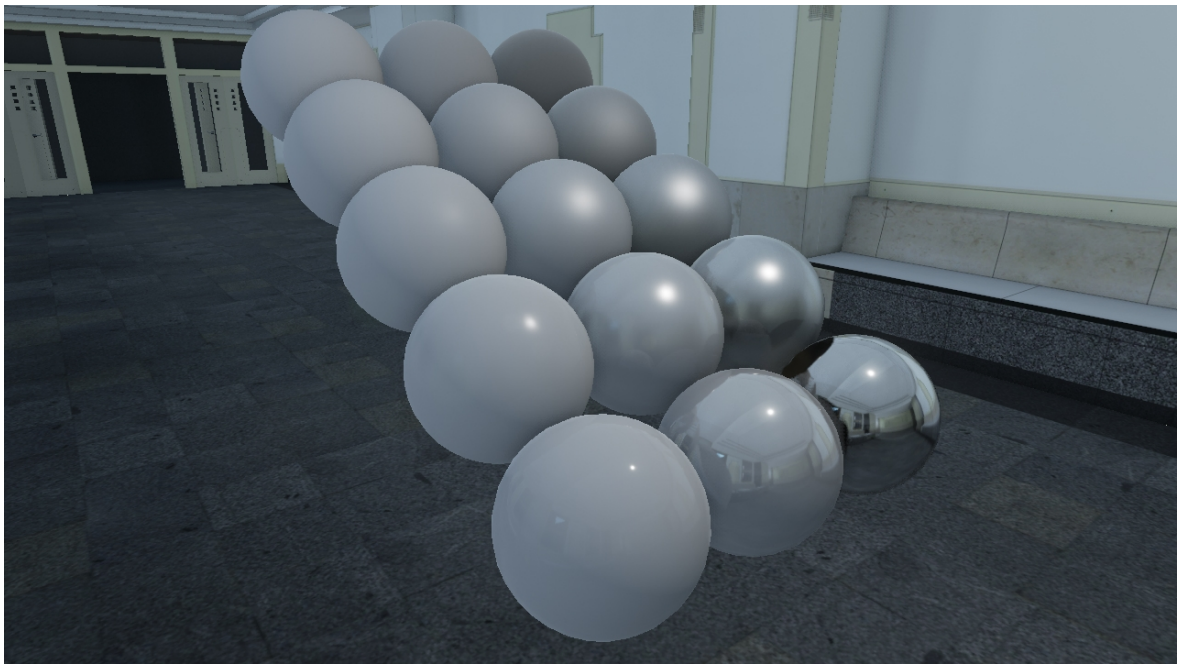
3.2. Definiranje materijala

Izgled materijala se također definira vizualnim jezikom koji se ponaša na sličan način kao i standardni jezici za opis programa za sjenčanje. Na početku materijal sadrži samo izlazni čvor gdje su pobrojani razni parametri, te se povezivanjem elemenata ostvaruje željeni izgled materijala. Na slici 5 je primjer takvih veza.



Slika 5. Sučelje za definiranje materijala

Osnovni parametri izlaznog čvora su: osnovna boja, metalni faktor, reflektirajući faktor, faktor hrapavosti, boja isijavanja, normale, pomak, transparentnost, neprozirnosti, faktor refleksije. Pažljivim odabirom tih parametara može se ostvariti veliki broj različitih materijala, od kojih su neki gotovo fotorealistični, a drugi fizički nemogući u stvarnom svijetu.



Slika 6. Utjecaj parametara materijala na refleksiju

Na slici 6 vidljiv je utjecaj različitih vrijednosti dva parametara na izgled materijala. Kugle su poredane tako da tri kugle na dnu imaju najmanji faktor hrapavosti, a tri kugle na vrhu najveći. Svaki od tri retka ima različit metalni faktor. Lijevi redak sadrži potpuno nemetalane materijale, a u desnom retku su potpuno metalni materijali. U sredini su polovično metalni materijali, koji se jako rijetko koriste jer su u stvarnosti materijali ili metalni ili nisu. Također je zanimljivo spomenuti da sve kugle imaju istu osnovnu boju, no na slici se može primijetiti da nisu sve iste svjetline. To je zbog toga što kugle koje reflektiraju više svjetlosti, odašilju manje difuznog svijetla jer manja količina svjetlosti probije površinu.

Za iscrtavanje scene Unreal Engine 4 koristi fizikalno zasnovane algoritme. Sustav je baziran na zakonu očuvanja energije i brine se da niti jedan objekt ne reflektira više svijetla nego što primi. Za osvjetljenje se koristi niz konvergentnih funkcija temeljenim na stvarnim fizičkim svojstvima koje računaju količinu primljene i odbijene svjetlosti za svaku jedinicu površine. Umjesto korištenja višestrukih difuznih i zrcalnih tekstura za svaki dio modela, dovoljno je definirati parametre materijala.

Za definiranje materijala korištene su dvije vrste tekstura. Prva tekstura sadrži informaciju o boji materijala. Druga tekstura definira tri osnova parametra koji definiraju ponašanje svijetla pri interakciji s materijalom. Tekstura je rastavljena je na RGB komponente. Svaki parametar ima raspon od 0 do 1. Gdje je 1 najviši intenzitet. U crvenom kanalu je definirano koliko je materijal metalan. Ovaj parametar se koristi jer metalna sredstva drugačije reflektiraju svijetlo od nemetalnih. Vrijednost parametra za 'čiste' homogene materijale će biti ili 0 ili 1. Nula odgovara potpuno nemetalnim površinama, a jedinica potpuno metalnim predmetima. Dok za hibridne površine poput zahrđalog metala vrijednost će biti između 0 i 1. U Zelenom kanalu teksture je definiran zrcalni faktor. On utječe na količinu refleksije, od manje je važnosti i ne koristi se često, pa je u većini slučajeva postavljen na sredinu 0.5. Treći faktor, koji je kodiran u plavom kanalu teksture je faktor hrapavosti. On služi za simuliranje iznimno malih nesavršenosti površine materijala. Hrapavi materijali raspršuju reflektiranu svjetlost više nego glatki predmeti, pa ovaj faktor utječe na zamućenje refleksije.

Neki modeli još koriste dodatne teksture za maskiranje, koje definiraju koja područja modela su vidljiva. Za stakla na prozorima definiran je posebni prozirni materijal. Uz to još su definirani i posebni generički materijali koji ne koriste teksture već unaprijed zadane vrijednosti. Ti materijali se koriste kod modela čija je cijela površina prekrivena istim materijalom, poput kromiranih kvaka na vratima i prozorima.

3.3. Osvjetljenje

Simulacija osvjetljenja jedan je od ključnih zadataka računalne grafike, pa je stoga u Unreal engine-u velika pažnja posvećena svijetlima. Izvori svijetla imaju dvije osnovne podijele. Podjela po obliku izvora, i podjela po načinu osvjetljenja. Izvori mogu biti točkasti (*point light*), paralelni (directional light), usmjereni (*spot light*) i svesmjerni (*sky light*). Druga podjela je na statične, nepokretne i pokretne izvore.

Za simulaciju realnog odbijanja svijetla u sceni Unreal Engine 4 koristi mape osvjetljenja (light map). Te mape se grade tijekom izgradnje scene i svaka naknadna promjena osvjetljenja zahtjeva gradnju novih mapa. Postupak računanja tih mapa je vrlo zahtjevan proces jer je potrebno simulirati kompleksne interakcije svijetla. Zrake svijetla se po nekoliko puta odbijaju od površina i sa svakim odbijanjem su slabije i raspršenije. Ta simulacija zna trajati i po nekoliko minuta. U jako velikim scenama i po nekoliko sati. Stoga nije pogodno za upotrebu u stvarnom vremenu. Taj sustav se zato koristi prilikom izgradnje scene te se mape osvjetljenja pohranjuju na disk. Rezultat su vrlo realistično osvjetljenje i sjene na statičnim objektima. Glavna mana takvog pristupa je što tako izračunato osvjetljenje se ne može mijenjati bez da se ponovni cijeli izračun. Odnosno svijetlo mora biti statično. Osim realističnog izgleda velika prednost je što nakon što se svijetlo izračuna, prikazivanje osvjetljenja s diska postaje trivijalno, pa ovaj način osvjetljenja daje najbolje performanse.

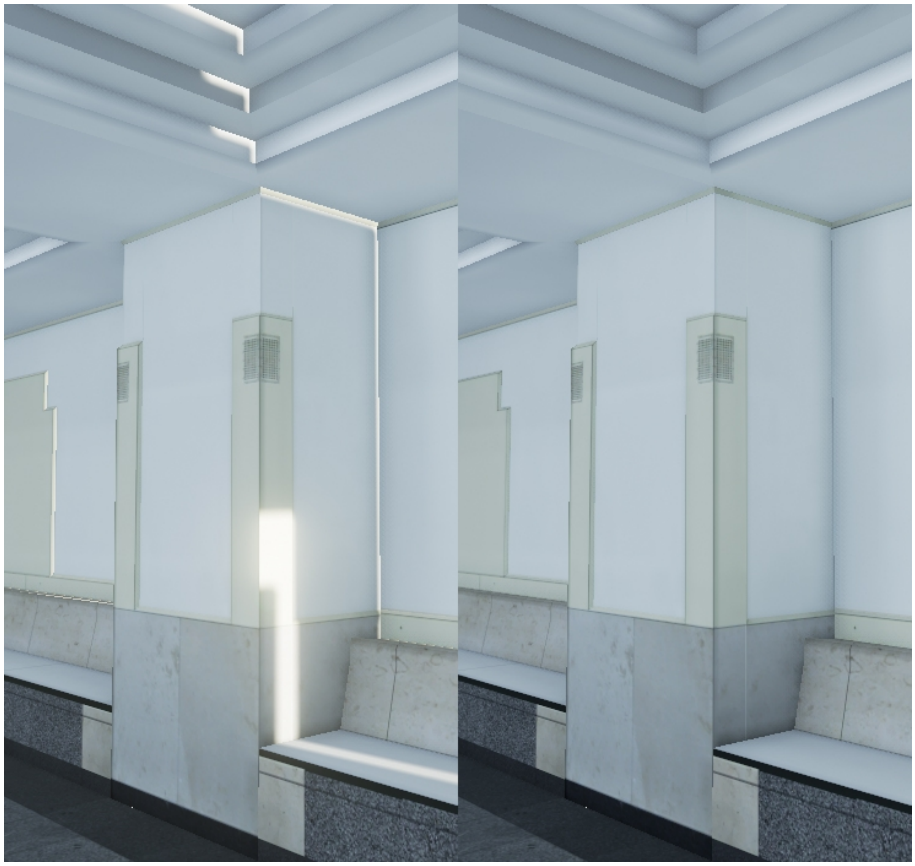
Drugi način osvjetljenja koristi se za svijetla koja su nepokretna. Ona se također ne mogu kretati, ali mogu mijenjati boju i intenzitet. Za takva svijetla se mape osvjetljenja računaju za svaki izvor pojedinačno. No tada problem nastaje kada se više izvora preklapaju i zbog toga je potrebno dosta ograničiti doseg tih svjetala. Unreal Engine trenutno podržava maksimalno preklapanje od 4 izvora, ako se među njima pojavi peti izvor, sustav će automatski ugasiti onaj izvor koji ima najmanji intenzitet. Nepokretna svijetla daju dobar kompromis između realističnog osvjetljenja i kontrole, no žrtvujući fleksibilnost jer se mora paziti na preklapanje izvora.

Posljednji način osvjetljenja su pokretni izvori. Njihovo svijetlo je u potpunosti dinamično i omogućuje mijenjanje parametara u stvarnom vremenu. No ta fleksibilnost čini ovo najsporijim način osvjetljenja.

U sceni su iskorišteni izvori iz svake od navedenih skupina. Kroz prozore su postavljena usmjerena statička svijetla koja simuliraju plavkasto difuzno dnevno svijetlo. Njima se dobivaju sjene u zaklonjenim dijelovima scene.

Direktna svjetlost od sunca simulirana je paralelnim svijetlom. Pošto je u simulaciji moguće mijenjati doba dana potrebno je moći pomicati sunce, pa je to svijetlo je dinamično. Zidne svjetiljke i ostala umjetna rasvjeta koristi nepokretne izvore kako bi se mogla paliti i gasiti tijekom izvođenja. Između njihovih izvora ima dovoljno prostora da se izbjegne preklapanje nepokretnih izvora.

Prilikom osvjetljavanja prostora dinamičkim izvorom primijećena je nedosljednost u osvjetljenju koja je nastala jer su modeli prostora jednostrani i svjetlost "curi" kroz stražnju stranu. Problem je uklonjen izgradnjom dodatnog modela koji obuhvaća čitav prostor scene, osim mjesta gdje bi svjetlost trebala prolaziti, poput prozora i vrata. Time su stražnje strane problematičnih modela zaklonjene od svijetla pa se problem ne pojavljuje. Rezultat je vidljiv na slici 7.

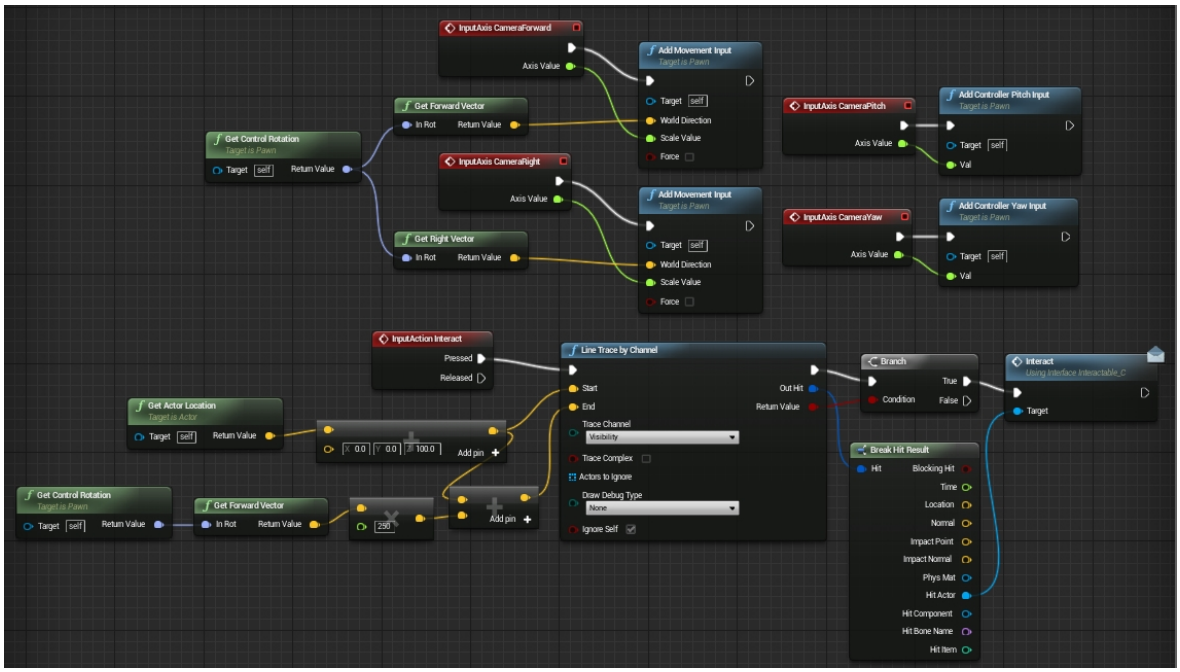


Slika 7. Pogrešno osvjetljenje zbog jednostranih modela prije i nakon ispravljanja

3.5. Kretanje kroz virtualni prostor

U aplikaciji je omogućeno kretanje kroz prostor korištenjem tipkovnice i miša. Virtualna kamera se kreće pomoću vizualnog skriptnog jezika unutar korisničkog aktera. Akter se sastoji od kamere i geometrijskog tijela za sudare u obliku kapsule. Kretanje se postiže dodavanjem vektora kretanja čiji smjer ovisi orijentaciji kamere. Kompletan nacrt za upravljanje kamere i interakciju prikazan je na slici 8.

Ulazni sustav prati stanja tipaka tipkovnice i pomake miše te ih prikazuje pomoću četiri ulazne osi. Koordinatnim osima x i y koje služe za kretanje prostorom upravlja se pomoću tipaka W, A, S i D. Druge dvije osi se kontroliraju pomacima miša i predstavljaju rotaciju kamere. Horizontalni pomak miša rezultira zakretanjem aktera oko z osi (os okomita na pod). Pomak miša naprijed nazad uzrokuje zakretanje kamere oko osi definirane vektorom koji je usmjeren desno od vektora smjera gledanja. To rezultira spuštanjem ili podizanjem pogleda.



Slika 8. Nacrt (*blueprint*) za upravljanje kamerom i interakciju

3.6. Interaktivni elementi

Uz navedene kontrole kretanja implementirane su dodatne tipke za interakciju s okolinom. Sve interakcije su ostvarene istim vizualnim skriptnim jezikom kao i kretanje kamere. Kao i kod svakog vizualnog skriptnog jezika, programiranje se ostvaruje manipulacijom programskih elemenata u grafičkom sučelju, te njihovo povezivanje u stabla, lance i čvorove.

Za razliku od tekstualnog sučelja vizualni jezik omogućuje lakše snalaženje među skriptnim elementima. Povezani elementi daju dobar prikaz uzročno-posljedične veze, no kod kompleksnijih sustava veze znaju biti dosta nepregledne, pogotovo kada se više veza i elemenata križaju unutar malog prostora. Kao što je preglednost i čitljivost koda važna kod tekstualnog programiranja, tako je i pri radu s vizualnim jezikom potrebno paziti na uredno pozicioniranje elemenata i veza. Upotreba vizualnih skriptnih jezika omogućuje brzu iteraciju i eksperimentiranje i vrlo je korisna za korisnike alata koji žele izraditi svoju aplikaciju, no možda nemaju puno iskustva s pisanjem koda.

U aplikaciji je moguće paliti zidne svjetiljke i otvarati vrata klikom na njih lijevom tipkom miša.. Također je moguće ubrzati tijek vremena pritiskom tipke plus na numeričkoj tipkovnici. Ta mogućnost je dodana radi lakše demonstracije dinamičnog osvjetljenja. Pomoću skripte su animirani i zidni satovi.

Za vrata su definirani parametri koji omogućuju da se akter prilagodi po potrebi. Osim što se može birati između dva različita materijala, moguće je odabrati i imaju li vrata u sebi staklo ili drvenu ploču, imaju li kvaku ili ne te da li ih je moguće otvoriti ili su zaključana. Uz to se može i podesiti maksimalni kut otvaranja. Animacija otvaranja i zatvaranja vrata nije unaprijed definirana već je u potpunosti pogonjena skriptom.

3.8. Dodatni efekti

Unreal Engine podržava razne specijalne efekte koji upotpunjuju prikaz scene i neki od tih efekata su korišteni u aplikaciji. Kod gledanja u jaki izvor svijetla na ekranu se javlja bljesak (*lens flare*) uzrokovan optikom leće virtualne kamere. Uz to simulira se i prašina na leći. Ti efekti se ne vide često u stvarnom svijetu, ali su popularni u modernim filmovima i igrama jer čine sliku interesantnijom. Drugi efekt koji proizlazi iz nesavršenosti kamera je zamagljenje brzih pokreta (motion blur).

Ljudskom oku potrebno je određeno vrijeme da se privikne na promjenu u osvjetljenju kada se prelazi iz svijetlog u tamni prostor i obrnuto. Taj efekt je lako zamijetiti prilikom ulaska i izlaska iz dugog tunela. To vrijeme prilagode se simulira dinamičkim mijenjanjem ekspozicije slike.

3.7. Zvukovna podloga

Osim grafike u računalnim simulacijama i igrama veliku važnost imaju i zvukovi. Zbog toga su u scenu su dodane audio komponente koje reproduciraju zvuk. Kao i u stvarnosti, glasnoća pojedinog zvuka ovisi o udaljenosti izvora od kamere. Zvukovi se mogu čuti kod interakcije sa svijetlom i vratima. U sceni se nalazi par zvukova okoline koji upotpunjuju ambijent i daju bolji ugođaj. Kraj južnog izlaza je moguće čuti zvukove prometa, a u hodnicima na sjeveru pjev ptica. Tijekom noći ti zvukovi se utišaju.

4. Rezultati

Za potrebe ovog rada izrađeno je oko 100 tekstura i 85 različitih modela s prosječnim brojem od 150 poligona. Istražene su mogućnosti Unreal 4 engine-a.
Razvijena je aplikacija za prikaz modela interijera. Prosječna brzina izvođenja aplikacije je 40 sličica u sekundi na prijenosnom računalu s i5 procesorom, 4 GB memorije i Nvidia 840M grafičkom karticom.
Kako bi se aplikacija lakše prilagodila prikazivanju na projektorskom zaslonu dodana je mogućnost postavljanja jačine osvjetljenja tipkama * i / na numeričkoj tipkovnici.
Aplikacija je postavljena na github servis i može se preuzeti na sljedećoj poveznici https://github.com/burst149/FerSimulator2015

5. Zaključak

Računalna grafika se koristi u mnogim granama ljudske djelatnosti. Jedna od primjena grafike je prikaz kompleksnih trodimenzionalnih scena. Prikaz takvih scena je koristan za prikaz modela prostora koji postoje u stvarnom svijetu. Uz lako modeliranje i prikazivanje takvih prostora, moderni alati omogućuju i interakciju sa scenom.

Svaki materijal se može definirati s nekoliko parametara opisanih skriptama, to je moguće kombinirati s dosjetljivim korištenjem tekstura kako bi se dobio jako fleksibilan način definiranja materijala.

Pri simulaciji osvjetljenja koristi se znanje iz optike i mnoštvo trikova za optimizaciju algoritama, no niti jedan algoritam nije optimalan za svaku primjenu.

Vizualni skriptni jezik olakšava lako programiranje ponašanja objekata u sceni.

Efekti koji su u stvarnom svijetu nepoželjni, poput nesavršenosti leća, u virtualnim sustavima se modeliraju kako bi scenu učinili realističnijom i interesantnijom.

6. Literatura

1. Blender Foundation, Blender Manual
   https://www.blender.org/manual/

2. Epic Games, Unreal Engine 4 Documentation
   https://docs.unrealengine.com/

3. Static Mesh from Blender
   https://wiki.unrealengine.com/Static_Mesh_from_Blender

4. Jeff Russel, Basic Theory of Physically-Based Rendering
   http://www.marmoset.co/toolbag/learn/pbr-theory

5. Physically-Based Shading Models in Film and Game Production
   http://renderwonk.com/publications/s2010-shading-course/

IZRADA SLOŽENIH MODELA INTERIJERA

SAŽETAK

U ovom radu su opisane tehnike izrade i prikaza virtualnih unutrašnjih prostora. Prvi dio sadrži pregled modeliranja interijera i dodatnih predmeta pomoću alata Blender. Opisan je postupak izrade potrebnih tekstura i priprema modela za daljnje korištenje. U sklopu rada razvijena je i aplikacija za prikaz i interakciju s izrađenim modelom. Drugi dio sadrži opis razvoja te aplikacije pomoću Unreal 4 engine-a. Rad opisuje metode sjenčanja materijala korištenjem fizikalno baziranog prikaza i razmatra nekoliko načina izračuna osvjetljenja. Za kretanje kroz scenu i interakciju s objektima korišten je vizualni skriptni jezik kojeg nudi Unreal Engine 4. U zadnjem dijelu rada opisane su neke dodatne mogućnosti poput specijalnih efekata i reprodukcije zvuka.

KLJUČNE RIJEČI:

računalna grafika, virtualni prostor, 3D modeliranje, Unreal Engine 4, fizikalno temeljen prikaz, vizualni programski jezik

DESIGN OF COMPLEX VIRTUAL INTERIORS

ABSTRACT

This thesis describes the creation and rendering of virtual interior environments. First part contains an overview of interior modeling and creation of additional assets using Blender. It also describes the creation of required textures and model preparation for later use. An application for displaying and interacting with the modeled scene was developed as part of the thesis. Second part describes the development of the application using the Unreal Engine 4. This thesis outlines the shading of materials using the physically based rendering algorithm and examines several lighting calculation methods. A visual scripting language provided by Unreal Engine 4 is used for movement and interaction within the scene. Final part of the thesis describes some additional features, like post processing effects and sound playback.

KEYWORDS:

computer graphics, virtual environment, 3D modeling, Unreal Engine 4, physically based rendering, visual programming language