SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

**DIPLOMSKI RAD br. 1238**

# Upravljanje elementima virtualnog sučelja uz kombinaciju uređaja Leap Motion i Oculus Rift

*Goran Bogovac*

Voditelj: *Prof. dr. sc. Željka Mihajlović*

Zagreb, lipanj 2016

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 10. ožujka 2016.

Predmet: **Računalna grafika**

# DIPLOMSKI ZADATAK br. 1238

Pristupnik: **Goran Bogovac (0023088617)**
Studij: Računarstvo
Profil: Programsko inženjerstvo i informacijski sustavi

Zadatak: **Upravljanje elementima virtualnog sučelja uz kombinaciju uređaja Leap Motion i Oculus Rift**

Opis zadatka:

Proučiti rad uređaja Leap Motion i Oculus Rift DK2. Proučiti mogućnosti njihove integracije u zajedničkoj sceni. Razraditi virtualni okoliš s elementima interaktivnog sučelja kojim se može upravljati uz upotrebu kombinacije dva proučena uređaja. Razraditi primjer komandne ploče i gesti kojima je omogućeno upravljanje navigacijom svemirskog broda. Na različitim primjerima prikazati ostvarene rezultate. Načiniti ocjenu ostvarenih rezultata i implementiranih postupaka.
Izraditi odgovarajući programski proizvod. Koristiti grafički pogon Unreal. 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: 18. ožujka 2016.
Rok za predaju rada: 1. srpnja 2016.

Mentor:

Prof. dr. sc. Željka Mihajlović

Predsjednik odbora za diplomski rad profila:

Prof. dr. sc. Krešimir Fertalj

Djelovođa:

Doc. dr. sc. Igor Mekterović

# Zahvale

Zahvaljujem svojoj obitelji na svim prilikama koje su mi pružili tijekom obrazovanja, svojim prijateljima na pomoći, svojoj mentorici koja je uvijek dostupna te svim profesorima i asistentima koji su mi obogatili znanje kroz studij računarstva odnosno programskog inženjerstva.

# Sadržaj


1. Uvod ..... 1

2. Korišteni uređaji Leap Motion i Oculus Rift ..... 2

2.1 Uređaj Leap Motion ..... 2

2.1.1 Kratki opis i povijest ..... 2

2.1.2 Specifikacije sklopovske opreme ..... 3

2.2 Uređaj Oculus Rift ..... 4

2.2.1 Uvod u virtualnu stvarnost ..... 4

2.2.2 Što je Oculus Rift? ..... 4

2.2.3 Tehničke specifikacije sklopovske opreme ..... 5

3. Unreal 4 razvojna okolina ..... 7

3.1 Sučelje ..... 7

3.1.1 Projekti ..... 7

3.1.2 Urednik ..... 9

3.2 Akteri i komponente ..... 11

3.2.1 Akteri ..... 11

3.2.2 Komponente ..... 13

3.3 Animacije ..... 14

3.3.1 Persona ..... 14

3.3.2 Ključni animacijski pojmovi ..... 15

3.4 Integracija Leap Motiona i Oculus Rifta u Unreal razvojnoj okolini ..... 16

3.4.1 Integracija Leap Motiona ..... 16

3.4.2 Integracija Oculus Rifta ... 17
4. Virtualna sučelja i prikaz scene ... 18
4.1 Prezentacija scene ... 18
4.2 Osnovni mehanizmi iscrtavanja scene ... 19
4.2.1 Geometrijski bazirani sustavi iscrtavanja ... 20
4.2.2 Ne geometrijski sustavi iscrtavanja ... 21
4.3 Interakcija s virtualnim sučeljem ... 22
4.3.1 Manipulacija ... 23
4.3.2 Navigacija ... 25
5. Implementacija ... 26
5.1 Modeliranje scene ... 26
5.1.1 Modeliranje svemirske sfere (okoline) ... 26
5.1.2 Modeliranje svemirskog broda ... 29
5.2 Modeliranje sučelja ... 32
5.2.1 Pomična vrata ... 32
5.2.2 Prateći tekst ... 35
5.2.3 Detektor objekta ... 40
5.3 Integracija uređaja Leap Motion i Oculus Rift ... 43
6. Zaključak ... 45
7. Literatura ... 46
Sažetak ... 48
Abstract ... 49

# 1. Uvod

Otkad se proizvelo prvo računalo (Eniac, 1946.), postojala je potreba za poboljšanje interakcije čovjeka i računala. Interakcija čovjeka i računala iznimno je važna grana računarstva koja se između ostalog bavi proučavanjem kako čovjeku što jednostavnije i vjerodostojnije omogućiti odnosno olakšati upravljanjem sve složenijim i složenijim sustavima. Veliki iskorak u interakciji napravio se uvođenjem grafičkog sučelja kakvog poznajemo danas u osnovnim elementima, a to su prozor, ikone, izbornici, pokazivač (engl. windows, icons, menus, pointer, WIMP). Virtualna sučelja bi upravo mogla biti sljedeći veliki korak u razvoju programske opreme te spomenute interakcije. Trenutno bilježimo golemu uporabu ekrana na dodir na svakom uglu, a jednog dana bi mogli zakoračiti u virtualni svijet. Velika prednost virtualnih sučelja je fizički prostor koji ne postoji tj. virtualni svijet je neograničen i drugačiji od svih dosadašnjih sučelja. Zbog prostorno beskonačnog potencijala, korisnik može izvršavati složenije naredbe i pratiti puno više stvari odjednom jer mu je jednostavno omogućen uvid u situaciju preko jedinstvenog prikaza. Virtualna stvarnost bilježi značajan rast u interesu i proizvodnji što omogućuje stabilnije tržište za područje razvoja virtualnih sučelja. Gotovo sve velike tvrtke za programsku opremu kao što su Microsoft i Sony imaju svoju inačicu uređaja za percipiranje virtualne stvarnosti. Prošlost je pokazala da ne postoji bolji indikator od pravca u kojem svijet tehnologije ide od činjenice da većina velikih tvrtki poveća svoja financijska sredstva u interesno područje. Možemo reći da su postavljeni virtualni temelji. Još uvijek se traži zlatna kombinacija percepcije virtualnog svijeta i korištenja idealnog uređaja za interakciju s tim svijetom. U ovom diplomskom radu pokušat ćemo iskoristi jednu potencijalnu takvu kombinaciju te simulirati jedno jednostavno virtualno sučelje pomoću uređaja Leap Motion i Oculus Rift u Unreal razvojnoj okolini verzije 4. Radi se o jednostavnoj simulaciji malog svemirskog broda gdje korisnik pregledava scenu preko Oculusa, a upravlja brodom pomoću Leap Motiona. Komande su razne geste poput pritiska gumba, pomicanje poluga i sl. Objašnjenje što su spomenuti uređaji je u sljedećem poglavlju.

# 2. Korišteni uređaji Leap Motion i Oculus Rift

U ovom dijelu opisat ćemo uređaje koje ćemo koristiti pri razvoju virtualnog sučelja. Vrlo je bitno razumjeti tehnologiju koja se koristi prije same integracije s razvojnom okolinom te konačno opisom implementacije proizvoda.

## 2.1 Uređaj Leap Motion

### 2.1.1 Kratki opis i povijest

Leap Motion je uređaj za detekciju pokreta ruku kojeg proizvodi istoimena američka tvrtka. Detekcija uključuje: položaj prstiju, šake, podlaktice, te predmeta koji imaju sličan izgled dijelu ljudske ruke (npr. cilindrična tijela kao što su štapići i olovke). Najčešće se koristi u razvoju računalnih igara ili simulacija, no još nije u širokoj upotrebi. Prototip Leap Motiona razvio se 2008. te se kroz sljedeće tri godine uložilo u njega oko 1.3M$, a u još dvije godine nevjerojatnih 30M$ [1]. Dana 21.5.2012 uređaj je službeno predstavljen javnosti, a 2014. godine izašla je v2 inačica. Na sljedećoj slici je izgled uređaja.


*Slika 1 Leap Motion uređaj*

Danas Leap Motion kompanija okreće se više pravcu virtualne i proširene stvarnosti što svjedoči izdavanje nove programske opreme nazvane Orion [2], čija je namjena isključivo za virtualnu stvarnost. Cijena Leap Motion uređaja kreće se oko 45$. Recenzenti uređaja ga više opisuju kao potencijalna tehnologija budućnosti, nego

trenutna, unatoč relativno pristupačnoj cijeni. Jedan od razloga takvih recenzija je to što Leap Motion ne može zamijeniti sučelje miša i tipkovnice odnosno ekrana osjetljivog na dodir čija je primjena vrlo zastupljena. Za potrebe razvoja programske opreme, dokumentacija je dostupna u programskim jezicima: JavaScript, C++, C#, Python, Java, i Objective-C.

### 2.1.2 Specifikacije sklopovske opreme

Leap Motion je mali uređaj dimenzija 70mm x 30mm x 11mm u duljini, širini i visini te težak je svega 32 grama [3]. Spomenute brojke čine ga lako prenosivim i praktičnim. Dizajn je namijenjen na način da se uređaj koristi položen uspravno na tvrdoj podlozi. Na dnu uređaja nalazi se crni panel s logom tvrtke dok je rub provučen aluminijskom vrpcom. Za pregled okoline, uređaj koristi tri infracrvene LED svjetiljke te dvije infracrvene monokromatske kamere [4]. Radijus detekcije ruku je otprilike jedan metar. LED svjetiljke generiraju infracrvenu svjetlost a kamere mogu proizvesti dvjesto slika u sekundi (engl. 200 frames per second, FPS). Na sljedećoj slici prikazani su slojevi sklopovske opreme uređaja [5].

*Slika 2 Slojevi sklopovske opreme Leap Motiona*

Instalacija uređaja vrlo je jednostavna. Sve što trebate je USB 3.0 kabel uključiti u svoje računalo, pustiti Leap Motion da detektira vaše ruke i spremni ste za korištenje.

## 2.2 Uređaj Oculus Rift

### 2.2.1 Uvod u virtualnu stvarnost

Prije nego što objasnimo što je Oculus Rift, potrebno je objasniti što je virtualna stvarnost. Virtualna stvarnost je računalno simulirano okruženje u kojem korisnik obavlja interakciju [6]. Korisnik u potpunosti nema doticaja sa stvarnošću, a s virtualnom stvarnošću komunicira preko sučelja koje mu je omogućeno. Međutim, najvažnije sučelje je ono koje mu omogućuje prikaz virtualne scene, a za to su najprikladniji naglavni ekrani (engl. Head-mounted display, HMD). Virtualna scena je „svijet“ u kojemu se odvija virtualna stvarnost. Ideja je simulirati korisnikovu prisutnost u tom „svijetu“ na različite načine. Virtualna stvarnost najviše se koristi u svijetu računalnih igara, vojnih simulacija i medicine. Računalne igre još nisu doživjele svoj procvat u virtualnoj stvarnosti jer za simulaciju virtualne scene potreban je nešto jača sklopovska oprema što prosječnom kupcu nije jednostavno za priuštiti na financijskom planu. Kao što je već spomenuto, najčešće korištena sklopovska oprema upravo je naglavni ekran, što je ujedno i Oculus Rift.

### 2.2.2 Što je Oculus Rift?

Oculus Rift je uređaj za prikaz virtualne stvarnosti preko naglavnog ekrana. Oculus Rift može detektirati pokret glave korisnika te time prikazuje scenu iz različitog kuta gledanja. Uređaj je proizvela kompanija Oculus VR te je prva tvrtka koja je uspješno lansirala proizvod ciljan isključivo za virtualnu stvarnost [7]. Tvrtku je 2014. godine kupio Facebook za velike dvije milijarde dolara. Uređaj je pušten u masovnu prodaju tek 28.3.2016 iako se dvije godine ranije mogao nabaviti bez većih problema. Također su izdane dvije verzije oprema za razvoj (engl. development kit, DK) kako bi se omogućilo programerima upoznavanje na vrijeme s Oculusom i njegovim mogućnostima. Na sljedećoj slici prikazan je uređaj.

*Slika 3 Oculus Rift*

### 2.2.3 Tehničke specifikacije sklopovske opreme

Oculus Rift ima OLED prikaz slike (engl. organic light-emitting diode) čija je rezolucija 1080x1200 po oku promatrača. Frekvencija osvježavanja slike je 90Hz s 110° kutom pregleda. U uređaju su integrirani zvučnici s 3D zvučnim efektom, praćenjem orijentacije i položaja. Sustav (osim zvučnog) za praćenje položaja zove se „Constellation“ i vrši ga USB infracrveni LED senzor koji se najčešće nalazi ispod stola korisnika (ili negdje na sličnoj poziciji) koji cijelu prostoriju obasja infracrvenim i LED svjetlom te time omogućuje stvaranje 3D scene za korisnika koji se može prividno kretati po njoj. Na sljedećoj slici prikazana je raslojenost sklopovske opreme Oculus Rift DK1 verzije.

*Slika 4 Oculus Rift DK1*

Za instalaciju Oculusa na računalo potrebno je imati sljedeće specifikacije sklopovske opreme: grafička kartica NVIDIA GTX 970 (ili neki ekvivalent), CPU ekvivalentan modelu Intel i5 i5-4590, 8GB RAM ili više, dva USB 3.0 porta i jedan USB 2.0 port, HDMI 1.3 video prikaz te instaliran Windows operacijski sustav (najranija verzija Windows 7 SP1 64-bit). Iz priloženog možemo zaključiti ono što smo već ranije spomenuli, a to je da VR nije zaživio još upravo zbog velikih zahtjeva sklopovske opreme. Postoje još i dodatni alati koje možete nabaviti uz Oculus kao što su Oculus gamepad i Oculus Touch. Oculus Gamepad je običan upravljački uređaj (engl. joystick ili gamepad) koji je samo programski kompatibilan za korištenje s Oculus Riftom zbog činjenice da dosta igara danas se igra koristeći neku vrstu gamepada. Upravo danas, zbog partnerstva Oculusa s Microsoftom (tvrtkom koja proizvodi Windows OS između ostalog), svaka kupovina Oculusa dolazi i s XBOX One bežičnim kontrolerom. Oculus Touch su zapravo dva kontrolera (jedan za svaku ruku) pomoću kojih se vrši detekcija pokreta unutar virtualne stvarnosti. Uređaje prati spomenuti Constellation sistem, tako da se pozicioniranje korisnika precizno bilježi i omogući prikaz ruku unutar stvarnosti. Naravno, uređaji nemaju precizno praćenje pokreta prstiju kao Leap Motion, no zasad su glavni alat kada je potrebno implementirati pokret ruku u kombinaciji s Oculusom.

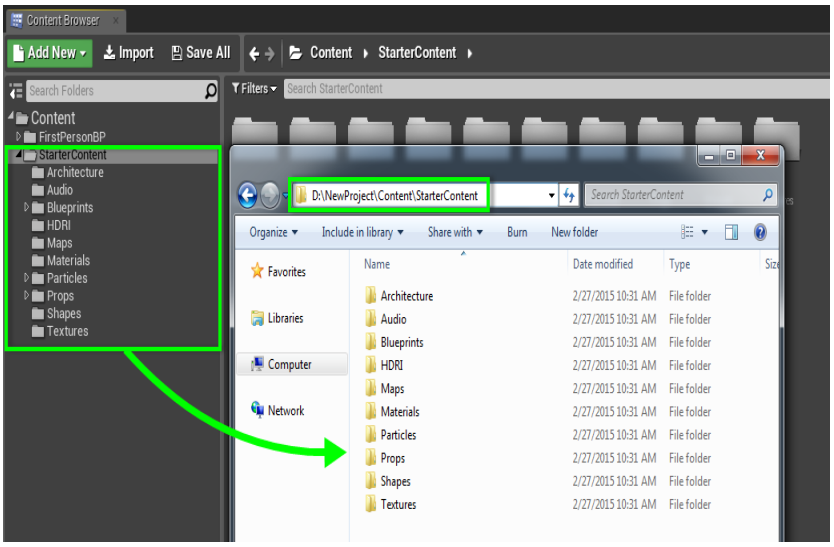
# 3. Unreal 4 razvojna okolina

Unreal 4 je kompletan paket alata koji se može koristiti za razvoj filmova, animacija, obrazovanje, vizualizaciju, arhitekturu i igri. U ovom radu bavit ćemo se aspektom razvoja igara. Sučelje okoline je robusno i zahtijeva osnovno poznavanje objektno orijentirane programske paradigme (poželjan jezik C++). Unreal okolina donosi čitav spektar grafičkih elemenata na korištenje, koji se mogu grupirati u veće cjeline i animirati. Animacije se kreiraju i povezuju pomoću Persona sučelja na interaktivan način. Sav rad pohranjuje se u projekt i omogućuje se rad na više projekata. U sljedećim poglavljima obradit ćemo osnovne dijelove Unreal 4 razvojne okoline, a to su: sučelje, akteri i komponente te grafika i animacije. Unreal 4 podržava napredne DirectX11 i DirectX12 funkcionalnosti iscrtavanja kao što su potpuna HDR refleksija scene, tisuću dinamičkih svjetlosnih snopova po sceni, sjenčanje temeljeno na fizici materijala itd. Ostale odlike verzije 4 su: kaskadni vizualni efekti, novi cjevovod materijala, skriptiranje i ispravljanje grešaka (engl. debugging) preko nacrta (bez pisanja linija koda, koristeći sučelje), preglednik sadržaja, „Persona“ animacije (skup alata za grafičke animacije i dizajn), „Matinee“ kinematika (skup alata za modeliranje toka scene), post-procesni efekti, puni pristup izvornom kodu, trenutni pregled igre, predefinirana umjetna inteligencija, zvukovni cjevovod i mogućnost integracije s međuslojem. U sljedećem poglavlju ćemo opisati osnovni rad s Unreal razvojnom okolinom.

## 3.1 Sučelje

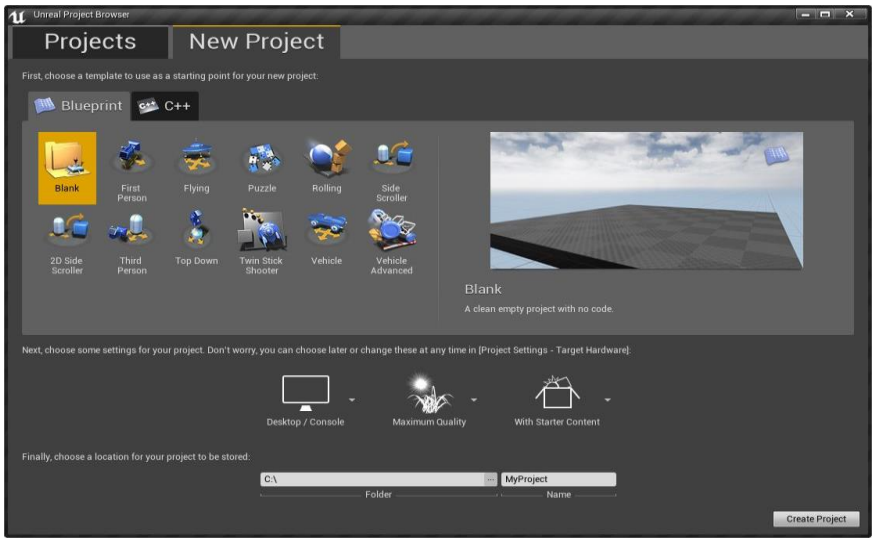
### 3.1.1 Projekti

Projekt je skup koda i sadržaja koji čine sve potrebne komponente za razvoj jedne igre, a pohranjuje se u obliku datotečnog sustava. Na slici nalazi se primjer kako izgleda u pregledniku sadržaja projekt koji je pohranjen na disku. Postoje dva predefinirana nastavka datoteka za Unreal projekte a to su .uproject koji služi za dodavanje, otvaranje i spremanje projekta te Project koji sadrži sve datoteke i mape vezane uz projekt.




*Slika 5 Preglednik sadržaja za projekt*

Unreal omogućuje rad na više projekata istovremeno pomoću preglednika projekata kao što je prikazano na sljedećoj slici.




*Slika 6 Preglednik projekata*

Prilikom prvog pokretanja okoline, pojavljuje se spomenuti preglednik. U pregledniku možete otvoriti ili stvoriti projekt. Ukoliko želite stvoriti novi projekt, možete odabrati prazni projekt gdje počinjete sve „od nule“ ili početne predloške koji se mogu podijeliti na C++ i nacrtne predloške. Razlika između predložaka je u inicijalnom okviru razvoja tj. za C++ se generiraju svi predlošci predviđeni za kodiranje u tom jeziku dok

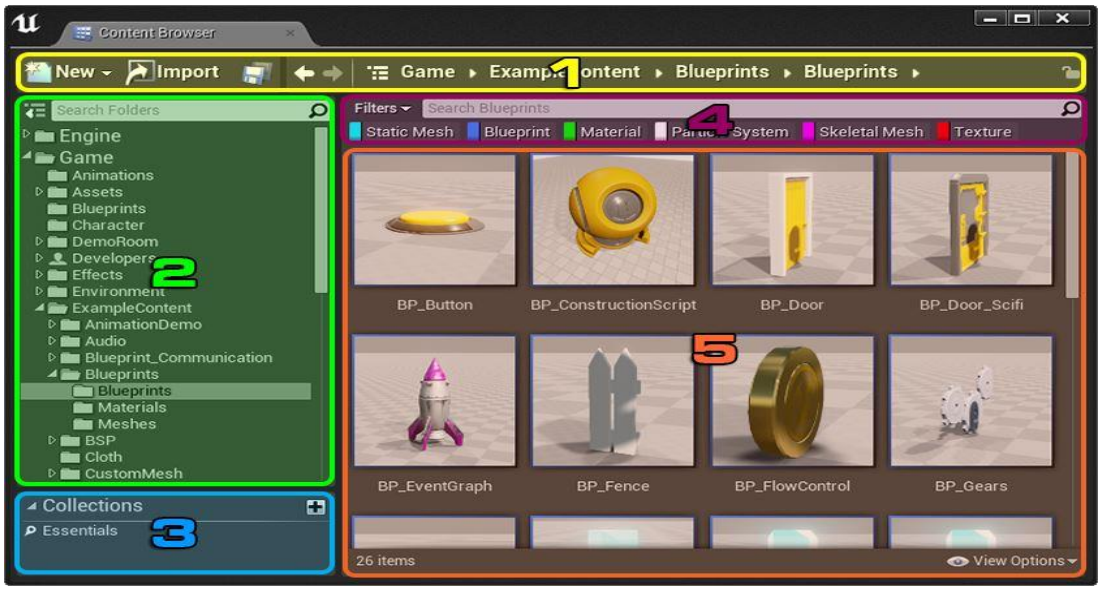
za nacrtne predloške se generiraju nacrti bez potrebe pisanja koda. To ne znači da se ne mogu dodavati C++ datoteke u nacrtnom predlošku i obrnuto, samo su drukčije inicijalizirani.

### 3.1.2 Urednik

Neki dijelovi urednika vidljivi su odmah, neki se pojavljuju prilikom označavanja određenih tipova objekata u sceni, a neki prilikom mijenjanja određenih vrsti svojstava. Ukoliko bi se sve komponente prikazivale odjednom, urednik bi bio nepregledan. Urednik se dijeli na četiri osnovna dijela: preglednik projekata (engl. Project Browser), preglednik sadržaja (engl. Content Browser), preglednik klasa (engl. Class Viewer) i globalni birač komponenti (engl. Global Asset Picker). Preglednik projekata objašnjen je u prethodnom potpoglavlju.

#### *3.1.2.1 Preglednik sadržaja*

Preglednik sadržaja primarno služi za stvaranje, dodavanje, organiziranje, pregled i mijenjanje komponenti sadržaja. Ukratko, ovaj dio sučelja može ostvariti interakciju i pretraživati sve komponente igre. Na slici označeni su njegovi dijelovi.



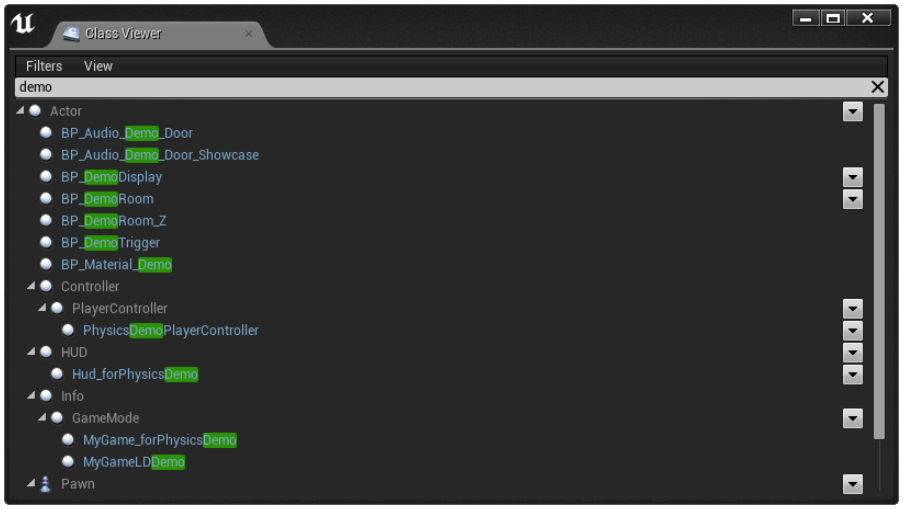

*Slika 7 Preglednik sadržaja*

Preglednik sadržaja sastoji se od:

1. Navigacijske trake (engl. Navigation Bar) – omogućuje navigaciju između mapa komponenti
2. Pogleda izvora (engl. Sources View) – sadrži sve mape vezane uz projekt
3. Ploča kolekcija (engl. Collections Panel) – omogućava brzi pristup svim kolekcijama
4. Područja upravljanja komponentama (engl. Asset Management Area) – omogućuje niz funkcionalnosti rada nad komponentama kao što su dodavanje, spremanje, pretraživanje itd.
5. Pogled komponenti (engl. Asset View) – prikazuje sve komponente unutar jedne mape u obliku mreže ili liste

#### *3.1.2.2 Preglednik klasa*

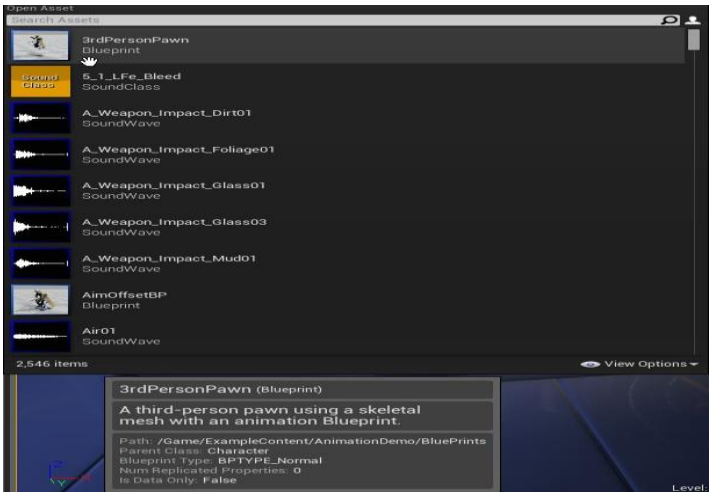
Preglednik klasa omogućava korisniku hijerarhijski uvid u klase koje se koriste u uredniku. Na slici prikazan je osnovni izgled preglednika. Najvažnije je primijetiti odnos roditelj-dijete u stablu klasa čija preciznost i pouzdanost prikaza je iznimno bitna.




*Slika 8 Preglednik klasa*

#### *3.1.2.3 Globalni birač komponenti*

Globalni birač komponenti omogućuje brzi pristup svim komponentama za razvoj igre. Tu komponente nisu prikazane hijerarhijski koje se odnose na vašu igru, već sve komponente unutar razvojne okoline. Na slici prikazano je kako to izgleda. Prozor globalnog birača zatvara se prilikom odabira komponente.




*Slika 9 Globalni birač komponenti*

## 3.2 Akteri i komponente

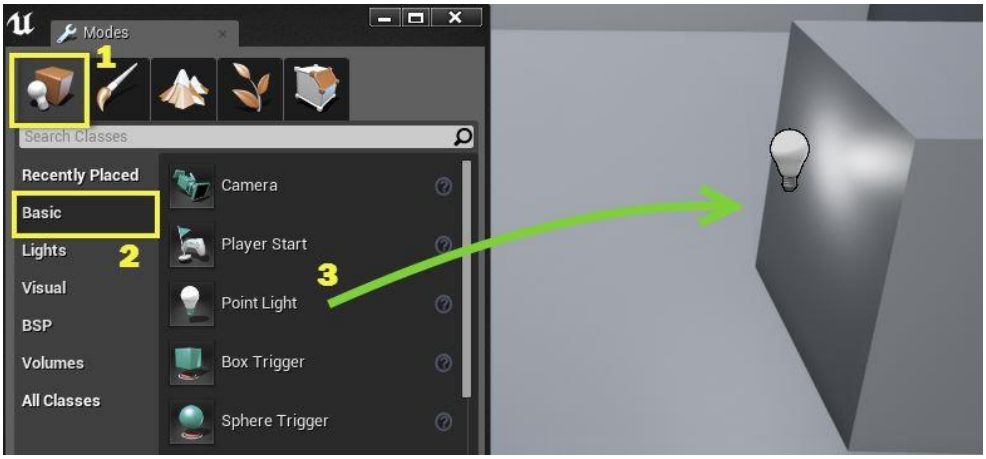
### 3.2.1 Akteri

Akter (engl. Actor) je naziv za bilo koji objekt u Unreal okolini koji se koristi u sceni igre. To je zapravo generička klasa koja podržava 3D transformacije kao što su translacija, rotacija i skaliranje. Postoji nekoliko tipova aktera, a neki od njih su: StaticMeshActor, CameraActor, i PlayerStartActor.

#### *3.2.1.1 Dodavanje i odabir aktera u scenu*

Postoje tri načina za dodati aktera u scenu: pomoću mjesnog načina rada (engl. Place Mode), pomoću preglednika konteksta (engl. Context Menu) i pomoću preglednika klasa (engl. Class View). U prvom načinu jednostavno povučete mišem objekt koji želite u sceni kao što je prikazano na slici. Drugi način je da odabere prvo aktera u pregledniku sadržaja. Kada ste ga odabrali, desni klik na njega i otvorit će

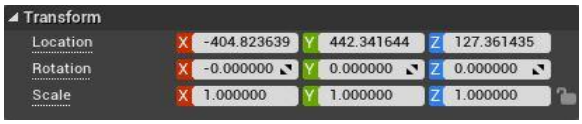
vam se preglednik konteksta i tu odaberete „Dodaj aktera“ (engl. add actor). Treći način je prvo da dođete do željenog aktera u pregledniku klasa te ga povučete mišem na scenu. Vrlo je bitno omogućiti brzu i jednostavno selekciju aktera u sceni. Najjednostavniji način je obični lijevi klik mišem, ali ovakvim načinom je otežan odabir aktera koji su prostorno raspršeni po sceni. Zato još postoji odabir pomoću nacrta scene (engl. World Outliner) i nadstrešnice (engl. Marquee). Postoje još i napredno označavanje gdje npr. desnim klikom na nekog aktera možemo odabrati opciju selekcije svih aktera istog tipa. Akteri se mogu grupirati u predefinirane grupe te obrađivati kao takve na toj razini. Grupiranje u Unreal okolini nije destruktivno što znači da je neograničen broj grupa u koju možete dodati jednog ili više aktera.




*Slika 10 Dodavanje aktera u scenu*

### *3.2.1.2 Transformacija aktera*

Aktere možemo translatirati, rotirati, skalirati ručno ili na interaktivan način. Ručne transformacije obavljamo pomoću ploče za transformaciju koja za tri spomenute transformacije nudi mogućnost mijenjanja parametra po x, y, z koordinati, prikazano na slici. Treba još naglasiti da parametri transformacije djece aktera su relativni s obzirom na poziciju roditelja.

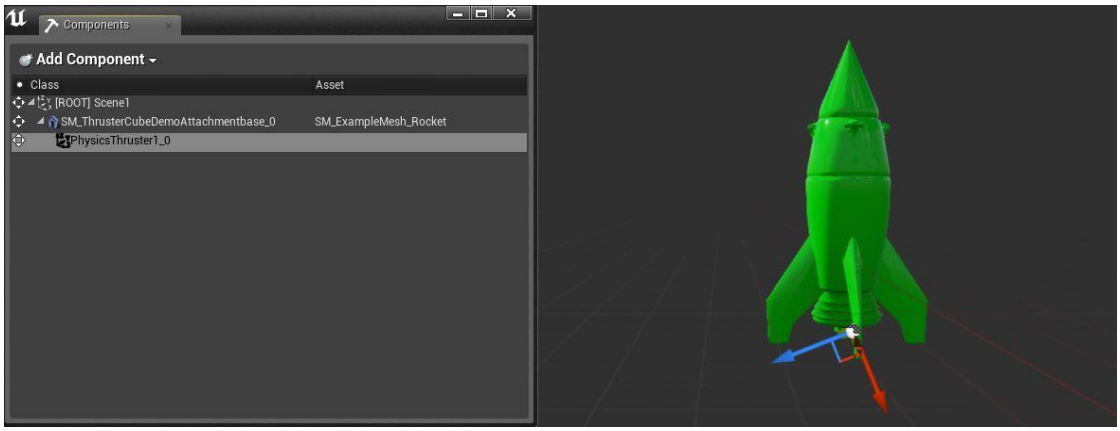



*Slika 11 Ručna transformacija*

Interaktivan način transformiranja aktera uključuje korištenje vizualnog alata (engl. widget) za translaciju, rotaciju i skaliranje. Prvo odaberete željenu transformaciju na alatnoj traci, te pomoću alata pomičete mišem do željenog rezultata. Kod interaktivnog načina rada moguće je postaviti i referentni koordinatni sustav (lokalni ili globalni) nad kojim ćete raditi transformacije. Ovo omogućuje veliku fleksibilnost pri postizanju željenog cilja izgleda aktera.

### 3.2.2 Komponente

Ovdje govorimo o komponentama kao dijelovima aktera, ne kao komponentama igre koji generalno mogu biti bilo kakvi objekti (engl. Asset) odnosno akteri. Komponenta u ovom smislu je zapravo dio funkcionalnosti koja se može vezati na aktera. Cilj je pomoću komponenti dobiti smislenu cjelinu za aktera. Npr. ako je akter automobil, onda bi njegove komponente bile: pedale, volan, olupina, motor, zvuk motora (bilo kakvi zvukovni ili svjetlosni elementi također spadaju pod komponente), kotači, ručna itd. Kad god se akter instancira u sceni, sve komponente vezane uz njega se također instanciraju. Postoji nešto manje od dvadeset tipova komponenti, neke od njih su: AI, Audio, Camera, Light, Movement, Navigation… Na slici je primjer komponente fizike (engl. Physics component).




*Slika 12 Komponenta fizike*

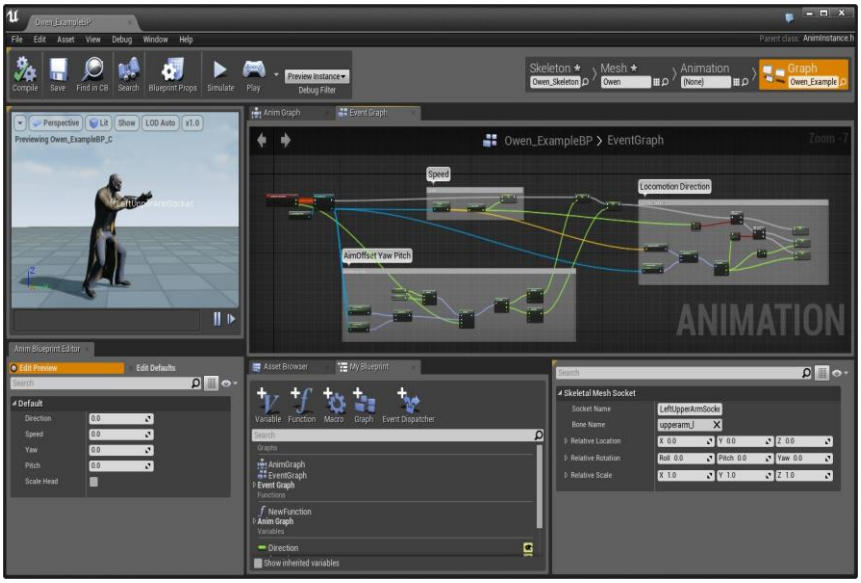
## 3.3 Animacije

Do sada smo pokazali osnovne funkcionalnosti upravljanja objekata u sceni, no još te objekte treba i animirati. Sustav animacija u Unreal okolini omogućuje visok stupanj kontrole aktera tako što koristi kombinaciju deformacije mreže kostura i vrhova. Kostur (engl. Skeleton) je hijerarhija lokacija kostiju i rotacija potrebnih za deformaciju mreže kostura. Kosturi su apstraktno razdvojeni od mreže kostura u svoje posebne kompon

ente (asset). To znači da se animacije primjenjuju na kostur, ne na mrežu kostura i upravo time se omogućuje višestruko dijeljenje istih animacija na razne mreže kostura.

### 3.3.1 Persona

Persona je skup alata za upravljanje animacijama u Unreal okolini. Gotovo sav posao oko animacija obavit će se u Personi. Persona se otvara pomoću preglednika sadržaja i sastoji se od četiri načina rada: kostura (engl. Skeleton) , mreže (engl. Mesh), animacije (engl. Animation) i grafa (engl. Graph). Svaki način rada odgovara specifičnom zadatku kojeg želite izvesti u sučelju. Na slici je prikazano Persona sučelje.




*Slika 13 Persona sučelje*

### 3.3.2 Ključni animacijski pojmovi

Animacijska sekvenca (engl. Animation Sequence) je jedinka animacijske komponente (asset) koja se može izvršiti na mreži kostura. Jedna ili više animacija čine jednu animacijsku sekvencu. Animacija može biti apsolutna ili aditivna u bilo kojem trenutku. Animacije se povezuju pomoću miješajućih čvorova (engl. blend nodes) i prostora (engl. blend spaces). Npr. animaciju za trčanje možete povezati s animacijom ciljanja puške i dobiti izlaznu animaciju kao kombinaciju tih dviju animacija. Time sustav omogućuje ponovno korištenje istih animacijskih sekvenci u više slučajeva.

Animacijska poza (engl. Animation Pose) je jedan uhvaćeni trenutak kostura sa svojom pozicijom i rotacijom svih njegovih kostiju, kao da ste pauzirali video u nekom trenutku. Neke animacije definirane su samo kao poze i služe kao predefinirano stanje kostura pod određenim uvjetima npr. kada igrač cilja s puškom, animacijska poza bi bila držanje puške. Poze mogu biti u lokalnom prostoru ili prostoru komponente. Lokalni prostor podrazumijeva transformaciju kosti s obzirom na samu kost, dok prostor komponente podrazumijeva transformaciju kosti u odnosu na mrežu kostura.

Meta oblikovanja (engl. Morph Target) je snimak koordinata vrhova za određenu mrežu u nekoj deformaciji. Npr. ako imate model čovjeka, njegove izraze lica možete izobličiti te taj oblik spremiti kao metu oblikovanja. Kasnije, tu metu oblikovanja možete iskoristiti da vaš čovjek promijeni izraz lica. Mete se mogu unijeti preko datoteke tipa FBX, a ubacuju se unutar animacijske sekvence. Ovo omogućuje vrlo jednostavno dodavanje složenih Morph Target animacija u Unreal okolini, jer možete imati proizvoljan broj meta u jednoj animaciji.

Animacijski nacrt (engl. Animation BluePrint) je specijalizirani nacrt čiji grafovi kontroliraju animaciju neke mreže kostura. Nacrt može stapati animacije, kontrolirati kosti kostura i generirati pozu za svaki okvir u iscrtavanju slike. Kontrolor dirigira svome pijunu (engl. Pawn) pokrete na temelju ulaznih događaja korisnika ili okruženja u toku igre. Svaki pijun ima svoju komponentu mreže kostura koja referencira mrežu kostura za animiranje i sadrži svoju instancu animacijskog nacrta.

Nacrt pristupa svojstvima pijuna pomoću dvaju grafova: EventGraph i AnimGraph. EventGraph služi za ažuriranje i računanje vrijednosti potrebnih za izvođenje animacije (npr. brzina i smjer). AnimGraph služi za dobivanje konačne animacijske poze za mrežu kostura u trenutnom okviru.

Prostori stapanja (engl. Blend Spaces) su posebne komponente (assets) koje se koriste u AnimGraph-u za stapanje animacija na temelju ulaznih podataka. Cilj je smanjiti potrebu za stvaranje ručno napisanih čvorova za obavljanje stapanja pod određenim uvjetima. Prostor stapanja je generički po prirodi i višestruko uporabljiv.

## 3.4 Integracija Leap Motiona i Oculus Rifta u Unreal razvojnoj okolini

U sljedećem dijelu opisat ćemo postupak integracije uređaja Leap Motion i Oculus Rift u Unreal razvojnoj okolini, odnosno što je sve potrebno napraviti i podesiti da bi se moglo krenuti u razvoj konkretnog proizvoda.

### 3.4.1 Integracija Leap Motiona

Potrebna okolina za Leap Motion dodaje se preko programskog dodatka priključka (engl. plugin) za Unreal 4 [8]. Taj dodatak dolazi s nekoliko predefiniranih nacrta (sjetite se što su nacrti objašnjeni u Unreal poglavlju) kojima se može pristupiti bilo preko sučelja ili koda u C++ programskom jeziku. Za bilo koji postojeći nacrt kojeg Leap Motion nudi, možete ga proširiti svojim funkcionalnostima dodavajući LeapMotionEvent sučelje pa LeapController komponentu u taj nacrt [9]. Dodatak nije ništa drugo nego integracija Leap programske opreme za razvoj (engl. software development kit, SDK) u Unreal okolinu. Dokumentacija za Leap aplikacijsko programabilno sučelje (engl. application programmable interface, API) dostupna je na Internet stranici Leap Motion tvrtke. Od Unreal verzije 4.11 Leap dodatak je već dostupan zajedno s instalacijom okoline. Sljedeći postupak opisan je upravo za tu verziju ili novije:

1. Otvorite ili stvorite novi projekt

2. Na izbornoj traci kliknite Windows -> Plugins. Pod kategorijom Input Devices trebao bi vam biti ponuđen Leap Motion.
3. Označite Enabled te pokrenite ponovo Unreal okolinu.
4. Dodatak bi trebao biti sada spreman za korištenje.

Postupak kako koristiti Leap Motion u Unreal okolini opisat će se u poglavlju implementacije.

### 3.4.2 Integracija Oculus Rifta

Skoro identičan postupak vrijedi za Oculus Rift kod integracije u Unreal razvojnoj okolini. Naime, postoji plugin koji je već dostupan unutar instalacije Unreal 4 [10]. Čak je i već omogućen u predefiniranim postavkama tako da je potrebno samo provjeriti detektira li okolina ispravno uređaj, a to se vrši na sljedeći način:

1. Pokrenite neku scenu ili igru.
2. Otvorite konzolu.
3. Upišite „showlog“.
4. Upišite „ovrversion“.
5. Verzija i datum biblioteke LibOVR bi se trebala ispisati u otvorenoj konzoli.

Ukoliko vam se ne ispiše ono što bi trebalo, provjerite je li vam Oculus uređaj priključen ispravno u računalo i imate li posljednju verziju programskog pogona (engl. drivera) instalirano na vašem računalu.

Konačno, sve je spremno za implementaciju virtualnog sučelja u Unreal razvojnoj okolini koristeći uređaje Oculus Rift i Leap Motion. Važno bi bilo napomenuti da bilo kakva dugoročna upotreba Oculus Rift uređaja na osobi može izazvati takozvani simulacijsku mučninu. Zbog postoje i službeni naputci koji pomažu programerima kada razvijaju aplikaciju za Oculus Rift [11].

# 4. Virtualna sučelja i prikaz scene

Kako bi u potpunosti razumjeli implementaciju, potrebno je još pojasniti kako se ostvaruju, dizajniraju i modeliraju. U samoj definiciji virtualne stvarnosti krije se mnoštvo mogućnosti za dizajniranje i modeliranje virtualnih sučelja, a u ovom radu prvenstveno se fokusiramo na interakciju pomoću opisanih uređaja Oculus Rift i Leap Motion, uz koje postoji mnoštvo drugih uređaja za interakciju čovjeka s virtualnim svijetom, odnosno njihovu implementaciju programske opreme. Za svaku implementaciju postoje određeni principi kojih bi se bilo poželjno pridržavati pri izradi programske opreme za podršku korištenja virtualnog sučelja ovisno o vrstama uređaja.

## 4.1 Prezentacija scene

Počnimo od tvrdnje da mi kao proizvođač, programer, dizajner itd. želimo predočiti korisniku našu ideju interakcije i što želimo od korisnika da napravi s danim elementima sučelja. Ispravno mapiranje ideje na prezentacijski objekt uvelike igra ulogu u razlici u kvaliteti sučelja kojeg pružate korisniku. Možemo to gledati kao da vi kao programer posjedujete informaciju koju želite prenijeti osobi te treba iskoristiti ispravan medij prijenosa inače se poruka neće uspješno prenijeti ili će se krivo protumačiti.

Postoji kvalitativna i kvantitativna prezentacija informacija. Kod kvantitativnih informacija je bitno korisniku prenijeti što više informacija u nekom smislenom formatu (npr. graf), dok je za kvalitativne informacije poanta što više utjecati na korisnika, dakle izrazito naglasiti podatke koje prezentirate. Na sljedećoj slici je primjer gumba iz vrlo popularne video igre Portal.

*Slika 14 Portal button*

Većina igrača je lako zaključilo da se radi o gumbu kojeg treba stisnuti, no što je zapravo dovelo ljude do takvog zaključka? Ispravna prezentacija scene, informacije, ideje programera da je to zapravo gumb, a ne nešto drugo u sceni koja se prezentira korisniku.

Time dolazimo do engleskog pojma *verisimilitude* što bi u prijevodu značilo sličnost s istinom. Naime, *verisimilitude* opisuje koliko je programer uspio prenijeti informaciju korisniku kao istinitu. Naravno, ne teži svaka aplikacija realističnosti prezentacije scene te za njih nije potrebno pridržavanju takvog pravila.

## 4.2 Osnovni mehanizmi iscrtavanja scene

Računalna grafika je područje računarstva koje se bavi postupcima iscrtavanja scene (engl. render). U pozadini računalne grafike krije se mnoštvo algoritama čiji se rad temelji na složenim matematičkim operacijama s matricama jer se pokazalo da pomoću višedimenzionalnih matrica dosta se grafičkih objekata može relativno jednostavno spremiti u memoriju računala. Ovdje ćemo spomenuti neke od osnovnih grafičkih elementa koji služe pri prikazu objekata u sceni.

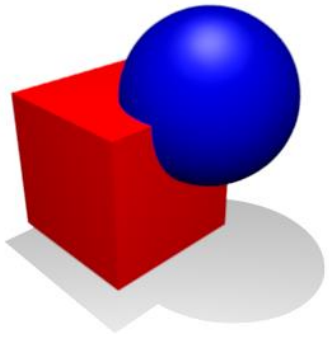
### 4.2.1 Geometrijski bazirani sustavi iscrtavanja

Tri najčešće korištenja geometrijska sustava su: poligonski, ne uniformni racionalni B splajn (engl. non uniform rational B spline, NURBS) te konstrukcijsko čvrsta geometrija (engl. constructive solid geometry, CSG).

Poligoni su planarni geometrijski likovi definiranu pomoću niza linijskih segmenta najčešće tri ili četiri. Poligonski model vrlo je jednostavan. Ideja je svaki složeniji objekt prikazati preko mreže međusobno povezanih poligona. Što više poligona se koristi, slika je preciznija, ali i performanse su lošije tako da uvijek postoji određeni kompromis kvalitete slike i brzine iscrtavanja. Mnoge današnje grafičke sklopovske komponente zbog široke uporabe koriste upravo ovu metodu iscrtavanja.

NURBS su parametarski definirani geometrijski oblici koji se najčešće koriste za prikaz krivuljastih objekata (npr. automobil). Prednost im jednostavno podešavanje parametara i podrška sklopovske opreme. Sastoje se od kontrolnih točaka koji određuju smjer krivulje te vektora uzlova koji definiraju gdje i kako kontrolne točke utječu na krivulju.

CSG objekti su geometrijski oblici nastali zbrajanjem i oduzimanjem osnovnih geometrijskih oblika poput sfere, prizme, piramide, kocke itd. Jednostavni geometrijski oblici koji u polazni gradivni materijali za model zovu se primitivi (engl. primitives). Njegova definicija zapravo je najveća prednost odnosno razlog zašto je popularan. CSG se također primjenjuje u Unreal razvojnoj okolini. Na sljedećoj slici je prikazan primjer modeliranja pomoću CSG-a.



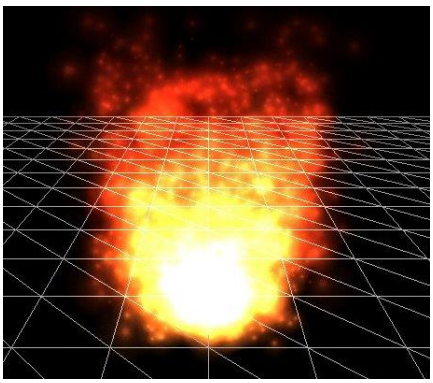
*Slika 15 CSG Unija*

### 4.2.2 Ne geometrijski sustavi iscrtavanja

Površine su jedne od primjera gdje iscrtavanje pomoću nekih od spomenutih geometrijskih oblika nije učinkovito zbog toga što se prikazuju kao čvrsti, transparentni objekti. Transparentnost predstavlja problem geometrijskom načinu iscrtavanja zbog materijala različitih gustoća koji se mogu nalaziti na površini (npr. pećnica u kojoj gori vatra). Ovdje razmatramo prostorni sustav te sustav čestica.

Prostorni ili volumni sustav je model iscrtavanja gdje se pomoću jedne ili više dvodimenzionalnih slika prikazuje 3D objekt u sceni. Takve slike zovu se teksture. Prikladno ga je koristiti kod poluprozirnih objekata čije boje i gustoća dosta variraju po površini. Prostorno iscrtavanje najčešće se ostvaruje pomoću priljeva zraka (engl. ray casting). *Ray-casting* definira zrake svjetlosti koje se odašilju iz izvora. Zrake svjetlosti se ponašaju u skladu s zakonima fizike te se reflektiraju, propuštaju ovisno o vrsti materijala na površini. Time dobijemo realističnu sliku složenih objekata. Problem priljeva zraka je složenost u izračunu pa time i zahtjevnost sklopovlja.

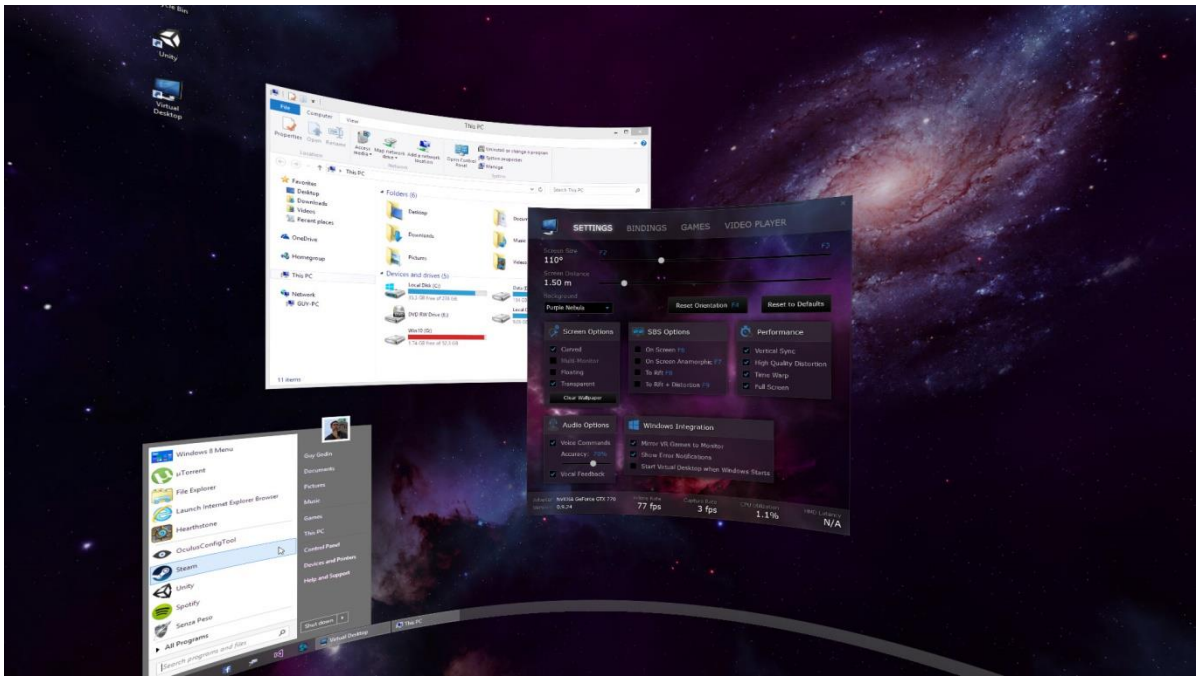
Sustav čestica (engl. particle system) je sustav u kojem se generira vrlo velik broj malih čestica koji imaju određena svojstva i pravila ponašanje te tako generiraju sliku *u sceni. Pogodni su za prikaz dinamike fluida, eksplozija tj. općenito bilo kakvih* geometrijski nepravilnih i vremenski promjenjivih pojava. Svaki sustav čestica ima svog odašiljača što je pozicija stvaranja (engl. spawn point) čestica. Svaka čestica ima svojstva, a najčešća su životni vijek, frekvencija titranja i boja. Ovdje također imamo problem složenosti izračuna pa time i performansi sklopovske opreme. Na sljedećoj slici dan je primjer sustava čestica vatre.



*Slika 16 Sustav čestica vatre*

## 4.3 Interakcija s virtualnim sučeljem

Interakcija čovjeka [12] i računala fokusira se na sučelja koje čovjek koristi i kako ih dizajnirati što bolje tako da budu intuitivna. Virtualno sučelje tehnički gledano je scena koja se iscrta, ali uloga te scene je realizirati sučelje. Ovdje ćemo objasniti koji su važni principi sučelja namijenjeno čovjeku. Pri realizaciji sučelja koriste se metafore. Metafora je preneseno značenje apstraktnih pojmova uz konkretno poimanje. Metafore se koriste pod pretpostavkom da korisniku time se približava poznavanje interakcije koju treba ostvariti. Najpoznatiji primjer metafore bio bi desktop [13] metafora prikazana na slici, gdje je cilj realizirati virtualno sučelje takvo da je što sličnije radnoj površini kompjutera jer je korisnik s njom dobro upoznat.




*Slika 17 Virtualna radna površina (desktop)*

Postoji mnoštvo metafora iz stvarnog svijeta koje se mogu učinkovito realizirati u virtualnom sučelju. Ključni dijelovi interakcije čovjeka s virtualnim sučeljem su manipulacija i navigacija.

### 4.3.1 Manipulacija

Manipulacija predstavlja sve radnje koje utječu na sustav nad kojim korisnik upravlja preko virtualnog sučelja. Svaka manipulacija sastoji se od selekcije i akcije. Selekcija je odabir neke funkcije sučelja, a akcija realizacija te funkcije. Često se istovremeno selekcija i akcija izvode. Selekcije se mogu vršiti pomoću pravaca, objekata ili numeričkih vrijednosti [14].

Manipulacija ima niz svojstava:

- Povratna informacija (engl. feedback) se koristi kako bi korisnik znao je li doista postigao interakciju s virtualnim sučeljem koju je htio.
- Cjepkanje interakcija (engl. ratcheting) je proces razbijanja složenih naredbi na više jednostavnih kako bi se olakšala interakcija s korisnikom. Npr. u metafori radne površine bi bilo pomicanje miša, pa klikanje, pa opet pomicanje miša…
- Ograničenja (engl. constraints) se primjenjuju zbog beskonačne prostorne mogućnosti virtualnog svijeta. Korisnik bi se izgubio ukoliko bi rascjepkali sučelje na velike virtualne udaljenosti. Ograničenje na udaljenost (engl. travel constraint) je najčešće korišten.
- Udaljenost (engl. distance) utječe na domet zone mogućnosti upravljanja nad virtualnom okolinom. Mogućnost upravljanja nad udaljenim objektima naziva se akcija u daljini (engl. action at a distance, AAAD).
- Područje pokazivača zrake (engl. pointer beam scope) je važno za sva sučelja koja koriste neku vrstu pokazivača (npr. mišev pokazivač) jer je potrebno definirati njegovo područje djelovanja.
- Histereza (engl. hysteresis) je svojstvo razlike učinka djelovanja i poništavanja neke akcije. Primjer bi bio kada korisnik želi označiti neki virtualni objekt koji je malen što otežava selekciju pa se problem rješava povećavanjem polja označavanja tog objekta ili vremenskom odgodom prije odznačavanja objekta.

- Okvir reference (engl. frame reference) je točka iz koje se generira virtualno sučelje. Okvir reference utječe na udaljenost, perspektivu i manipulaciju.
- Lokacija kontrole (engl. location control) je pozicija dijela unutar virtualnog sučelja. Češće korištene kontrole obično su bliže pozicionirane korisniku.
- Vidljivost kontrola (engl. control visibility) govori o vidnom polju korisnika koje obuhvaća virtualno sučelje.
- Formula gibanja (engl. movement formula) je svojstvo dijela sučelja da ima definiran način gibanja u virtualnoj sceni. Neki objekti se pomiču u sceni po potrebi dok drugi mogu biti stacionarni.

Postoje četiri manipulacijske metode: direktna kontrola korisnika, fizička kontrola, virtualna kontrola, kontrola agenta. U ovom radu promatramo virtualnu kontrolu. Virtualna kontrola je kontrola koja se manifestira isključivo u virtualni svijet. Konkretno, Leap Motion uređaj generira objekt ruku unutar scene te se pomoću tog virtualnog objekta upravlja drugim objektima. Virtualne kontrole vrlo često projiciraju stvarne fizičke u virtualnu stvarnost jer je korisnik navikao na takve pokrete (npr. zamah teniskim reketom, širenje dlana ruke i sl.). Velika prednost virtualnih kontrola je mogućnost sakrivanja po potrebi, dakle ne zauzimaju višak virtualnog prostora ako za time nije potrebno. Mana virtualnih kontrola bila bi potreba za stvarnim posrednikom koji će aktivirati kontrolu. To može biti miš, tipkovnica, gamepad… U našem slučaju to je Leap Motion.

Sve manipulacijske operacije svode se na sljedeće tipove:

- Pozicioniranje i skaliranje objekata
- Djelovanje sile nad objektom
- Modificiranje atributa objekata
- Modificiranje globalnih atributa
- Promjena stanja virtualnih kontrola
- Kontroliranje puta

### 4.3.2 Navigacija

Navigacija opisuje načine kretanja u virtualnoj sceni. Kod virtualnih sučelja navigacija predstavlja snalaženje korisnika u cjelokupnom prostoru sučelja. Navigacija se sastoji od putovanja (engl. travel) i pronalaska puta (engl. wayfinding).

*Travel* opisuje kako se korisnik kreće kroz virtualni prostor. Gotovi svaki virtualni sustav zahtijeva neki oblik putovanja. Vrsta putovanja trebala bi se prilagoditi dizajnu virtualnog sučelja koji se koristi. Npr. ako je sučelje raspršeno na većem prostoru, potrebna je kontrola koja će omogućiti brzo kretanje po takvom sučelju.

*Wayfinding* opisuje metode pomoću kojeg korisnik određuje svoju željenu poziciju u prostoru i vremenu virtualne scene. Prilikom pronalaženja puta korisnik stvara takozvani mentalni model, nešto slično mentalnoj mapi iz psihologije, pomoću kojeg pamti ključna mjesta u sceni. Postoje četiri učestale strategije pri stvaranju kognitivnih mapa:

- Podijeli pa vladaj se primjenjuje na način da se teritorij podijeli na manje cjeline, prouči njihove uloge te međusobne poveznice.
- Globalna mreža je strategija u kojoj se koriste obilježja značajnih teritorija (engl. landmark) te povezuje u globalnu sliku scene.
- Progresivna ekspanzija je strategija gdje korisnik pokušava zapamtiti prvo manju cjelinu mape, te polako širi opseg poznavanja mape. Ova strategija pokazala se često neuspješnom.
- Narativna elaboracija je strategija gdje korisnik koristi priče pomoću kojih pokušava zapamtiti dijelove. Ovo je najneuspješnija strategija.

Postoje razne pomoćne metode u pronalasku puta a neke od njih su praćenje puta, mape, obilježja teritorija, kompas…

# 5. Implementacija

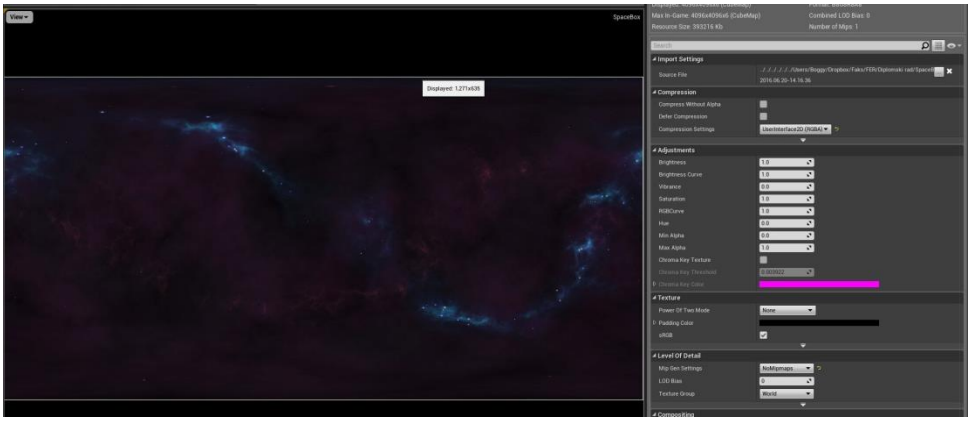
Konačno, u ovom poglavlju upoznat ćemo se konkretnom implementacijom virtualnog sučelja. Ideja je napraviti stacionarni svemirski brod koji se nalazi u nekom dijelu svemira i unutar njega korisnik (pilot, astronaut…) na različite načine preko virtualnog sučelja ostvaruje interakciju s okolinom. Korištena terminologija je iz Unreal razvojne okoline (opisana u trećem poglavlju). Kompletna implementacija smještena je u jednu razinu (engl. level) nazvan *Space*.

## 5.1 Modeliranje scene

Za modeliranje okoline korišten je početnički paket (engl. starter pack) Unreal razvojne okoline. Princip modeliranja je CSG koje je objašnjeno u prethodno poglavlju.

### 5.1.1 Modeliranje svemirske sfere (okoline)

Okolina se modelira pomoću takozvanog *skybox blueprint* koji će biti objašnjen u nastavku. Korištena tekstura je kubna mapa generirana pomoću programske opreme *Spacescape* [15]. Generirana tekstura je datoteka formata *DirectDrawSurface* (skraćeno .dds), rezolucije 4094x4094 i veličine 384MB. Pomoću opcije import u Unreal okolini učita se tekstura (potrebno je neko vrijeme da se učite pošto je velike rezolucije, otprilike 30 min). Na sljedećoj slici prikazana je tekstura svemirske sfere.

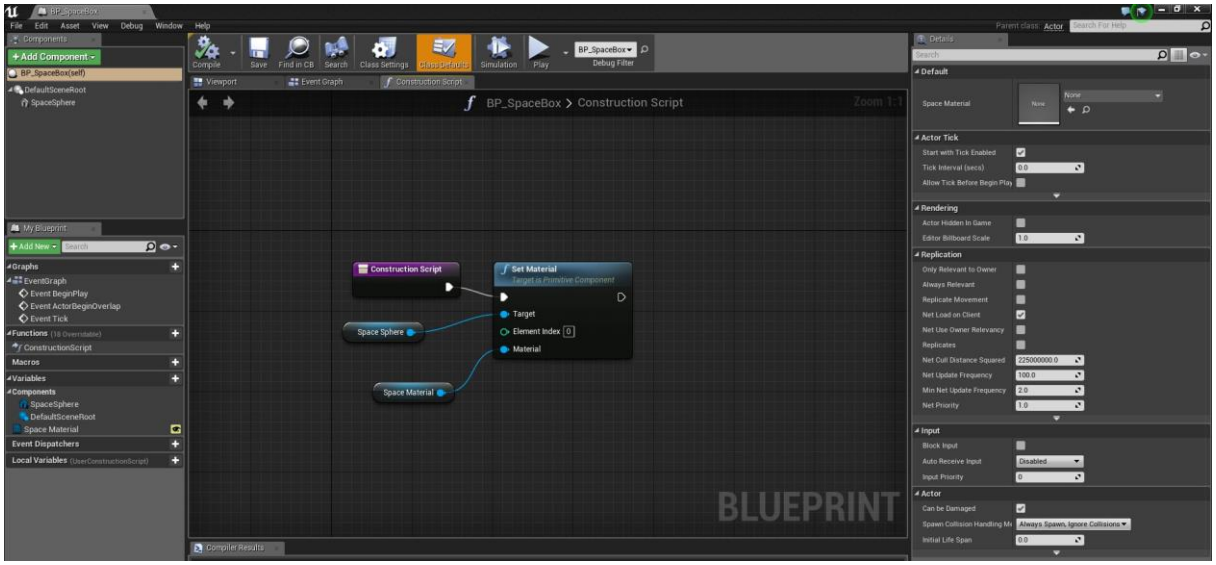


*Slika 18 Space box tekstura*

Tekstura pri čistom importu se čini zamagljena, pikselizirana te je potrebno promijeniti od predefiniranih postavki sljedeće:

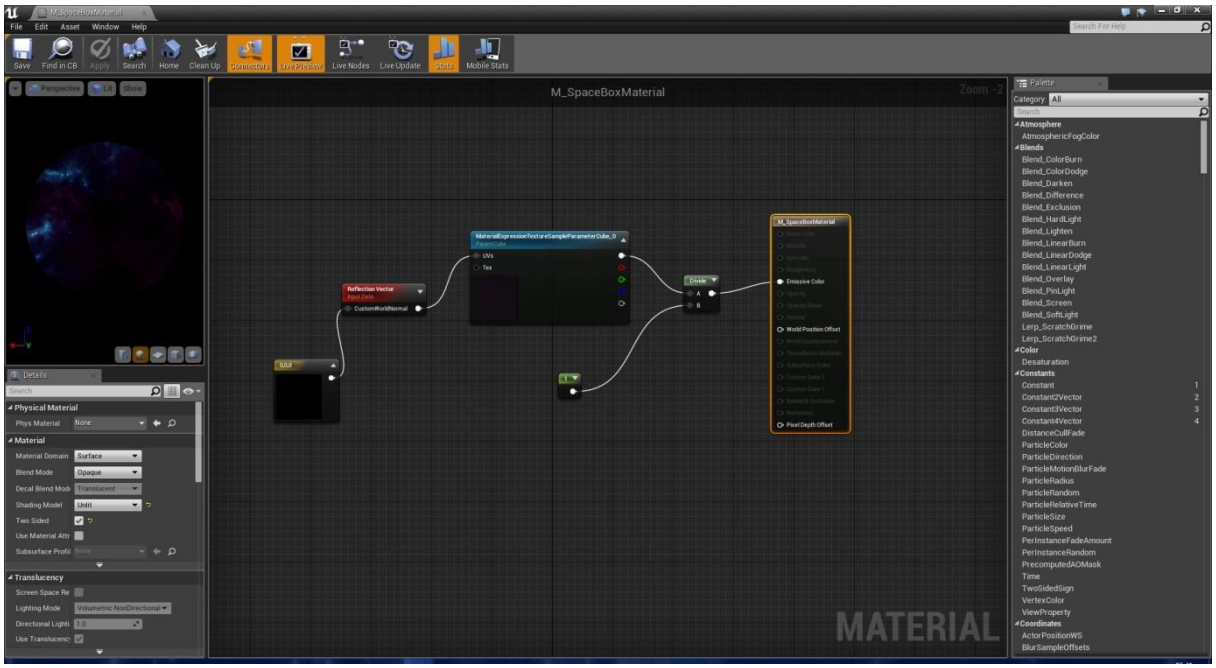
- *Compression settings* postaviti na *UserInterface2D*
- *MipGenSettings* postaviti na *NoMipmaps*
- Postaviti *TextureGroup* na *SkyBox*
- Po potrebi uključiti ili isključiti *sRGB* (u ovoj implementaciji je isključen)

Nakon toga stvorit ćemo novu *blueprint* klasu tipa *Actor*. Unutar uređivača klase dodajemo komponentu *Static Mesh* te je namjestimo na vrijednost *SM_SkySphere* (to je predefinirana statička mreža od Unreal okoline). U *ConstructionScript* dijelu uređivača (sučelje gdje definiramo što se događa pri instanciranju klase) definiramo javnu člansku varijablu *SkyMaterial* tipa *Material*. Na sljedećoj slici prikazana je konstrukcijska skripta klase svemirske kutije.



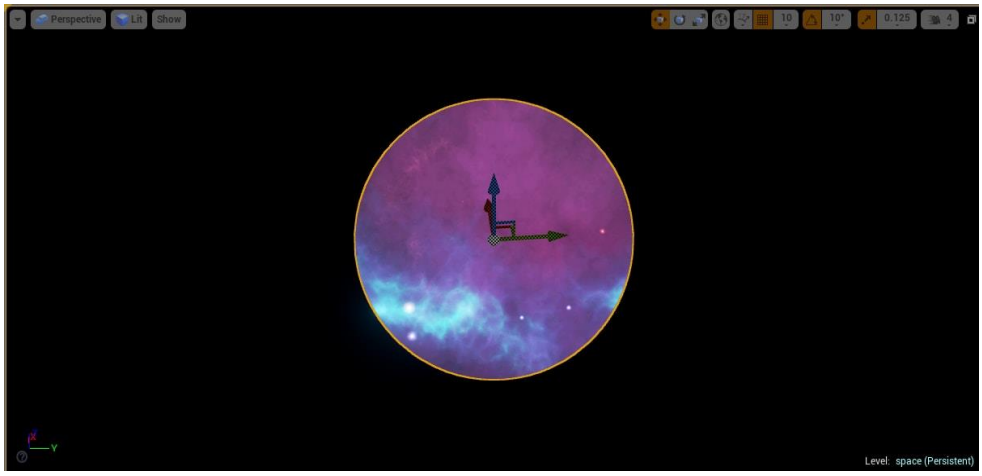

*Slika 19 Konstrukcijska skripta za Spacebox*

Dodajemo dva GET čvora za *SkyMaterial* i *SkySphere* te ih postavimo kao ulaz funkcije čvora *Set Material*. U prijevodu, dohvatimo materijal i vežemo ga za sferu pri svakoj konstrukciji objekta tipa *Spacebox*. Naposljetku moramo napraviti materijal koji će koristiti učitanu teksturu. Na sljedećoj slici prikazana je struktura materijala.




*Slika 20 Spacebox materijal*

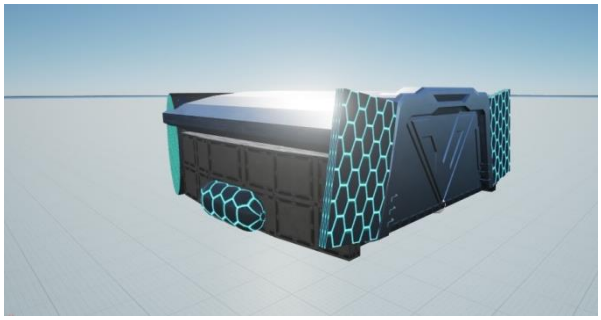
Od postavki s lijeve strane, *Shading Model* promijenjen je na *Unlit* i *Two Sided* je uključen. Ne želimo nikakvu refleksiju oko svemirske sfere, stoga je refleksijski vektor postavljen na vrijednost (0, 0, 0). Potrebno je dodati čvor *TextureSampleCube* jer smo učitali teksturu kao kubnu mapu te vezati njegov izlaz na parametar *Emissive Color*. *M_SpaceMaterial* je konačni čvor koji se prikazuje na izlazu. Postupak je gotov, potrebno je još samo ubaciti *Spacebox* u scenu i izgledat će kao na sljedećoj slici.




*Slika 21 Spacebox objekt u sceni*

### 5.1.2 Modeliranje svemirskog broda

Vanjština svemirskog broda sastoji se od: podnožja, krova, jednog stražnjeg motora, dva motora sa strane, dva prednja motora, pokretnih vrata te oklopnih dijelova koji to sve povezuju. Na sljedećoj slici prikazan je brod iz dva različita kuta.



*Slika 22 Model svemirskog broda izvana*

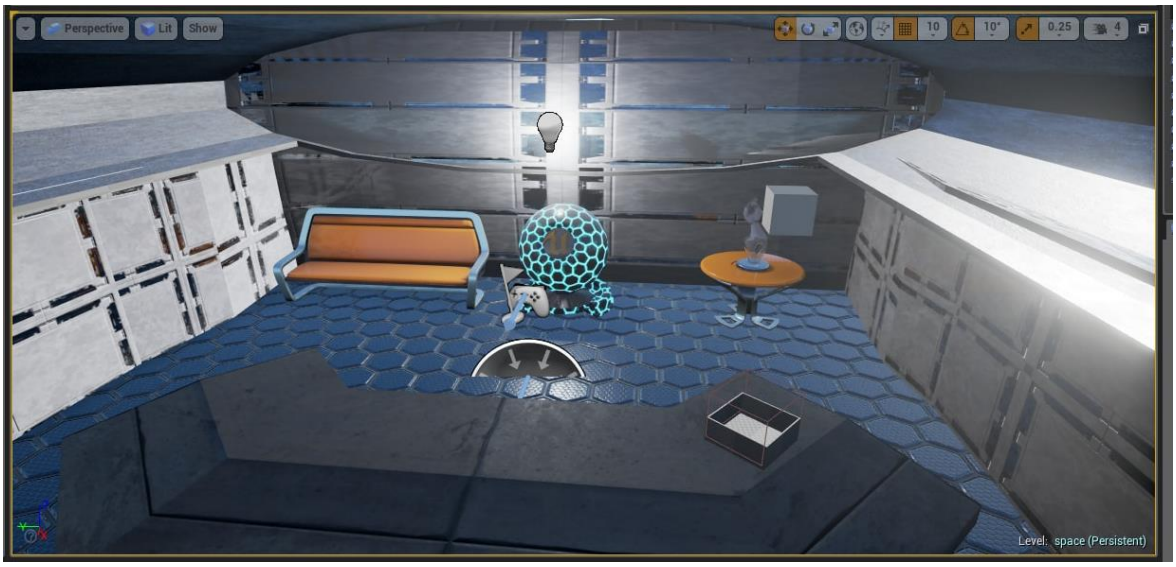
Korišteni materijal za motore je *M_Hex_Tile_Pulse*, a za olupinu broda je *M_Tech_Panel* iz početnog paketa. Krov broda je zapravo izduženi šuplji valjak koji unutar sebe ima *Roof Brush* što je geometrijska komponenta presjeka. Dakle, gdje god se spomenuti geometrijski objekt preklapa s ostalim objektima, taj dio objekta neće se iscrtati. Objekt presjeka dodaje se tako što se odabere oduzimajući način iscrtavanja prije dodavanja objekta u scenu, svaki *Brush* objekt može biti dodavajući (engl. additive, normalno se iscrtava) ili oduzimajući (engl. substractive, „reže“ ostale objekte). Dva bočna motora su objekti od klase *Cylinder* što je modelirana klasa opreme (engl. prop). Stražnji motor, baš kao krov je izduženi valjak, ali popunjen te osim *HexTilePulse* materijala ima na sebi *M_Hex_Tile* materijal. Ostatak olupine, uz dva prednja motora su objekti kvadra te su objekti tipa *Static Mesh* što nije isto što i Brush objekt. Naime, na *Static Mesh* objekte ne utječe obris oduzimajućih objekata u preklapanju.

Vanjskog izvora svjetla nema već je namještena bijela boja u postavkama scene u potkategoriji *Lightmass* umjesto početne crne boje svjetlosti. Time je postignuto dovoljno svjetline za svemirsku okolinu. Spomenuta potkategorija postavki obuhvaća ponašanje širenja svjetlosti kroz scenu. S ovim smo zaokružili kompletan eksterijer aplikacije.

Unutrašnjost broda je glavni dio virtualne scene. Zasada ćemo ignorirati neke objekte koji su vidljivi u sceni (oni koji igraju direktnu ulogu u virtualnom sučelju). Na sljedećoj slici prikazan je prednji dio unutrašnjosti broda.

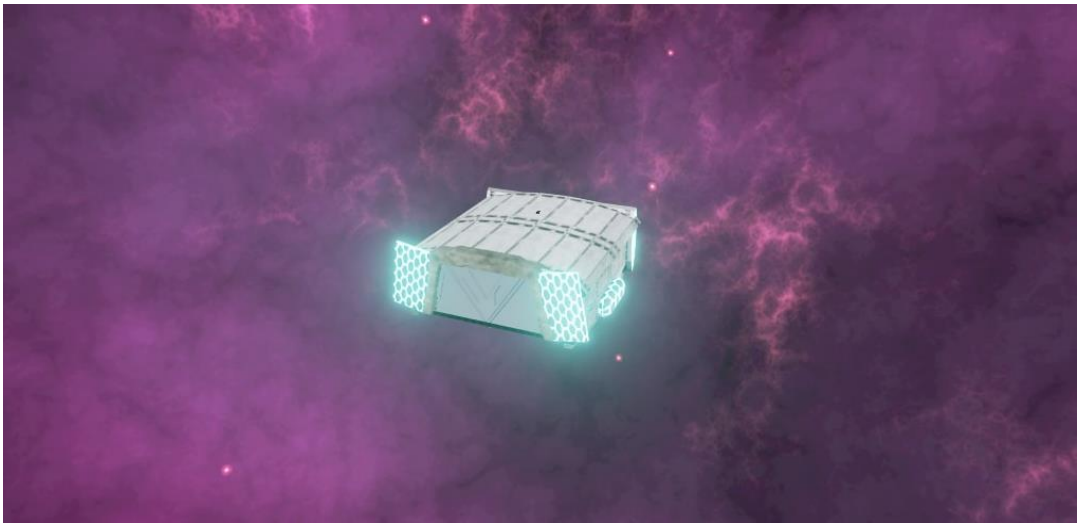

*Slika 23 Prednji dio broda*

U prednjem dijelu broda imamo pokretna vrata, stol za sučelje (improvizirana kontrolna ploča), s desna na lijevo detektor ruku, jedan crveni i zeleni gumb, kocka i kutija. Svi objekti koji se nalaze na kontrolnoj ploči bit će naknadno objašnjeni, zasada razmatrajmo statičke objekte. Kontrolna ploča je objekt tipa Shape_Trim_90_Out iz mape *Shapes* početnog sadržaja s pridruženim materijalom *M_Concrete_Tiles*. S prednje strane još se nalazi *Spotlight* izvor svjetla koji obasjava cijelu prostoriju. Sve komponente koje se nalaze na radnoj površini su dio komponente kontrolne ploče što znači da se njihov koordinatni sustav pozicije i rotacije relativno uspoređuje s koordinatama kontrolne ploče koji je roditelj komponenta. Dakle, kad bi postavili poziciju bilo koje komponente djeteta na (0, 0, 0), dobili bi mjesto gdje se nalazi ishodište objekta radne ploče. Razlog takvom grupiranju je zbog finog i brzog podešavanja pozicije radne ploče. Kada se roditelj translatira u sceni za neku poziciju, djeca komponente se translatiraju za isti vektorski iznos. Njihov relativni odnos u prostoru mora ostati isti. Međutim, ako probate translatirati ili rotirati dijete, ostali element ostat će isti. Na sljedećoj slici prikazan je stražnji dio unutrašnjosti broda.




*Slika 24 Stražnji dio broda*

Stražnji dio broda uključuje izduženi naslonjač, stol sa staklenom skulpturom i generator. Svi dijelovi su iz mape *prop* početnog sadržaja. Tu je još točkasti izvor svjetla (engl. point light) koji obasjava prostoriju. Svi spomenuti objekti imaju čisto estetsku ulogu u prostoru. S ovime smo zaokružili cjelinu modeliranja scene te ćemo sada opisati modeliranje virtualnog sučelja u sceni zajedno s njegovim funkcionalnostima. Konačna verzija broda integrirana sa svemirskom sferom prikazana je na sljedećoj slici.



*Slika 25 Konačna verzija nebeskog broda u svemirskoj sferi*

## 5.2 Modeliranje sučelja

### 5.2.1 Pomična vrata

Prva izvedena funkcionalnost sučelja je mogućnost otvaranja vrata broda pomoću stavljanja objekta kocke u kutiju. Model vrata dostupan je na službenoj *wikipedia* stranici Unreal razvojne okoline. Početni korak izrade je napraviti *blueprint* klasu tipa *Actor* koja će predstavljati naša vrata [16]. U implementaciji ime klase je *BP_Tutorial_Door*. Unutar uređivača dodajemo Static Mesh komponentu *S_LT_Doors_SM_Door05* što je statička komponenta okvira vrata te postavimo varijablu imena *DoorFrame* (okvir vrata) koja se odnosi na tu komponentu. Sada dodamo statičku komponentu koja predstavlja pokretni dio vrata, imena *S_LT_Doors_SM_DoorWay05* te postavimo varijablu imena *Door* (vrata). Zbog toga što smo napravili varijable za dijelove pomičnih vrata, možemo ubaciti u scenu koliko god vrata želimo i za svaka imati različite materijale i ostala svojstva. Također se može primijetiti da okvir i klizni dio vrata ne moraju biti od istog materijala jer su to razdvojene statičke komponente. Komponenta *Door* postavljena je kao dijete komponente *DoorFrame* zbog efektivne translacije i vezanja objekata kao što je prikazano na slici.

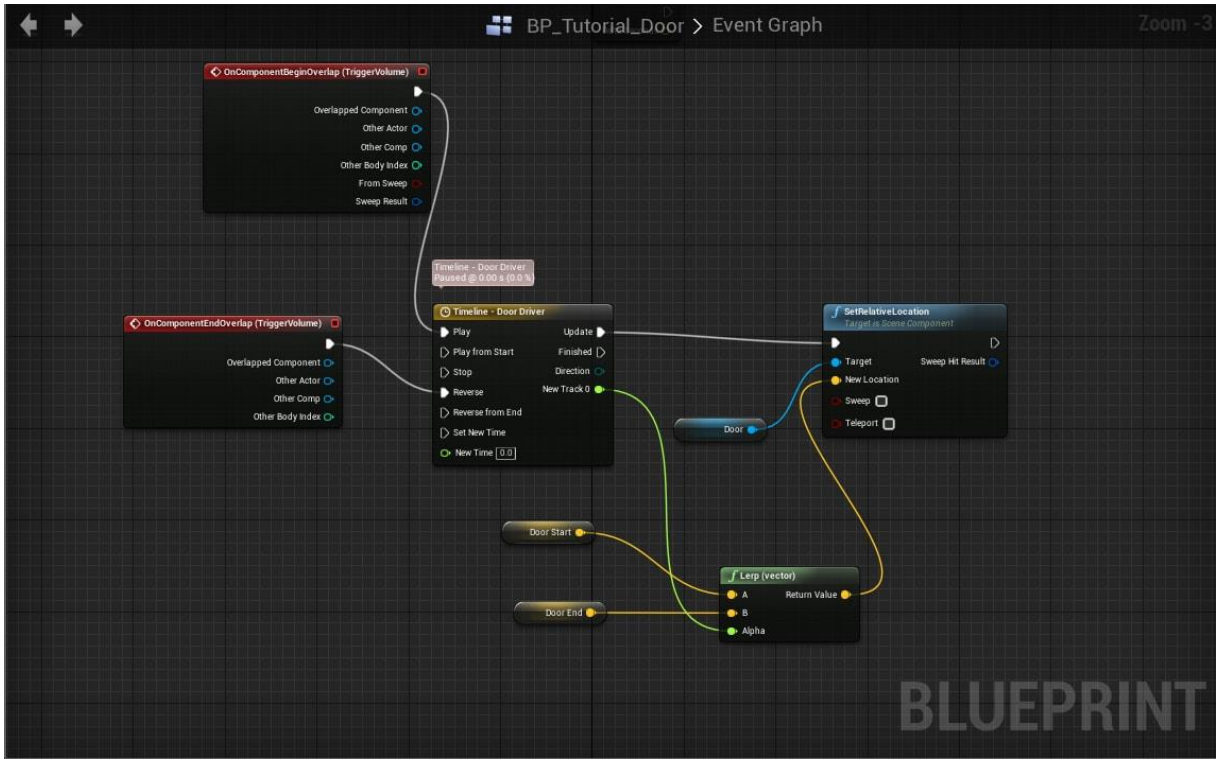

*Slika 26 Isječak svojstava pomičnih vrata*

Nakon toga dodajemo komponentu kocke okidača (engl. box collision) i nazvat ćemo je *TriggerVolume* (prostor okidača). U postavkama okidača u kategoriji *Shape* namjestimo parametar *Box Extent* na (192, 192, 128), u kategoriji *Transform* lokaciju (0, 0, 128) te u kategoriji *Collision* (kolizija, sudar) parametar *Collision Enabled* na *Query only* te da je odaziv ili reakcija (engl. *responsiveness*) na kolizije preklapanje (engl. overlap). Programeru je dana potpuna sloboda ovdje da odredi točno koja vrsta objekata smije izazvati kakvu reakciju kod volumnog okidača. Npr. možete postaviti da samo glavni igrač generira događaj preklapanja, dok ostale okidač ignorira. U ovom trenutku možete dodati na sceni objekt pomičnih vrata, no još nemamo nikakvu funkcionalnost otvaranja i zatvaranja na senzor.

Otvorimo karticu *Event Graph* u uređivaču. Prvo dodajemo čvor *OnComponentBeginOverlap* za varijablu *TriggerVolume*. U prijevodu, želimo reagirati u slučaju kada dolazi do preklapanja komponente čija uloga je okidač, a to je naša varijabla *TriggerVolume*. Sada trebamo postaviti sustav linearne interpolacije [17] (skraćeno *lerp*). Prvo dodajemo čvor *lerp (vector)* koji ima dva ulaza A i B. Ulaz A promoviramo u varijablu što znači da nećemo ručno unositi vrijednost (barem ne direktno unutar grafa), već preko varijable koju nazivamo *DoorStart*. Također, ulaz B promoviramo u varijablu *DoorEnd* po istom principu s tim da će ova varijabla biti javna. Zbog urednosti, varijable *DoorStart* i *DoorEnd* stavimo pod kategoriju *Door Setup* (postavljanje vrata), te namjestimo vrijednost varijable *DoorEnd* na (0, 0, 288).

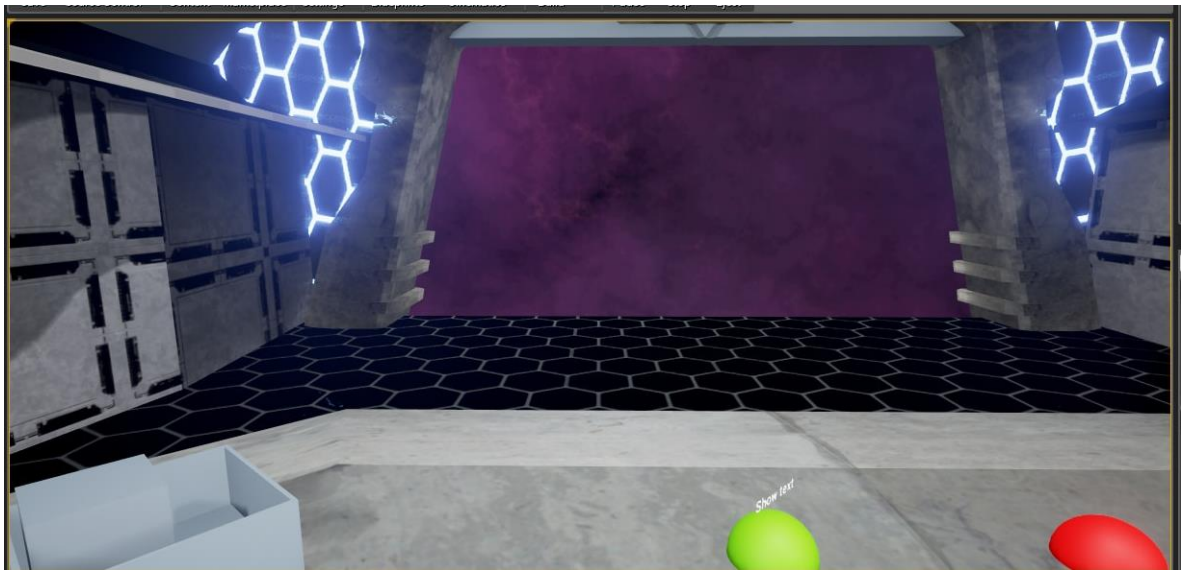
Skoro sve imamo spremno za otvaranje vrata, a sad je potrebno animirati proces otvaranja (kasnije zatvaranja). Dodajemo u grafu novi čvor vremenske crte (engl. Timeline) te ga imenujemo *Door Driver*. U kartici *Timeline* editor možemo podesiti naš vremenski čvor. Tamo dodajemo dva čvora (ne čvor grafa, nego čvor na vremenskoj crti), jedan s vremenskom vrijednošću 0 i ulazom 0, drugi s vremenskom vrijednošću 1.5 i ulazom 1. Namjestimo još *Length* parametar na 5, ovo znači da će cjelokupna animacija trajati pet sekundi. Povežemo izlaz Driver na ulaz *Alpha* od *lerp vector* čvora.

Još nam samo nedostaje dio koji će otvoriti vrata, stoga dodajemo čvor koji dohvaća varijablu *Door* (običan GET) te njega vežemo na novi čvor *Set Relative Location* čija funkcija je postaviti novu relativnu ulaznog objekta. Postoji poseban ulaz gotovo svakog čvora za događaj, označen je bijelom bojom. Na taj ulaz spajamo izlaz *lerp vector* čvora. To je to, imamo vrata koja se otvaraju, ali ne zatvaraju. Ukoliko želimo zatvoriti vrata, možemo to protumačiti kao obrnuti postupak zatvaranja. Dakle, dodajemo novi čvor *OnActorEndOverlap* za člansku varijablu *TriggerVolume* te rezultat tog događaja spajamo na ulaz istog vremenskog čvora ali *Reverse*. Time smo napokon ostvarili pomična vrata i sljedeća slika prikazuje konačan izgled grafa događaja klase.




*Slika 27 Graf događaja pomičnih vrata*

Okidač za vrata smješten je unutar kutije u koju treba ubaciti objekt kocke. Sve dok se kocka preklapa s okidačem, vrata će biti otvorena te u protivnom slučaju vrata će se zatvoriti. Sljedeća slika prikazuje situaciju gdje su vrata otvorena.



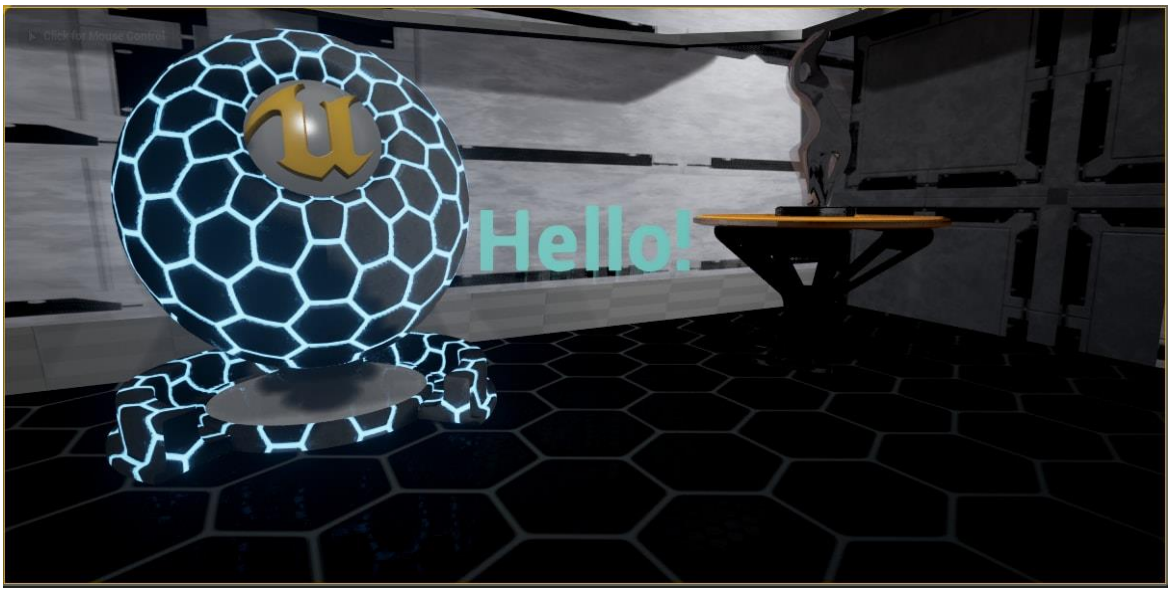

*Slika 28 Otvorena vrata svemirskog broda*

Time smo ostvarili jednu vrstu interakcije s virtualnim sučeljem, a to je preko konkretnog objekta u sceni. Sljedeća interakcija je s gumbima.

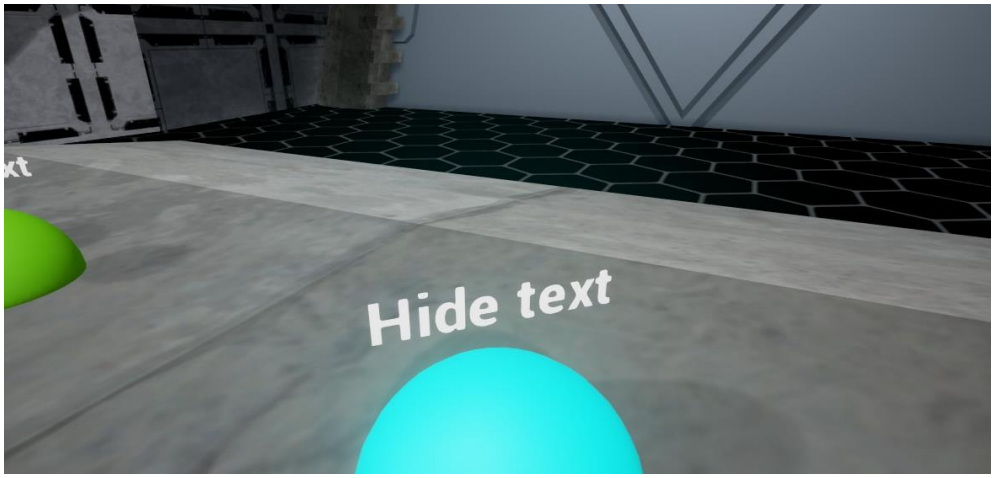
### 5.2.2 Prateći tekst

Sljedeća funkcionalnost virtualnog sučelja je prikaz teksta koji je uvijek centriran u odnosu na oko promatrača te orijentiran prema promatraču scene. Pritiskom zelenog gumba tekst se pojavljuje ispred korisnika, a pritiskom crvenog gumba se sakrije. Prije opisa implementacije, na sljedećim slikama je izgled konačne implementacije.


*Slika 29 Početna pozicija teksta*

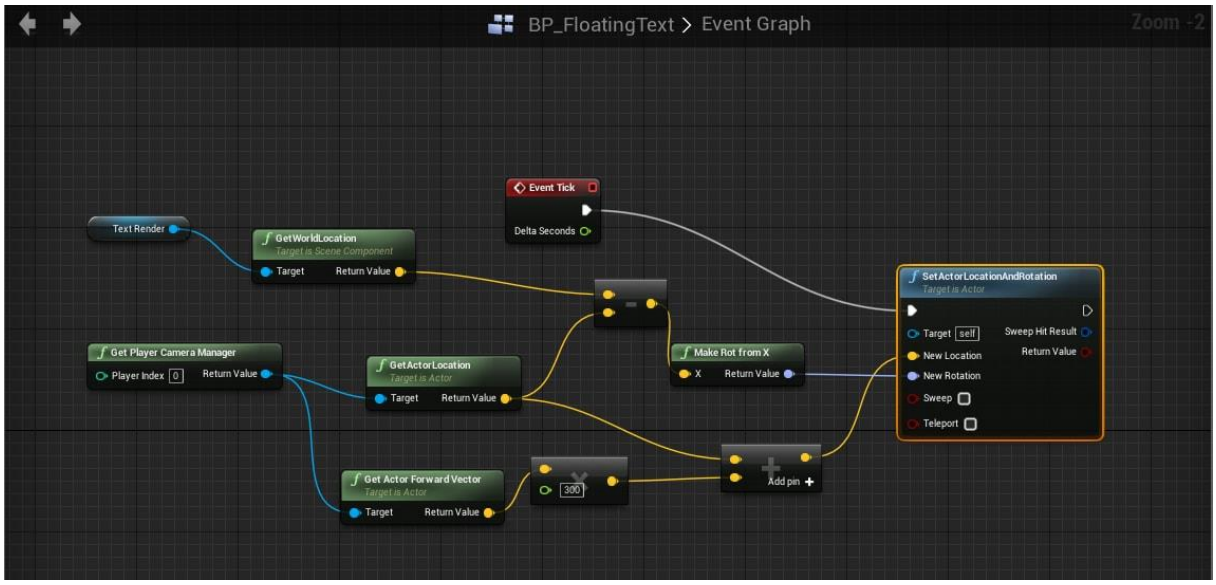



*Slika 30 Promjena orijentacije promatrača*




*Slika 31 Pritiskom gumba za skrivanje, tekst se više ne prikazuje*

Krenimo prvo od implementacije samog teksta za prikaz. Dodajemo novi *blueprint* tipa *actor* koji predstavlja naš tekst. U klasi dodajemo Billboard komponentu te na nju dodajemo novu komponentu za iscrtavanje teksta. U postavkama te komponente stavite tekst po želji, u ovom slučaju to je „*Hello*!“. Pošto je tekst postavljen u smjeru pozitivne x osi, a nama u sceni treba biti u smjeru negativne x osi, potrebno je rotirati oko Z osi objekt teksta za 180°. Više ništa ne treba dodavati u konkretnoj klasi već podesiti graf događaja kao što je prikazano na slici.

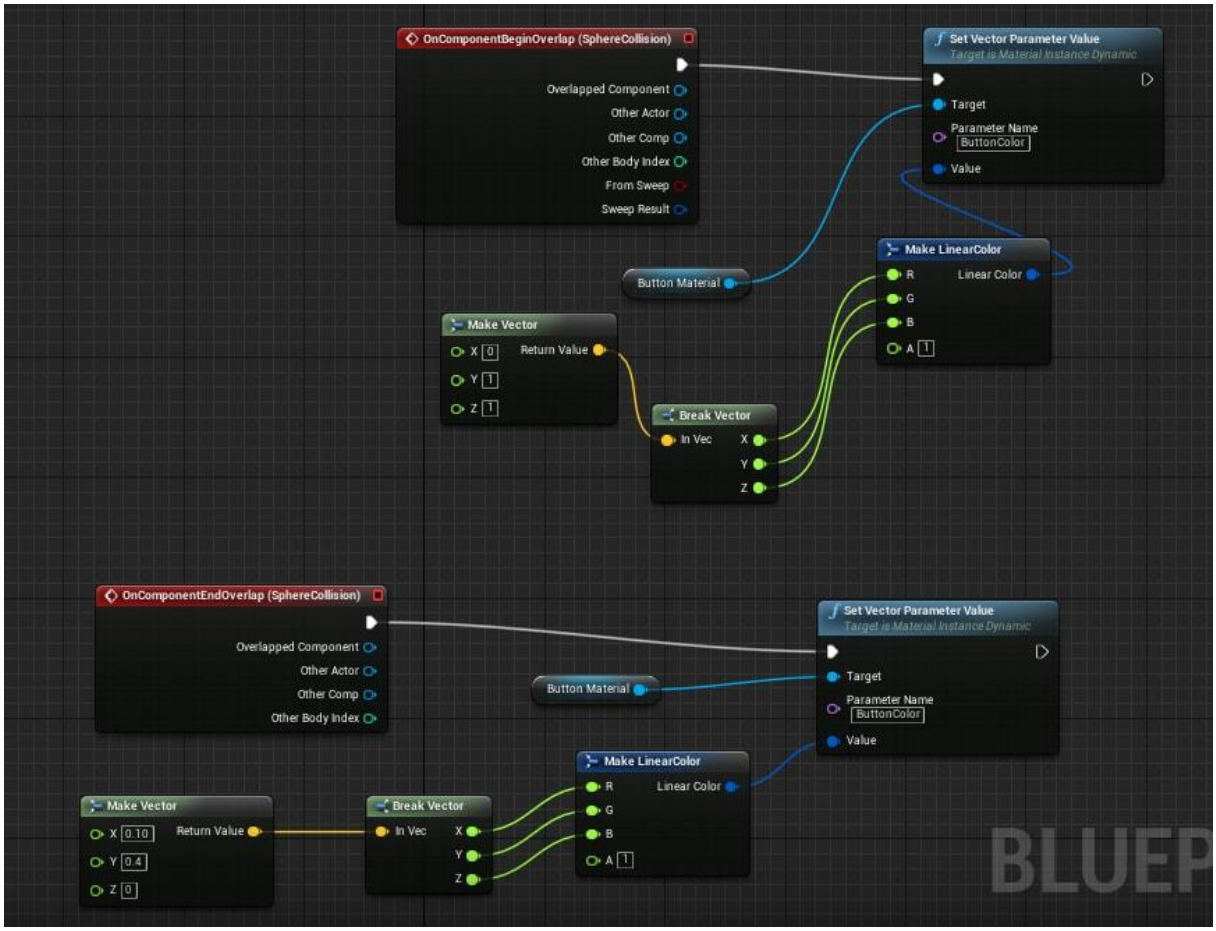



*Slika 32 Graf događaja za prikazani tekst*

Krenimo redom, graf možemo podijeliti na dva dijela od kojih se jedan brine za novu poziciju, a drugi za novu rotaciju objekta teksta. Ako želimo da nam se tekst uvijek pozicionira ispred promatrača, trebamo dobiti vektor koji gleda prema naprijed od pozicije promatrača. Čvor *Get Player Camera Manager* dohvaća objekt kamere igrača. Iz tog objekta imamo gotove metode za dohvat vektora usmjeren pravo i lokaciju aktera. Za vektor je čvor *Get Actor Forward Vector*, a za lokaciju je čvor *Get Actor Location*. Sve što nam preostaje je uvrstiti lokaciju promatrača u vektor usmjerenja te dodati nekakav skalarni iznos koji predstavlja željenu udaljenost prikaza teksta od promatrača. Zbrojem lokacije i vektora promatrača dobijemo konačnu željenu koordinatu točke teksta za iscrtavanje koju usmjerimo u ulaz New Location konačnog čvora. Čvor koji obavlja dužnost postavljanja nove pozicije i orijentacije našeg objekta teksta je *Set Actor Location and Rotation*.

Za orijentaciju potrebno je dobiti vektor dobiven koordinate promatrača i objekta teksta u smjeru promatrača. Čvor *Get World Location* dohvaća lokaciju objekta teksta, a već iskorišteni čvor za dohvat lokacije promatrača možemo opet iskoristiti. Njihovi izlazi povezuju se s čvorom za oduzimanje vektora te rezultat pohranjujemo u ulaz *New Rotation* konačnog čvora. Time smo riješili pitanje orijentacije teksta prema promatraču.

Iscrtavanje teksta nam je spremno. Potrebno je dodati kontrole koje će upravljati pokazivanjem objekta u sceni. Imat ćemo dvije praktički identične *blueprint* klase zvane BP_Show_Text i BP_Hide_Text. Oba objekta sastoje se od tijela sfere obojane crvenom bojom za skrivanje odnosno zelenom bojom za prikaz te sfernog okidača koji je za malu razliku većeg radijusa od objekta obojane sfere. Na sljedećoj slici prikazan je graf događaja za obje *blueprint* klase jer je identičan.

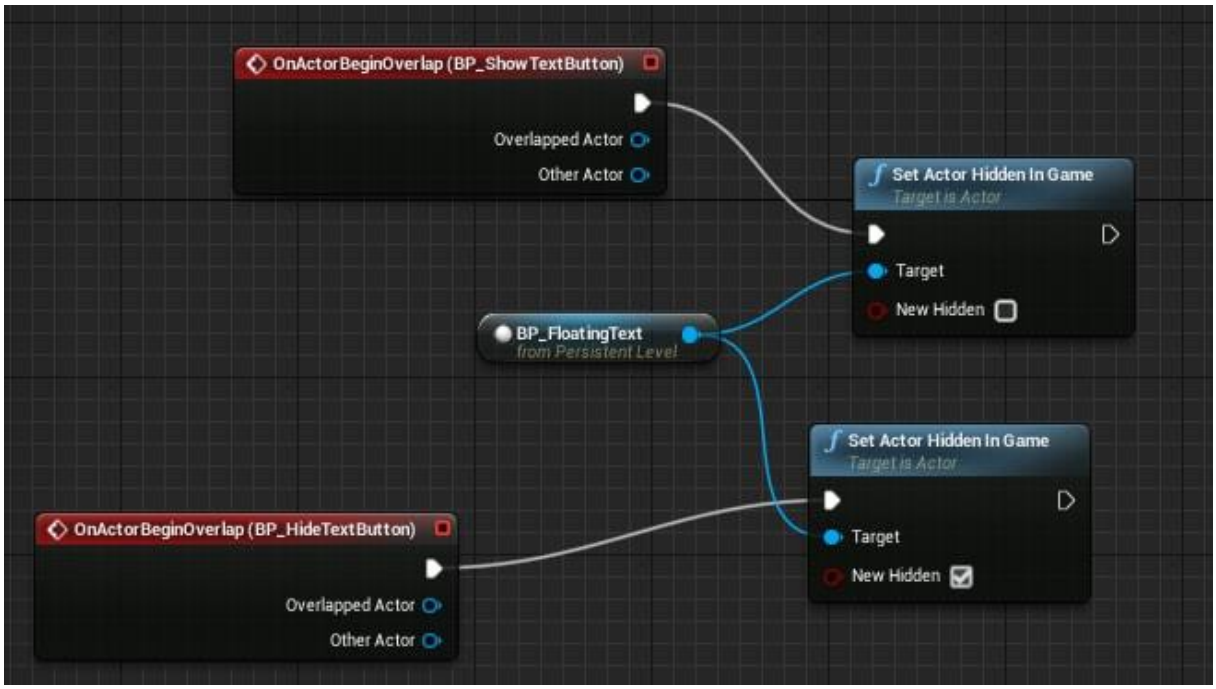



*Slika 33 Show/Hide text blueprint*

Gornja polovica grafa opisuje reakciju na događaj početka preklapanja gumba. Želimo sve dok korisnik pritišće gumb da svijetli plavom bojom. Prvo imamo već dobro poznati čvor *On Actor Begin Overlap* koji predstavlja događaj početka preklapanja te kad se on dogodi želimo postaviti vrijednost parametara *ButtonColor* (koji smo definirali u postavkama klase) na svijetlo plavu boju (čvor *Set Vector Parameter Value*). Plavu boju generiramo pomoću čvorova *Make Vector* (stvori nam

vektor 0, 1, 1) te od vektora stvorimo boju pomoću čvorova *Break Vector* i *Make Linear Color*. *Break Vector* razbije dobiveni vektor opet na x,y,z koordinate, a čvor *Make Linear Color* pretvara dobiveni vektor u *RGBa* zapis. *RGBa* zapis prosljeđuje se ulazu *Value* čvora za promjenu vrijednosti vektorskog parametara boje. Gotovo identičan proces događa se u donjem dijelu grafa, jedina razlika je što je događaj *On Actor End Overlap*, a vrijednost boje zelena ili crvena ovisno o ulozi gumba.

Trenutno imamo dobro ponašanje objekta teksta i gumba, ali ne i poveznicu za generiranje ili brisanje teksta pri interakciji s gumbom. Svaka razina u Unreal okolini ima svoj vlastiti *blueprint*, a pošto svaki *blueprint* ima graf događaja, tako i razina posjeduje svoj graf događaja. Graf događaja razine je specifičan po tome što ovdje baratamo objektima isključivo u sceni. Na sljedećoj slici gledamo isječak grafa koji odgovara ovoj funkcionalnosti virtualnog sučelja.

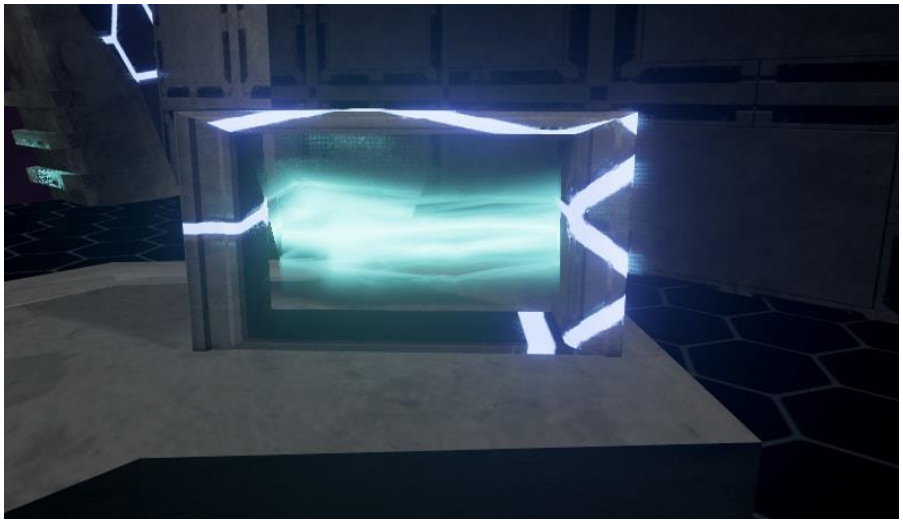



*Slika 34 Graf sučelja za prikaz teksta*

Imamo dva događaja za početak preklapanja gdje svaki odgovara jednom gumbu. Ovisno o ulozi gumba svojstvo *Hidden* se mijenja pomoću Čvora *Set Actor Hidden In Game* nad objektom klase *BP_FloatingText* koji dobijemo metodom GET čvora. Sada sve radi kako treba i kako je bilo prikazano ranije na slikama.

### 5.2.3 Detektor objekta

Detektor objekta sastoji se od titrajuće svjetlosne zrake realizirane preko sustava čestica te statičkog okvira s materijalom od motora broda. Na sljedećoj slici prikazan je izgled detektora. Detektor u sebi sadrži komponentu *Box Trigger* koja služi za detektiranje preklapanja objekata.

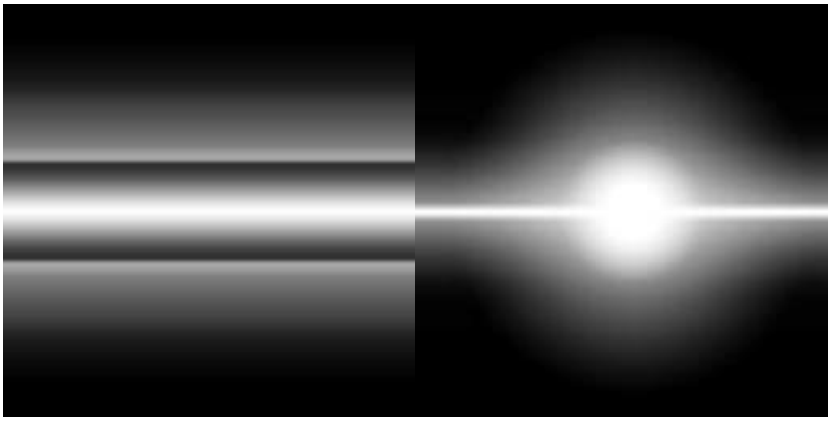


*Slika 35 Detektor objekata*

Pri svakom prolasku objekta kroz detektora događa se eksplozija na sredini broda ispred komandne ploče kao što je prikazano na slici. Eksplozija se nalazi u početnom sadržaju tj. njezina *blueprint* klasa i to je gotova implementacija sustava čestica.

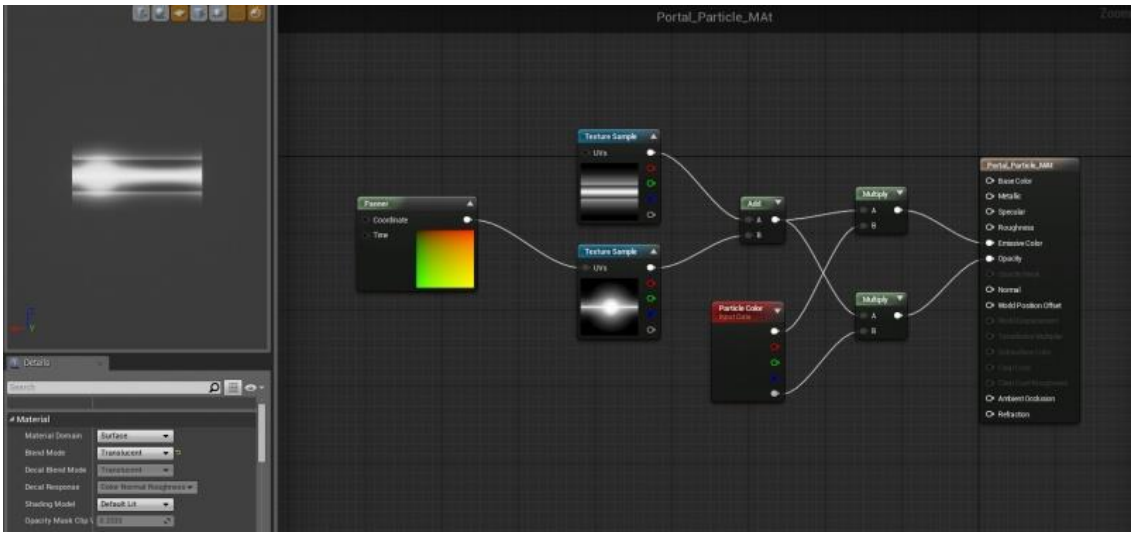
*Slika 36 Eksplozija izazvana detektorom objekta*

S druge strane sustav čestica laserske zrake je ručno implementiran stoga će se opisati postupak. Prije svega, trebaju nam dvije teksture čestice (slika) koji tvore zajedno materijal odašiljača (engl. emitter) [18].



*Slika 37 Teksture čestica svjetlosti*

Te dvije teksture složene su zajedno u materijal koji je prikazan kao na slici.




*Slika 38 Materijal sustava čestica*

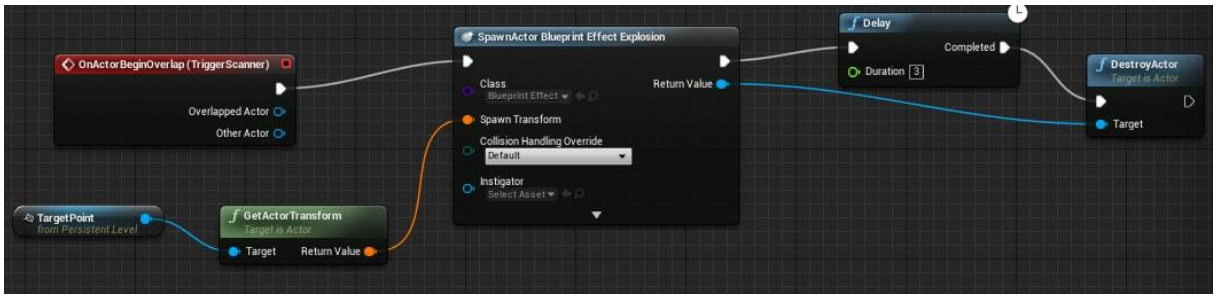
Prvi čvor *Panner* vežemo za površinsku teksturu čestice titraja se na nju dodajemo drugu teksturu. *Particle color* čvor služi za podešavanje boje materijala te konačni rezultat se sprema u ulaze *Emissive Color* i *Opacity*. *Opacity* predstavlja svojstvo popunjenosti bojom, a *Emissive Color* predstavlja samu boju materijala. Sustav čestica u Unreal okolini sastoji se od modula. Svaki modul obuhvaća jednu skupinu

svojstava koja se mogu namještati po potrebi. Moduli se mogu brisati i dodavati iz sustava čestica. Dodajmo novi sustav čestica (engl. particle system) nazvan *PS_Beam* te ga otvorimo u editoru. Prvo treba dodati modul *BeamData* koji obuhvaća svojstva kao što su brzina, broj interpolacijskih točaka, najveći broj zraka i najbitnije svojstvo za ovu implementaciju *Beam Method* koje govori koju metodu koristiti pri iscrtavanju zrake. Za naš slučaj, ta metoda je Target jer želimo definirati početnu i krajnju točku između kojih će se vršiti gibanje laserske zrake. Sljedeći je modul *Noise* koji omogućuje titranje naše zrake. Tu definiramo frekvenciju, amplitudu i sve ostale pojmove vezane uz titranje čestica. Još je potrebno definirati dvije članske varijable *BeamSource* i *BeamTarget*.

Ulogu izvora i odredišta zrake mogu odraditi bilo koji akteri. Za ovu implementaciju korišteni su *Note Actor* objekti. Kada smo ubacili u scenu zraku, treba dodati dva Instance parametar za izvor i za odredište. Podesimo da je izvor *SourceNote* objekt, a odredište *TargetNote* objekt i sada imamo situaciju gdje zraka prati dva Note aktera.

Potrebno je definirati mjesto gdje želimo da se dogodi eksplozija, a to se radi pomoću aktera tipa *TargetPoint* tako što ga ubacimo u scenu na željeno mjesto.

Konačno trebamo još u grafu događaja razine dodati situaciju kada se dogodi preklapanje okidača unutar detektora objekta. Isječak grafa odgovornog za taj dio prikazan je na sljedećoj slici.



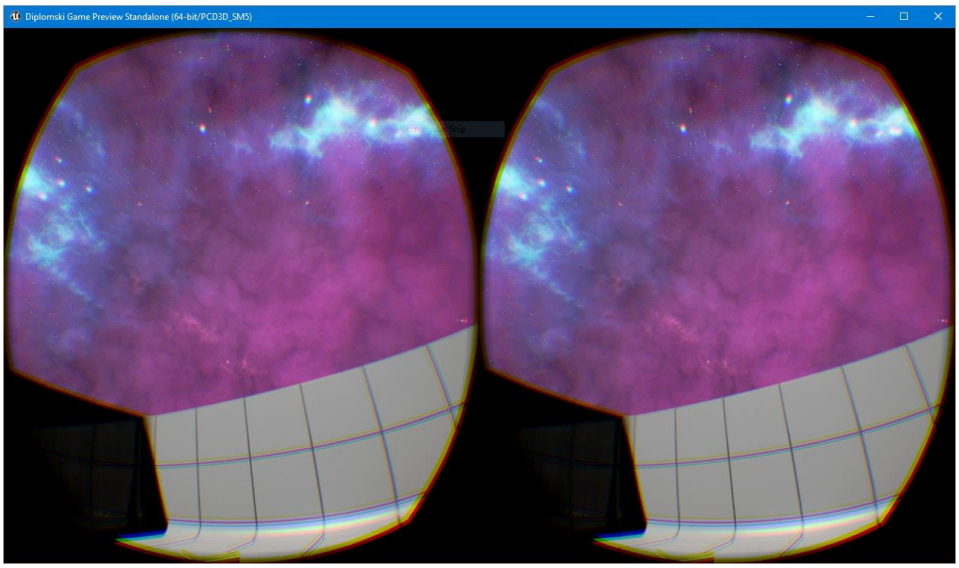

*Slika 39 Generiranje eksplozije na preklapanje objekta*

Nakon što smo dohvatili *TargetPoint* objekt pomoću njegove GET metode čvora, želimo izvršiti transformaciju stvaranja objekta na tu lokaciju. Čvor *Spawn Actor* radi upravo  to tako što mu dodijelimo željenu *blueprint* klasu čije objekte želimo stvoriti.

Pošto ne želimo da se efekt eksplozije zadržava na sceni, nakon tri sekunde (čvor *Delay*) ga brišemo iz scene pomoću čvora *Destroy Actor*.

## 5.3 Integracija uređaja Leap Motion i Oculus Rift

Kao što smo već ranije ukazali, Oculus Rift je već integriran u Unreal razvojnoj okolini te za njegovo korištenje u gotovoj raspakiranoj je dovoljno raspakirati aplikaciju za Windows operacijski sustav. Kada pokrenete aplikaciju, pritiskom tipki *alt+enter* bi trebalo rezultirati iscrtavanjem aplikacije kroz uređaj Oculus. Zajednica programera Unreal okoline još uvijek ima neke primjedbe za takav način izrade aplikacije specifične za virtualnu stvarnost, ali takvo je trenutno stanje. Tijekom razvoja aplikacije za Oculus, testiranje je vrlo jednostavno. Dovoljno je samo pokrenuti scenu preko VR pregleda (engl. preview) i dobit ćete prikaz preko Oculus uređaja kao što je prikazano na slici.

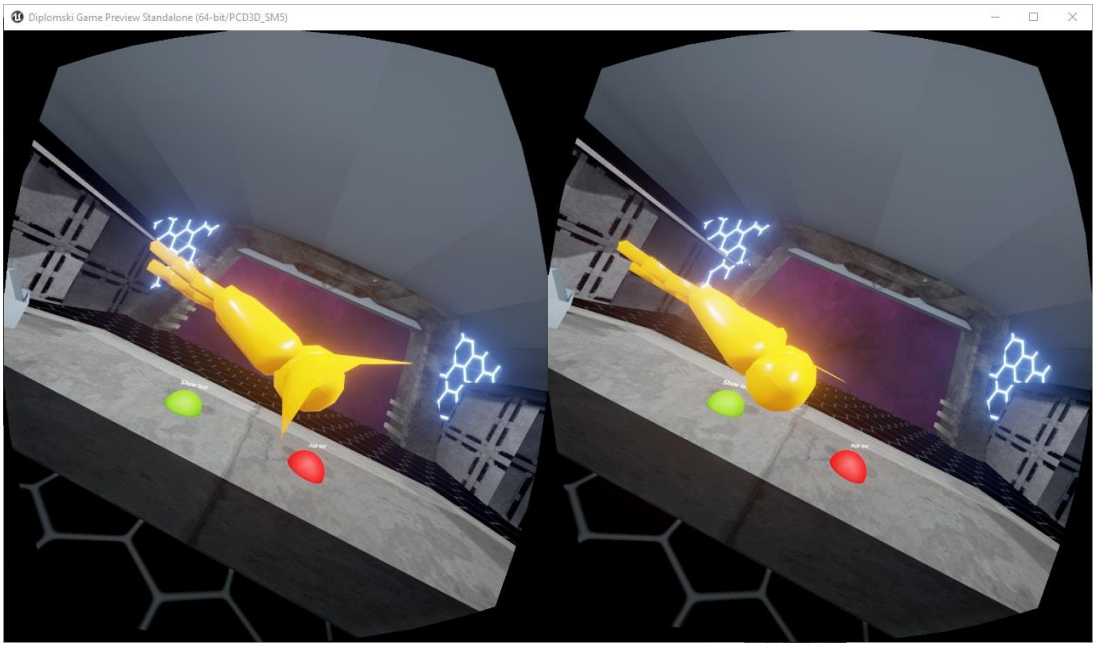



*Slika 40 Testiranje aplikacije preko Oculus uređaja s VR Preview*

Što se tiče *Leap Motiona*, za njega ne treba posebno raspakiranje jer je on integriran preko dodatka kao *Gamemode* svojstvo igre. Dakle, potrebno je stvoriti novu *blueprint* klasu tipa *Gamemode* te unutar namjestiti svojstvo Character na neki od predefiniranih likova Leap Motion dodatka. U implementaciji je korišten *LeapBasicRiggedCharacter* odnosno njegova kopija preimenovana u

*DiplLeapBasicRiggedCharacter*. Uzeta je kopija jer je u postavkama potrebno uključiti generiranje događaja preklapanja pri koliziji s drugim objektima. Bez toga gore spomenuti objekti detektori će ignorirati Leap Motion pokrete ruku. Na sljedećim slikama prikazana je LeapMotion integracija u „običnom“ i VR pokretanju.


*Slika 41 Leap Motion u ruka u sceni*




*Slika 42 Integracija Leap Motion i Oculus Rift*

S time smo namjestili sve što se trebalo namjestiti za ovaj projekt. Zadnja bitna napomena je da kretanje igrača kroz scenu potrebno je dodati *3D Character* objekt koji će se vezati na početnu poziciju scene.

# 6. Zaključak

U ovom radu možemo zaključiti da je potencijal korištenja virtualnih sučelja za široku primjenu velik. Sve velike tvrtke koje se bave računalnim igrama počinju ulagati u virtualnu stvarnost te pokazuju sve više i više interesa, a to je uvijek jedan od boljih pokazatelja u kojem smjeru svijet tehnologija ide.

Ovaj rad bavi se problematikom virtualnih sučelja, sučelja koja su realizirana u virtualnom svijetu te se vrši interakcija preko određenih medija. Ti mediji su u ovom slučaju uređaji Oculus Rift kao naglavni uređaj prikaza virtualne stvarnosti te uređaj za detekciju pokreta ruku Leap Motion. Korištena razvojna okolina je Unreal verzija 4.12. Unreal ima integriranu podršku za Oculus Rift uređaj, dok za Leap Motion je potrebno skinuti dodatak.

Za kvalitetnu realizaciju virtualnih sučelja potrebno je poznavati osnovne principe interakcije čovjeka i računala te osnove računalne grafike. Većina problematike koja je „izronila“ bila je vezana uz problematiku računalne grafike. Rad je obuhvatio tri osnovna načina upravljanja virtualnim sučeljem: pritiskom gumba, pomicanjem objekata, detekcijom pokreta objekta. Sva tri načina prikladna su za različite interakcije sa sučeljem. Npr. nelogično bi bilo odvojiti funkcionalnost otvaranja i zatvaranja vrata svemirskog broda na dva gumba koja su značajno razmaknuti u prostoru.

U pozadini logike implementiranja sučelja leži programska paradigma bazirana na događaje (engl. event-driven programming paradigm) gdje je ideja da postoje događaji, pretplatnici na događaje te ostali objekti u sceni na kojih se može utjecati. Pretplatnici na događaje reagiraju ako se izvrši događaj koji oni „slušaju“. Primjer u implementaciji bio bi kocka okidač (engl. box trigger) za otvaranje vrata.

Sve u svemu, uređaj Oculus Rift pokazao je iznimnu laganu implementaciju u Unreal razvojnoj okolini unatoč tome što je zahtjevan na programsko sklopovlje. S druge strane, Leap Motion uređaj ipak nije oduševio zbog činjenice da mu je preuzak domet skeniranja ruku kako bi se iskoristio u potpunosti za realizaciju virtualnog sučelja. Svejedno, za jednostavna virtualna sučelja to je sasvim dovoljno.

# 7. Literatura


[1] Wikipedia, Leap Motion, https://en.wikipedia.org/wiki/Leap_Motion, 15.6.2016

[2] Leap Motion Inc, Leap Motion, https://www.leapmotion.com/about-us, 15.6.2016

[3] Leap Motion Controller Review, https://www.engadget.com/2013/07/22/leap-motion-controller-review/, 15.6.2016

[4] Damir Ciganović Janković, Upravljanje gestama 3D objektom pomoću uređaja Leap Motion, 1.7.2015, http://www.zemris.fer.hr/predmeti/ra/Magisterij/15_CiganovicJankovic/Final_0036461068_56.pdf, 15.6.2016

[5] Leap Motion Controller Packaging Software, https://www.amazon.com/Leap-Motion-Controller-Packaging-Software/dp/B00HVYBWQO, 15.6.2016

[6] Virtual Reality, https://en.wikipedia.org/wiki/Virtual_reality, 16.6.2016

[7] Oculus Rift, https://en.wikipedia.org/wiki/Oculus_Rift, 16.6.2016

[8] Leap Motion VR, https://www.leapmotion.com/product/vr, 16.6.2016

[9] Leap UE4 Plugin, https://github.com/getnamo/leap-ue4, 16.6.2016

[10] Oculus Rift Overview, https://wiki.unrealengine.com/Oculus_Rift#Overview, 16.6.2016

[11] Developer VR Latest Concepts, https://developer.oculus.com/documentation/intro-vr/latest/concepts/bp_intro/, 16.6.2016

[12] Human Computer Interaction,

https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction, 26.6.2016

[13] Virtual Desktop, https://en.wikipedia.org/wiki/Virtual_desktop, 26.6.2016

[14] William R. Sherman & Alan B. Craig, Understanding Virtual Reality, http://www.sciencedirect.com/science/book/9781558603530, 26.6.2016

[15] Alex C. Peterson, Spacescape, http://www.alexcpeterson.com/spacescape/, 26.6.2016

[16] Blueprint Animated Door Tutorial, https://wiki.unrealengine.com/Blueprint_Automated_Door_Tutorial, 26.6.2016

[17] Linear Interpolation, https://en.wikipedia.org/wiki/Linear_interpolation, 27.6.2016

[18] Beam Particle Tutorial, https://wiki.unrealengine.com/Beam_Particle_(Tutorial), 27.6.2016

## Sažetak (Upravljanje elementima virtualnog sučelja uz kombinaciju uređaja Leap Motion i Oculus Rift)

Uređaji Leap Motion i Oculus Rift kao par nude mnoštvo mogućnosti u virtualnom svijetu, pa tako i u interakciji s virtualnim sučeljem. Olakšana integracija sklopovlja za Unreal razvojnu okolinu uvelike doprinosi mladim programerima da se upuste u nove tehnologije i istraživanja mogućnosti virtualnog svijeta. Virtualno sučelje polazi od istih principa kao i svako drugo sučelje u interakciji čovjeka i računala te ako se poštuju neki osnovni principi, kvaliteta programske opreme znatno je veća. U ovom radu realizirane su tri vrste upravljanja virtualnim sučeljem: upravljanje objektom, pritisak gumba i detekcija objekta. Sva tri načina upravljanja prikladni su za različite situacije u realizaciji manipulacije virtualnom okolinom.

**Ključne riječi:** Oculus Rift, Leap Motion, Unreal, virtualna stvarnost, virtualno sučelje, modeliranje, događaji

## Abstract (Controlling virtual interface elements with devices Oculus Rift and Leap Motion)

Devices Oculus Rift and Leap Motion offer a wide range of possibilities to use in virtual reality as well as in virtual intefaces. Easy to use integrated support for Unreal engine greatly contributes the cause of researching new technologies and trying them out in virtual worlds. Virtual intefaces follow the same principles of human computer interaction as any other interface. This leads to quality software solutions and implementations of virtual interfaces. In this thesis there are three types of virtual interface interactions: object manipulation, button press and object detection. Every interaction has its own situation spectrum in which optimal use is accomplished.

**Key words:** Oculus Rift, Leap Motion, Unreal engine, virtual reality, virtual interface, modelling, events