Usporedba platformi za razvoj računalnih igara Unreal i Unity

Ivor, Gradiški-Zrinski

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ivor Gradiški- Zrinski

Usporedba platformi za razvoj računalnih igara Unreal i Unity

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ivor Gradiški- Zrinski

Matični broj: 44056/15–R

Studij: Informacijski sustavi

Usporedba platformi za razvoj računalnih igara Unreal i Unity

ZAVRŠNI RAD

Mentor/Mentorica:

Dr. sc. Robert Kudelić

Varaždin, srpanj 2019.

Ivor Gradiški-Zrinski

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Rad se temelji na identificiranju razlika između dviju popularnih platformi za razvoj računalnih igara. Ideja rada je da čitatelja postepeno uvede u temu. Platforme za razvoj računalnih igara su glavna tema rada te sam krenuo s idejom da korisnik uhvati smisao istih. Igre treba razvijati po određenim metodologijama i to je mukotrpan posao. Naglasak je stavljen na razvoj, ali također i na niži nivo s ciljem dobivanja boljeg uvida u funkcioniranje posla. Uz to sam obradio par metodologija vezanih uz razvoj. Slijede generalne stvari vezane uz platforme Unreal i Unity koje su glavni dio rada te glavna tema rada.

U glavnom dijelu rada obrađuje se paralelna usporedba grafičkog sučelja. Platforme imaju jako široku primjenu te ih je teško usporediti. Odlučio sam se za usporedbu njih dviju na mogućnosti koja se zove prepoznavanje sudara. Prepoznavanje sudara je jako kompleksna tema za koju je potreban uvod koji je odrađen na jednostavnom primjeru. Nakon prepoznavanja sudara slijedi općenita usporedba u nekim bitnim komponentama, a to su grafika, korisničko iskustvo te podržane platforme.

Rad završava konačnim osvrtom na ranije spomenute dvije platforme i odlučivanje koja je bolja, barem prema mojem ukusu.

Ključne riječi:

Unreal

Unity

Prepoznavanje sudara

Wheel Colission

Računalne igre

Razvoj računalnih igara

Platforme za razvoj igara

Sadržaj

1. Uvod

Video igre su se od svojih početaka pa sve do danas drastično promijenile. Kako se mijenjaju one tako se mijenjaju te poboljšavaju načini njihove izrade. Od malih nogu sam dosta zainteresiran za računala, a pogotovo za računalne igre. Kako su godine prolazile počelo me je sve više zanimati ne samo igranje, nego i način izrade igara. To je zapravo i jedan od razloga zašto sam upisao ovaj fakultet.

Video igre danas vidimo u mnogo različitih oblika, a neki od njih su računalne, mobilne i igre za konzolu, no tu nije kraj. Za izradu pojedine igre uz početnu ideju najvažnija je vrhunska implementacija. Problem nastaje kada ideju treba pretočiti u konkretan proizvod jer bez vrhunske implementacije nema vrhunskog proizvoda. Ovdje nastupaju razne platforme koje omogućuju izradu. U raznim zanimanjima i poslovima nudi se mnogo različitih opcija za razvoj no treba znati odabrati ispravnu. Mišljenja sam da ljudi gledaju na izgradnju igara kao vrstu zabave, no danas je to preraslo u vrlo ozbiljan posao. Gledajući jednu suvremenu računalnu igru od početka te sudjelujući u testiranju iste narasla mi je želja za detaljnijim analiziranjem i istraživanjem razvoja igara.

Vođen tom radoznalošću i željom za razumijevanjem izrade računalnih igara odabrao sam ovu temu. Osim FPS-a (*eng. First Person Shooter*), obožavam i igre utrkivanja posebno Formulu 1. Na Formuli 1 je kvalitetno napravljeno sudaranje bolida. Osim sudaranja osjet podloge je također nevjerojatan, a naravno implementacija je na vrhunskom nivou. S time sam bio doista impresioniran te ću pokušati napraviti nešto poput toga na dvjema platformama, a one su Unreal i Unity. Svaka ima svoje prednosti i nedostatke koje ću pokušati pronaći tijekom izrade.

Rad će se temeljiti na usporedbi. Osim detekcije sudara naglasak će biti na korisničkom iskustvu i prednostima te nedostatcima svake od platformi. Osim tuđih osvrta i mišljenja pokušat ću dati i svoje s ciljem dobivanja boljeg dojma i šire slike o platformi.

2. Metode i tehnike rada

Tema koju sam odabrao je dosta široka i može se koristiti ne samo za završne već i diplomske radove. Platforme za razvoj su jako veliki i kompleksni programi te sam zato krenuo s istraživanjem o njima u svrhu pronalaženja najboljih područja za usporedbu. Za to sam trebao pronaći kvalitetnu literaturu što iziskuje mnogo vremena.

Pronađenu literaturu sam prvo proučavao da utvrdim da li je dovoljno kvalitetna i sadrži li ono što tražim. Kvalitetna literatura je tada preuzeta te je krenulo detaljno čitanje i proučavanje. Osim literature trebalo je kreirati korisne bilješke i ciljeve koji će biti odrađeni u radu. Tu se vidjela prava kompleksnost teme. Osim što treba biti dobro upoznat s literaturom, također treba znati i praktični dio da se jednostavne računalne igre uspješno kreiraju. Kreirao sam probne računalne igre na temelju literature i vlastitog iskustva. Te sam to pokušao spojiti u jedno. Ideja je bila dati svoje osvrte te ih potkrijepiti dokazima iz literature.

Korišteni alati:

- Unreal Engine 4
- Unity 5
- Microsoft Word
- Github

3. Općenito o računalnim igrama i njihovom razvoju

Glavni cilj ovog poglavlja je zainteresirati i uvesti čitatelja u svijet računalnih igara. Počinje s opisivanjem računalnih igara i njihovim nastankom. Budući da računalnu igru treba stvoriti slijedi opis kako se to radi u par jednostavnih koraka.

3.1. Računalne igre

Računalne igre imaju veoma dugu povijest. Korijeni računalnih igara sežu iz 1947. godine. Thomas T Goldsmith Jr. i Estle Ray Mann su dizajnirali prvu igru na računalima s katodnim cijevima. U šezdesetim godinama prošlog stoljeća računalne igre su bile relativno jednostavne i dvodimenzionalne. Pac-Man je vrlo dobar primjer nekadašnje igre. Osamdesetih godina nastupio je velik i brz napredak računalnih komponenti što se naravno odražava i na igre. Brža i jača računala bila su odlična podloga trodimenzionalnim igrama. Kao vjetar u leđa na tržište su dolazile naprednije konzole ukomponirane s boljim kontrolama te se smatraju korijenima današnjih igara.

(F. W.B.Li, 2018.,str. 1)

Danas je vrlo jednostavno naći igre koje se zadržavaju na tržištu više od 10 godina. Razlog tome je rastuća potražnja za igrama. Današnji razvoj tehnologije dovodi nas do toga da smo okupirani igrama. Teško je pronaći kućanstva u kojima nitko nije odigrao neku računalnu ili mobilnu igru.

(F. W.B.Li,2018., str. 2)

Sve današnje računalne igre su kreativniji i bolje kreirani nasljednici prvih računalnih igara. Razlikuju se ponajviše u kompleksnosti. Računala nekad i danas se doslovno ne mogu uspoređivati, a to vrijedi i za računalne igre. U današnje vrijeme na računalne igre se implementira jako kompleksna grafika s vrlo složenim teksturama, sjenama i drugim efektima. Osim slika dodaju se svi multimedijski efekti. Nekadašnji 8 bitni zvuk je neusporedivo gori od danas vrlo zastupljenog MP3 formata. Uklopiti sve te dijelove u jedan kvalitetan proizvod je zahtjevan proces za kreatore igre, ali i za današnja računala. Koriste se razni algoritmi koji taj proces olakšavaju, a implementirani su u platformama za razvoj računalnih igara. Platforme za razvoj služe da bi se optimizirao uložen trud i vrijeme s ciljem da olakšaju kreiranje konačnog proizvoda.

(F. W.B.Li,2018., str. 2)

Gledajući razne tipove igara poput akcijskih igara, avanturističkih, strategija, dobiva se dojam njihove jake podjele. Grublje se igre mogu podijeliti na igre s jednim ili pak više igrača. Mišljenja sam da igre više umreženih igrača stvaraju bolji ugođaj. Osim što pridonose kvaliteti igre one i unaprjeđuju komunikacijske vještine. Većina takvih igara tjera ljude na komuniciranje, suradnju i razmišljanje. Računalne igre s više igrača temelje se na stvarima iz normalnog života što je jedan veliki plus.

Važno je napomenuti da igre mogu biti profesija i sredstvo zarade te postoje veliki turniri u raznim igrama, a u nekim zemljama se igre smatraju sportom. Igre ponekad djeluju loše na edukaciju no pojavio se trend igrifikacije u raznim obrazovnim ustanovama u svijetu.

Slušajući predavanja o igrifikaciji koje je bilo planirano kao blagi uvod u temu dalo je dobar uvid i razumijevanje smisla igrifikacije. Iskusio sam ju na predmetu Web dizajn i programiranje i moje mišljenje je da taj trend još nije na željenoj razini, no početak je dobar. Gledajući dobre strane računalnih igara treba se zapitati kako to sve nastaje i koji je postupak, a o tome ćemo u nadolazećim poglavljima.

3.2. Metodologije razvoja računalnih igara

Razvoj računalnih igara je proces koji počinje s idejom ili konceptom. Dizajneri igara u kreiranje igre kreću s početnim skicama, pričom ili konceptom. Tvrtke imaju razne pristupe za dizajn igara ovisno o njihovim potrebama. Važno je napomenuti kako ne postoji standardizirana metoda već nekolicina raznih metoda za razvoj računalnih igara.

(S.Aslan, 2016.)

3.2.1. Vodopadna metoda

Vodopadna metoda je sekvencijalni razvojni model. Izvodi se po fazama. Zahtjev treba biti jasan prije odlaska na novu fazu jer nema povratka na prijašnju. Svaka faza se izvodi bez preklapanja s nekom prethodnom i moraju biti završene u danom vremenskom roku. Svaka faza je zaleđena prije odlaska na novu što dovodi do kasnog pronalaženja grešaka. Testiranje se izvodi samo u danoj fazi, a ne kroz cijeli projekt što dovodi do zahtjevnog i skupog otklanjanja grešaka koje mogu nastati tijekom izrade cjelokupnog projekta.

(S.Balaji,Dr.M.S.Marugaiyan,2019.)

Prednosti:

- Zahtjevi su jasni prije početka razvoja
- Svaka faza završava u određenom vremenskom razdoblju
- Linearan model koji je lako implementirati
- Minimalna količina resursa potrebna za provedbu modela

- Svaka faza prati dokumentaciju s ciljem bolje kvalitete

Nedostatci:

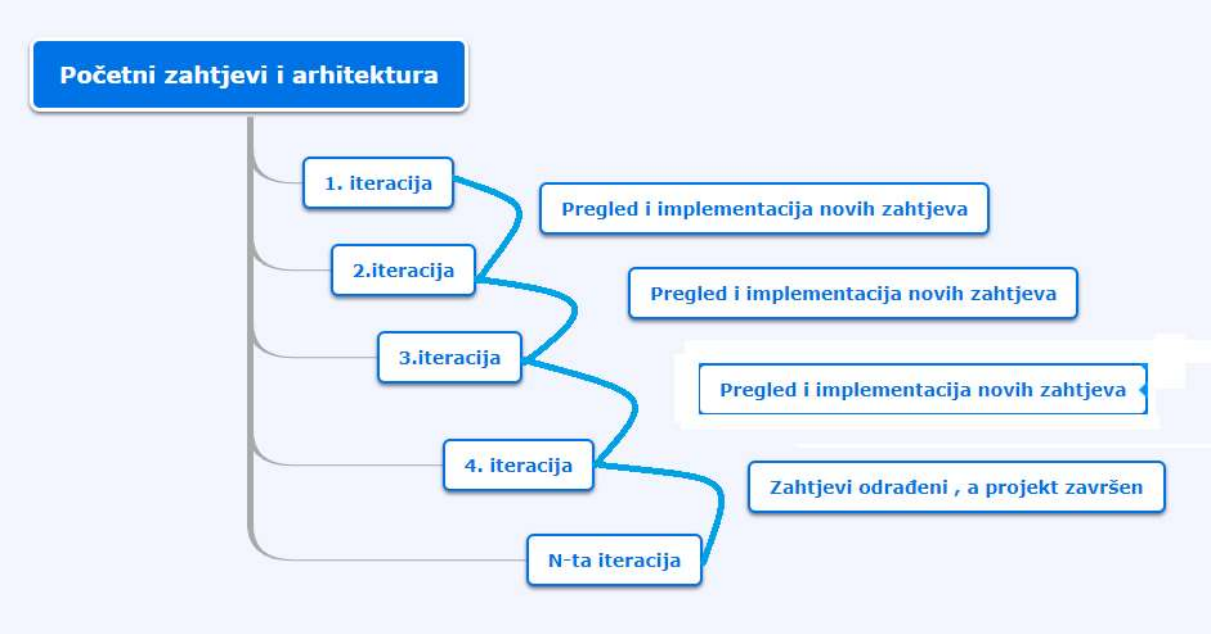
- Problemi s određenom fazom se jako teško riješe u potpunosti i vuku se nakon odjave te faze, što rezultira loše strukturiranim sustavom
- Ako klijent izmijeni zahtjev to se neće implementirati u trenutni razvojni proces što dovodi do problema

(S.Balaji,Dr.M.S.Marugaiyan,2019.)

3.2.2. Agilni razvoj programa

Agilni razvoj programa se temelji na vrlo prilagodljivom timu koji je sposoban odgovoriti na promjenjive zahtjeve. Radni program se isporučuje vrlo često s ciljem da korisnici budu zadovoljni, a ako nisu odgovor na željene promjene je vrlo brz. Bitnije je da su kupci zadovoljni, a ne strogo držanje svojih principa.

(S.Balaji,Dr.M.S.Marugaiyan,2019.)



Slika 1. Agilni razvoj programa [12.]

Prednosti:

- Sposobnost reagiranja na promjenjive zahtjeve
- Komunikacija s kupcem licem u lice te kontinuirani odnos u svrhu napretka projekta

Nedostatci:

- Teško je procijeniti vrijeme i napore kod velikih projekata te dolazi u pitanje profitabilnost projekta
- Iskusni programeri su u poziciji da donesu odluke potrebne za agilni tip razvoja što ostavlja vrlo malo mjesta za početnike

(S.Balaji,Dr.M.S.Marugaiyan,2019.)

3.2.3.SCRUM

SCRUM je poboljšanje iterativnog i inkrementalnog pristupa isporuci objektno orijentiranog programa ili nekog većeg sustava. SCRUM je metodologija upravljanja, poboljšanja i održavanja postojećeg sustava ili prototipa proizvodnje. Sastoji se od serija sprinteva te je laganog okvira. U usporedbi s tradicionalnim pristupima SCRUM ima manje dokumentacije i manje je kompliciran.

SCRUM se bazira na sljedećim varijablama:

- Zahtjevi kupaca – kako trenutni sustav poboljšati
- Vremenski pritisak – koji je vremenski okvir potreban da bi se stekla konkurentska prednost
- Konkurencija- tko je konkurencija i što je potrebno da ih se nadmaši
- Kvaliteta- što je potrebna i ciljana kvaliteta u odnosu na gornje varijable
- Vizija- koje su promjene potrebne u trenutnoj fazi za ispunjenje vizije sustava
- Resursi- koje osoblje i sredstva imamo na raspolaganju

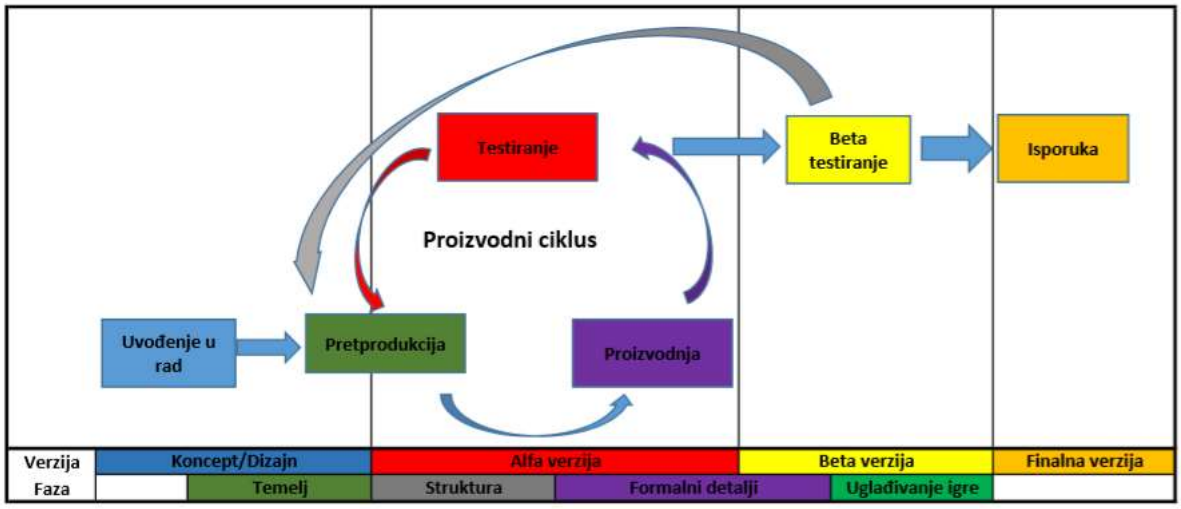
(K.Schwaber,2019.)

Tablica 1. Usporedba vodopadne i SCRUM metodologije

Vodopadna metoda	SCRUM
Fokus je na projektu	Fokus je na proizvodu
Model s konstantnim fazama	Model sa sprintevima
Ne prihvaća dobro promjene	Očekuje i dobro prihvaća promjene
Mnogo dokumentacije	Malo dokumentacije
Planiranje troškova tijekom planiranja projekta	Troškovi su izračunati tijekom projekta
Mala vjerojatnost uspjeha	Velika vjerojatnost uspjeha
Limitiranje kreativnosti i fleksibilnosti timova	Nema limita kod kreativnosti i fleksibilnosti timova

(Izvor: M. Mahalakshmi, DR. M. Sundararajan,2013.)

3.3. Razvoj računalnih igara



Slika 2. Razvojni ciklus[16.]

Na slici je prikazan životni ciklus razvoja računalni igara koji je izrađen po sljedećim principima:

- Izrađen je po ključnim aktivnostima u svakoj fazi razvojnog ciklusa
- Razvojni ciklus primjenjuje iterativni pristup koji pak omogućava veći stupanj fleksibilnosti prema promjenama
- Ovaj razvojni pristup je stvoren kako bi se riješio i ustanovio kriterij kvalitete svake razine prototipa s ciljem postizanja visoke kvalitete finalnog proizvoda

(R.Ramadam, Y.Widyani)

3.3.1. Uvođenje u rad

Prvi korak u kreiranju igre je stvaranje grubog koncepta. U konceptu se opisuje vrsta igre koja se kreira. Kao rezultat faze kreira se koncept i jednostavan opis igre.

(R.Ramadam,Y.Widyani)

3.3.2. Pretprodukcija

Pretprodukcija je prva te jedna od najvažnijih faza u ciklusu. Uključuje stvaranje i reviziju dizajna igre. Osim dizajna, kreira se i prototip igre. Glavni fokus dizajniranja je definiranje žanra igre, mehanike igre, priče, likova, tehničkih elemenata te dokumentacije vezane uz igru (engl. *Game Design Documentation*). Nakon kreiranja dokumentacije analizira se prototip na temelju kojeg se procjenjuje cjelokupna ideja. U daljnjim iteracijama prototipovi se unapređuju i ispravljaju se pogreške.

(R.Ramadam,Y.Widyani)

3.3.3. Proizvodnja

Proizvodnja je temeljni proces. Stvaraju se kodovi vezani uz izvođenje igre te se implementiraju u finalni proizvod zajedno sa svim ostalim bitnim dijelovima. Naglasak proizvodnje u ranijoj fazi je na usavršavanju raznih formalnih detalja, dodavanja novih mogućnosti i unapređivanja performansi. Aktivnost koja se također radi je balansiranje igre tako da težina odgovara igračima.

(R.Ramadam,Y.Widyani)

Kasnija faza je orijentirana na cijeli prototip. Cilj ove pod faze je pristupačnost i zanimljivost računalne igre. Ne bi se smjele raditi velike promjene da se ne naruši temeljna ideja. Smisao faze je jednostavnost, razumljivost i zanimljivost igre.

(R.Ramadam,Y.Widyani)

3.3.4.Testiranje

Testiranje se provodi u svrhu procjene upotrebljivosti i igrivosti igre. Svaka faza ima svoje metode ispitivanja. Ispitivanje formalnih detalja procjenjuje kvalitetu, funkcionalnost raznih detalja i pronalazi njihove poteškoće u izvođenju te greške. Kada su greške u radu pronađene treba ih kvalitetno analizirati i dokumentirati. Kod testiranja je bitna širina pristupa i razni načini testiranja da se osigura kvalitetan konačan proizvod.

(R.Ramadam,Y.Widyani)

Kod testiranja naprednijeg prototipa ispituju se stvari koje su se htjele postići u kasnijoj fazi proizvodnje, a vezane su za pristupačnost i zanimljivost igre. Kao rezultat testiranja kreira se izvještaj o greškama i zahtjevi za promjenama. Rezultat daje uvid u spremnost igre. Ako je igra spremna ide u fazu beta testiranja. U slučaju da nije dovoljno dobra vraća se u fazu proizvodnje gdje se dalje usavršava.

(R.Ramadam,Y.Widyani)

3.3.5.Beta test igre

U beta testiranju koriste se vanjski ispitivači. Metode ispitivanja su jednake onima u ranijoj fazi testiranja, no sada je povezano u jedan proizvod. Postoje također otvoreno i zatvoreno testiranje. U zatvorenom testiranju testeri su samo pozvane osobe odabrane od strane proizvođača. Kod otvorenog testiranja odabiru se osobe koje su se prijavile sa željom za testiranjem igre. U beta fazi testeri dobivaju više vremena za testiranje igre i otkrivanje raznih grešaka u radu. Izlaz iz ove faze čine izvješća o greškama i povratne informacije korisnika. Beta završava kada testeri daju svoja konačna izvješća ili kada je rok za beta testiranje gotov. Ovisno o rezultatu beta testiranja igra se može vratiti natrag u proizvodnju ili se kreće u objavljivanje igre.

(R.Ramadam,Y.Widyani)

3.3.6. Isporuka

Kada igra zadovolji dane kriterije i stigne u završnu fazu sprema se isporuka igre. Pod fazom isporuke smatra se lansiranje proizvoda na tržište zajedno sa njegovom dokumentacijom. Osim isporuke planira se održavanje i daljnje širenje igre.

(R.Ramadam,Y.Widyani)

4. Platforma za razvoj računalnih igara

Ovo poglavlje govori o platformama za razvoj računalnih igara. One su veoma kompleksni programi sa složenom arhitekturom. Osim općenitog opisa u poglavlju se govori o smislu platformi za razvoj igara te njihovoj arhitekturi.

4.1. Arhitektura platforma

Glavni cilj platformi za razvoj igara je razvoj raznih računalnih igara. Danas gotovo nema jednostavne igre koja nije napravljena u nekoj platformi. Platforma za razvoj računalnih igara sastoji se od podsustava, a glavni od njih su:

- Podsustav za zvuk
- Podsustav za korisnički unos
- Podsustav za fiziku
- Grafički prevoditelj
- Umjetna inteligencija
- Jezgra platforme
- Podsustav za skriptiranje
- Podsustav za mrežnu komunikaciju

(Nilson i Söderberg,2007.,str.3)

Čitajući i prolazeći kroz literaturu naišao sam na još 2 bitna podsustava koji će biti obrađeni, a to su:

- Podsustav za animiranje
- Podsustav za naknadno procesiranje

4.2. Podsustav za zvuk

Podsustav za zvuk ima zadatak da apstrahira zvučnu karticu koja je u računalu s ciljem da je mogu koristiti drugi podsustavi. Zadaća podsustava je da ostali podsustavi mogu kreirati i reproducirati zvuk neovisno o zvučnoj kartici. Podsustav za zvuk zapravo stvara objekt koji u interakciji sa zvučnom karticom kreira sučelje koje koriste svi ostali podsustavi. Zvučni podsustav koristi razne algoritme koji pomažu kod učitavanja, modificiranja te reprodukcije zvuka. Napredniji podsustavi mogu proračunati i reproducirati Dopplerov učinak, jeku te razne oscilacije zvuka.

(Nilson i Söderberg,2007.,str.12)

4.3. Podsustav za korisnički unos

Podsustav za korisnički unos ili ulazni podsustav odnosi se na načine reagiranja sustava prilikom korisničkog unosa. Pod korisnički unos se smatraju pritisci na tipkovnicu ili mišu. Poput podsustava za zvuk i podsustav za korisnički unos ima sučelje koje komunicira s ulaznim jedinicama. Opet razlikuje se u tome što drugi podsustavi ne smiju vidjeti njegove klase. Kod njegovog kreiranja treba paziti da je optimiziran za sve vrste unosa, jer bi u suprotnom moglo doći do problema kod unosa.

(Nilson i Söderberg,2007.,str.13)

Mislim da bez ovog podsustava igranje igara nije moguće. Najviše dolazi do izražaja kod računalnih igara. Na većini trkaćih igara može se upravljati vozilom na više raznih načina. Tu nastupa ovaj podsustav koji određuje i kontrolira način upravljanja koji smo odabrali. To je od velike važnosti za korisničko iskustvo, jer bez ispravnih kontrola igra ne pruža sve svoje mogućnosti.

4.4. Podsustav za fiziku

U realnom svijetu nema stvari koja nije nekako vezana uz zakone fizike. U igrama je sve mašti na volju te nam fizika nužno ne treba ili samo služi kako dobra motivacija. Postoje situacije kada se kreiraju realne situacije gdje zakoni fizike moraju biti interpretirani na idealan način. Tome služi ovaj podsustav, a njegova ne tako jednostavna zadaća je da prenese zakone fizike na objekte u računalnoj igri. Najjednostavnije rečeno nudi nam nekolicinu funkcija za simuliranje raznih sila, sudara i slično.

(Nilson i Söderberg,2007.,str.10).

4.5. Grafički prevoditelj

Grafički prevoditelj obrađuje apsolutno sve što se prikazuje na ekranu tijekom izvođenja igre. Od raznih tekstova, slika do animacija. Radi slično kao podsustav za zvuk. Stvara sučelje u kooperaciji s grafičkom karticom. To sučelje je odgovorno za slanje grafike na neki ekran ili u drugu klasu. Klase pak mogu biti razni ekrani na koje je moguće slati grafiku.

(Nilson i Söderberg,2007.,str.12)

4.6. Umjetna inteligencija

Popularnost umjetne inteligencije svakodnevno raste, ponajviše zadnjih nekoliko godina. Kvalitetno kreirana umjetna inteligencija čini samu igru mnogo boljom jer nudi veći izazov i povećava zanimanje za igru. Kvalitetni podsustavi poput umjetne inteligencije troše više računalne snage i to predstavlja veliku prepreku razvojnim programerima. Cilj je da umjetna inteligencija bude što je više moguće napredna, a da ne kompromitira igru. Umjetna inteligencija se razlikuje od podsustava kao što su fizika i grafika jer se njezin kod u nekim dijelovima može ponovno iskoristiti. Umjetna inteligencija je građena od različitih jedinica s raznim algoritmima.

(Nilson i Söderberg,2007.,str.11)

Mišljenja sam da danas svaka kvalitetnija računalna igra ima poneki segment umjetne inteligencije. Kvalitetan primjer su računalne igre utrkivanja. Na njima su izvrsno odrađeni aspekti umjetne inteligencije. Brzina i inteligencija vozača nije jednaka te se svaki ponaša drugačije. Sljedeći primjer su ratne igre gdje postoje razni likovi koji također nisu jednaki i svaki „razmišlja“ na sličan, ali ne jedinstven način.

4.7. Jezgra platforme

Gotovo svaki složeniji sustav ima potrebu za nekom vrstom koordinatora. U slučaju platformi za razvoj računalnih igara tu ulogu preuzima jezgra. Ona organizira komunikaciju među dijelovima i poduzima sve s ciljem izbjegavanja ne željenih grešaka. Ona je naj apstraktniji dio platforme za razvoj računalnih igra. Osim upravljanja s komponentama također upravlja memorijom. Pokretanje svake igre i svega vezanoga za igru kreće u jezgri.

(Nilson i Söderberg,2007.,str.4)

4.8. Podsustav za skriptiranje

Važan dio svake platforme za razvoj igara je da korisnici lako kreiraju i koriste sadržaj. Taj postupak se radi pomoću skriptiranja . Platforme za razvoj računalnih igara se najčešće sastoje od 2 dijela, a to su tvrda i meka arhitektura. Tvrda arhitektura vrši komunikaciju s raznim računalnim komponentama, dok je meka arhitektura zadužena za posebnosti u našoj igri. Pod posebnosti se misli na grafiku, zvukove i samu igrivost. Smisao meke arhitekture je da nju sami stvaramo i usavršavamo te nije unaprijed kreirana ni zadana.

(Nilson i Söderberg,2007.,str.14)

4.9. Podsustav za mrežnu komunikaciju

Danas velika većina računalnih igara nudi mogućnost mrežnog igranja. Mrežno igranje zahtijeva mrežnu komunikaciju. Dobar mrežni kod pomaže i doprinosi kvaliteti i boljoj igrivosti. Jedan takav podsustav se zove Torque Game Engine. Prednost Torque Game Enginea je da je mrežni kod robustan i omogućava komunikaciju sa gotovo svakim računalom na internetu. Radi na UDP protokolu.

(Nilson i Söderberg,2007.,str.15)

4.10. Podsustav za animiranje

S gledišta arhitekture podsustav za animiranje se sastoji od tri različita sloja. Ti slojevi su:

- Animacijski cjevovod
- Stanje stroja u radu (engl. Action state machine, ASM)
- Upravitelj animacije (engl. Animation controllers)

(J.Gregory,2016., str.604)

4.10.1. Animacijski cjevovod

Animacijski cjevovod preuzima jedan ili više isječaka i ulaznih informacija za svaki animirani lik te ih spaja u jedno. Time generira pozu kostura lika za kojeg se kreira animacija. Također izračunava i globalnu poziciju tog lika na temelju kostura. Nakon izračunate pozicije i kreiranog kostura dodaje liku vanjski izgled uz pomoć grafičkog prevoditelja. U ovom sloju najviše do izražaja dolaze kinematika i fizika lutke.

(J.Gregory,2016., str.604)

4.10.2. Stanje stroja u radu

Razni postupci koje radi lik igre poput trčanja, skakanja, hodanja i ostalog su najčešće najbolje modelirani uz pomoć ASM-a. ASM je zapravo na vrhu cjevovoda i pruža animacijska sučelja. Omogućava likovima fluidni prijelaz u razna stanja. Osim toga, danas je sasvim normalno da lik radi više različitih pokreta u istom trenutku. Dobar primjer toga je pucanje iz oružja tijekom trčanja. To se postiže s više neovisnih strojeva stanja koji kontroliraju isti lik u igri.

(J.Gregory,2016., str.604)

4.10.3. Upravitelj animacije

U mnogim platformama za razvoj igara ponašanje igrača kontrolira napredni sustav upravitelja animacije. Svaki upravitelj je prilagođen za upravljanje likom u određenom načinu rada. Može se realizirati da jedan upravitelj radi dok se lik u igri kreće, a drugi radi dok je lik u zaklonu ili vozi automobil. Upravitelji omogućuju izvrsnu fluidnost te ne usporavaju igru, ne troše puno računalne snage niti smetaju radu umjetne inteligencije.

(J.Gregory,2016., str.604)

4.11. Podsustav za naknadno procesiranje

Kada su na kostur implementirane jedna ili više animacija i kada su ti rezultati povezani u jedno pomoću linearne interpolacije često je potrebno mijenjati položaj prije grafičkog prevođenja. Za to se koristi podsustav za naknadno procesiranje. Neke od vrsta naknadnog procesiranja su sljedeće:

- Proceduralne animacije
- Inverzna kinematika
- Krpene luke (engl. *Rag Dolls*)

(J.Gregory,2016., str.594)

4.11.1. Proceduralne animacije

Proceduralna animacije je svaka animacija koja je generirana tijekom izvođenja neke igre. Nekada ručne animacije rade kod pozicioniranja kostura lika, a nakon toga se poze dalje mijenjaju proceduralnom animacijom što je zapravo naknadno procesiranje. Proceduralne animacije se mogu koristiti umjesto animiranih isječaka. Kao primjer može se uzeti drveće ili trave koje pomiče vjetar. Proceduralna animacija se koristi sama ili uz pomoć ručne animacije. Pomoću ručne animacije kreira se početni položaj drveća. Vjetar se simulira kreiranjem sinusoida koje će micati zglobove kostura drveća. Rezultat toga je pomicanje drveća dok puše lagan povjetarac. (J.Gregory,2016., str.595)

4.11.2. Inverzna kinematika

Recimo da dizajniramo animaciju na kojoj lik u igri želi pokupiti neku stvar s poda. Kreirajući tu animaciju u programu Maya, animacija izgleda izvrsno no tijekom izrade i implementacije u igri pod nije savršeno ravan ili lik jednostavno promaši objekt. Zbog te greške se mora srediti finalna pozicija kostura lika da bi bila na idealnoj poziciji za dohvaćanje željene stvari. Inverzna kinematika služi za ispravljanje takvih grešaka, a to je također vrsta naknadnog procesiranja.

(J.Gregory,2016., str.596)

Slika 3. Inverzna kinematika (6.,str.596)

4.11.3. Rag Dolls

Tijelo lika kada umre ili se onesvijesti počinje padati kao i u realnom životu. Kod takve situacije u igri cilj je da tijelo dobro reagira u dodiru s okolinom. Rag doll je kolekcija krutih tijela na kojima su primijenjeni zakoni fizike. Tijela su ograničena zglobovima kao i kod čovjeka s ciljem bolje simulacije beživotnog tijela. Prijenos podataka iz fizikalnog sustava u kostur lika je naknadno procesiranje.

(J.Gregory,2016., str.597)

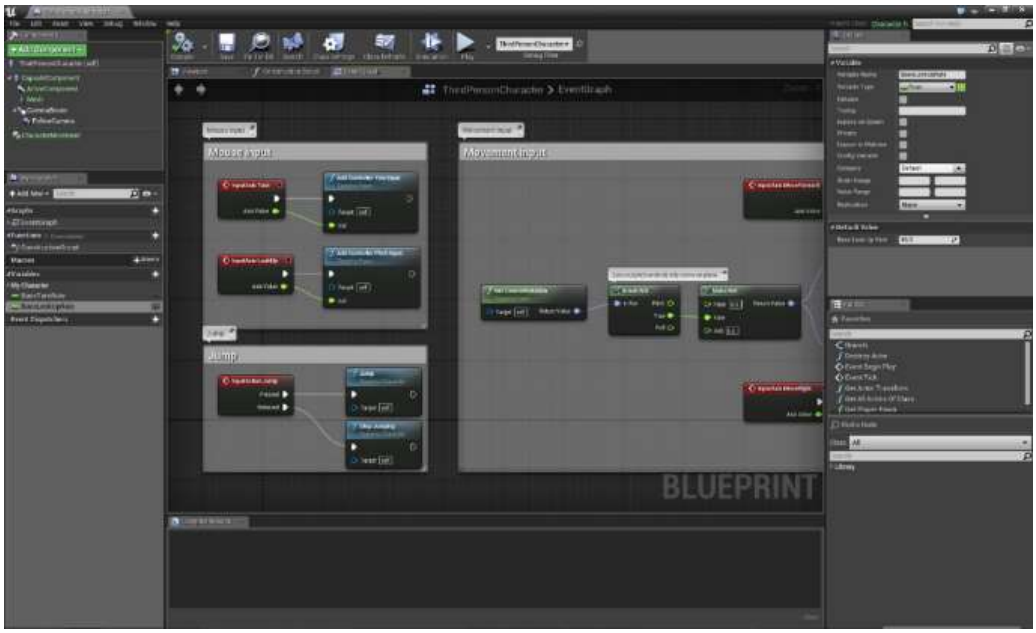
5. Platforme Unity i Unreal

U poglavlju koje slijedi upoznat ćete se s osnovnim pojmovima, izgledom i mogućnostima platforma za razvoj raznih igara.

5.1. Unreal

Unreal nudi zanimljivo i jednostavno sučelje za skriptiranje pomoću korištenja čvorova koji će kasnije kreirati cijelu skicu. Usko je povezano s objektno orijentiranim programiranjem. U shematskom planu mogu se koristiti razne klase, funkcije, događaji i naravno varijable. Namijenjen je za osnovna povezivanja no postiže i razine gdje se mogu kreirati kompleksni elementi igre. Svaki plan se koristi isključivo za jedan nivo igre. Planovi se ne koriste samo za izradu nivoa igre nego i svih ostalih stvari koje će se na nivou događati. Kada kreiramo nekog lika, njegov plan nudi pogled na razne atribute koji karakteriziraju taj lik. Zanimljiv način primjene je kod sklopki ili vrata u nekoj računalnoj igri.

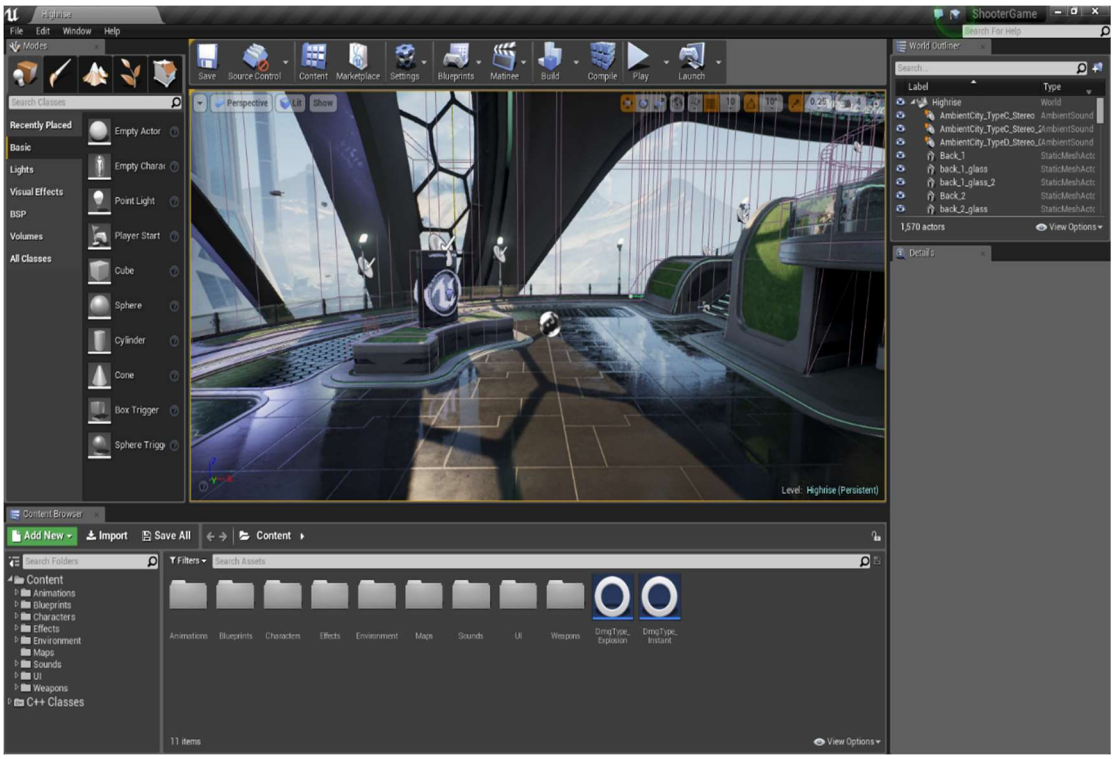
(Epic Games,2018.).



Slika 4. Shematski plan [autorski rad]

Mišljenja sam da je instalacija jednostavna i ne predstavlja nikakve probleme. Pod time smatram brzinu instaliranja i jednostavnost. Nakon početne instalacije prikazan je preglednik gdje je smještena mogućnost preuzimanja platforme Unreal. Kreatori Unreala imaju i druge

proizvode sadržane u tom izborniku kojima je lako pristupiti. Nakon uspješne instalacije cjelokupne platforme otvara se preglednik projekata i rad može početi.



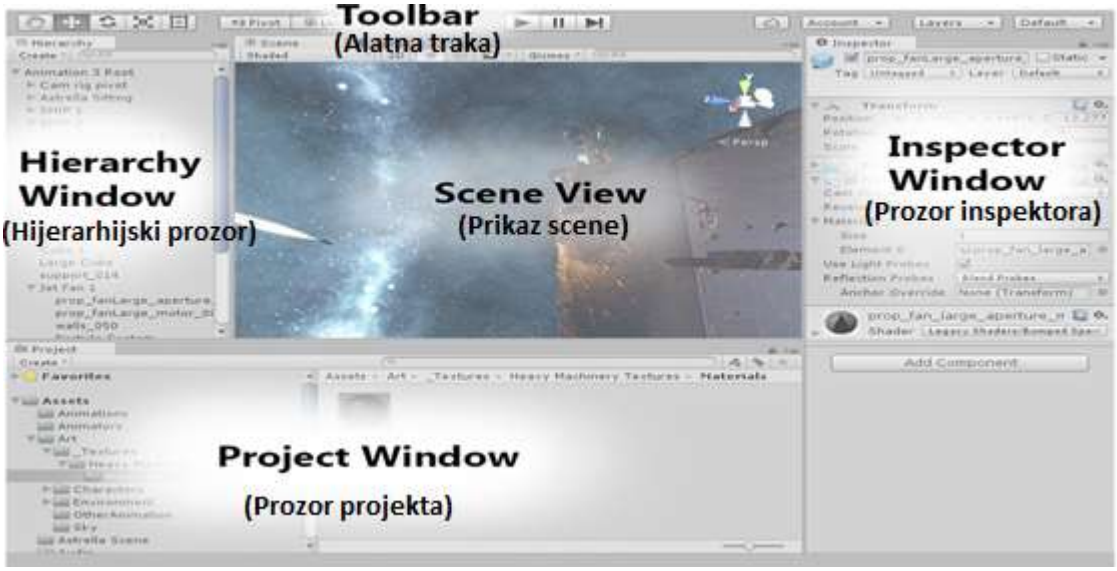
Slika 5. Unreal sučelje [1.]

U gornjem lijevom kutu prikazana je alatna traka s pokojom mogućnošću za otvaranje i spremanje projekta na kojem radimo. Pod alatnom trakom nalazi se Mode prozor gdje su sadržani razni svjetlosni i ostali efekti koji se jednostavno prenose na potrebno mjesto u projektu. U središnjem dijelu sučelja nalazi se pogled viewport koji daje uvid u cijeli projekt. Slijedi content browser gdje se pronalaze svi bitni dijelovi i datoteke potrebne za naš projekt. S krajnje desne strane imamo objekte koji su hijerarhijski povezani, a važno je napomenuti da je sve jako pregledno i jednostavno u tom dijelu. Posljednji dio je Details koji je sada prazan, a popunjava se pokretanjem projekta gdje se tada mogu vidjeti važni detalji o određenom objektu.

5.2. Unity

Unity je alat za izradu igara ne samo za računalo, nego i mobitele, Nintendo, Play Station i mnoge druge.

Mišljenja sam da Unity ima vrlo intuitivno sučelje. Naravno podijeljeno je na više dijelova koji se mogu grupirati i mijenjati po želji. Sučelje funkcionira na vrlo jednostavan način, a to je postignuto tako što se na scenu koju gledamo postavljaju razni objekti potrebni za igru. Nakon iniciranja i postavljanja tih objekata možemo isprobati njihovu međusobnu interakciju pokretanjem simulacije igre.



Slika 6. Izgled sučelja [11.]

5.2.1. Unity sučelje

Na slici 6 su prikazani najčešći i najkorisniji prozori. Svaki od njih će biti malo detaljnije odrađen.

Prozor projekta (*eng. Project Window*) je prozor koji koristimo za pristupanje raznim objektima koje koristimo u projektu. Gledajući sliku može se vidjeti sličnost s Windows Explorerom, a naravno radi i na sličan način. Sadrži ploču na lijevoj strani koja prikazuje hijerarhiju projekta. Odabirom neke mape njezin sadržaj se prikazuje na desnom dijelu prozora. Također klikom na mapu dobiva se pogled i na ugniježđene mape. Kada se određeni sadržaj otvori u desnoj strani pišu nam podaci o njemu. Prikazano je da li je to neka skripta, materijal ili slično. Također ako neki sadržaj često koristimo može se dodati u favorite radi bržeg pristupa istom.

Prikaz scene (*eng. Scene Window*) je prozor koji nam nudi pogled na scenu koju trenutno radimo. Uz njega odmah dolazi i Game Window koji prikazuje kako se igra izvodi na odabranoj sceni. Pogled na scenu se koristi za postavljanje svih objekata te manipuliranje njima. Osim toga nudi i postavljanje svjetla prema našim željama. Game Window prikazuje produkt naših kamera prethodno postavljenih u pogledu na scenu. Sadrži i gumb za pokretanje koji prikazuje kako se naša igra odvija što je odlično za pronalaženje grešaka.

Hijerarhijski prozor (*eng. Hierarchy Window*) sadrži popis objekata u trenutno odabranoj sceni. Pomoću ovog prozora lako se pristupa objektima na sceni. Osim lakšeg pristupa prikazuje nam objekte roditelje i objekte djecu što je jako bitan dio kod dizajniranja igre.

Prozor inspektora(*eng. Inspector Window*) je prozor koji prikazuje razne podatke za odabrani objekt. Osim pogleda na podatke i vrijednosti objekta nudi i njihovo mijenjanje. Uz sve gore navedeno ovaj prozor prikazuje i skripte koje su vezane na taj objekt.

Alatna traka(*eng. Toolbar*) omogućuje pristup najvažnijim radnim značajkama. Na njoj se nalaze alati za manipuliranje scenom i objektima. Sadrži i neke važne gumbe poput gumba za pristup raznim Unity uslugama i izbornik za promjene vidljivosti slojeva.

(Unity Documentation,2019.)

Mišljenja sam da je to jedna moćna mogućnost i velika prednost Unitya. U srednjoj školi sam radio s CAD (*eng. Computer-aid-design*) programima. Neki od njih su imali mogućnost slične implementacije slike u projekt, no ne na tako visokom nivou. Zbog tog nedostatka mislim da Unity može naći primjenu u raznim strukama koje nemaju veze s dizajnom igara. Možda nije dovoljan da se stvori neki novi proizvod, ali za prototip može vrlo dobro poslužiti.

6. Prepoznavanje sudara

Projektiranje učinkovitog sustava za prepoznavanje sudara je pomalo nalik na slaganje slagalice odnosno puno komada mora biti posloženo na pravi način da bi se vidjela prava slika. Kod prepoznavanja sudara imamo nekoliko čimbenika koji utječu na odabir vrste prepoznavanja sudara, a oni su:

- Domena aplikacije – geometrijski prikazi korišteni za scene i objekte imaju izravan utjecaj na korištene algoritme. Kada su postavljeni s manje ograničenja treba koristiti više općenitijih prepoznavanja sudara koji mogu imati posljedice na performanse
- Vrste upita - što su upiti detaljniji to nam je potrebno više računalne snage. Osim upita treba pripaziti i na strukture podataka koje se koriste jer svaka ne podržava iste upite
- Parametri simulacije okoline- simulacija sadrži par važnih parametara koji izravno utječu na prepoznavanje sudara. To uključuje broj objekata, njihovu veličinu, položaj, kretanje te da li su kruti ili fleksibilni
- Performanse sustava – prepoznavanje sudara u stvarnom vremenu radi pod strogim vremenskim ograničenjem. Bitan je dobar kompromis i balans da se zadovolje tražene performanse
- Robusnost- sve aplikacije nemaju jednake zahtjeve za simuliranje zakona fizike. Uzmimo na primjer igru izgradnje kuća. Tamo i najmanja greška može napraviti veliku finalnu grešku ako jedna cigla prodre u drugu. U sportskim igrama pak ako lopta slučajno završi usred mreže to nije neki velik problem i ne utječe na daljnji tok igre

(C.Ericson,2019.,str.9)

6.1. Slijedno i istovremeno pomicanje objekata

U stvarnom vremenu se objekti kreću istovremeno tijekom određenog vremenskog okvira. Računalna simulacija mora odrediti dolazi li do sudara dva objekta u nekom vremenu. Simuliranje uzastopnog pomicanja objekta može postati vrlo skupo, a ako neki objekt već leži na podlozi odnosno već je u željenom sudaru tada dolazi do problema. U sljedećem koraku će se prepoznati sudar objekta i podloge i ta simulacija bi testirala jednake korake zauvijek. Jedno od rješenja za ovaj problem je korištenje široke faze za identificiranje objekata koji bi mogli

djelovati unutar grupe, a ignorirali bi objekte iz drugih grupa. S tim se dobiva mogućnost da se simulacije unutar grupa izvode u različitim brzinama. U oba slučaja istodobna ažuriranja i provjere ostaju skupe i često su rezervirane samo za čvrsta tijela. Igre nisu samo simulacije krutih tijela stoga bi se previše vremena i napora izgubilo na simulaciju ovim načinom. Alternativa za to je slijedno pomicanje.

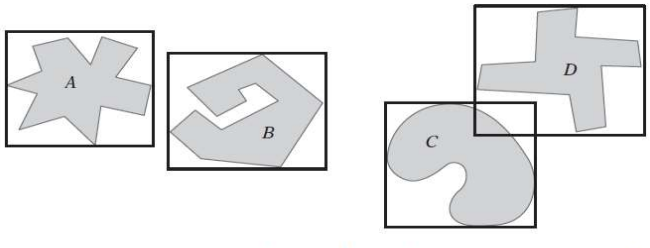
Kod slijednog pomicanja objekti se pomiču jedan po jedan u nekom određenom vremenskom okviru, a sudari se detektiraju prije odlaska na novi korak. Slijedno pomicanje nije uvijek točno jer se objekti mogu sudariti s nekima koji još nisu pomaknuti. Ako se taj sudar pojavi dovoljno je poništiti taj zadnji korak i usporediti ga s istovremenim pomicanjem radi preciznije detekcije sudara i problem je riješen.

(C.Ericson,2019.,str.16)

Slika 7. Sudar objekata [17.]

6.2. Granični volumen

Izravno testiranje geometrije dva objekta je vrlo skupo i zahtjevno pogotovo kada se objekti sastoje od mnogo različitih geometrijskih tijela. Da se smanje troškovi testira se granični volumen objekta. Granični volumen je jednostavna enkapsulacija jednog ili više objekata složenije prirode. Ideja je da se pomoću graničnog volumena izvode jeftinija i brža ispitivanja. Kada je sudar prepoznat testovi traju duže jer se vrše s većom preciznošću.



Slika 8. Granični volumen[15.]

6.2.1. Željene karakteristike graničnog volumena objekta

Svi geometrijski oblici ne mogu biti jednako efektivni kao granični volumen objekta, a ovo su neke od stvari koje moraju zadovoljiti:

- Što više odgovarati obliku objekta
- Biti jednostavne za izračun
- Biti jednostavne za rotaciju i transformaciju
- Koristiti malo memorije

Temeljna ideja je da granični volumen smanji troškove i prethodi kompliciranijim testovima. Osim da prethodi geometrijskim testovima cilj je da i on kao test bude što precizniji. Ta preciznost pak povlači za sobom kompromis između troškova i točnosti. Granični volumen se izračunava po koraku prethodne obrade, a ne u vrijeme izvođenja.

(C.Ericson,2019.,str.77)

Slika 9. Granični volumen [15.]

6.3. Wheel Collider

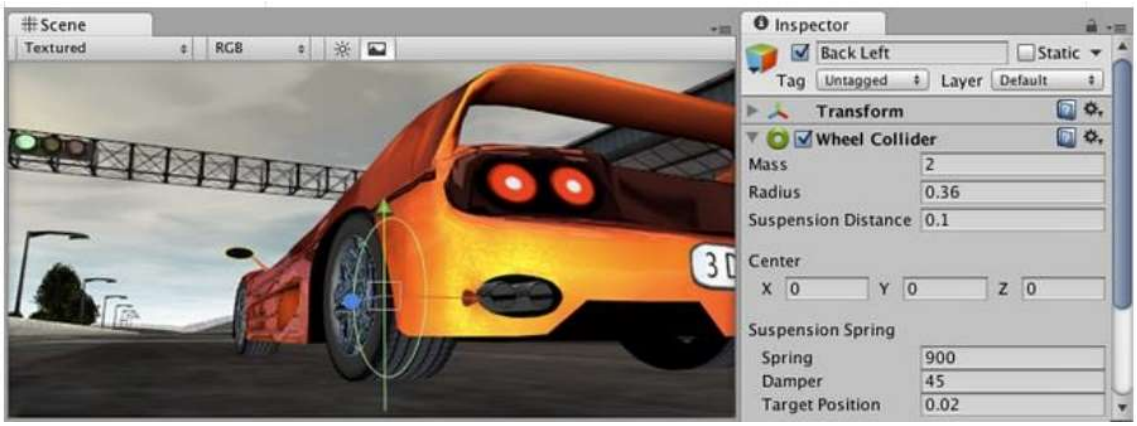
Wheel Collider je jedna od instanci prepoznavanja sudara koja je predviđena za vozila koja se kreću po tlu. Njegov je cilj da prepozna sudaranje između nekog automobila odnosno njegovih kotača s podlogom.

Wheel Collider je nastao kao kombinacija detekcije sudara, fizike kotača i klizanja kotača po podlozi. Naravno može se koristiti i za druge objekte, ali mu je primarna uloga korištenje kod vozila s kotačima.

(TerraUnity,2018.)

Ostvaruje interakciju automobila ili nekog drugog zamišljenog vozila s njegovom podlogom. Ta mogućnost je stvarno moćna jer preslikava mnogo realnih faktora. Traže se podaci poput

mase vozila, promjera kotača, koeficijenta odbijanja kotača koji u simulira koliko su pune gume na vozilu. Osim kotača postoji i simuliranje ovjesa poput dužine amortizera i njihove tvrdoće.



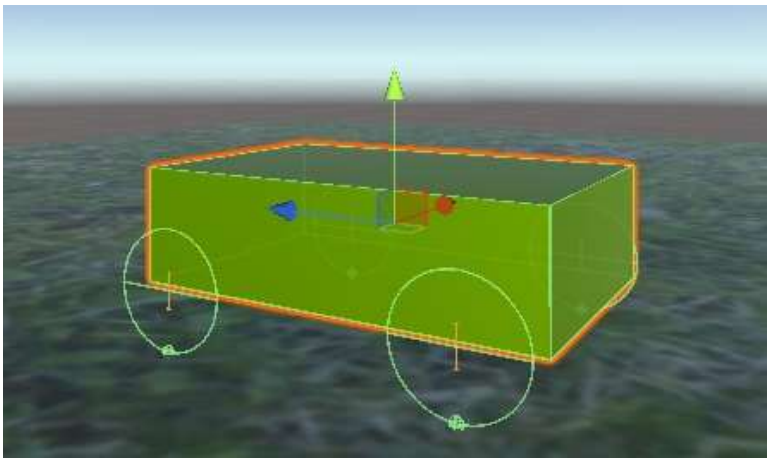
Slika 10. Wheel Collider [3.]

7. Wheel Collider u platformi Unity

U ovom poglavlju govorit će se o prednostima i nedostacima platforme Unity. Prednosti i nedostatci analizirat će se na ranije spomenutoj mogućnosti zvanoj Wheel Collider. Struktura ovog poglavlja je takva da će se izgraditi model automobila, a tijekom izrade će se objašnjavati stvari te problemi koji nastaju putem kreiranja primjera.

7.1. Kreiranje automobila

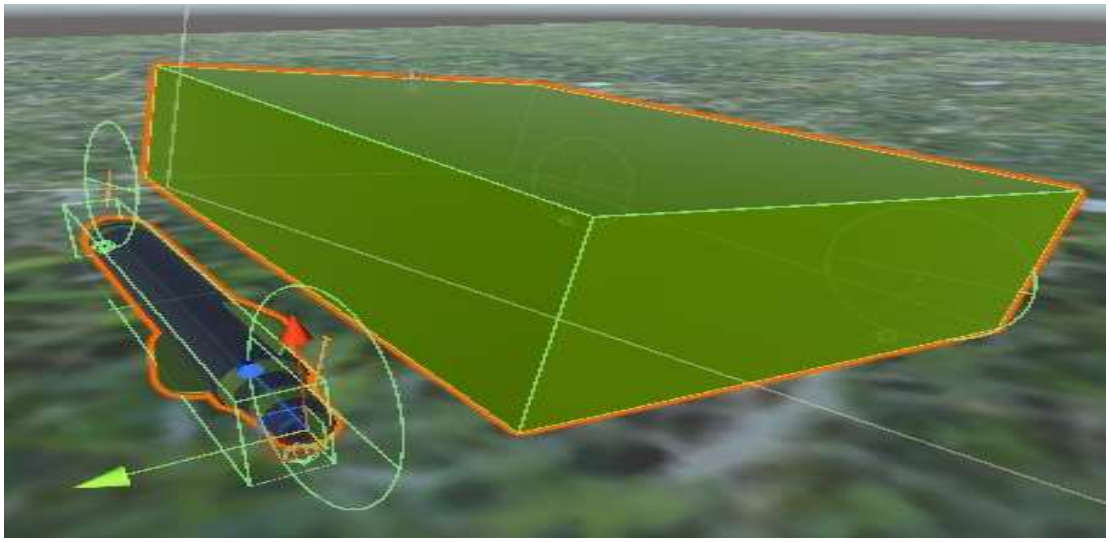
Kod kreiranja automobila dizajn nije od presudne važnosti jer je cilj prikazati sudaranje. Automobil simbolizira kvadar s priključenim kotačima. Kotači su zamišljeni kao djeca kvadra koji je šasija. Važno je napomenuti da Unity podržava kreiranje objekata u nekim drugim 3D programima, što je iznimno korisno. U ovom dijelu Unity platforma djeluje jako jednostavno te nudi mnogo slobode.



Slika 11. Šasija automobila [autorski rad]

Travnata podloga je osnova na koju se naslanja vozilo u svrhu simulacije. Na oblik su prvotno dodani „Box Collider“ i „Rigid Body“. Box Collider je jednostavan za korištenje i služi za prepoznavanje sudara dvaju objekta, a Rigid Body služi za simuliranje težine vozila i utjecaja gravitacijske sile na vozilo. Kod Unreala je mnogo jednostavnije kreirati početni model sličan ovome i uštedi se ponešto vremena.

Za testiranje prvotnih postavki potrebno je kreirati prepreke za izrađeno vozilo. Iz slike koja slijedi vidi se prepoznavanje sudara radi. Pokretanje i zaustavljanje vozila još nije realizirano te se vozilo pomiče pomoću koordinatnih osi. Slika dokazuje da vozilo prepoznaje i reagira na objekt koji mu se našao na putu.



Slika 12. Prepoznavanje sudara [autorski rad]

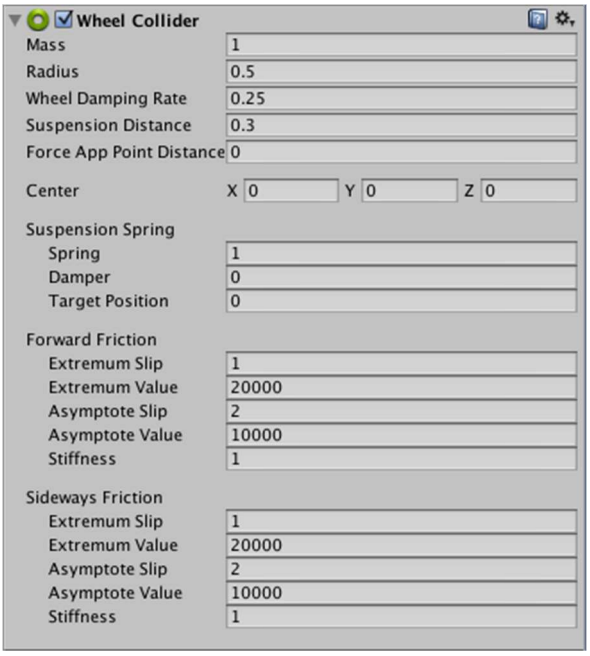
7.2. Wheel Collider u platformi Unity

Tablica 2. Prikaz svojstava za Wheel Collider

Svojstvo:	Funkcija:
Mass	Ukupna masa kreiranog vozila
Radius	Promjer kotača
Wheel Damping Rate	Vrijednost odskakivanja kotača
Suspension Distance	Najveća vrijednost širenja ovjesa
Force App Point Distance	Parametar koji opisuje gdje će se primijeniti sila na kotač. Gleda se od baze kotača. Standardno je zadana s 0, ali postoje neka vozila gdje se mora dati malo više od njegovog centra mase s ciljem bolje simulacije
Center	Sredina kotača kada se gleda objekt
Suspension Spring	Sila kojom ovjes pokušava doći do zadane pozicije, najčešće kod poravnatog automobila ili drugog vozila
Spring	Sila kojom ovjes pokušava doći do zadane pozicije, što je veća to brže dolazi do ciljane pozicije. Može se usporediti s tvrdoćom ovjesa.
Damper	Brzina kojom ovjes dolazi do zadanog položaja. Upravlja silom ovjesa.
Target Position	Ciljana pozicija u koju se ovjes želi postaviti
Forward/Sideways Friction	Svojstva vezana uz trenje guma. Velika sila kojom guma djeluje na podlogu rezultira malim poduprav ljanjem i obrnuto

(Izvor: Unity Documentation, 2018.)

Za pokretanje vozila korištena je skripta koja se nalazi pod prilogom 1. Vozilo se pokreće bez problema, no prepreke na kojima je planiran sudar ne reagiraju pravilno zbog ne točnih parametara. Postavljanje istih također nije bilo baš lako. Parametri kojima se određuje reakcija vozila na ostale objekte su vrlo kompliciranі. Mišljenja sam da je početniku vrlo teško savladati postavljanje parametara. Naravno stvar je moguće srediti da radi izvrsno što zahtijeva poveće znanje i iskustvo. Osobno nisam uspio kvalitetno posložiti ovjes da dobro odgovara na prepreke. U svrhu testiranja koristila se skripta u prilogu 2. Nije previše kompliciranа no daje kvalitetne rezultate za testiranje.



Slika 13. Wheel Collider u platformi Unity [3.]

Scena za testiranje je puna raznih prepreka i skokova s ciljem utvrđivanja reakcije vozila na razne stvari. Zakoni fizike pomoću ove dvije skripte i korištenjem Rigid Body mogućnosti nisu savršeno preneseni na igru, no naravno nisu ni toliko kompliciranі. Mišljenja sam da vozilo realno ponaša na skoku i da prelazi prepreke dovoljno dobro. Poguralo je također lakše objekte od sebe, a naišlo na probleme kada je objekt bio teži. Kao što sam rekao zbog više raznih faktora nisam uspio napraviti realan prikaz zakona fizike i sudaranja no za neki početni pokušaj radi na relativno visokoj razini.

8. Wheel Collider u platformi Unreal

U nadolazećem poglavlju govori se o Wheel Collideru u platformi Unreal. Pokušat će se analizirati mogućnosti platforme, težina rada u njoj i njezine mogućnosti. Slijedi testiranje Wheel Collidera u svrhu usporedbe s istim u platformi Unity.

8.1. Kreiranje automobila

Unreal je po ovome pitanju iznimno prilagodljiv. Ima mnogo načina za kreiranje raznih objekata. Najjednostavnije no naravno i najlošije je kreiranje pomoću zadanih objekata. Dobro je za neki primjer i razna testiranja, no mislim da nije za ništa više od toga. To nije razlog lijenosti ljudi koji su tu platformu radili nego su je uspjeli dobro prilagoditi programima za izradu 3D modela. Pod tim programima ima i onih koji su besplatni pa je razvoj time omogućen svim pojedincima. Mišljenja sam da je Unreal kompleksniji i teško je shvatiti na prvu te treba lagano krenuti u njega uz korištenje raznih pomoći.

Kreiranjem automobila koji sliči prvom primjeru u Unityu mislim da ovdje Unreal ima prednost jer je dizajniranje u njemu na puno višem nivou, jest da je teže no nudi razne mogućnosti. Izgradnja iziskuje mnogo vremena i pripreme što početnicima može predstavljati problem. No ako neko samo želi vidjeti kako automobil radi u globalu nudi se mogućnost korištenja predloška. Predložak nije vrhunski, no nudi dobar pregled i široke mogućnosti testiranja. Zbog manjka znanja u području 3D modeliranja također sam ga i sam koristio.

Slika 14. Automobil u Unrealu [autorski rad]

Zanimljiva stvar na svim ostalim objektima u Unreal platformi su već primijenjeni zakoni fizike. Osobno mi se sviđa ta mogućnost jer nije kompliciranа za modificiranje. Gledajući neke

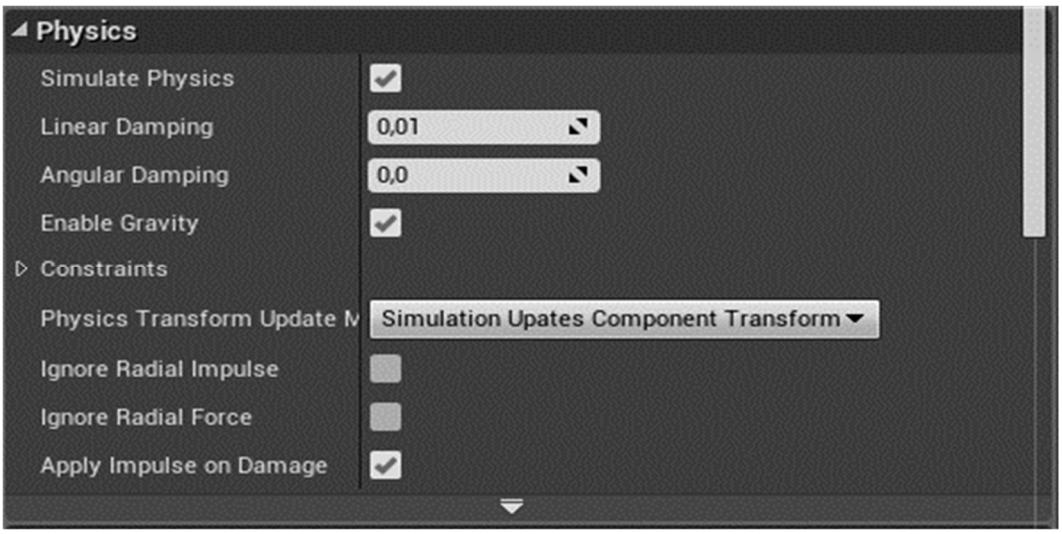
mogućnosti koje se kod sudaranja mogu mijenjati nailazim na dosta sličnosti s Unity platformom počevši od mase vozila, ovjesa i sličnog.

Slika 15. Prepoznavanje sudara [autorski rad]

Na slici 15 vidi se interakcija vozila, kotača i podloge. To su sve već ranije postavljene postavke. Nakon nekih testiranja i izmjene parametara mislim da je Unreal ovdje jači. Ne samo po reakcijama raznih elemenata u testiranju nego i sa širinom koju nudi.

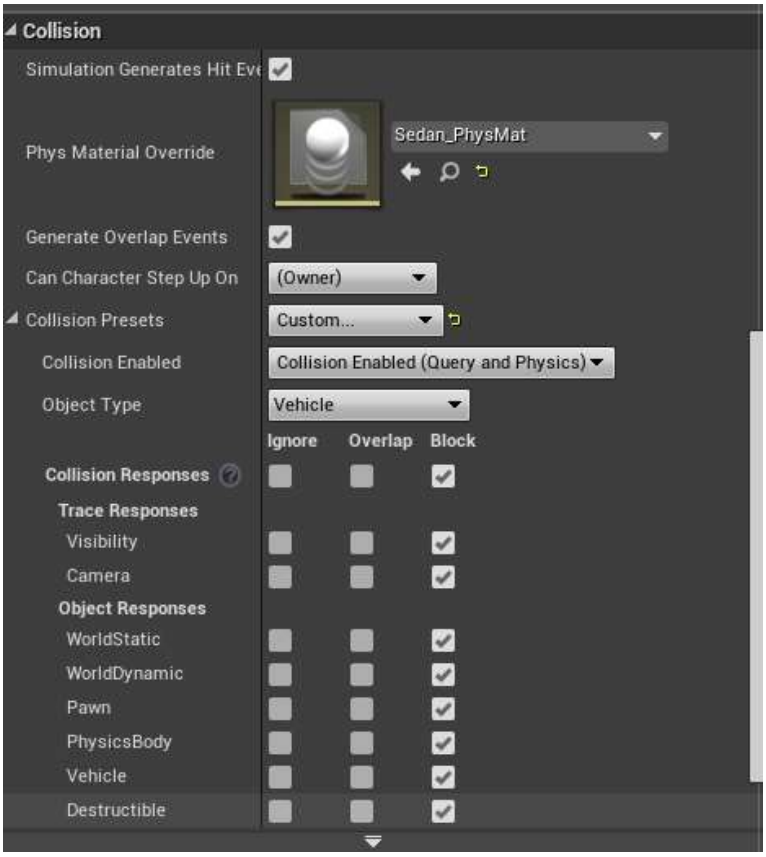
8.2. Wheel Collider u platformi Unreal

Komponenta za pomicanje vozila je već bila ugrađena no može se mijenjati bez problema. Za potrebu vožnje ne treba dodavati nove mogućnosti osim ako to ne želimo. Upravljanje i izmjena postavki je napravljena kvalitetnije nego u ranije spomenutoj platformi Unity. Unreal također ima više komponenata za simulaciju. Prva bitna je Physics, a prikazana je pod ovim tekstom na slici 16.



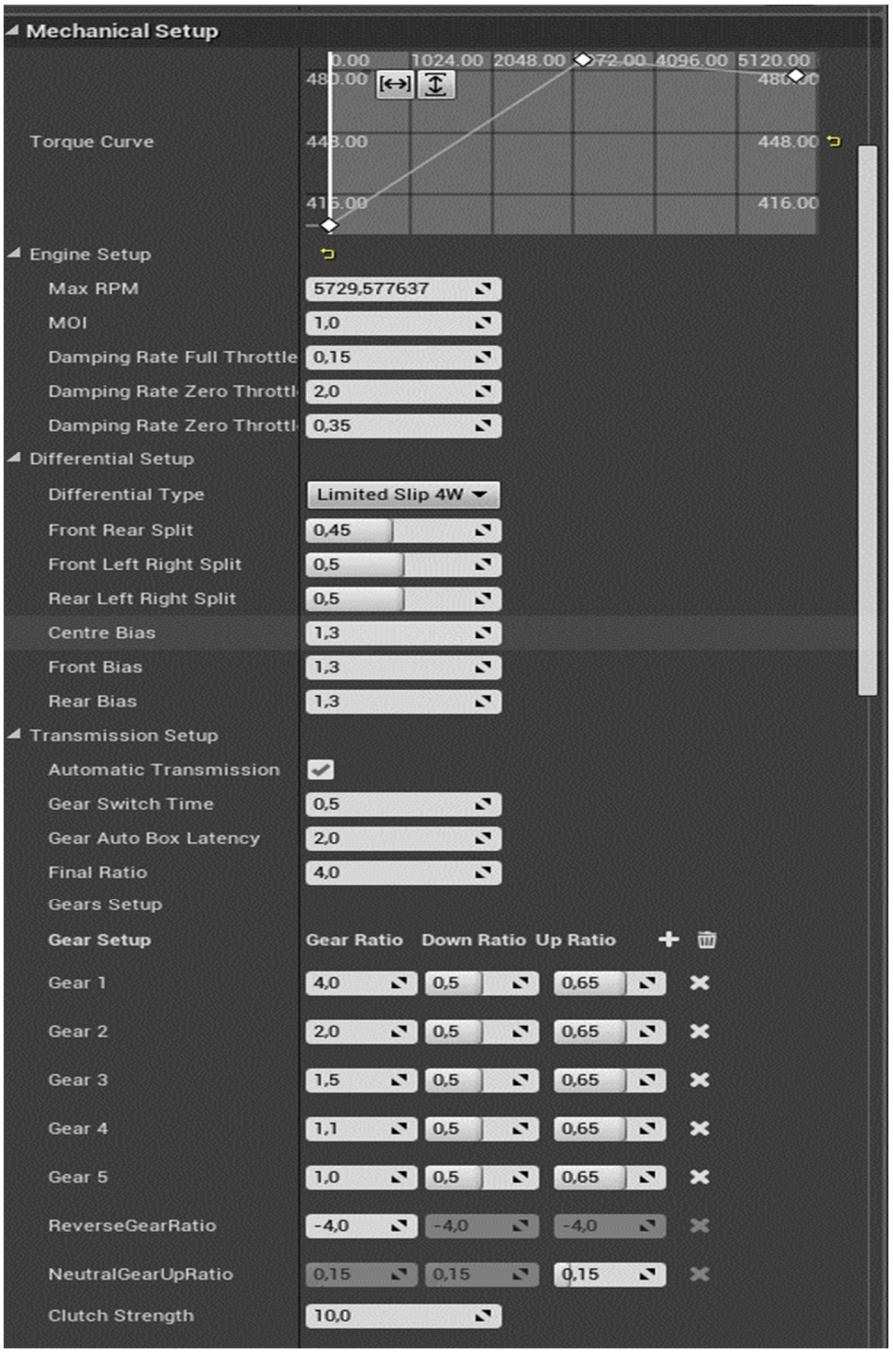
Slika 16. Fizika u Unreal platformi [autorski rad]

Kada se radi o sudaranju objekata također postoje razni predlošci za vozila i ostale objekte. Osim toga nudi se i mogućnost da ručno odaberemo kako će se naš objekt reagirati u koliziji s drugim objektima.



Slika 17. Collision u Unreal platformi[autorski rad]

Stvari poput mase dodaju se na drugim mogućnostima koje slijede u nastavku. Ovaj dio je jako jednostavno napravljen i prema mojem iskustvu radi vrlo dobro. Iznimno mi se sviđa što se može određivati broj okretaja automobila, pogon koji želimo, omjer između brzina. Komponente koje se ovdje mijenjaju ne razlikuju se mnogo od neke dobre računalne igre utrkivanja gdje se traži da igrač zna naći idealne postavke za svoje trkaće vozilo. Važno je napomenuti da se svaka promjena dobro osjeti što olakšava testiranje vozila.



Slika 18. Mehaničke postavke [autorski rad]

Detaljnije slika 18 počinje s krivuljom momenta što svako vozilo ima i vrlo je realno napravljeno. Slijedi broj okretaja stroja odnosno mašine u automobilu koji se također jednostavno odredi i mijenja. Kada se rade igre utrkivanja diferencijal je dosta bitan pogotovo kod vozila s različitim pogonima i performansama. Ovdje je to detaljno odrađeno, no s naglaskom na jednostavnost. Na mjenjaču se odabire tip koji može biti automatski ili manualni kao i na realnim vozilima. Valja napomenuti omjere među brzinama. Kao i sve do sada izmjene su lagane i brze.

9. Općenita usporedba platforma Unity i Unreal

Poglavlje koje daje odgovore na cjelokupnu usporedbu performansi platformi. Nakon prepoznavanja sudara koja je jedna mogućnost tih platformi sada je cilj još malo se osvrnuti na sve što nude i u čemu je koja bolja. Ovdje ću se vratiti na neke stvari iz uvoda koje sam već spomenuo ranije radi kvalitetnije konačne usporedbe.

9.1. Programiranje

Unity ima mogućnost kreiranja skripti za objekte u C#. C# je relativno jednostavan za korištenje kod Unitya. Koristi se Start() za pokretanje neke scene ili igre dok Update() služi za pomicanje nekog objekta. Kod Unreala postoji već ranije spomenut sustav shematskih planova. Jako je jednostavan za slijediti. Za ljude koji nisu baš spretni u programiranju shematski planovi uvelike olakšavaju kreiranje igre. Velika prednost Unreala je jednostavnost pronalaženja raznih novih mogućnosti.

(A. Hajalie,2015.,str.1)

Moja prednost u ovom segmentu ide platformi Unity iz sljedećih razloga. Jednostavnija je za početnike i lakše je doći do nekog početnog proizvoda koji tada stvori daljnju motivaciju za rad. Naravno podrška i razni materijali su mnogo dostupniji. Također C# ima veću širinu u odnosu na C++ kod Unreala. Shematski plan je kvalitetna mogućnost, ali ne dovoljna da Unreal u ovom segmentu pobijedi.

Ocjena:

Unity: 5

Unreal: 3

9.2. Grafičko sučelje

Na prvi pogled slična su oba grafička sučelja no postoji i podosta razlika. Problem kod Unreala je manjak pogleda na igru. Vidi se stvarno malo što ne ostavlja dobar dojam o izgledu igre. Problem se javlja kod uređivanja kuta svjetlosti na nekom objektu. Postoji pregled scene, ali to nije dovoljno da se vidi na pravi način. Kod Unity platforme taj problem je izbjegnut. Postoji pogled na scenu i pogled na igru. Za grafiku naravno pomaže, no kod otkrivanja grešaka je gotovo savršeno. Osim toga Unreal nailazi na probleme kada se želi pratiti i nalaziti greška na nekom objektu. Unity može pratiti svaki objekt koji se jednostavno odabere dok se u platformi Unreal igra mora zaustaviti i istraživati.

(A. Hajalie,2015.,str.2)

Ja se također slažem da su sučelja slična kao i pretraživanje. Unity mi je u ovom dijelu mnogo jači i bolji jer gdje god postoji razlika između sučelja tamo je sučelje Unitya bolje. Mislim da je preglednost svakog objekta bolja, čak se i lakše snalaziti po raznim scenama. Unity radi onako kako bi svako grafičko sučelje trebalo. Osim toga i otkrivanje grešaka je mnogo lakše pomoću odabira objekta.

Ocjena:

Unity: 5

Unreal:3

9.3. Grafika

Slika 19. Usporedba grafike[18.]

Grafičke mogućnosti su mnogo jače na platformi Unreal. Nudi mnogo raznih materijala, ali i kreiranje novih ovisno o potrebi. Svi materijali se mogu po potrebi lako modificirati. Slično je programiranju, a naravno sve na temelju shematskih planova. Ima razne varijable koje se koriste, a neke od njih su : boja, metalik boja, oštrina, prozirnost, odsjaj… Kod Unity platforma također postoji kreiranje materijala i rad sa svojstvima, ali na nešto nižem nivou.

(A. Hajalie,2015,str.3)

Moj dojam je da Unreal vodi u grafici, ali i svjetlosti. Što se materijala tiče također jer se u Unityu sve mora naknadno preuzimati ili kreirati što je dosta kompliciranо u početku. Kreiranje

i postavljanje kamera bilo mi je nešto malo praktičnije u Unrealu no skoro su identični. Gledajući neke kreacije na slici 19 vidi se koja je platforma bolja. Ovo su moje ocjene, a slike provjerite sami. Važno je napomenuti da danas Unity može parirati Unrealu u dosta segmenata izgleda scena, no problem je što se na većini konzola i računala ne može tečno pokrenuti igra kada su detalji odrađeni na visokom nivou.

Ocjena:

Unity: 3

Unreal: 5

9.4. Sadržaj za korisnike

Nevjerojatna stvar kod Unitya je dućan. Prepun je kvalitetnih objekata i stvari za razvoj igara. Pod to spadaju razni 3D modeli, animacije, zvukovi, skripte te ostalo. Naravno neki sadržaj se plaća no još uvijek je mnogo toga besplatno. Korisno je kada trebamo nešto specifično ili već bezbroj puta korišteno. Umjesto traženja po internetu jednostavno se legalno preuzme. Unreal također ima svoj dućan s raznim sadržajima. Problem je u tome što nije toliko opširan.

(A. Hajalie,2015,str.4)

Mislim da je Unity u ovom dijelu mnogo jači. Koristio sam oba dva dućana i mišljenja sam da je Unity dućan stvarno moćan. Prepun besplatnih i korisnih stvari koje u samo par aktivnosti završe u igri. Naravno sve je legalno, a i ljudi su dosta susretljivi. Dobio sam dojam da je Unity bolji za početnike i nezavisne programere. Unreal je stariji od Unity platforme i koriste ga iskusniji izdavači. To je jedan od razloga da je dućan sirov i oskudan. Naravno da iskusniji izdavači kreiraju sve za svoje potrebe na način koji im odgovara te da to ne dijele okolo.

Ocjena:

Unity: 5

Unreal: 4

9.5. Podržane platforme

Tablica 3. Podržane platforme

Vrsta platforme za igru	Unreal	Unity
računala	Windows, Mac OS, Linux, HTML5	Windows, Mac OS, Linux, WebGl
mobiteli	iOS, Android	iOS, Android, WindowsPhone (Universal Windows Platform)
konzole	PlayStation 4, Xbox One, Nintendo Switch	Nintendo Switch, PlayStation 4, Xbox One
virtualna stvarnost	Playstation VR, Oculus Rift, SteamVR, Samsung Gear VR, Magic Leap Google VR	Windows Mixed Reality. Oculus Rift. PlayStation VR, Daydream, Samsung Gear VR, Apple ARkit, Google ARCore Magic Leap
televizori	tvOS	Android TV tvOS

(Izvor: Unity Technologies 2019., Epic Games, 2019.)

Gledajući dvije platforme vrlo je teško pronaći razliku kojom bi neka od njih dobila prednost u ovom segmentu. Naravno detaljnije gledajući tablicu može se zamijetiti lagana prednost Unity platforme.

Ocjena:

Unity: 5

Unreal: 4

10. Zaključak

Usporedba platformi za razvoj računalnih igara Unreal i Unity je jako široka, ali zanimljiva tema. Rekli smo da su računalne igre danas vrlo dobro zastupljene te me također jako zanimaju. One se rade na platformama koje sam pokušao što bolje opisati. Slijedio je uvod u Unity pa nakon toga i Unreal. Istražujući te platforme došao sam do nekih novih saznanjima o njima.

Malo po malo krenula je izrada testnih primjera nekih igara. Cilj je bio napraviti funkcionalan automobil u obje platforme. Gledajući Unity dizajn nije težak, ali sam s time bio i prije upoznat. Kod Unreala sam imao više problema, ali je njihov predložak odradio dio posla. Pošto sam već prije bio upoznat s Unity platformom mogu reći da sam se tamo bolje snalazio i možda imao malo blaži pogled na nju. Zbog toga sam za detaljniju usporedbu uzeo sudaranje kotača. Nisam imao priliku raditi s time, ali moram priznat da je to opširna tema o kojoj nisam znao gotovo ništa. Radeći u obje platforme bilo je dosta sličnih dijelova i kada sam nešto napravio u jednoj nije bilo problema kod druge platforme no također bilo je i razlika od kojih neke nisam mogao jednostavno savladati. U sudaranju kotača pobjedu bi dodijelio platformi Unreal jer je imala mnogo više raznih mogućnosti koje su implementirane tako da traže da se ide u dubinu. To je bilo izvrsno jer uz proučavanje svih tih stvari usput i učimo.

Gledajući općenite dijelove svake od platformi za mene je pobjednik Unity. Ne kažem da je Unreal lošiji nego da mi Unity više odgovara. Uspoređujući sam i čitajući neke radove pokušao sam dati svoj pogled na platformu u usporedbi s nekom drugom osobom. Svakako prije uporabe neke platforme treba se upoznati s njezinim prednostima i nedostatcima te odabrati onu koja nam više odgovara. Na kraju krajeva obje su besplatne i izbor je na nikome osim na nama.

11. Popis literature

[1.] Epic Games, Unreal Engine Wiki.[Na internetu]. Dostupno na: https://wiki.unrealengine.com/Main_Page [Pristupljeno: 22.7.2018.]

[2.] F. W.B.Li., Computer Games. [Na internetu]. Dostupno na: https://community.dur.ac.uk/frederick.li/paper/game.v2.pdf [Pristupljeno: 20.7.2018.]

[3.] Unity Documentation, Wheel Collider. [Na internetu]. Dostupno na: https://docs.unity3d.com/560/Documentation/Manual/class-WheelCollider.html [Pristupljeno: 2.8.2018.]

[4.] A. Hajalie, Unity vs Unreal Engine 4. [Na internetu]. Dostupno na: http://web.eecs.umich.edu/~sugih/courses/eecs441/common/Unity-Unreal.pdf [Pristupljeno: 3.8.2018.]

[5.] B. Nilson, M. Söderberg, Game Engine Arhitecture. [Na internetu]. Dostupno na: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.459.9537&rep=rep1&type=pdf [Pristupljeno: 29.7.2018.]

[6.] J.Gregory, Game Engine Arhitecture 2nd Edition, CRC Press,2015.

[7.] S. Aslan, Methodologies for Development and Software Quality.[Na internetu]. Dostupno na: https://vtechworks.lib.vt.edu/bitstream/handle/10919/73368/Aslan_S_D_2016.pdf [Pristupljeno 28.7.2018]

[8.] R.Ramadam,Y.Widyani, Game Development Life Cycle Guidlines, [Na internetu]. Dostupno na: https://www.researchgate.net/publication/271548605_Game_development_life_cycle_guidelines [Pristupljeno 28.7.2018.]

[9.] Unity Technologies, Unity Features, [Na internetu] Dostupno na : https://unity3d.com/unity [Pristupljeno : 11.6.2019.]

[10.] Epic Games, Platform Support,[Na internetu] Dostupno na: https://www.unrealengine.com/en-US/what-is-unreal-engine-4 [Pristupljeno : 11.6.2019.]

[11.] Unity Documentation, Learning the interface ,[Na internetu] Dostupno na: https://docs.unity3d.com/Manual/LearningtheInterface.html [Pristupljeno: 11.6.2019.]

[12.] S.Balaji,Dr.M.S.Marugaiyan, Waterfall Vs V-model vs Agile, [Na internetu] Dostupno na: http://jitbm.com/Volume2No1/waterfall.pdf [Pristupljeno 11.6.2019.]

[13.] K.Schwaber, SCRUM Development Process, [Na internetu] Dostupno na: https://link.springer.com/chapter/10.1007/978-1-4471-0947-1_11 [Pristupljeno: 11.6.2019.]

[14.] M.Mahalakshmi, DR.M.Sundararajan, Traditional SDLC Vs Scrum Methodology, [Na internetu] Dostupno na : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.413.2992&rep=rep1&type=pdf [Pristupljeno 11.6.2019.]

[15.] C.Ericson, Real Time Collision Detection, [Na internetu] Dostupno na: http://www.r-5.org/files/books/computers/algo-list/realtime-3d/Christer_Ericson-Real-Time_Collision_Detection-EN.pdf [Pristupljeno : 12.6.2019.]

[16.] Razvojni ciklus [Slika] Dostupno na: https://personanonymous.wordpress.com/2013/07/17/a-guidelines-of-game-development-life-cycle-v2-0/ [Pristupljeno:28.7.2018.]

[17.] Sudar objekata [Slika] Dostupno na: https://fifaforums.easports.com/en/discussion/175610/will-the-frostbite-engine-fix-the-body-glitch-examples-inside [Pristupljeno 12.6.2019.]

[18.] Usporedba Unreal i Unity [Slika] Dostupno na: https://youtu.be/Hr5IOEQI7eg?t=777 Vrijeme pojavljivanja:12:57 [Pristupljeno 12.6.2019.]

12. Popis slika

13. Popis tablica

14. Prilog 1. Skripta za pokretanje

```
using UnityEngine;
using System.Collections;
public class RearWheelDrive : MonoBehaviour {
	private WheelCollider[] wheels;
	public float maxAngle = 30;
	public float maxTorque = 300;
	public GameObject wheelShape;

	// here we find all the WheelColliders down in the hierarchy
	public void Start()
	{
		wheels = GetComponentsInChildren<WheelCollider>();

		for (int i = 0; i < wheels.Length; ++i)
		{
			var wheel = wheels [i];

			// create wheel shapes only when needed
			if (wheelShape != null)
			{
				var ws = GameObject.Instantiate (wheelShape);
				ws.transform.parent = wheel.transform;
			}
		}
	}
	public void Update()
	{
		float angle = maxAngle * Input.GetAxis("Horizontal");
		float torque = maxTorque * Input.GetAxis("Vertical");

		foreach (WheelCollider wheel in wheels)
		{
			if (wheel.transform.localPosition.z > 0)
				wheel.steerAngle = angle;

			if (wheel.transform.localPosition.z < 0)
				wheel.motorTorque = torque;

			// update visual wheels if any
			if (wheelShape)
			{
				Quaternion q;
				Vector3 p;
				wheel.GetWorldPose (out p, out q);

				// assume that the only child of the wheelcollider is
the wheel shape
				Transform shapeTransform = wheel.transform.GetChild
(0);
				shapeTransform.position = p;
				shapeTransform.rotation = q;
			}

		}
	}
}
```

15. Prilog 2. Skripta za ovjes

```
using UnityEngine;
using System.Collections;

[ExecuteInEditMode()]
public class EasySuspension : MonoBehaviour {
	[Range(0, 20)]
	public float naturalFrequency = 10;

	[Range(0, 3)]
	public float dampingRatio = 0.8f;

	[Range(-1, 1)]
	public float forceShift = 0.03f;

	public bool setSuspensionDistance = true;

	void Update () {
		// work out the stiffness and damper parameters based on the
better spring model
		foreach (WheelCollider wc in
GetComponentsInChildren<WheelCollider>()) {
			JointSpring spring = wc.suspensionSpring;

			spring.spring = Mathf.Pow(Mathf.Sqrt(wc.sprungMass) *
naturalFrequency, 2);
			spring.damper = 2 * dampingRatio * Mathf.Sqrt(spring.spring
* wc.sprungMass);

			wc.suspensionSpring = spring;

			Vector3 wheelRelativeBody =
transform.InverseTransformPoint(wc.transform.position);
			float distance = GetComponent<Rigidbody>().centerOfMass.y
- wheelRelativeBody.y + wc.radius;

			wc.forceAppPointDistance = distance - forceShift;

			// the following line makes sure the spring force at maximum
droop is exactly zero
			if (spring.targetPosition > 0 && setSuspensionDistance)
				wc.suspensionDistance = wc.sprungMass *
Physics.gravity.magnitude / (spring.targetPosition * spring.spring);
		}
	}
```