Upute za razvoj jednostavne 3D videoigre u Unreal Engine sustavu

2022./2023. god

Unreal Engine 5.1.

Koraci razvoja:

1. Odabir okruženja

2. Ubacivanje asseta i izrada mape

3. Stvaranje više levela

4. Programiranje blueprintima

5. Osvjetljenje

6. Interakcija s okolinom

7. Implementacija fizike

8. Kolizija objekata

9. Jednostavnije animacije

10. Zvukovi

11. Korisničko sučelje

Pojmovi koji će ponekada biti korišteni u uputama:

**Quixel Bridge** – besplatan dodatak razvojnom okruženju Unreal Engine iz kojega se mogu nabaviti različiti objekti i teksture za izgradnju mape naše video igre. Ako smo prijavljeni u Epic sustav preuzimanje je jednostavno. Ili pritisnemo na gumb za preuzimanje te odaberemo lokaciju spremanja ili još jednostavnije, samo prevučemo mišem u našu igru.

Teksture se razlikuju od objekata po tome što su to samo materijali koji se moraju staviti na već postojeći objekt.

**Asset** – engleski naziv za objekt korišten u igri

**Level** – razina u videoigri

Prečaci na tipkovnici / keyboard shortcuts

**Space** – mijenjanje stanja koja će akcija biti provedena nad odabranim objektom.

Moguće opcije su:

1. Pomicanje objekta,

2. Rotacija objekta i

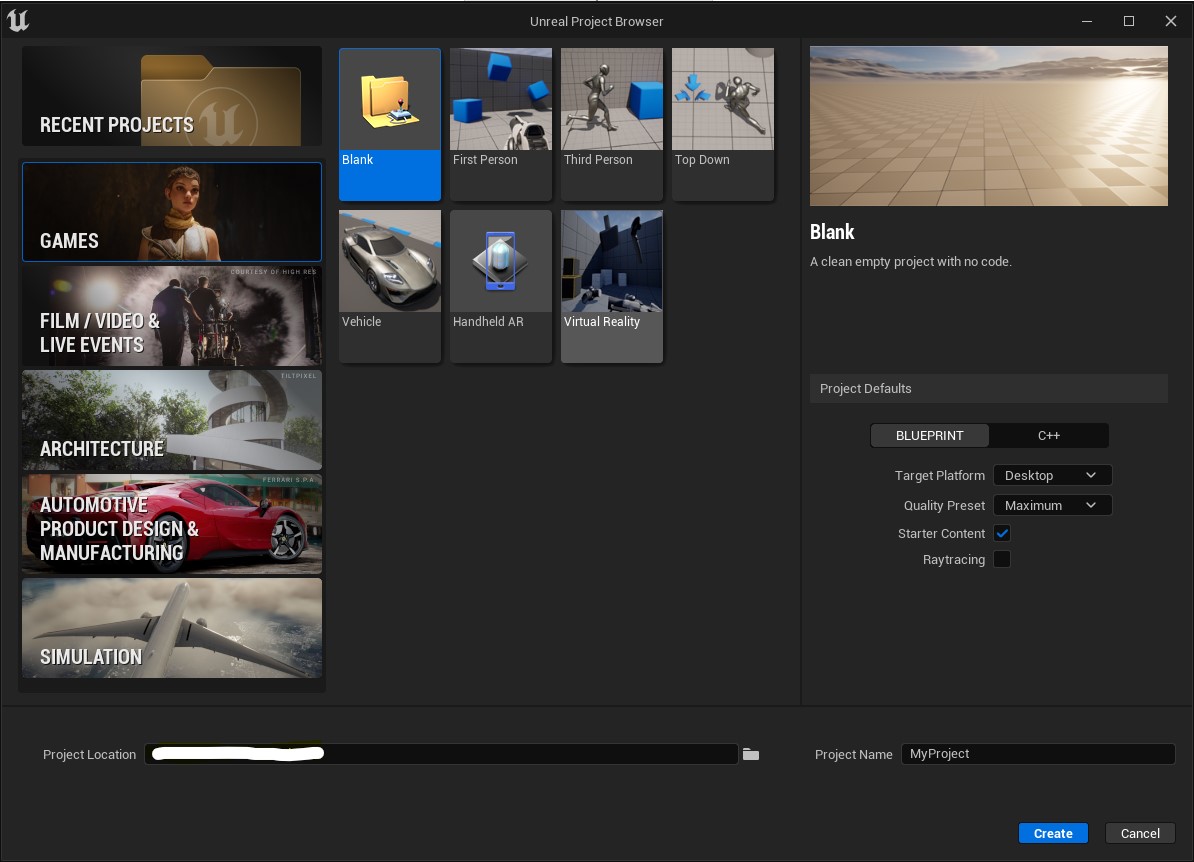
3. Promjena veličine objekta

**Ctrl + d** – dupliciranje odabranog objekta

**Držanje klika miša** – držanje desnog klika miša omogućava nam navigiranje u prostoru. Okretanjem miša se rotiramo dok se tipkama WASD krećemo po prostoru.

1. ODABIR OKRUŽENJA

Nakon prvog pokretanja Unreal Engine studija odabiremo opciju Games te u sklopu ovog rada opciju Third Person.



2. Ubacivanje asseta i izrada mape

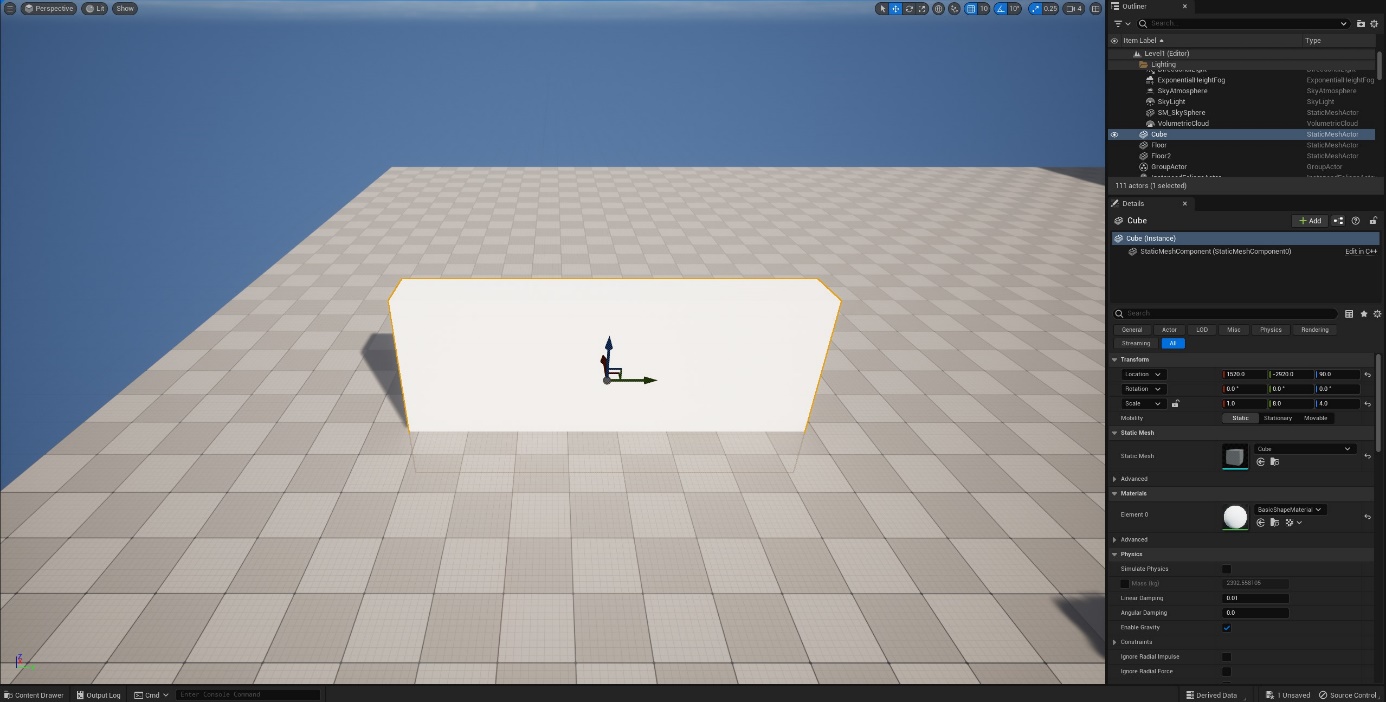
Projekt se učitava i počinjemo s radom.

Za stvaranje objekata koji će se koristiti u video igri postoje dva načina stvaranja:

**a) Stvaranje basic objekta i dodavanje teksure**

Na ovaj način izbjegavamo neke probleme koji bi se mogli pojaviti korištenjem vanjskih asseta.





Sada pomoću opcije Quixel Bridge tražimo željenu teksturu za u ovom slučaju zid.

[*Slika zida sa strukturom]*

**b) Uvoz objekata direktno sa sustava Quixel Bridge**

U sklopu ovog rada za objekte će se većinom koristiti već gotovi uvezeni objekti zbog njihove realističnije strukture.

Krećući sa zidom, uvozimo objekt te vidimo da nije obostran. Objekt kopiramo i rotiramo te namještamo sa prethodno stvorenim. Tada pomoću opcije (naći kak se zove) dva elementa spajaju se u jedan za daljnje korištenje.

*[Slike]*

Sada kada imamo gotov objekt idemo složiti glavnu strukturu mape naše igre – labirint. Dizajn osmislimo sami ili pronađemo na internetu i krećemo s radom. Objekt zida rotiramo te mijenjamo širinu po potrebi.

Kako se sama struktura proširuje širenjem zida, nekada je bolja opcija postavljanje zidova tako da prolaze jedan kroz drugi.

*[slika]*

Isto ponavljamo i za tlo, možemo teksturu dodati na tlo koje je stvoreno u sklopu projekta.

**c) Uvoz objekata iz sustava Blender**

Blender je popularan besplatan alat za modeliranje 3D objekata. Nekolicina objekata u ovom projektu modelirana je u Blenderu (npr. lampe u labirintu). Pri završetku modeliranja određenog objekta isti izvozimo (engl. export) u FBX format. Takav objekt onda uvozimo (engl. Import) u Unreal Engine.

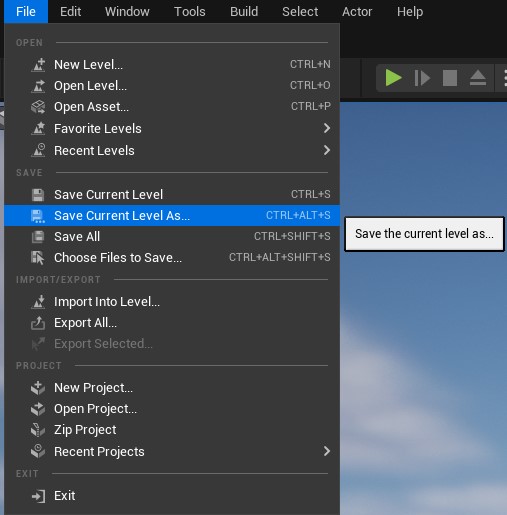


Primjer modela lampe uvezenog u Unreal Engine

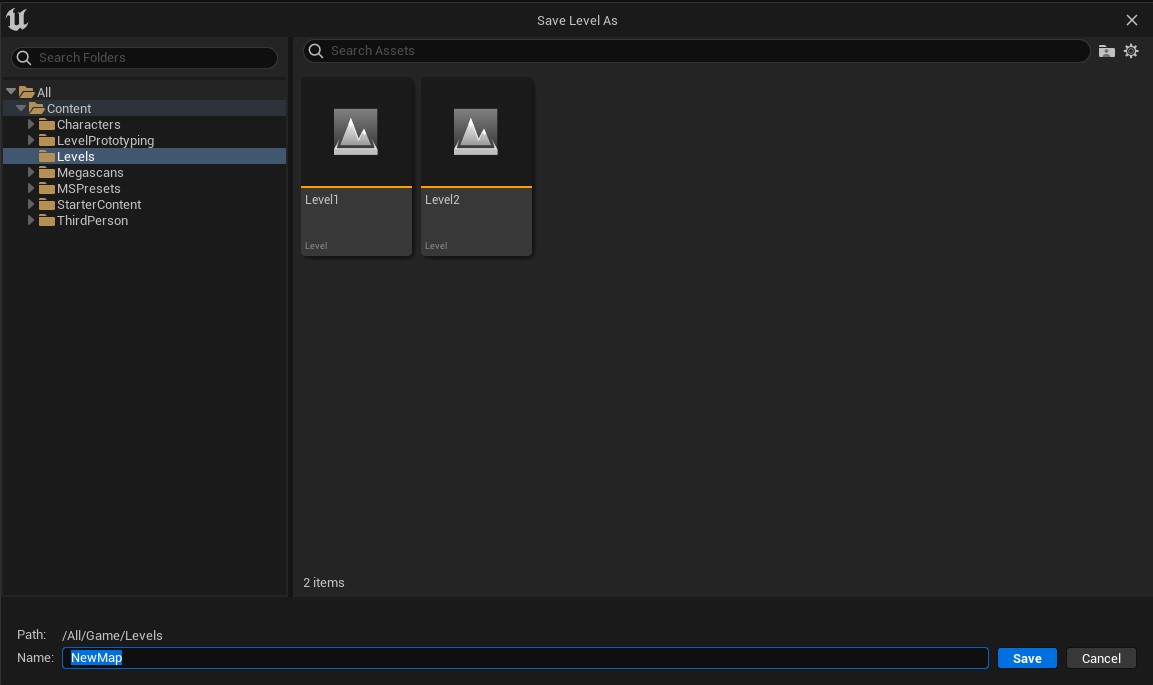
3. Stvaranje više levela

Kako igra ne bi bila prekratka te bi imala neku strukturu želimo stvoriti više razina/levela.

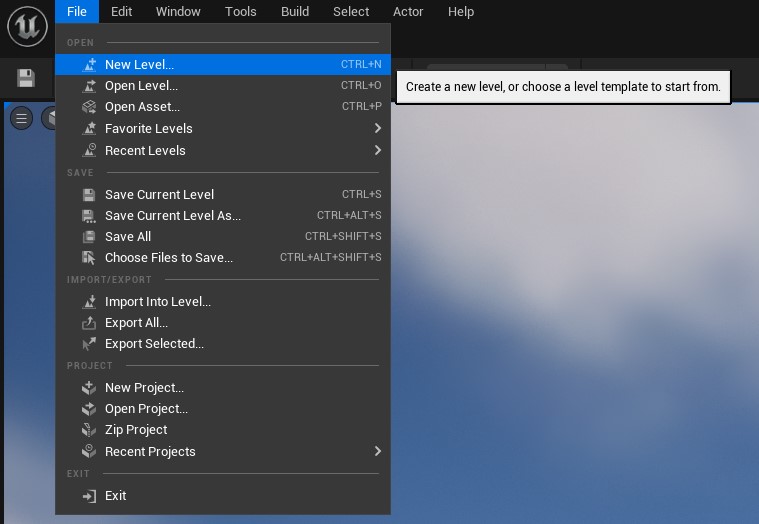
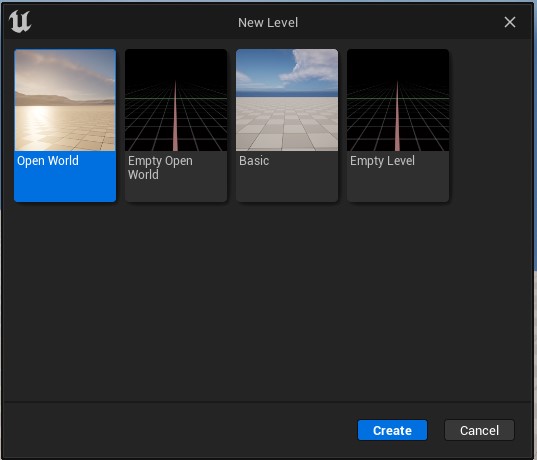
To radimo na način da prvotno spremimo prethodno stvoreni level.



Za bolju organizaciju preporučljivo je stvoriti dodatni folder namijenjen za tu svrhu.



Tada odabiremo opciju New Level iz izbornika i odabiremo željeni tip mape, u ovom slučaju *Basic*.

4. Programiranje blueprintima

Kako bismo dodali funkcionalnosti u igricu poput animacija, fizike, radnji i kretanja Unreal Engine 5.1 nudi programiranje tzv. blueprint sistemom koji se zasniva na povezivanju widgeta čvorovima i posljedičnim stvaranjem vizualne skripte umjesto pisanja tradicionalnog koda.

Za kreiranje tzv. Blueprint Classa najbolje je u folderu Content kreirati novi folder naziva Blueprints u koji ćemo ih smjestiti i desnim klikom na prazan prostor Content Drawera u folderu otvoriti izbornik u kojem ćemo izabrati Blueprint Class.

Graphical user interface, application

Description automatically generated

Dodavanje Blueprint Classa

**Blueprint Class** predstavlja složeno okruženje sastavljeno od Viewporta, Contruction scripta i Event Grapha. Unutar Viewporta stvaramo dodajemo komponente (npr. objekti, osvjetljenje, itd.), a unutar Event Grapha realiziramo logiku pomoću vizualnog skriptiranja blueprintima. Construction Script služi za incijalizacijske akcije prilikom stvaranja instanci Blueprinta.

Third person template mapa nudi ključnu blueprint klasu (BP\_ThirdPersonCharacter) u kojoj se nalazi kod unutar grafa događaja (engl. Event Graph), a definira kretanje dobivenog lika.

Ideja u video igri je da igrač može sakupljati kamenje, tako da će dodavanje nove varijable u blueprint (klikom na Add u sekciji My Blueprint) čiji je tip cijeli broj (engl. Integer) signalizirati igraču koliko trenutno ima kamenja i na temelju toga dozvoliti ili zabraniti igraču da baci kamen, na primjer naziva Rock Amount.

Graphical user interface, text, application

Description automatically generated

Stvorena varijabla Rock Amount

U Event Graphu vidljivo je par triggera koji se pokreću kada se određeni pritisak gumba na tipkovnici registrira. Predefinirane tzv. Input Axise može se izmijeniti ili odlaskom na postavke projekta (engl. Project Settings) i podešavanjem vrijednosti u kategoriji Bindings sekcije Input.

Graphical user interface

Description automatically generated

Primjer blueprinta za dodavanje kretnje liku pri pomicanju u svakom smjeru

5. Osvjetljenje

Osvjetljenje je važan faktor u izradi video igre. Srećom, Unreal Engine 5 nudi puno opcija za modificiranje emitiranog svjetla i stvaranja novih izvora svjetlosti.

Najbolji i najbrži način za osvijetliti prostor proizvoljnim izvorom svjetlosti je kreiranjem Blueprint Classa u kojeg uvozimo objekt iz kojeg želimo emitirati svjetlost. Zatim u Components tabu klikom na Add dodajemo novu komponentu u blueprint. U kategoriji Lights izabiremo željenu vrstu izvora svjetlosti.

Text

Description automatically generated

Ponuđene opcije izvora svjetlosti

U pokaznu video igru dodali smo Point Light kako bismo rekreirali osvjetljenje koje dolazi iz lampe. Pomicanjem objekta u područje iz kojeg bi se trebala emitirati svjetlost i podešavanjem raznih opcija poput intenziteta, boje svjetlosti, radijusa emitiranja, itd.

A screenshot of a video game

Description automatically generated

Prikaz objekta i opcija za podešavanje efekata izvora svjetlosti

Ovime je izrada Blueprint Classa završena i isti se može pozicionirati unutar svijeta. Pozicioniranjem lampi kroz cijelo igrivo područje dobivamo potpuno osvijetljeni prostor.

Ako se video igra odvija u vanjskom prostoru potreban je potrebno je u svijet uvesti usmjereno svjetlo (engl. Directional Light). Ono će osvijetliti cijeli prostor na način na koji sunce osvjetljava zemlju. I za usmjereno svjetlo postoje brojne opcije koje se mogu podesiti, iako su prvotne dovoljno dobre. Opcije uključuju temperaturu, boju svjetlosti, određivanja hoće li svjetlost stvarati sjene, intenzitet, itd.

Unrealove template mape za brži početak realizacije projekta pružaju skup osvjetljenja u mapi Lighting koje su podešene na optimalan način. Osim usmjerenog svjetla, postoji i exponencijalna visina magle (engl. Exponential Height Fog), nebeska atmosfera (engl. Sky Atmosphere), nebesko svjetlo (eng. Sky Light) te volumetrični oblaci (engl. Volumetric Cloud) i nebeska sfera (engl. Sky Sphere) koji dodaju realističniji izgled igre dodavanjem oblaka, prividnog osjećaja sfere, horizonta, itd.

A picture containing sky, outdoor

Description automatically generated

Prikaz vanjskog dijela mape i navedenog osvjetljenja

6. Interakcija s okolinom

U mapi se nalazi puno kamenja koje se igrač može sakupiti. Kreiranjem novog Blueprint Classa zvanog Rock možemo dodati komponentu Static Mesh naziva Rock\_SM, a za sam mesh izabrati kamen dobiven u Starter Content mapi (SM\_rock). Također, moramo dodati sfernu koliziju (engl. Sphere Collision). Sferna kolizija, nazovimo je InteractionCollision, bit će namijenjena za detekciju igrača u blizini kamena i imati veliki domet jer je ideja da se omogućuje sakupljanje kamena kada se igračev mesh preklopi (engl. overlap) s kolizijom, odnosno kada je u blizini kamena. Međutim, da bismo postigli efekt prolaska kroz koliziju, ali istovremenog generiranja događaja preklapanja potrebnog za vizualno skriptiranje moramo podesiti postavke kolizije u sekciji kolizija (engl. collision) u detaljima. Potrebno nam je kreirati posebnu vrstu kolizije koja se preklapa samo s igračem.

Graphical user interface

Description automatically generated0

Konačne postavke kolizije za InteractionSphere

Dobiveni blueprint lika već sadrži koliziju čije su postavke namještene tako da se ponaša kao igrač, što uključuje opciju generiranje događaja preklapanja (engl. Generate Overlap Events) koja je omogućena. Ista opcija mora biti uključena u našoj koliziji kamena kako bi se mogli davati signali blueprintima.

U svrhu radnje prikupljanja kamena potrebno je u već spomenute postavke projekta u sekciji input dodati novu mapiranje za akciju (engl. Action Mapping) koje će služiti za sakupljanje kamena. Nazvat ćemo ga Interact i dodijeliti mu proizvoljnu tipku, na primjer F.

Prije same detekcije interakcije vrlo je važno omogućiti kamenu da zaprimi pritiske tipki kontrolera igrača (engl. Player Controller). Indeks igrača je prema zadanim postavkama postavljen na 0.

A picture containing text, indoor

Description automatically generated

Važan preduvjet za omogućavanje interakcije između korisnika i kamena

Sada možemo generirati događaje specifične za InteractionCollision desnim klikom na istu i iz Add Event menija izabrati događaje OnComponentBeginOverlap (koji se pokreće kada se igrač preklapa sa sfernom kolizijom) i OnComponentEndOverlap (koji se pokreće kada igrač napusti koliziju).

Ostale metode možemo dodavati desnim klikom bilo gdje u Event Graph i pretraživanjem i izabiranjem iz ponuđenog menija. Tim postupkom dodajemo Cast To BP\_ThirdPersonCharacter metodu koja će, ako se točno igrač detektira u ulazu Object nastaviti izvršavanje. Izlaz događaja za početak preklapanja Other Actor (referenca na objekt koji se preklapa) spajamo u Object ulaz Cast To metode. Zatim, dodajemo događaj za pritisak novostvorenog mapiranja F (za interakciju) i spajamo ga u Gate koji dopušta prolaz dalje samo ako je otvoren, a podešavamo da ga početak interakcije otvara, a završetak zatvara.

Još je samo preostalo dodati jedan kamen igraču uvećavanjem varijable RockAmount za jedan ako se uspješno prođe Gate i pomoću metode Destroy Actor uništiti kamen (opcija Target ostavlja se na self).

A screenshot of a computer

Description automatically generated with medium confidence

Prikaz dovršene logike za sakupljanje kamena

7. Implementacija fizike

Želimo da igrač ima mogućnost bacanja kamena pritiskom na tipku F ako u posjedu ima jedan ili više kamena. U tu svrhu dodajemo novo mapiranje akcije Throw Object kojem pridjeljujemo neku tipku, na primjer E.

Ono što nam je potrebno za bacanje kamena je:

* stvaranje kamena ispred igrača,
* oduzimanje jednog kamena iz posjeda igrača i
* bacanje kamena.

Da bi na kamen utjecala fizika, potrebno je omogućiti opciju Simulate Physics. Međutim, važno je razumjeti da iako imamo mesh kamena, on nema granice kolizije i potrebno ih je uvesti kako bismo mogli omogućiti simuliranje fizike.

U Rock Blueprint Class dodat ćemo novu sfernu koliziju (engl. Sphere Collision) zvanu RockCollision koju ćemo postaviti kao roditelja (engl. parent) svim ostalim komponentama i proizvoljno odrediti vrijednosti za Linear damping (sila trenja dodana da smanji linearno kretanje) i Angular damping (sila trenja dodana da smanji rotaciju) i masu.

Graphical user interface

Description automatically generated

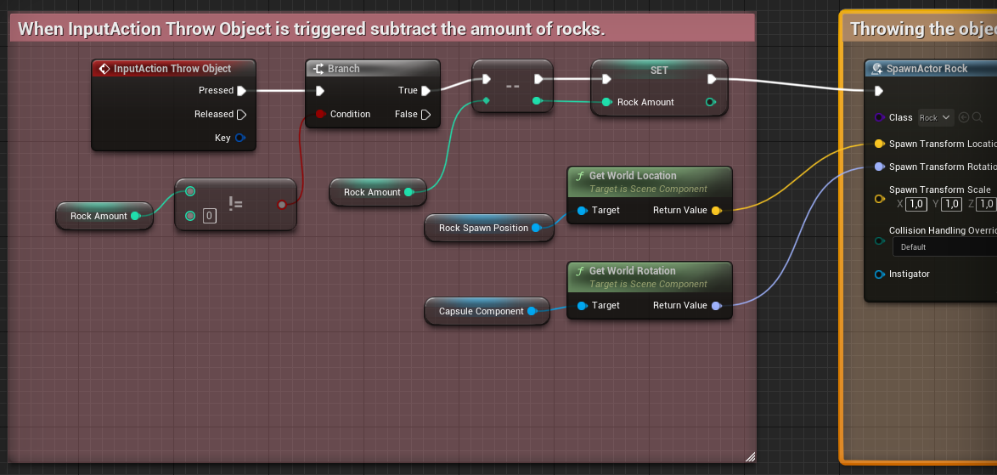
Postavljene Physics vrijednosti za RockCollision

**a) Stvaranje kamena**

Kako bismo dobili referencu na položaj malo ispred igrača moramo stvoriti neki objekt koji čija će nam lokacija služiti kao referenca za stvaranje kamena. Dovoljno je u blueprint igrača dodati sferu (engl. Sphere), pozicionirati je proizvoljno negdje ispred igrača i postaviti ga u detaljima da je skriven u igri (sekcija Rendering -> Hidden in Game). Sferu ćemo nazvati, na primjer, RockSpawnPosition.

**b) oduzimanje jednog kamena iz posjeda igrača**

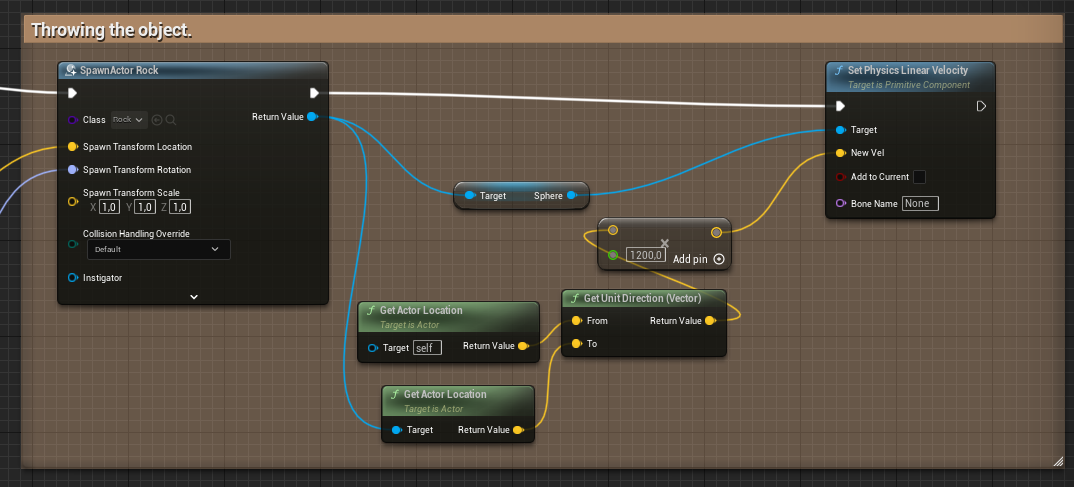
Zatim stvaramo događaj za tu akciju u Event Graphu blueprinta igrača. Prvo provjeravamo ima li igrač jedan ili više kamena u svom posjedu (provjeravamo stanje varijable RockAmount) i ako ima dozvoljavamo prolaz, umanjujemo varijablu za 1 i stvaramo blueprint kamena pomoću metode SpawnActor. Za lokaciju uzimamo referencu na World Location sfere koja ispred lika (dovlačenjem njene reference iz sekcije Components iz metodom get, a za rotaciju referencu na World Rotation sferne kapsule lika.



Oduzimanje jednog kamena pritiskom na tipku E

**c) bacanje kamena**

Izlazni čvor iz SpawnActor metode vraća nam stvoreni objekt kojemu moramo povećati linearnu brzinu (engl. Linear Velocity) tako da stvorimo metodu SetPhysicsLinearVelocity i u nju spojimo stvoreni objekt, a za lokaciju uzimamo jedinični smjer (GetUnitDirection) u čiji From čvor spajamo Actor Location trenutnog blueprinta lika, a u To čvor Actor Locationa stvorenog objekta. Rezultantni vektor množimo s nekom visokom vrijednošću (npr. 1200) kako bismo dodali dovoljno sile za simuliranje bacanja predmeta te spajamo kao ulazni vektor u metodu SetPhysicsLinearVelocity.

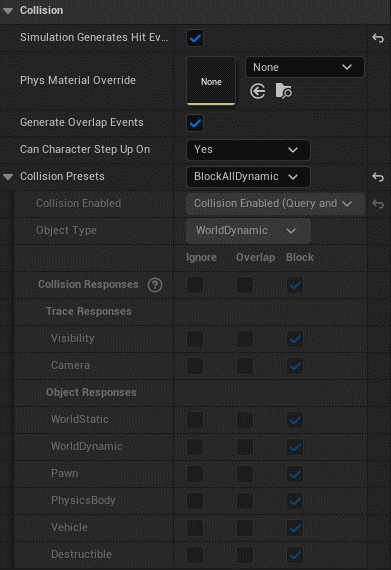


Stvaranje kamena ispred igrača i dodavanje jedinične brzine kamenu

8. Kolizija objekata

Pri bacanju kamena u igri želimo da se on odbija od solidnih objekata poput zidova i poda, što znači da moramo namjestiti koliziju samog objekta kamena unutar Blueprint Classa Rock.

Za sfernu koliziju postavit ćemo postavke kao što su prikazane na slici kako bismo dobili koliziju koja blokira sve, odnosno odbija se od svega, ali i generira događaje udarca (što je bitno ako želimo da njegov udarac u neki drugi objekt omogući komuniciranje s blueprintom drugog objekta javljajući mu detekciju udarca).



Postavke kolizije za Rock\_SM

Bitno je da kao Collision Presets postavimo BlockAllDynamic kako bi dobili efekt odbijanja od svih solidnih površina, ali u isto vrijeme mogli definirali objekt kao dinamičan, odnosno pokretan.

U pokaznoj video igri igrač mora baciti kamen tako da pogodi polugu koja će mu omogućiti daljnji prolazak. Ovo je moguće realizirati na sličan način kao što smo napravili sakupljanje kamena (promatrajući događaje preklapanja sfera), ali to bih značilo da kamenu moramo kreirati još jednu sfernu koliziju. Bolja i preciznija varijanta je da pomoću već spomenutih generiranja događaja udarca (koje smo uključili za RockCollision) pomoću tzv. Hit Eventa komuniciramo s Blueprint Classom poluge.

Kreiramo Blueprint Class poluge naziva Switch u koju ubacujemo Static Mesh poluge. Želimo da udarac kamena u bilo koju geometriju poluge može generirati unutar blueprinta događaj Event Hit. Međutim, kao što je već spomenuto, mesh sam po sebi nije dovoljan jer nema granice kolizije, tako da dodajemo novu koliziju tipa box (engl. Box Collision) naziva LeverCollision, postavljamo je kao roditelja svih komponenti i dajemo joj iste postavke kolizije kao i za RockCollision.

A picture containing graphical user interface

Description automatically generated

Prikaz hijerarhije Blueprint Classa Switch i Viewporta

Ovime smo osigurali da se udarcem ovih dvaju objekata događaj Event Hit izvrši na blueprintu poluge.

9. Jednostavnije animacije

Unreal Engine 5.1 nudi alat Level Sequence koji pruža vrlo dobar način kreiranja jednostavnijih animacija objekata unutar levela.

Kreiranjem novog Level Sequencea naziva WallMovement možemo simulirati pomicanje zidova kada se poluga pogodi kamenom. U Sequencer se može dodati bilo koji objekt u levelu i simulirati njegovo kretanje. Ako dodamo jedan zid u Sequencer i izaberemo da pratimo njegov Transform, možemo pomoću keyframeova kroz vrijeme napraviti animaciju kretanja zida.

Graphical user interface

Description automatically generated

Prikaz rada sa Sequencerom

Važno je napomenuti da Level Sequencer može komunicirati samo sa Level Blueprintom jer označavaju funkcionalnosti vezane za određeni level, tako Sequencer ne može biti referenciran unutar blueprint klasa. Sada se javlja pitanje - kako izvršiti komunikaciju između Level Blueprinta i Level Sequencera?

**Level Blueprint** predstavlja blueprint klasu vezanu uz određeni level, a može mu se pristupiti klikom na ikonu vizualnog skriptiranja u gornjoj alatnoj traci.

Graphical user interface, text, application

Description automatically generated

Lokacija Level Blueprinta

Kako bismo uspješno izvršili komunikaciju, prije je potrebno postaviti Game Mode. U panelu World Settings možemo definirati GameMode Override i tu ćemo izabrati već definirani BP\_ThirdPersonGameMode. Tu su definirane zadane vrijednosti za objekte poput igrača, korisničko sučelje, kontrolora igrača, itd.

A picture containing text, road, monitor, screen

Description automatically generated

Prikaz odabranog Game Mode Overridea

Otvaranjem navedenog Game Modea možemo dodati tzv. Event Dispatcher u njega.

**Event Dispatcher** služi kao odašiljač događaja preko kojeg Blueprint Class može poslati poruku do Level blueprinta. Kreirat ćemo novi Event Dispatcher naziva OnLeverTriggered koji će se pozvati kada pogodimo polugu kamenom.

Uz preduvjet uspješno postavljene kolizije (opisane u pretodnom poglavlju), kod za komunikaciju s Level Blueprintom preko Game Modea izgleda kao na sljedećoj slici:

A screenshot of a computer

Description automatically generated with medium confidence

Pozivanje eventa OnLevelTriggered kada se za polugu aktivira Event Hit

Sada je potrebno unutar Level Blueprinta pretplatiti se na događaj OnLeverTriggered tako da događaj Event BeginPlay spojimo na Cast To BP\_ThirdPersonGameMode te se iz njega pretplatimo na OnLeverTriggered kojega delegiramo na tzv. Custom Event koji ćemo nazvati Move Walls. Zatim kreiramo Level Sequence Player u kojem izaberemo napravljeni WallMovement Level Sequencer i reproduciramo ga. Još je preostalo dodati odgodu u duljini jednakoj duljini animacije nakon čega možemo uništiti zid kako se ne animacija ne bi opet reproducirala. Referencu na zid možemo dobiti preko Level Sequencera i ukopčati je u Target metode Destroy Actor.

A screenshot of a computer

Description automatically generated with medium confidence

Postupak kojim se Level Blueprint preko Game Modea pretplaćuje na događaje

10. Zvuk

Unreal Engine 5.1 nudi uvoz i reproduciranje .wav audio datoteka. Audio zapisi mogu se dodati kao komponenta bilo kojem Blueprint Classu, uz brojne opcije za domet i glasnoću zvuka.

Ako želimo da se pri udarcu poluge osim animacije zida reproducira i audio zapis (na primjer nekakav zvuk klika), jednostavno dodamo Audio komponentu u Switch Blueprint Class i omogućimo opciju Override Attenuation kako bismo dobili pristup dodatnim opcijama i manualno mogli namjestiti domet i način gubljenja zvuka kroz daljinu.

Graphical user interface

Description automatically generated

Postavke zvuka

Malim preinakama događaja Event Hit unutar blueprinta Switcha možemo postići da se prvo reproducira zvuk, a zatim nakon kratke odgode reproducira i animacija.

A screenshot of a video game

Description automatically generated

Preinaka Event Hit događaja uključivanjem zvuka u događaj

Zvuk micanja zida možemo još lakše implementirati tako da se vratimo u kreirani Level Sequencer WallMovement i klikom na gumb Track dodamo novi Audio Track i izaberemo željeni zvuk pomicanja zida. Zatim pomoću opcije Attach možemo izabrati objekt u Level Sequenceru za koji želimo prikačiti (engl. attach) željeni zvuk.

Rectangle

Description automatically generated

Prikaz dodanog Audio Tracka prikačenog za mesh zida