

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
ANALIZA I RAZVOJ PROGRAMA

AIRBENDER VR

UNREAL ENGINE
Naputci za izradu

Dorian Čajko Ivan Huzjak Denis Jocković Filip Novački Luka Štefanić

VARAŽDIN

SADRŽAJ

1	UVOD U INDUSTRIJU, STVARANJE IGARA TE OKRUŽENJE	6
1.1	DIZAJN VIDEO IGARA	6
1.2	VR	7
1.3	UNREAL ENGINE	8
1.4	KREIRANJE PROJEKTA	8
1.5	ZAKLJUČAK I OTVORENA PITANJA	8
2	KREIRANJE I KRETANJE GLAVNOG LIKA	9
2.1	VR <i>blueprint</i>	9
2.1.1	Postavljanje visine igrača te postavke VR uređaja	9
2.1.2	Stvaranje i vezanje kontrolera za VR uređaj	10
2.1.3	Proces teleportacije	11
2.1.4	Rukovanje kontrolerima	13
2.1.5	Postavljanje rotacije mjesta teleportacije	14
2.2	KONTROLER <i>blueprint</i>	15
2.2.1	Dobivanje lokacije i rotacije za mjesto teleportacije	15
2.2.2	Aktiviranje vizualizacije mjesta teleportacije	16
2.2.3	Dektiviranje vizualizacije mjesta teleportacije	17
2.3	ZAKLJUČAK	17
3	DETEKCIJA POKRETA KONTROLERA	18
4	STVARANJE META	19
5	NAPADAČKE MOĆI	20
6	STVARANJE PRIJETNJI ZA IGRAČA I PRIKAZ ZDRAVLJA (HP)	21
7	OBRAMBENI MEHANIZMI ZA IGRAČA	22
8	BODOVANJE I PRIKAZ BODOVA	23
9	IZRADA KRAJOLIKA NIVOVA	24
10	IZBORNICI	25
11	”UMJETNA INTELIGENCIJA” – PROTIVNIK SE KREĆE	26
12	VIZUALNI EFEKTI	27
13	ZVUK	28

UVOD

Ovaj priručnik za programiranje projekta namijenjen je prvenstveno studentima tehničkih područja kao nit vodilja kroz stvaranje jednog projekta. Igra koja se stvara ovim putem temelji se na popularnoj crtanoj seriji Avatar, a ime, u originalu *Avatar, the Last Airbender*, vrlo se prikladno poklopilo s imenom kolegija, *Analiza i razvoj programa*, iliti skraćeno AIR.

Podrazumijeva se da čitatelj ovog priručnika poznaje osnovne koncepte programiranja te se oni ovdje neće u detalje objašnjavati. Naglasak će se staviti na koncepte koji se upotrebljavaju u izradi igre, dakle oni vezani za Unreal Engine, razvoj igara itd.

Glavni tekst sadržavat će objašnjenje postupaka koji se koriste u izradi igre. Moguće je da se koraci malo promijene u određenim verzijama softvera koji se koriste, no pretpostavlja se da će sve ostati slično jer se ne koriste jako opskurni koncepti.

Na kraju priručnika nalazi se kazalo pojmova kako bi se određeni pojam mogao lakše pronaći ukoliko je samo spomenut negdje u tekstu, a nije iz naslova jasno o kojem se pojmu točno radi.

Dodatna objašnjenja mogu se vidjeti izdvojena sa strane kako bi se dodatno povezalo objašnjeno s drugim konceptima.

1 UVOD U INDUSTRIJU, STVARANJE IGARA TE OKRUŽENJE

Programiranje igara se po mnogočemu razlikuje od programiranja poslovnih aplikacija. Pristup i razmišljanje su drugačiji. Dok za poslovne aplikacije je važno usredotočiti se na podatke, u igrama je važno i misliti na tzv. *delivery*, odnosno kako će korisnik doživjeti aplikaciju. Grafiku, fiziku, objekte i druga zajednička obilježja objedinjava *game engine*. Unreal Engine je jedan od najpopularnijih rješenja za razvijanje igara te ćemo se u ovom priručniku koristiti njime.

Jedna od posebnosti Unreal Enginea je što koristi tzv. *blueprinte*, alat kojim se odnosi između objekata programiraju ne kodom, nego povlačenjem odnosa između objekata koji su reprezentirani vizualno što olakšava predočavanje te dodatnu razinu apstrakcije od programskog jezika C++.

Prije nego što će biti objašnjeni detalji o Unreal Engineu objasniti će se i osnove dizajna video igara (en. *game design*), kako teče proces izrade igara te na što se sve treba pripaziti kod izrade igara.

U Unreal Engineu moguće je u potpunosti programirati u C++-u, no zbog kompleksnosti jezika i pristupačnosti *blueprinta* uglavnom ćemo se baviti vizualnim skriptiranjem.

Kao dodatna motivacija za stvaranje igara je činjenica da se ta industrija u zadnjih osam godina tržišna vrijednost igara udvostručila, a predviđa se da će se u iduće tri godine (2020. – 2023.) vrijednost utrostručiti. Broj aktivnih igrača u svijetu raste velikom brzinom te se u tim podacima prepoznaje perspektiva te industrije. Iz tog je razloga dobro poznavanje ovog sektora, ako ne zbog želje za radom u njoj, barem zbog opće kulture.

1.1 DIZAJN VIDEO IGARA

Dizajn video igara kao proces teško je definirati. Dizajn obuhvaća sve ono što se događa za vrijeme stvaranja igre, dakle počinje idejom i temom, nastavlja razvitkom i na kraju stvaranje verzije igre koja se izdaje i distribuira igračima.

Stvaranje ideje Ideja se razvija na razne načine – može doći kroz razgovor s bliskim osobama, može se razviti kroz *brainstorming*, može doći kroz bljesak inspiracije ili na neke druge načine. Ono što je obično veći problem je doći do jedinstvenog i kvalitetnog sadržaja jer je do dana današnjeg stvoren ogroman broj igara.

Osnovne funkcionalnosti i mehanike Mehanike nam govore kako će objekti međusobno reagirati, koja je njihova interakcija, što će sve likovi raditi u igri itd. Funkcionalnosti su alati kojim se rješavaju neki problemi, npr. kretanje glavnog lika, obrana lika od napada, umjetna inteligencija itd.

Mehanike igre govore kako će se igra ponašati u određenim trenucima, odnosno na neke korisničke akcije. Mehanike se uvelike razlikuju između različitih žanrova i to ih često čini specifičnima. Primjeri mehanika su izazovi na *bossovima*, način na koji igrač ima interakciju s raznim objektima u igri, kako koristi objekte, kako se objekti ponašaju prema njemu itd.

Skica i razrada igre U ovom dijelu procesa dizajna video igre kreiraju se grube skice buduće igre i donose se razmatranja kako će izgledati pojedini element igre. Skice nisu detaljne, ali nam uvelike olakšavaju daljnji rad u nekom od alata. Tu se razvijaju likovi, razine, moći itd.

Game Design Document Nakon što je uvodni dio napravljen, vrijeme je da se pojedini elementi malo detaljnije razrade. Taj dokument naziva se *Game Design Document*, ili kratko – GDD. To je detaljan dokument koji, između ostalog, sadrži:

- naziv igre
- sažetak igre
- funkcionalnosti
- mehanike
- opis likova
- dizajn razina

Cilj dokumenta je olakšati svima razvoj igre tako da lakše zajedno surađuju. U njemu su opisane sve glavne komponente

1.2 VR

Igra-projekt kojeg će ovaj priručnik opisati bit će implementiran za VR. VR je skraćenica na engleskom od *virtual reality* što znači da se imitira stvarnost u virtualnom okruženju. Cilj VR-a je korisniku stvoriti osjećaj kao da je u stvarnom svijetu podražujući više osjetila. Glavni uređaj koji je svojevrsno obilježje VR koncepta su naočale, odnosno *headset* koji prekriva cijelo vidno polje i korisniku omogućuje da vidi sliku kao realnost tako da svako oko vidi posebnu sliku. Ovaj projekt koristit će i kontrolere koji omogućuju korisniku pokretanje.

Pomoću kontrolera moguće je pratiti pokrete ruku. U nekim igrama to je već iskorišteno za precizno bacanje projektila, predmeta, uzimanje predmeta itd., a u našem projektu to će biti glavni okidači za usmjereno bacanje moći.

U sklopu ovih lekcija neće se raditi GDD, ali za bilo koji ozbiljan projekt dobro je imati taj dokument kao zamjenu za dokumentaciju kako bi se olakšalo snalaženje u projektu i kodu.

VR može vrlo lako učiniti neiskusnog igrača omamljenog, odnosno može osjećati glavobolju, vrtoglavicu i slične simptome ukoliko nije naviknut na virtualnu stvarnost. Postoje mnoge tehnike kako se to može ublažiti. U ovom projektu će se pokušati voditi računa o tome koliko god je moguće.

1.3 UNREAL ENGINE

Kako bi se instalirao Unreal Engine, potrebno je otići na mrežnu stranicu stranicu unrealengine.com. U gornjem desnom kutu su opcije **Sign in** te **Download**. Prijava je potrebna za pokretanje Unreal Enginea tako da se korisnik mora registrirati prije ili kasnije.

Kod odabira licence potrebno je pripaziti koja se odabire. *Publishing license* ona je koja se odabire ukoliko se proizvod namjerava prodati, a *Creators license* ukoliko se namjerava raditi nemonetiziran rad. Studenti su navedeni u obje kategorije jer se *engine* ne mora plaćati ako se koristi za projekt koji još ne stvara profit.

Dalje je potrebno registrirati se te slijediti uputstva kod instalacije. Unreal Engine radi na operacijskim sustavima Linux te Windows.

Instalacija za Linux ponešto je složenija te je potrebno kompajlirati cijeli projekt. To uzima poprilično vremena i resursa, a sadrži i nešto više koraka koji se ovdje neće opisati jer se orijentiramo na Windows operacijske sustave

1.4 KREIRANJE PROJEKTA

1.5 ZAKLJUČAK I OTVORENA PITANJA

Ovo je bio uvod i treba nas pripremiti za sve ono što dolazi u idućoj lekciji, a to je postavljanje prvih objekata u svijet.

Ono čime se ovdje može dodatno obogatiti projekt su:

- otvori prostor za dodatna poboljšanja prvi primjer
- prijedlog drugog poboljšanja
- za učinak koji želimo treći primjer

2 KREIRANJE I KRETANJE GLAVNOG LIKA

Unreal engine ima *preset* za svaku popularnu vrstu video igre pa se tako populariziranjem VR igri razvio i VR *preset*. U tom *presetu* već je implementirana funkcionalnost kretanja, odnosno teleportacije, no i dalje je važno razumjeti na koji način kretanje funkcionira kako bi smo mogli lakše implementirati ostale funkcionalnosti u skladu s onime što već imamo.

Kretanje je implementirano u dva odvojena objekta koji međusobno komuniciraju. Jedan objekt je kamera kroz koji igrač prati svijet, dok je drugi objekt kontroler, odnosno kontroleri na kojima se nalazi tipka za kretanje.

Ta su dva objekta povezana zato što se pomoću vektora kontrolera i kolizije određuje mjesto na koje će se postaviti igrač nakon uspješne teleportacije. Kretanje, odnosno teleportacija se izvodi kroz niz petlji.

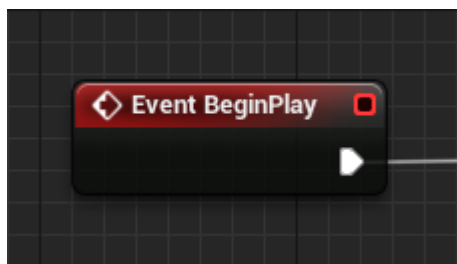
2.1 VR *blueprint*

Element, odnosno **actor** iz naslova je VR uređaj u virtualnom prostoru. Taj **actor** predstavlja stvarni VR uređaj iz realnog svijeta virtualnom svijetu. U nastavku slijedi opis svih funkcionalnosti koje su postavljene pomoću blueprinta u navedenom actoru.

2.1.1 POSTAVLJANJE VISINE IGRAČA TE POSTAVKE VR UREĐAJA

Sustav događaja je sustav koji omogućuje programerima da se neka funkcija pokrene nakon što se nešto dogodi.

Sve što će se sada opisati okida se na događaj početka igre. Događaja u Unreal engineu ima mnogo. Za sada će se koristiti događaji početak igre, *tick* ili *input* od strane igrača.



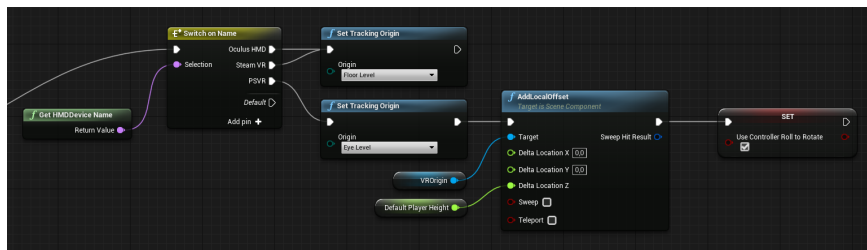
Slika 1: Događaj za početak igre

Prvi skup čvorova koji slijedi nakon čvora početka igre služi za prilagodbu visine igrača ukoliko se radi o PSVR uređaju jer PSVR ne zna gdje se nalazi pod te može sam odrediti visinu na kojoj se nalazi igrač. Praćenje (eng. *tracking*) *headseta* i kontrolera se na VR uređajima vrši pomoću namjanje

Play Station Virtual Reality

dviju kamera koje onda mogu uspoređivati dvije iste scene iz različitog kuta gledanja te na taj način dobiti informacije o trodimenzionalnosti prostora kojeg pokrivaju.

Ovisno o uređaju određuje se ime uređaja koji se kasnije koristi za identifikaciju scenskog elementa kojem je taj uređaj pridružen. Drugim riječima, stvarni uređaj se veže s virtualnim te oni postaju isti element. Na taj način pokretanjem glave koje mijenjaju lokaciju fizičkog uređaja izazivaju i mijenjanje lokacije u virtualnom prostoru. Sljedeći čvor prikazuje implementaciju opisanog.



Slika 2: Čvor koji prati pokrete glave i sinkronizira pokrete s igrom

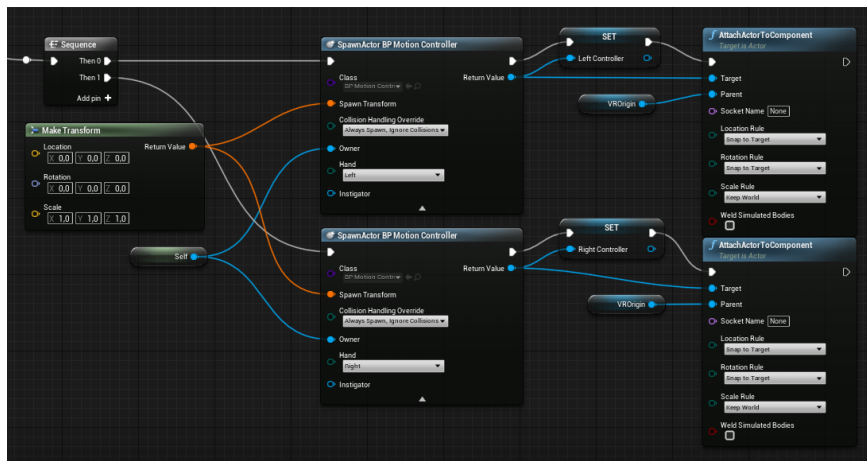
2.1.2 STVARANJE I VEZANJE KONTROLERA ZA VR UREĐAJ

Kontroleri (ruke) nalaze se kao odvojeni *actor* u Unreal Engineu kao i u drugim *game engineima* jer se kreće od pretpostavke da u projektu može biti i samo HMD uređaj što obično i smatramo VR uređajem bez svojih kontrolera. To su uređaji poput Samsung Gear VR, Google Cardboard, spomenuti PSVR itd. Ukoliko uređaj ima svoje kontrolere, a najpopularniji VR uređaji poput HTC Vive, Oculus Rift S i Quest te Valve Index imaju kontrolere, onda je te kontrolere potrebno spojiti s uređajem na način da im objekt uređaja bude vlasnik. U nastavku se može vidjeti da se nakon početka igre stvaraju dvije instance kontrolera koje odgovaraju lijevoj i desnoj ruci.

Kod oba kontrolera vlasnik je **Self** što je referenca na *blueprint* u kojem se trenutno izrađuju čvorovi. Prisjetimo se da je to *blueprint* od **MotionControllerPawna**, odnosno *blueprint* od samog VR uređaja te ovdje sa **Self** označavamo upravo njega. Postavljaju se vrijednosti unutar objekata **Left Controller** i **Right Controller** te im se pridružuje roditeljska klasa koja je zapravo sam VR uređaj. Stvaranje ruku u VR okruženju odvija se bez obzira na kolizije koje se mogu dogoditi jer uvijek želimo da se ruke stvore iz tog razloga što su ruke na neki način tipkovnica i miš VR uređaja uz njega samog.

PSVR uređaj ima samo jednu kameru koja prati intenzivne izvore svjetlosti na PSVR uređaju te po tome može znati samo trodimenzionalnu rotaciju uređaja i nema prostornu osvještenost o poziciji samog uređaja u odnosu na bilo koji drugi element u prostoru.

HMD – Head Mounted Display



Slika 3: Pripasivanje kontrolera i HMD-a objektima

2.1.3 PROCES TELEPORTACIJE

Teleportacija je najkompleksniji element ove igre pa se neće moći jednostavno prikazati i objasniti u jednom skupu čvorova. Krenut ćemo od elementa kojeg je najjednostavnije razumjeti jer sadrži funkciju za teleportaciju. Izvršavanje teleportacije događa se nakon provjere određenih sigurnosnih i praktičnih uvjeta. Važno je imati u vidu da se izvršavanje ovog dijela funkcionalnosti ne izvršava „odmah“ nakon korisnikovog zahtjeva za teleportacijom (po pritisku tipke), već je ovo normalan slijed operacija nakon uspješnih provjera. *Odmah* je napisano pod navodnicima jer se korisniku čini da je u tom trenutku, ali zapravo se cijeli niz događaja odradi prije teleportacije.

U samoj funkcionalnosti teleportacije postoje još dvije provjere:

- nalazi li se već igrač u procesu teleportacije kada traži teleportaciju
- je li mjesto na koje se želi teleportirati valjano

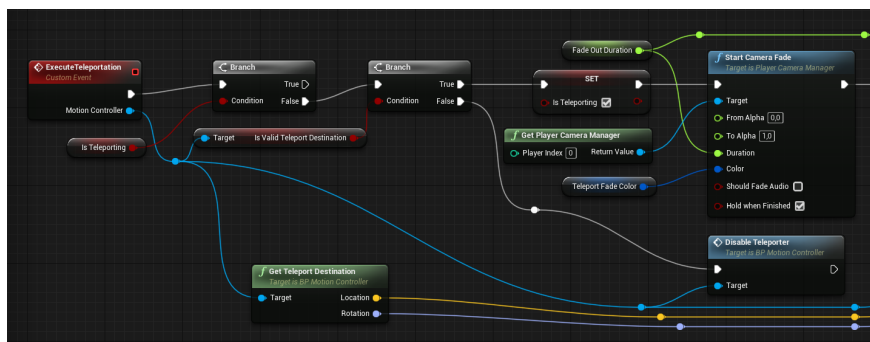
Ukoliko su zadovoljena prethodna dva uvjeta zabranjuje se prolaz novih zahtjeva za teleportaciju tako da se proces teleportacije postavlja na vrijednost **True**. Kad je prvi uvjet zadovoljen (**False** ovom slučaju) a drugi nije, briše se vizualna reprezentacija teleportacije.

Zadovoljavanje oba uvjeta pokreće funkciju koja započinje zatamnjenje (eng. *fade out*) ekrana an VR uređaju. Zatamnjenje traje isto onoliko vremena koliko traje i funkcija odgode vremena tako da se teleportacija dogodi točno nakon potpunog zatamnjenja ekrana.

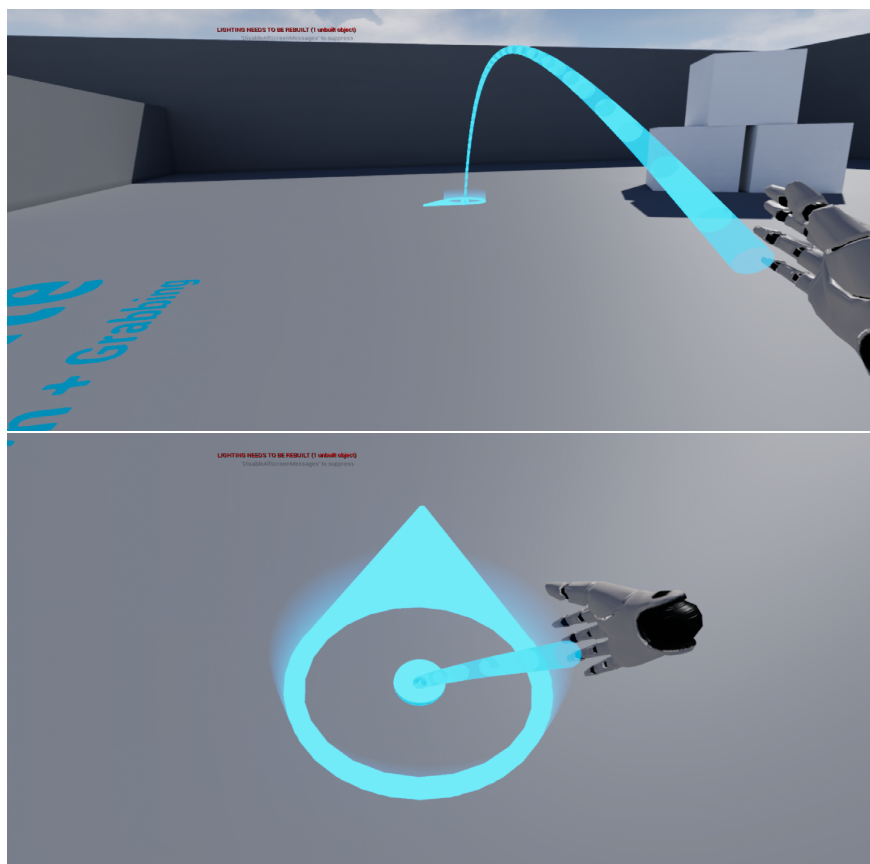
Valjano mjesto za teleportaciju ili kretanje je mjesto kojeg objekt slobodnog kretanja označava kao slobodnog za kretanje, a mjesto na kojeg se igrač teleportira prikazano je indikatorom mjesta teleportacije (v. sliku 5). Svaki *pawn* može imati svoj objekt slobodnog kretanja.

fade out – padanje mraka na oči

pawn – fizička realizacija igrača u svijetu



Slika 4: Prikaz procesa teleportacije



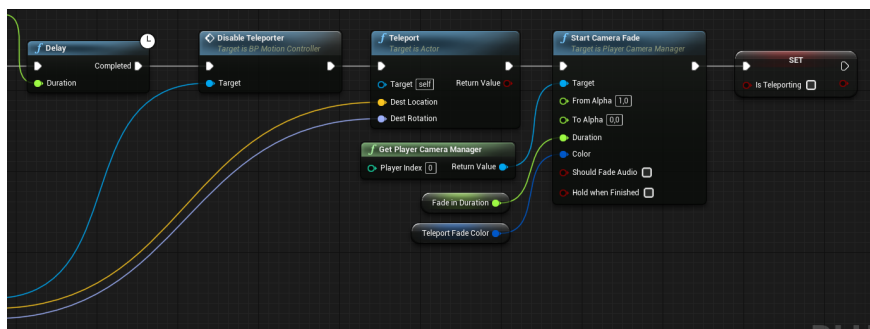
Slika 5: Indikator mjesta teleportacije

Objekt zabrane kretanja su nevidljivi zidovi u videoigrama dok *actor* može biti (vidljiv) zid kroz kojeg se ne može kretati.

Nakon teleportacije započinje proces vraćanja crnog ekrana u normalan pogled (eng. *fade in*) VR uređaja. Teleportacija može bez problema raditi i bez korištenja zatamnjenja u ove dvije spomenute funkcije i ne čini nikakvu razliku u funkcionalnosti teleportiranja, ali poboljšava korisničko iskustvo. Obična teleportacija bez korištenja zatamnjenja ekrana djeluje neprikladno, odnosno igraču se može činiti kao da dolazi do grešaka. Zatamnjenje ov-

dje teleportaciju čini prirodnijom koliko god se proces teleportiranja može nazvati prirodnim.

U međuvremenu se briše vizualna reprezentacija indikatora mjesta teleportacije i sve završava postavljanjem procesa teleportacije na **False**. Takva zaokruženost funkcionalnosti ispitivanjem istinitosti varijable zapravo čini jedan semafor koji ne dopušta novo izvršavanje funkcionalnosti dok prethodno izvršavanje nije došlo do kraja. Funkcija teleportacije lokaciju i rotaciju teleportacije dobiva od funkcije za dobivanje željenog mjesta za teleportaciju koja se može vidjeti na prethodnoj slici.



Slika 6: Prikaz završetka teleportacije

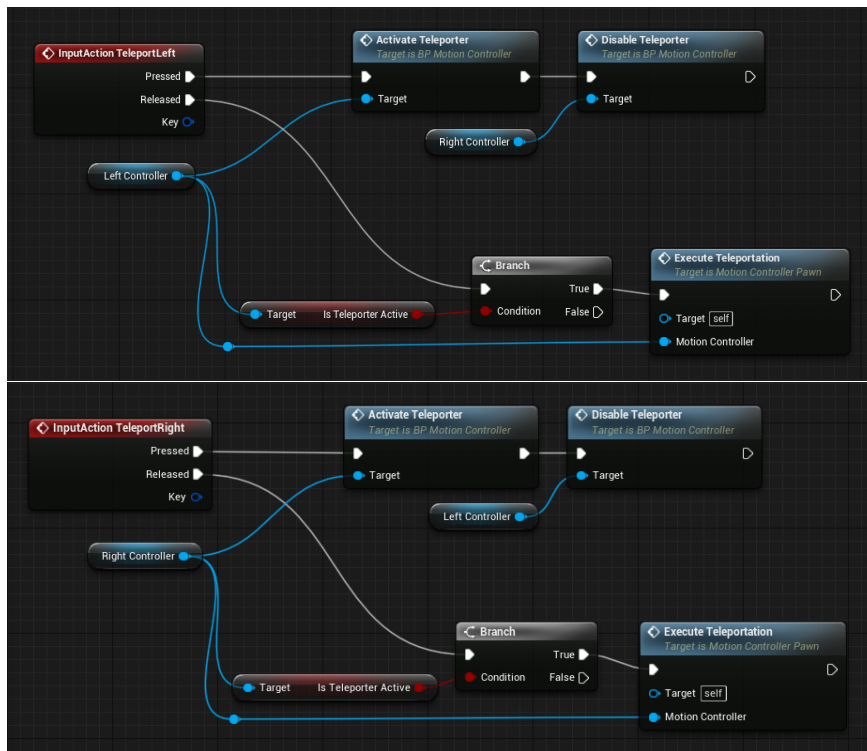
2.1.4 RUKOVANJE KONTROLERIMA

Sljedeće funkcionalnosti okidaju se na igračev unos preko kontrolera. Početni događaj koji to okida može se vidjeti prikazan crvenom bojom na slici 7. Element na kojeg utječemo je objekt kojeg smo definirali kod spajanja kontrolera s VR uređajem. Pošto je svaki kontroler odvojeni objekt moramo definirati što se događa i za jedan i za drugi kontroler pritiskom na odgovarajuću tipku. Ako i na jednom i na drugom kontroleru želimo implementirati funkcionalnost teleportacije to znači da je kod ili u ovom slučaju skup čvorova isti osim što se referenciramo na različiti kontroler. Zbog tog razloga će se u nastavku opisati slučaj za samo jedan kontroler jer je implementacija za drugi kontroler manje više ista.

Na pritisak tipke kontrolera izvršava se funkcija za aktivaciju teleportacije, a to je funkcija koja stvara vizualnu reprezentaciju za mjesto teleportacije na kontroleru na kojem je pritisnuta tipka. Nakon toga se onemogućava vizualna reprezentacija za mjesto teleportacije na drugom kontroleru. Tako je napravljeno kako igrač ne bi mogao izazvati grešku na način da pritisne tipku na jednom kontroleru i dok je drži pritisnutu, pritišće tipku i na drugom kontroleru. Na ovaj način to se izbjegne jer aktivacija na jednom kontroleru automatski znači deaktivaciju na drugom kontroleru.

Na otpuštanje pritiska tipke provjerava se je li postavljen uvjet iz funkcije aktivacije vizualne reprezentacije teleportacije. Ukoliko je, odlazi se u funk-

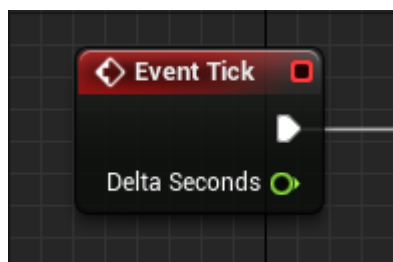
ciju teleportacije koju smo već opisali. Na slikama se usporedbom može vidjeti kako se u obje slike radi o istoj funkcionalnosti te je jedina razlika u objektu kontrolera za kojeg želimo aktivirati vizualnu reprezentaciju mjesta teleportacije i objektu kontrolera za kojeg želimo deaktivirati vizualnu reprezentaciju mjesta teleportacije.



Slika 7: Prikaz upravljanja kontrolerima, redom lijevi pa desni

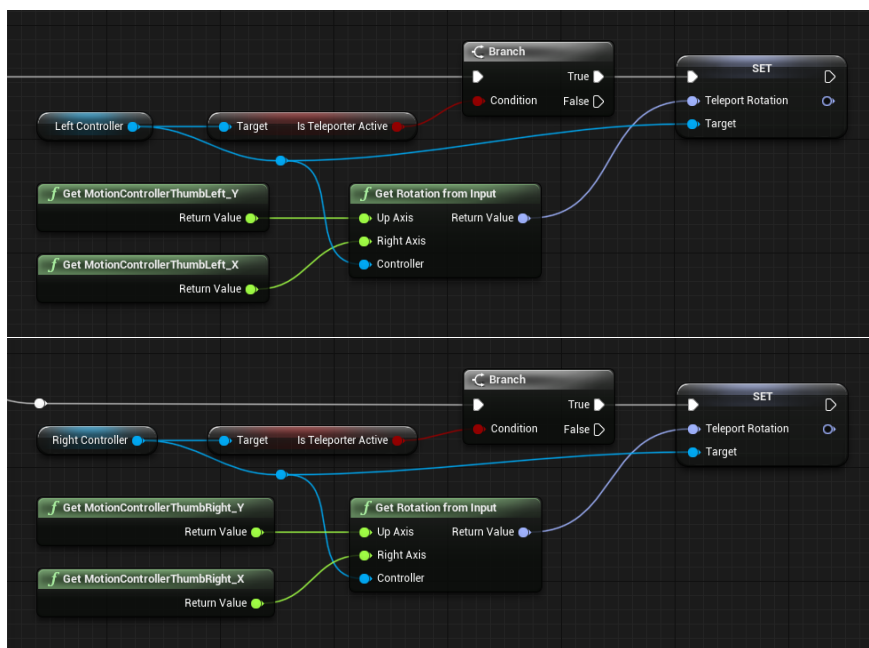
2.1.5 POSTAVLJANJE ROTACIJE MJESTA TELEPORTACIJE

Funkcionalnost koja slijedi izazvana je događajem *tick* koji se konstanto pojavljuje dok je igra pokrenuta. To je jedan od spomenutih događaja okidača koji je također označen crvenom bojom u *blueprintovima*. Iste funkcionalnosti vrijede i za lijevi i za desni kontroler tako da se analogijom mogu poopćiti.



Slika 8: Slika događaja *tick*

Na objektu odabranog kontrolera ispituje se aktiviranost vizulane reprezentacije mjesta teleportacije te ako je aktivna, postavlja se varijabla rotacije teleportacije preko x i y osi na gljivici na očitano vrijednost. Prisjetimo se da se do ovog dijela funkcionalnosti dolazi svakim *tickom* što znači da se mogućnost promjene rotacije teleportacije događa konstantno odnosno dokle god je valjan uvjet ulaska u petlju, a to je da je aktivirana vizualna reprezentacija mjesta teleportacije. Isti slučaj je i kod drugog kontrolera pa to nećemo posebno objašnjavati osim možda napomenuti očito – razlikuje se objekt nad kojim se vrši funkcionalnost.



Slika 9: Prikaz rotacije nakon teleportacije

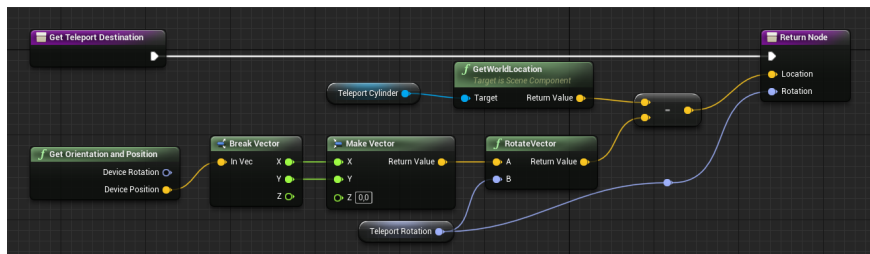
2.2 KONTROLER *blueprint*

Sada prelazimo na dio koda koje se nalazi u *blueprintu* od kontrolera. Prvo ćemo krenuti od dobivanja lokacije i rotacije za mjesto teleportacije. Zbog VR uređaja koji nemaju lokacijsku osviještenost neki su detalji malo složeniji.

2.2.1 DOBIVANJE LOKACIJE I ROTACIJE ZA MJESTO TELEPORTACIJE

Ova funkcionalnost za VR uređaje je veoma jednostavna jer se iz lokacije mjesta teleportacije dobiva vektor lokacije te se iz postavljene rotacije preko gljivice postavlja rotacija. Međutim, zbog kompatibilnosti sa uređajima poput PSVR prolazi se kroz još nekoliko koraka. Uzima se trenutna pozicija

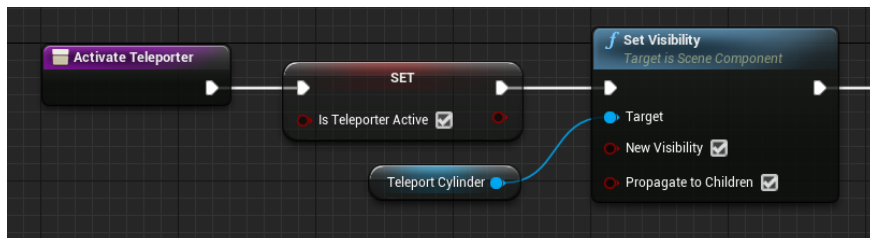
VR uređaja te se od toga uzimaju samo x i y vrijednosti. Zatim se taj vektor rotira prema rotaciji gljivice te se dobiveni vektor oduzima od vektora mjesta teleportacije. Ovo se ponovo izvršava u svrhu prilagođavanja VR uređaju koji ne poznaje visinu jer je na ovakav način visina nebitna te će se postaviti na onu prijašnje utvrđenu udaljenost od poda kao novu visinu.



Slika 10: Prikaz rotacije nakon teleportacije

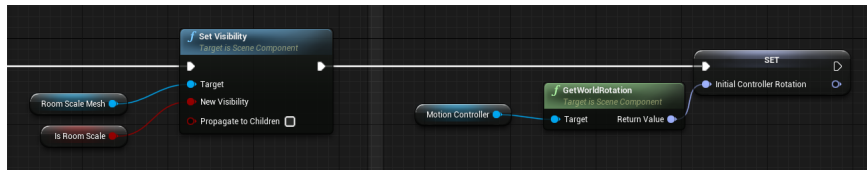
2.2.2 AKTIVIRANJE VIZUALIZACIJE MJESTA TELEPORTACIJE

Poziv ove funkcije već smo koristili kod pritiska tipke kontrolera na *blueprintu* VR uređaja. Ova funkcija postavlja istinitost varijable aktivacije vizualizacije mjesta teleportacije na **True** te postavlja vidljivost strelice koja vizualizira lokaciju i rotaciju na vidljivo. Jednostavnost i briljantnost ovog skupa čvorova može se vidjeti na slici 11.



Slika 11: Prikaz prikaza vizualizacije mjesta teleportacije

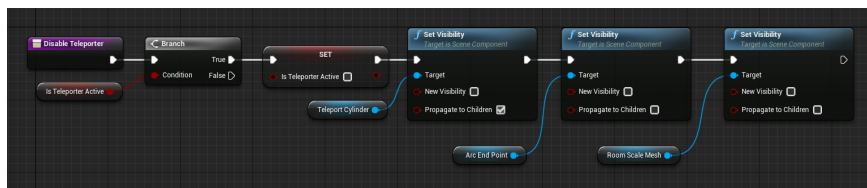
Postoje i dodatne funkcionalnosti koje nisu potrebne za normalno funkcioniranje teleportacije na Oculus Rift S uređaju, no za neke druge uređaje je korisno. Svi VR uređaji koji koriste *light-house* praćenje mogu se poslužiti i vidljivosti scenske komponente koja je prostor u kojem se igra. Ovdje to ne možemo ispitati pa nećemo detaljnije ni obrađivati. Još jedna dodatna funkcionalnost koja je praćenje rotacije kontrolera odnosno zapešća. Ova rotacija se može koristiti u igrama, ako želimo da igrač određuje rotaciju teleportacije preko rotacije kontrolera odnosno svojeg zapešća. Nekada može biti zanimljivo no više je umrajaće od rotacije preko gljivice zato što je neprirodno te ga mi nećemo koristiti. Ovdje se može vidjeti opisani dio.



Slika 12: Aktivacija indikatora mjesta teleportacije

2.2.3 DEKTIVIRANJE VIZUALIZACIJE MJESTA TELEPORTACIJE

Funcionalnosti koja uklanja vizualizaciju mjesta teleportacije je ona koja je također potrebna kako se vizualizacija ne bi stalno vidjela. Kako smo vrijednost varijable aktiviranosti vizualizacije teleportacije postavili na **True**, ovdje provjeravamo tu varijablu te ju postavljamo na **False**. Na taj način provjeravamo je li igrač ušao u proces aktivacije jer se ne bi smjelo ništa isključiti ako nije uključeno. Postavlja se vidljivost indikatora lokacije i rotacije na nevidljivo isto kao i vidljivost kraja luka za teleportaciju. Na kraju se postavlja vidljivost za *room scale* element koji ponovo nemamo kod našeg VR uređaja, no zbog kompatibilnosti ćemo ga samo spomenuti.



Slika 13: Deaktivacija vizualnog indikatora mjesta teleportacije

2.3 ZAKLJUČAK

Kroz ovu lekciju pokazali smo kako funkcioniraju osnovne funkcionalnosti kretanja u prostoru pomoću VR uređaja te kontrolera. Ove pretpostavljene funkcionalnosti napravljene su tako da većina VR uređaja radi na *plug'n'play* način te tako da se ne moraju stalno iznova programirati osnovne stvari.

Kao dopunu ove lekcije korisno bi bilo i istražiti koncepte:

- *light-house* praćenje i *inside-out* praćenje
- *room-scale*
- praćenje rotacije zapešća za smjer teleportacije umjesto gljivice – ponekad neprirodno, ali korisno za istražiti

Prostor za bilješke:

Oculus Riftovi imaju tzv. *inside-out* praćenje što znači da za praćenje fizičke pozicije kontrolera i samog VR uređaja nisu potrebni vanjski elementi tj. *light-house* na dva suprotna kraja prostora za igru. VR uređaji koji koriste *inside-out* praćenje imaju kamere na samom VR uređaju koje preko izraženih točaka u stvarnom prostoru poput ruba stola, točke nagle promjene u boji tepiha i sl. određuju vlastite točke praćenja u stvarnom svijetu te na taj način određuju fizičku poziciju VR uređaja i kontrolera što se na kraju preslikava i u virtualno okruženje.

3 DETEKCIJA POKRETA KONTROLERA

4 STVARANJE META

5 NAPADAČKE MOĆI

6 STVARANJE PRIJETNJI ZA IGRAČA I PRIKAZ ZDRAVLJA (HP)

7 OBRAMBENI MEHANIZMI ZA IGRAČA

8 BODOVANJE I PRIKAZ BODOVA

9 IZRADA KRAJOLIKA NIVOVA

10 IZBORNICI

11 "UMJETNA INTELIGENCIJA" – PROTIVNIK SE KREĆE

12 VIZUALNI EFEKTI

13 ZVUK

KAZALO

actor, 12	Uvod, 5
Controller, 10	Valve Index, 10
dogadjaj, 9	visina, 9
event, 9	VR, 7
fade in, 12	
fade out, 11	
gljivice, 14, 15	
Google Cardboard, 10	
Head Mounted Display, 10	
HMD, 10	
HTC Vive, 10	
indikator mjesta teleportacije, 13, 17	
inside-out praćenje, 17	
kontroler, 13	
kretanje, 9	
light-house praćenje, 16, 17	
lokacijska osviještenost kontrolera, 15	
mehanike, 6	
mrak na oči, 11	
nevidljiv zid, 12	
Oculus Rift, 10	
pawn, 11	
postavke, 9	
PSVR, 9, 10, 15	
room-scale, 17	
rotacija mjesta teleportacije, 14	
rotacija zapešća, 17	
rukovanje kontrolerom, 13	
Samsung Gear VR, 10	
teleportacija, 9, 11, 13	