LAB University of Applied Sciences

Pelifysiikat Unreal Enginessä

LAB-ammattikorkeakoulu
Tekniikan ammattikorkeakoulututkinto
2022
Roni Niittumäki

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Roni Niittumäki	Opinnäytetyö, AMK	2022
	Sivumäärä	
	31	

Työn nimi

Pelifysiikat Unreal Enginessä

Tutkinto ja koulutusala

Insinööri (AMK), Tieto- ja viestintätekniikka

Toimeksiantajaorganisaatio (jos opinnäytetyöllä on toimeksiantaja)

Pixtell Oy

Tiivistelmä

Opinnäytetyön tavoitteena on tutkia Unreal Engine pelimoottorin pelifysiikka ominaisuuksia ja todeta optimaalisin tapa käyttää niitä opinnäytetyön toimeksiantajan kontekstissa eli avoimen maailman videopelissä, jossa pelifysiikkoja käytetään taistelumekaniikoissa. Opinnäytetyöhön kehitettiin taistelumekaniikat videopeliin, jossa käytetään Unreal Enginen tarjoamia pelifysiikkatyökaluja simuloimaan pelifysiikoita pelihahmoissa tiettyjen törmäystapahtumien tapahtuessa.

Työ on jaettuna teoriaosaan, jossa käsitellään pelimoottoria, pelifysiikoita ja videopelien yleisiä taistelumekaniikkoja. Työn käytännön osassa rakennettiin pelin taistelumekaniikat kuin myös pelifysiikat, jotka aktivoituvat pelin taisteluosioiden yhteydessä.

Työn tekemiseen käytetään Unreal Engine pelimoottoria ja sen tarjoamia editoreita, kuten fysiikkaeditori ja visuaalinen, C++ sukuinen ohjelmointikieli Blueprint.

Asiasanat

Unreal Engine, pelimoottori, pelifysiikat

Abstract

Author(s)	Type of Publication	Published
Roni Niittumäki	Thesis, UAS	2022
	Number of Pages	
	31	

Title of Publication

Game physics in Unreal Engine

Degree, Field of Study

Engineer (UAS), Information and communication technology

Organisation of the client (if the thesis work is commissioned by another party)

Pixtell Oy.

Abstract

The goal of this thesis is to investigate game physics features of Unreal Engine game engine and find the most optimal solution to use them in the context of thesis's client, i.e., an open world video game where game physics are used in combat mechanics. For the thesis, combat mechanics were developed for a video game, where game physics tools provided by Unreal Engine are used to simulate game physics in game characters when certain collision events occur.

The thesis is divided into a theory part, which deals with the game engine, game physics and general combat mechanics of video games. In the practical part of the thesis, the combat mechanics of the game were developed, as well as the game physics, which are activated during the combat sections of the game.

The work is done using the Unreal Engine game engine and the editors it provides, such as the physics editor and the visual, C++ related programming language

Keywords

Unreal Engine, game engine, game physics

Sisällys

Sanasto

Blender: 3D-mallinnusohjelma

Blueprint: Unreal Enginen visuaalinen ohjelmointikieli

Blueprint komponentti: Objektityyppi mikä voidaan liittää muihin komponentteihin aliobjektiksi

Boolean: Muuttuja mikä voi olla tosi tai epätosi

Epic Games: Yhdysvaltalainen pelitalo

Event Tick: Tapahtuma, joka laukaistaan jokainen kuvasekunti

Float: Desimaalilukumuuttuja

Hitbox: Törmäyslaatikko mikä hoitaa vastustajan vahingoittamisen tai vahingon ottamisen

Mixamo: Adoben palvelu ilmaisille animaatioille

Muuttuja: Ominaisuus, joka sisältää arvon tai viittauksen objektiin

PhysX: Nvidian kehittämä fysiikkamoottori

Ragdoll: Fysiikka simulaation tila, jossa hahmo on velttona

Root motion: Juuriluulla ohjattu animaatio

Sidescroller: Sivustapäin kuvattu videopeli

Solmu: Blueprint visuaalisessa ohjelmoimisessa käytettävä tapahtuma tai toimintokutsu

Törmäysvasteet: Kun kaksi peliobjektia on vuorovaikutuksessa törmätessään

Unreal Engine : Epic Gamesin julkistama pelimoottori.

Zbrush: 3D-mallinnusohjelma

1 Johdanto

Videopelifysiikoita voidaan käyttää monilla eri tavoin videopeleissä. Tässä opinnäytetyössä käsitellään videopelifysiikoita Unreal Engine pelimoottorissa ja yritetään löytää optimaalisimmat tavat hyödyntää niitä pelimekaniikoissa. Tutkimus tehdään Pixtell oy:lle, joka kehittää Last Drop– Legend of Seppo nimistä videopeliä ja pelimekaniikat kehitetään myös kyseiseen videopeliin. Pixtell oy on videotuotantoa tekevä yritys, joka sijaitsee Jyväskylässä.

Pelifysiikoita käytetään tässä kontekstissa pääosin pelissä kehitteillä olevissa taistelumekaniikoissa. Työkaluina käytetään Unreal Enginen tarjoamia editoreita ohjelmoimaan tarvittavat pelimekaniikat fysiikoiden aktivoimiseen, kuin myös itse fysiikoiden toimivuus. Ohjelmoimiseen käytetään C++ -sukuista ohjelmointikieltä Blueprint ja pelifysiikoiden muokkaamiseen käytetään Unreal Enginen fysiikkaeditoria. Opinnäytetyön tavoitteena on tutkia pelifysiikoita ja etsiä optimaalisimmat ja parhaat menetelmät toimeksiantajan videopelin taistelumekaniikkoihin.

Työn teoriaosuudessa kerrotaan yleisesti pelimoottoreista ja itse työssä käytettävästä Unreal Enginestä. Tässä osuudessa käsitellään myös mitä Blueprint -pohjainen ohjelmointi on ja miten Unreal Enginen fysiikkaeditorit toimivat ja miten näitä työkaluja voidaan käyttää peliohjelmoinnissa. Käytännön osuudessa analysoidaan käytännön esimerkkejä pelin taistelumekaniikoista ja fysiikoiden toimivuudesta näissä pelimekaniikoissa.

Pixtell Oy toimii opinnäytetyön toimeksiantajana. He ovat videotuotantoa tekevä yritys ja heillä on yli 25-vuotta takana videotuotannon kanssa. Viimeisten vuosien aikana yritys on siirtänyt katseensa videopelituotantoon ja aloittanut Last Drop – Legend of Seppo videopelin tuotannon. Se on avoimen maailman seikkailupeli, jossa pelaaja ohjaa Jyväskyläläistä Seppo päähenkilöä. Pelin tarkoituksena on raitistua alkoholismista ja kokea pelin värikäs tarina.

Työssä käytetään Adoben tarjoamia ilmaisjakelussa olevia rojaltivapaita animaatioita Mixamo palvelusta.

2 Pelimoottorit

2.1 Pelimoottorit yleisesti

Pelimoottorilla tarkoitetaan ohjelmistokehystä, jonka ensisijaisena tarkoituksena on tehdä videopelien kehittämisestä helpompaa pelikehittäjille. Ne ovat usein voimakkaita ohjelmistoja, joilla on virtaviivaistetut toiminnot luoda pelimekaniikkoja ja fysiikkasimulointeja. Pelimoottorit itsessään usein ovat voimakkaita fysiikkamoottoreita. Niiden ominaisuuksiin sisältyy myös pelien renderöinti, törmäystapahtumien laskenta, tekoälyn kehitys peliä varten ja myös monia muita työkaluja, jotka helpottavat pelikehittäjien työtä. Pelimoottorien tarjoamien työkalujen ansiosta pelikehittäjät kykenevät pistämään enemmän aikaa muihinkin pelikehityksessä tarvittaviin alueisiin kuin pelkästään ohjelmoiminen, kuten tekstuurit tai 3D-mallit. Suurin osa videopeleistä nykypäivänä luodaan pelimoottorien avulla. Pelimoottoreita on markkinoilla useita ja ne ovat joskus kehitetty palvelemaan eri tarpeita pelikehittäjille. Suosituimpien kärjessä ovat pelimoottorit Unreal Engine, Unity ja CryEngine. Nämä kolme kyvykästä pelimoottoria tarjoavat laajat työkalut pelikehitykseen niin 3D- kuin 2D-peleille. On myös pienempiä pelimoottoreita kuten Amazon Lumberyard joka tarjoaa työkalut vain 2D-pelien kehitykselle. (Tyler 2022 a.)

Pelimoottoreita ei kuitenkaan välttämättä aina käytetä pelkästään pelien kehitykseen. Molempia Unreal Engineä ja Unityä on käytetty useiden sovellusten tuotantoon. Esimerkiksi Unreal Engineä on käytetty niin autojen suunnittelussa kuin myös suurien TV-sarjojen kehityksessä. Unreal enginen tehokasta renderöinti moottoria käytettiin valtavan suositussa Star Wars TV-sarjassa The Mandalorian luomaan realistisia taustoja sarjan monipuolisille ja värikkäille planeetoille ja ympäristöille. (Farris 2020.)

2.2 Unreal Engine

Unreal Engine on yhdysvaltalaisen pelifirman Epic Gamesin kehittämä tehokas pelimoottori, joka tarjoaa laajat ja suoraviivaistetut työkalut niin pelikehitykseen kuin myös muihinkin tuotantoihin. Unreal Enginen ensimmäinen versio julkaistiin vuonna 1998 nimellä Unreal Engine, mutta uudempien versioiden jälkeen tämä versio tunnetaan laajasti nimeltä Unreal Engine 1. Pelimoottoria on käytetty monien suosittujen ja menestyneiden pelien kehitykseen. Näistä esimerkkeinä on Rocsteady Studiosin kehittämä videopeli Batman: Arkham Asylum, 2K Bostonin ja 2K Australian kehittämä ensimmäisen persoonan seikkailupeli BioShock ja BioWaren kehittämä tieteisfiktio roolipeli Mass Effect. Listaan mahtuu myös itse Epic Gamesin kehittämä valtavan suosittu videopeli Fortnite. Nämä kaikki suositut ja menestyneet eri genren videopelit toimivat todisteina Unreal Enginen kyvykkyydestä luoda erilaisia, monimutkaisia ja menestyneitä videopelejä. (Tyler 2022 a.)

Unreal Enginen käyttö on ollut maksutonta vuodesta 2015 asti ja sitä käyttävät monet suuret pelitalot kuin myös indie-pelikehittäjät. Vaikka pelimoottorin käyttö on ilmaista, Epic Games ottaa rojalteja pelimoottorilla julkaistujen pelien tuotoista. Kaupallisten pelien julkaisuista pelikehittäjien pitää lähettää julkaisulomake Epic Gamesille, jonka jälkeen pelin kehittäjät voivat ansaita noin $1 000 000 bruttotuloja. Tämän ansainnan jälkeen Epic Games lunastaa 5 % tuloista rojalteina, jotka tulee ilmoittaa Epic Gamesille joka neljännesvuosittain. Tämän ansiosta Unreal Engine on hyvinkin sopiva vaihtoehto jopa pienille peliyrityksille tai jopa yksityishenkilöiden peliprojekteille. (Epic Games a.)

Kuten jo aiemmin mainittu, Unreal Engine sopeutuu hyvin muihinkin, kuin pelkästään videopeli projekteihin. Se on suosittu työkalu myös elokuvaprojekteihin ja virtuaalitodellisuus projekteihin. John Favreun ohjaama TV-sarja The Mandalorian käyttää Unreal Enginessä luotuja taustoja kuvauksissa. Sarja kuvattiin osittain oikealla lavalla, jota ympäröi massiiviset LED näytöt, joissa toistettiin kohtauksia varten luodut dynaamiset taustat. Tämän ansiosta näyttelijät pystyvät reagoimaan tapahtumiin reaaliaikaisesti, toisin kuin normaalissa CGI:tä vaativissa väriavainnus lavoilla. Tällaista kuvaustapaa ei ollut oikeastaan ennen käytetty nopeatempoisissa kuvauksissa. Led-näyttöjen taustoja varten tarvittiin neljä täysin synkronoitua tietokonetta pyörittämään Unreal Engineä yhdessä. Saman aikaisesti kolme operaattoria manipuloivat luotuja kohtauksia reaaliaikaisesti. Näillä menetelmillä pystyttiin eliminoimaan kuvauspaikkojen vaatimukset ja se teki VFX-kuvauksista helpompaa reaaliaikaisesti. The Mandalorian -sarjan onnistuvuuden myötä on selvää, että Unreal Enginellä on paikka muissakin kuin vain videopeliprojekteissa. Kuvassa 1. on Led-näyttöjen ja Unreal Engine -kohtauksen käyttöä The Mandalorianin kuvauksissa. (Farris 2020.)

Kuva 1. The Mandalorian TV-sarjan kuvaukset (Farris 2020)

Kirjoitushetkellä uusin versio pelimoottorista on Unreal Engine 5. Se julkaistiin kehittäjien käyttöön 2022 ja vähitellen korvaa aiemman version Unreal Engine 4. Projektissa käytetään kuitenkin vielä Unreal Enginen versiota 4.26.2, sillä peliprojektin siirtyminen Unreal Enginen uudempaan versioon ei ole vielä valmis. Unreal Engine 5 tarjoaa suuria päivityksiä pelimoottoriin, kuten paranneltu reaaliaikainen renderöinti. Tämän ansiosta Unreal Engine kykenee hahmottamaan maailmoja reaaliaikaisesti ja kyvykkäämmin kuin aiemmissa versioissa. Tähän sisältyy Lumen, joka on kokonaan dynaaminen ratkaisu maailmanlaajuiselle valaistukselle kehittäjän maailmassa. Lumen tarjoaa mahdollisuuden realistisen uskottavalle valaistukselle, joka reaaliaikaisesti mukautuu geometrian mukaan. Tämän avulla kehittäjien ei tarvitse enää luoda monimutkaisia UV-kuvia, valokarttoja tai heijastuskaappauksia. Valojen luominen ja muokkaus tapahtuu kätevästi Unreal editorissa ja valaistus pysyy samana kehittäjille kuin pelaajille pelin ajon jälkeen. Toisena uutena ominaisuutena on Nanite, joka on virtualisoitu mikropolygonigeometriajärjestelmä. Sen avulla voidaan luoda pelejä, joissa on valtavia määriä yksityiskohtia geometriassa. Se tarjoaa mahdollisuuden käyttää malleja, joissa on jopa miljoonia polygoneja. Nämä voivat olla joko fotogrammetrisesti skannattu tai 3D-veistoksia, jotka on luotu 3D-mallinnusohjelmistolla, kuten ZBrush tai Blender. Nämä suuret mallit voidaan asettaa peliin jopa miljoonia kertoja ilman että riskeerataan reaaliaikaisen kuvanopeuden suorituskykyä. Videopelien laatustandardit ovat olleet nousussa ja nykypäivänä vaaditaan suuria resoluutioita ilman että uhrataan 60 kuvaa

sekunnissa rajaa. TSR, eli Temporal Super Resolution on Unreal Enginen ratkaisu tähän ongelmaan. Se on pelimoottoriin sisäänrakennettu järjestelmä ja se on alustasta riippumaton. Se tarjoaa parempaa suorituskykyä mahdollistamalla pelin toiston pienemmällä resoluutiolla mutta käyttäen samalla pikselitarkkuutta kuin kehykset, jotka on renderöity suuremmalla tarkkuudella. Tämä ominaisuus on kuitenkin vielä vasta Beta-vaiheessa. (Epic Games b.)

Unreal editorin yksi tärkeimmistä ominaisuuksista on sen työkalut luoda avoimia pelimaailmoja kätevästi. Uusi maailmanjakamisjärjestelmän avulla maailmojen hallinta on helpompaa luomalla automaattisesti ruudukot soluiksi. Moottorissa on myös hyvät työkalut yhteistyölle. Kehittäjät voivat työskennellä reaaliaikaisesti saman maailman parissa ilman että häiritsevät toisten tekemää työtä. Moottori tarjoaa myös sisäänrakennetut työkalut hahmojen ja animaatioiden muokkaamiselle. Animaatioiden kehitys ja mukautus on mahdollista Unreal Enginen sisällä ja animaatioiden siirtäminen uudelleen kohdistuksen avulla on mahdollista hahmojen välillä. (Epic Games b.)

3 Taistelumekaniikat videopeleissä

Videopeleissä on usein hyvinkin erilaiset taistelumekaniikat. Niiden kehittämiseen kuluu paljon aikaa ja resursseja, sillä mekaniikoiden pitää tuntua hyvältä pelaajalle ja myös olla tasapainotettuja pelin vaikeustason mukaan. Taistelumekaniikat voivat joko toimia pelin keskeisimpänä mekaniikkana, tai ne voivat olla pienemmässä roolissa, jos peli ei keskity paljoa taisteluihin. Näissä peleissä taistelumekaniikat rakentavat koko pelin itsessään ja ovat pelien tärkein pelimekaniikka. Tällaisissa taistelupeleissä vastustajien pelihahmot asetetaan vastakkain ja on keskeistä yrittää torjua tai väistää vastustajan iskuja samalla kun pelaaja yrittää itse osua omilla iskuillaan vastustajaan, kunnes vastustajan elämäpisteet vähentyvät nollaan. Taistelut videopeleissä yleensä voitetaan vähentämällä vastustajan elämäpisteet nollaan. Eri pelihahmoilla voi olla uniikkeja kykyjä ja yleensä peliohjaimen syötteet pitää antaa tarkasti ja hyvin ajoitetusti. Peleihin yleensä kuuluu myös hyvä määrä strategiaa, kuten oman pelihahmon asettaminen pelimaailmassa kohtaan, josta saadaan etulyöntiasema vastustajaan tai erikoishyökkäysten laukaiseminen juuri oikeaan aikaan. Vastustajana voi toimia myös toinen palaaja, eikä välttämättä tietokonevastustaja. (Iyer 2021.)

4 Pelifysiikat

4.1 Pelifysiikat peleissä

Pelifysiikat ovat videopeleissä simuloituja fysiikan lakeja. Pelifysiikkojen luomiseen käytetään usein fysiikka moottoreita ja edistynyttä ohjelmointilogiikka. Tarkoituksena ei välttämättä ole aina kuitenkaan simuloida oikean maailman fysiikkoja, vaan fysiikkojen laatu riippuu usein siitä, millainen videopeli on kyseessä. Peleissä, jotka koittavat simuloida oikeaa elämää usein yritetään luoda hyvin realistiset fysiikat kuten Rockstar Gamesin julkaisemassa ja massiivisen suositussa avoimen maailman ammuskelupelissä Grand Theft Auto V, mutta vähemmän realistisuutta vaativissa videopeleissä fysiikat voivat palvella paremmin pelin pelattavuutta, ja mekaniikkoja jos ne ovat epärealistisemmat, kuten Psyonix:in ajoneuvoihin perustuvassa urheilupelissä Rocket League, joka on myös kehitetty Unreal Engine pelimoottorilla. Kun nykyajan videopelit ovat jo graafisesti vaikuttavia, pelifysiikat tuovat mukaan uuden kerroksen immersiota pelaajille ja ne auttavat upottamaan pelin pelaajat videopelin maailmaan. Videopelifysiikat yrittävät jäljitellä oikean maailman fysiikoita, mutta eivät tee sitä aivan täysinäisesti. Tähän on yleensä syynsä pelin pelattavuuden puolella. Täysin realistinen peli ei olisi välttämättä hauska pelata, joten pelinkehittäjät ottavat tiettyjä vapauksia pelifysiikkojen kanssa saadakseen pelistä hyvän tuntuisen pelata. (Tyler 2022 b.)

Fysiikkojen ohjelmoiminen peliin parantaa siis pelin immersion arvoa laajasti. Unreal Engine käyttää PhysX fysiikkamoottoria sen fysiikkasimulaatioiden laskemiseen. PhysX on Nvidia Corporationin kehittämä fysiikkamoottori ja se toimii useilla eri alustoilla, kuten Microsoft Windows, konsolialustat ja myös useat mobiilialustat. Valmiiksi ohjelmoituja fysiikkamoottoreita asennetaan pelimoottoreihin syystä, että ne vapauttavat pelikehittäjät suurelta määrältä ylimääräistä työtä ohjelmoida omat klassiset Newtonin fysiikkalait peleihinsä. (Nvidia Developer.)

4.2 Fysiikka simulaatiot

Pelifysiikat jakautuvat Unreal Enginen sisällä moniin eri alajärjestelmiin. Näihin sisältyy ominaisuudet niin fysiikkojen simuloimisesta kuin myös törmäystapahtumien laskeminen. Fysiikka simulaatioita varten objekteilla pitää olla omat fysiikka resurssit. Nämä voivat olla asetettuna objekteihin suoraan tai mukautettuina, itse luotuina materiaaleina. Näitä fysiikkaresursseja käytetään ohjaamaan fysiikka simulaatioita kuin myös niiden ominaisuuksien määrittämistä. Simulaatiolla tässä kontekstissa tarkoitetaan sitä, kun objekti on dynaamisesti vuorovaikutuksessa pelimaailman kanssa. Tämä voi tarkoittaa niin törmäystapahtumia kuin myös ragdoll tyylisiä fysiikoita. Ragdoll -fysiikoilla tarkoitetaan käytännössä tapahtumaa, kun pelihahmossa aktivoidaan täydet fysiikat, ja hahmo putoaa velttona lattialle

räsynukkemaisesti. Tätä ominaisuutta käytetään usein pelihahmojen kuolemistapauksissa. (Epic Games c.) Kuvassa 2. on pelihahmo ragdoll tilassa.

Kuva 2. Pelihahmossa on aktivoitu ragdoll simulaatio (The Last Drop 2022)

Tähän sisältyy myös fysiikkakehot, eli physics bodies. Näillä tarkoitetaan yksinkertaistettuja 3D-meshejä, joita käytetään fysiikkasimulaatioissa. Niillä voi olla monia eri muotoja, kuten laatikkoina, kapseleina tai palloina. Näiden käyttöön on pääosin syynä 3D-mallien jatkuva kehitys ja monimutkaisuus, joten yksinkertaisia muotoja on hyvä käyttää fysiikka simulaatioiden ohjaamiseen. (Epic Games d.)

Unreal Enginessä on myös fysiikka työkalu kaaos tuhoutuminen. Kaaos tuhoutuminen on järjestelmä, jonka avulla voidaan luoda tuhoamisominaisuuksia reaaliajassa. Tällä tuhoamisjärjestelmällä kehittäjät kykenevät määrittämään tarkasti miten erilaiset pelin objektit ja geometria hajoavat niiden tuhoutuessa. Nämä geometriat luodaan etukäteen hajoaviksi simulaation aikana. Nämä objektit ovat siis aluksi kokonaisia objekteja, mutta ne voivat murentua simulaation aikana riippuen ympäristön vaikutuksista, kuten törmäykset. Esimerkiksi pelimaailmassa voi olla jäykkä kuutio, joka on asetettu tuhoutumaan, kun pelihahmo lyö sitä. Ne käyttävät kerääntymismallia, joka ohjaa miten geometria ja objektit kiinnittyvät toisiinsa. Tämä tuhoutumisjärjestelmä sisältää myös oman välimuistijärjestelmän, jotta tuhoutumismekaniikat toimisivat mahdollisimman sujuvasti pelin ajon aikana. (Epic Games e.) Kuvassa 3. on näkymä tuhoutumisjärjestelmästä räjähdysefektien kanssa.

Kuva 3. Pelin sisäinen tuhoutumisjärjestelmä räjähdysefektin kanssa (Epic Games e.)

4.2.1 Fysiikka pohjaiset animaatiot

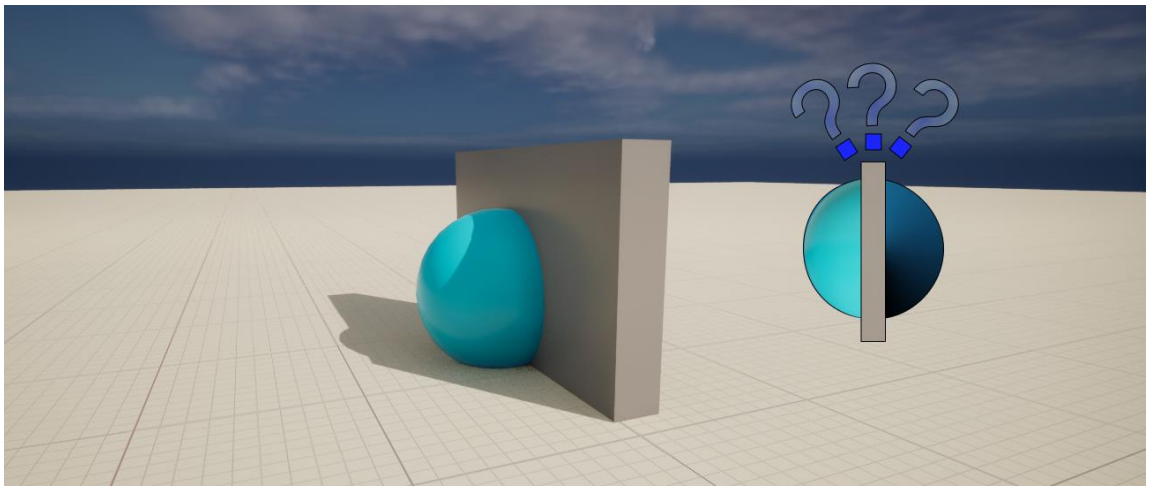
Unreal Engine mahdollistaa animaatioiden ja fysiikka simulaatioiden yhdistämisen toisiinsa. Tämän päätarkoituksena on luoda sulavia simulaatioita, jotta voidaan luoda realistisen näköisiä efektejä hahmoihin. Nämä fysiikat ovat yhdistettyinä pelihahmojen luurankoihin, jotka ohjaavat hahmojen animaatioiden kulkua, ja on myös huomioitava, että luurangolla on oltava oma fysiikkaresurssi, jotta fysiikkasimulaatiot toimivat. Fysiikka simulaatiota kutsutaan usein yleisesti ragdolliksi. Tässä on syynä se, että kun fysiikka simulaatio aktivoidaan hahmomallissa, se putoaa veltoksi. Tätä ominaisuutta voidaan käyttää monillakin tavoin, mutta yleisin on esimerkiksi pelihahmojen kuolema tapahtumat. Näissä tapahtumissa pelihahmolla aktivoidaan ragdoll, kun niiden elämäpisteet menevät nollaan ja hahmo ”kuolee”. Ragdollia voi käyttää myös esimerkiksi osumisreaktioissa. Osumisreaktioilla tarkoitetaan reaktiota, joka pelihahmoilla on, kun niihin osutaan jollain tavalla, kuten esimerkiksi vastustajan lyönti. Ragdoll voidaan näissä tapauksissa aktivoida vain tietyissä osissa pelihahmojen kehoa, luoden erilaisia reaktioita riippuen siitä, mihin kohtaan pelihahmoa osutaan. (Epic Games f.)

4.2.2 Törmäysvasteet

Collision, eli törmäysvasteet ja kuin myös jäljitysvasteet käsittelevät törmäyksiä ja säteitä Unreal Enginessä ajon aikana. Törmäyksessä objekti on suorassa vuorovaikutuksessa toisen objektityypin kanssa, ja kyseisen objektin asetukset määrittävät miten ne käyttäytyvät keskenään. Jokainen objekti voidaan tässä asettaa käyttäytymään tietyllä tavalla, kun törmäys- tai päällekkäisyystapahtuma tapahtuu. Objektit voivat estää toisia objekteja, mennä päällekkäin tai olla huomioimatta toisiaan. (Epic Games g.)

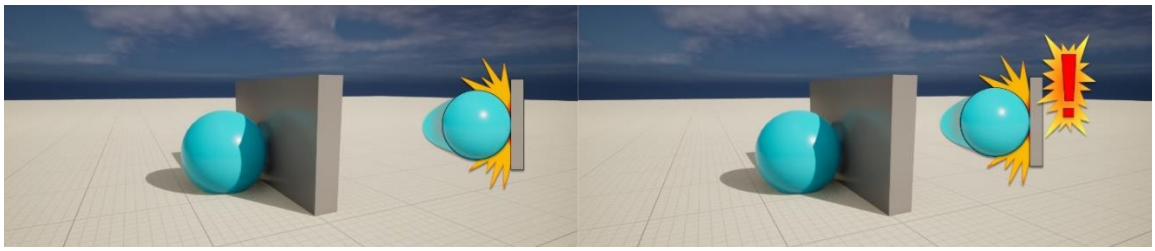
Törmäysten käsittelyssä on sääntöjä, joita pelikehittäjien tulee pitää mielessä. Nämä ovat törmäysten asetukset, joissa voidaan määrittää objektien käyttäytyminen eri törmäystilanteissa. Tämä liittyy vahvasti pelin taistelumekaniikkojen lyöntien rekisteröimiseen ja täten myös pelin fysiikkojen aktivoimiseen oikeaan aikaan. Estääkseen objektin pitää molemmilla

objektityypeillä olla asetettuna asetus ”estä”. Osumistapahtuma simulaatiot tulee olla käytössä, jotta nämä tapahtumat voidaan suorittaa oikein. Kehittäjien tulee myös huomioida objektien päällekkäin asettelun ja objektien huomioimatta ottamisen erot, vaikka ne saattavat vaikuttaa aluksi hyvinkin samanlaisilta. Suurimpana erona näissä kahdessa asetuksessa on se, että päällekkäin asettelussa voidaan esimerkiksi laukaista koodia tapahtuman lauetessa, toisin kuin objektien huomioimatta ottamisessa, jossa objektit vain yksinkertaisesti läpäisevät toisensa. Objektien päällekkäin asettelua voidaan käyttää esimerkiksi rekisteröimään lyöntien osuminen pelin taistelumekaniikoissa. Kuvassa 4. on tapahtuma, kun pallo-objekti läpäisee seinän ja luo joko päällekkäin asettelu tapahtuman, tai huomiotta ottamisen tapahtuman. (Epic Games g.)



Kuva 4. Päällekkäin asettelu tapahtuma (Epic Games g.)

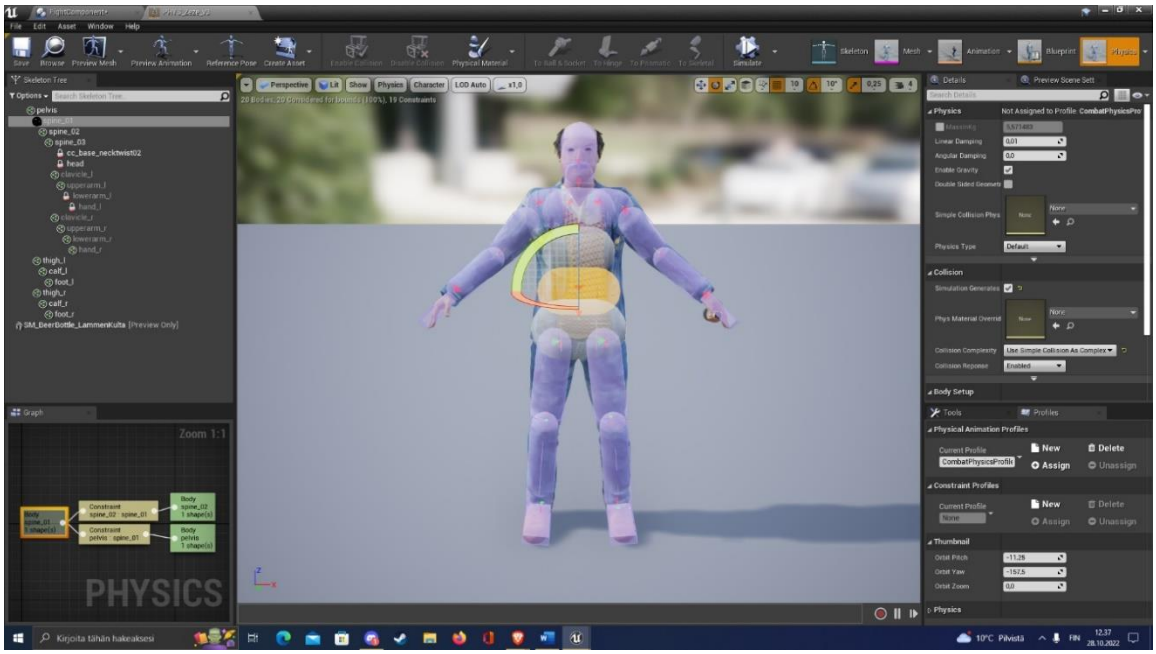
Päällekkäisyys tapahtumia voidaan tehdä myös vaikka objektit olisi asetettu estämään toisensa. Tämä tapahtuu usein, jos objekti liikkuu hyvin suurilla nopeuksilla. Pitää myös olla tarkkana asetusten kanssa, sillä molempien objektien asetukset pitää asettaa oikein halutun käyttäytymisen saavuttamiseksi. Esimerkiksi jos kuvan pallo-objektin asetukset on asetettu menemään päällekkäin seinän kanssa, mutta seinän asetukset on asetettu ottamaan pallo huomiotta, niin päällekkäin asettelun tapahtumaa ei koskaan tapahdu. Toinen esimerkki on, jos kuvan pallo-objekti on asetettu menemään päällekkäin, mutta seinä on asetettu estämään pallo-objekti, niin päällekkäisyys tapahtuma tapahtuu, eikä esto tapahtuma. Jos molemmat objektit on asetettu estämään toisensa, voidaan laukaista koodia onHit -tapahtuman avulla. Tämä tapahtuma siis rekisteröi koska objektit törmäävät toisiinsa ja laukaisee halutun koodin tämän tapahtuessa. Kuvassa 5. vasemmalla pallo-objekti törmää seinän kanssa. Tämä on normaali törmäystapahtuma jossa seinä estää pallon liikkeen. Kuvassa oikealla pallo törmää myös seinän kanssa, mutta siinä tapahtuu myös OnHit -tapahtuma, jonka tapahtuessa voidaan laukaista koodia. Näitä törmäysvasteita kutsutaan usein peleissä nimellä hitbox (Epic Games g.)



Kuva 5. Törmäys ja OnHit -tapahtuma (Epic Games g.)

4.3 Unreal Enginen fysiikkaeditori

Unreal Engine tarjoaa kehittäjille hyvän fysiikkaeditorin, jolla voi muokata niille asetettujen objektien fysiikkojen ominaisuuksia. Hahmon sisällä olevat kuplat toimivat törmäyslaatikkoina, jotka määrittävät miten hahmo käyttäytyy fysiikkasimulaatiossa. Itse hahmon 3D-mallissa ei siis ole törmäyslogiikkaa, vaan se on mallin sisälle asetettuna törmäyskuplina. Käytännössä nämä kuplat ovat siis ne, jotka törmäävät objekteihin. Kuplat ovat asetettuna kiinni hahmon luurangon osiin, joten ne liikkuvat hahmon animaatioiden ja fysiikkasimulaation mukana. Jos kuplia ei olisi kiinnitetty luihin, ne putoaisivat hahmon läpi maahan simulaation alkaessa. Editorissa on oikealla lukuisia asetuksia, joilla voidaan määritellä fysiikkojen ominaisuuksia. Näihin kuuluu muun muassa massa kilogrammoissa, joka nimensä mukaan määrittää kilogrammoissa objektin massan. Tämä vaikuttaa esimerkiksi siihen kuinka paljon voimaa tarvitaan objektin liikuttamiseen (Epic Games h). Tärkeänä ominaisuutena tässä editorissa on kuitenkin fyysiset animaatio profiilit. Tämä antaa kehittäjille mahdollisuuden luoda useita asetusprofiileja fysiikoille, joita voidaan vaihtaa laukaisemalla koodia. Tämä on tärkeää projektissa, sillä simuloituja fysiikkoja käytetään muissakin ominaisuuksissa kuin vain taistelumekaniikoissa. Tätä varten pitää luoda oma fysiikkaprofiili vain taistelufysiikkoja varten, sillä niillä voi olla eri vaatimukset kuin muilla fysiikoilla. Luotu CombatPhysicsProfile laukaistaan taistelun lyöntien tapahtuessa koodin avulla, ja palautetaan takaisin normaaliin tilaan pienen ajan kuluttua. Profiileja voi olla niin monta kuin niitä tarvitaan. (Epic Games i.) Kuvassa 6. on Unreal Enginen fysiikkaeditori, jossa muokataan pelaajahahmon fysiikkoja ja testataan fysiikkasimulaatioita.



Kuva 6. Unreal Enginen fysiikkaeditori (The Last Drop 2022)

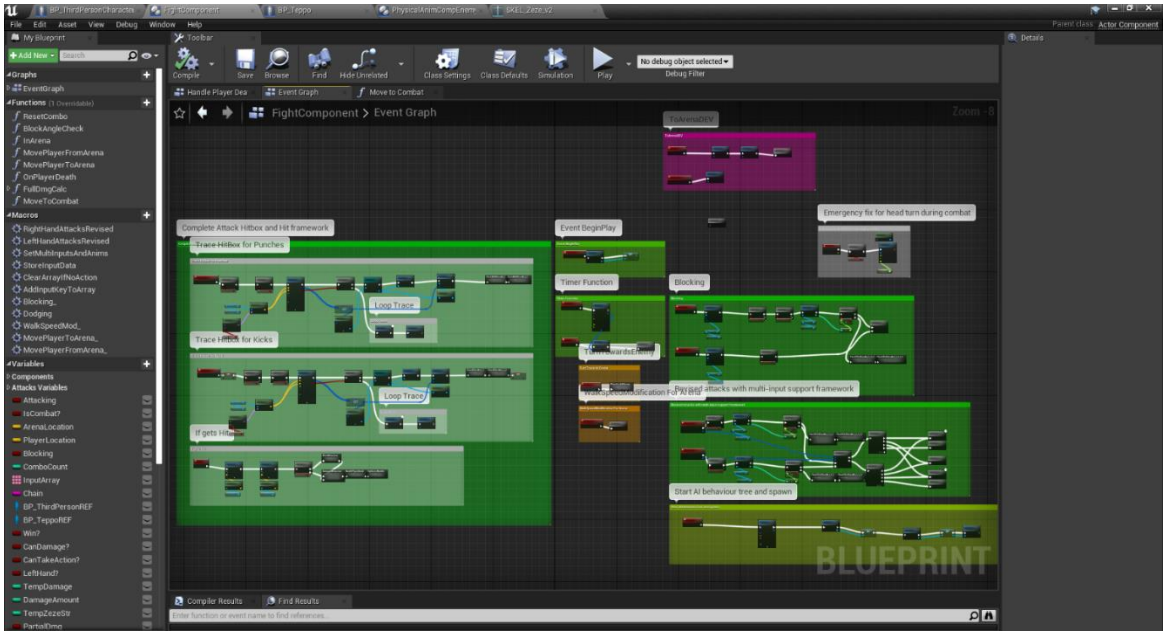
4.3.1 Fysiikkamateriaalit

Fysiikkamateriaaleilla tarkoitetaan luotuja resursseja, jotka liitetään suoraan objekteihin, joissa halutaan simuloida pelifysiikoita. Näillä resursseilla voidaan ohjata ja muokata objektin fysiikkojen ominaisuuksia ja määrittää miten se käyttäytyy simulaation aikana. Näitä fysiikkaresursseja voidaan luoda suoraan Unreal Enginen sisällä ja asettaa kiinni objekteihin, jotka niitä tarvitsevat. Jokaisella fyysisellä objektilla on olemassa jo oma vakio fysiikkamateriaali, mutta luomalla uuden voidaan helposti mukauttaa sen ominaisuuksia. Näitä voidaan käyttää määrittämään, miten objekti käyttäytyy eri tavoin eri tapahtumien tapahtuessa fysiikka simulaatiossa, kuten esimerkiksi miten pelihahmon ruumis käyttäytyy sen kuollessa, tai miten jalkapallo käyttäytyy sitä potkaistaessa. (Epic Games c.)

Näissä fysiikkaresursseissa voidaan muokata monia fysiikkoihin liittyviä ominaisuuksia objekteissa. Näihin kuuluu muun muassa kitka, joka määrittää kuinka helposti objekti voi liukua pinnoilla. Asetuksissa voidaan myös muokata, kuinka pomppiva objekti on. Siinä määritetään energian määrä, jota säilytetään, kun objekti törmää toisen pinnan kanssa. Objektin massaa voidaan myös muokata riippuen siitä, kuinka suuri objekti on. Objektin suurentuessa sen massaa voidaan myös lisätä. Liittyen myös tuhoutuviin objekteihin asetuksissa voidaan säätää, kuinka paljon vahinkoa objektin tarvitsee ottaa ennen kuin se hajoaa. Ominaisuuksissa on myös omat asetukset pelin ajoneuvojen ominaisuuksille, kuten renkaiden kitka. (Epic Games j.)

4.3.2 Blueprintit

Unreal Engine tarjoaa uudenlaisen ja uniikin tavan ohjelmoida mekaniikkoja videopeleihin. Tämä on C++ -pohjainen visuaalinen ohjelmointikieli blueprint. Tämä järjestelmä perustuu täysin solmupohjaisen käyttöliittymän käyttöön, jota käytetään pelimekaniikkojen luontiin ilman että tarvitsee kirjoittaa oikeaa koodia. Kuten monet muutkin ohjelmointikielet, blueprinttiä käytetään oliopohjaiseen ohjelmointiin. Blueprinttien tarkoituksena on tarjota joustava ja tehokas työkalu suunnittelijoille pelimekaniikkojen nopeaan ja joustavaan luontiin, jotka voidaan myöhemmin täydellistää toisella perinteisellä ohjelmointikielellä. Näin suunnittelijat voivat käyttää samanlaisia työkaluja kuin ohjelmoijat, auttaen lopullisen tuloksen hahmottamisessa. Se toimii myös toisin päin laajentamaan valmiiksi kirjoitettua C++ -koodia. Käytännössä blueprint ohjelmointikieli toimii yhdistämällä solmuja ja toimintoja toisiinsa kiinni johdoilla. Tämä mahdollistaa monimutkaisten mekaniikkojen luonnin visuaalisesti. Näitä solmukaavioita voidaan käyttää muun muassa objektien kuin myös yleisten pelitapahtumien luontiin. (Epic Games k.) Kuvassa 7. on Unreal Enginen näkymä blueprint -editorista. Vasemmalla ikkunassa on luodut funktiot, makrot ja muuttujat, ja keskellä on graafinen näkymä visuaalisista solmuista, joita käytetään ohjelmoimisessa. (Epic Games l.)



Kuva 7. Näkymä Blueprint editorista

Blueprinttejä on erityyppisiä, joista yleisimmät kehityksessä ovat taso blueprint ja blueprint -luokat. Taso blueprint on pelin tasolle määritelty kaavio. Sillä voi viitata ja muokata objekteja kyseisellä tasolla vapaasti. Se on siis globaalisti toimiva Blueprint ja jokaisella tasolla on oma taso blueprint. Uusia taso blueprinttiejä ei voi kuitenkaan manuaalisesti luoda. Tähän sisältyy myös tason videoiden toisto ja myös pelin tarkistuspisteet. Taso blueprint voi

olla myös vaikutuksessa tasoon kiinnitettyjen muiden blueprint -luokkien kanssa. Näin voidaan kutsua esimerkiksi muuttujia tai funktioita toisista luokista ja laukaista niitä taso blueprintissä. (Epic Games m). Blueprint -luokat taas ovat blueprinttejä joita kehittäjät voivat luoda manuaalisesti eri pelissä oleville asioille. Nämä luokat luodaan kätevästi Unreal editorin avulla ja ne voidaan myöhemmin asettaa pelitasoihin kuten mikä tahansa muu peliobjekti toimimaan instanssina. Blueprint tyyppejä on monenlaisia eri tarkoituksiin, joka määritetään valitsemalla sen yläluokka. Nämä yläluokat ovat näyttelijä, pelinappula, hahmo, pelaajan ohjain ja pelitila. näyttelijäluokka on yksinkertaisesti objekti, joka voidaan synnyttää pelimaailmaan. Pelinappula taas on ylempi versio näyttelijästä, sillä se voidaan ”omistaa” ja täten vastaanottaa pelaajan ohjausta. Hahmo on taas yksi ylemmäs pelinappulasta, koska sillä on lisätty kyky liikkua. Pelaajan ohjain on myös ylempi versio näyttelijästä, sillä se on vastuussa pelaajahahmon ohjaamisesta, joka on myös pelinappula luokan tyyppi. Ja viimeisenä pelitila on luokka, joka määrittää peliin itseen liittyvät asiat kuten säännöt, pisteet ja muut globaalit pelin säännöt. Yllä mainitut luokat ovat yleisimmät käytetyt blueprint -luokat, mutta olemassa on muitakin ja jokaista niistä voidaan käyttää samoin ylempänä luokkana. (Epic Games n.)

4.3.3 Blueprint komponentit

Komponentteja käytetään eristääkseen pelin taistelumekaniikat pelaajahahmon omasta blueprintistä. Tähän on syynä se, että peliprojektissa on monia eri kehittäjiä, jotka aktiivisesti haluavat muokata pelaajahahmon blueprinttiä. Tästä syystä on työnkulun kannalta helpompaa luoda komponentit omille tekemille mekaniikoille, joita voidaan helposti kutsua pelaajan blueprintissä, jotta ei tule liian paljon konflikteja myöhemmin version hallinnassa, kun omaa tekemää työtä tallennetaan GitHubiin. Komponentti on käytännössä erityinen objektityyppi, joka voidaan liittää näyttelijöihin aliobjektina. Näin komponentin sisällä kirjoitettuja mekaniikkoja voidaan kutsua helposti siihen liitetyn näyttelijän sisällä, ja näin laukaista haluttuja funktioita. Komponentit ovat erityisen hyödyllisiä myös yhteisten pelimekaniikkojen jakamiseen eri näyttelijöiden välillä. Tämä tarkoittaa käytännössä sitä, että kaksi näyttelijää voivat käyttää yhteisesti yhtä koodia, kuten vaikka samaa logiikkaa pelifysiikoiden aktivoimiselle. Esimerkiksi komponenttiin voidaan kirjoittaa pelifysiikkojen koodi, ja sitä voidaan käyttää sen jälkeen useammassa näyttelijässä. (Epic Games o.)

Komponentteja on kolme eri tyyppiä:

- Näyttelijäkomponentti, joka on komponenttien perusluokka. Lähes kaikki minkä kanssa pelaaja on vuorovaikutuksessa pelissä, on jonkinlainen komponentti.

- Kohtauskomponentti, joka on kuin näyttelijäkomponentti, mutta se on asetettuna pelimaailmaan fyysisesti. Kohtauskomponentit voivat toimia ”juurena” useille näyttelijäkomponenteille.

- Primitiiviset komponentit, jotka ovat kohtauskomponentteja, mutta ne sisältävät jonkinlaista geometriaa. Tätä geometriaa käytetään pääosin törmäystapahtumiin tai renderöintiin. Näihin sisältyy yleisesti geometriat kuten laatikko- ja kapselikomponentit. Nämä geometriat voivat olla kokonaan näkymättömiä törmäystapahtumia varten.

Fysiikkatila on tarpeellinen, jotta näyttelijäkomponentti kykenee olemaan vuorovaikutuksessa fysiikkasimulaation kanssa. Oletuksena näyttelijä- ja kohtauskomponenteilla ei ole fysiikkatilaa, tämä on oletuksena vain primitiivisillä komponenteilla. (Epic Games o.)

5 LAST DROP – Legend of Seppo videopeli

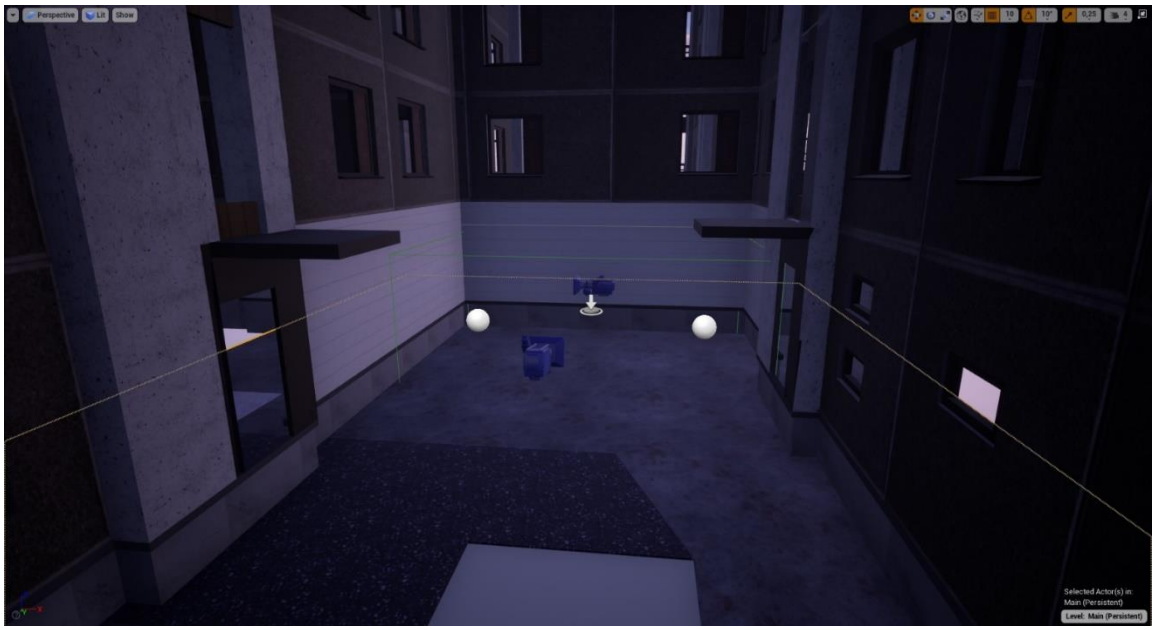
5.1 Last Drop -pelin taistelumekaniikat

Taistelumekaniikat tehtiin Pixtell oy:n avoimen maailman videopeliin. Pixtell oy on videotuotantoa tekevä yritys, joka on lähivuosina aloittanut myös pelikehityksen ja Last Drop -peli on heidän ensimmäinen peliprojektinsa. Pelin taistelusta haluttiin tehdä areenamainen, mikä tarkoittaa sitä, että taistelu ei tapahdu pelin avoimessa maailmassa, vaan suljetulla alueella, jossa pelin kamera on asetettuna kuvaamaan molempaa pelaajaa ja vastustajaa sivusta. Tähän toimi inspiraationa osittain Bandai Namcon kehittämä taistelupeli Tekken kuin myös vähemmän tunnettu Distinctive softwaren kehittämä nyrkkeilypeli 4D Sports Boxing. Toisena vaatimuksena oli, että taistelussa käytettäisiin fysiikkoihin perustuvia animaatioita. Tämä tarkoittaa käytännössä sitä, että kun pelaajan lyönti osuu viholliseen vihollisella ei ole valmiiksi nauhoitettua animaatioita reaktiota varten, vaan reaktion hoitaa Unreal Enginen fysiikkamoottori. Tämänlainen järjestelmä ottaa löysästi inspiraatiota Boneloafin kehittämästä pelistä Gang Beasts, jossa simuloidut fysiikka-animaatiot ovat hyvin sidottuna pelin mekaniikkoihin. Kuvassa 8. on demoalue Last Drop -pelistä kuvattuna Unreal Enginen editorista.

Kuva 8. Last Drop -pelin demoalue (The Last Drop 2022)

Pelin taistelussa olevat hahmot ovat asetettu vastakkain ja pelin normaalisti oleva kolmannen persoonan kamera siirtyy kuvaamaan molempia pelaajaa ja vastustajaa sivusta dynaamisesti. Kamera on ohjelmoitu osoittamaan pelaajan ja vastustajan välissä laskettua keskipistettä, ja se loittonee ja lähestyy riippuen siitä, kuinka kaukana pelaaja ja vastustaja ovat

toisistaan. Kamera on lähempänä, kun pelaaja ja vastustaja ovat lähekkäin ja kauempana kun pelaaja ja vastustaja ovat kauempana toisistaan. Vaikka kamera on vakituisessa kohdassa, taistelun liike ei ole silti samanlaista kuin sidescroller tyypin peleissä. Pelaaja ja vastustaja ovat lukittuna katsomaan toisiinsa päin, jotta pelaaja ei voi vahingossa kääntyä katsomaan väärään suuntaan. Näin taistelu pidetään suoraviivaisempana. Kuvassa 9. on taisteluareena kuvattuna Unreal Enginen editorista.



Kuva 9. Taisteluareena (The Last Drop 2022)

Pelaajalla on yhteensä kuusi mahdollista lyöntiä. Nämä ovat molemmat kolmen yhdistelmiä molemmille käsille. Pelaaja voi myös yhdistää eri käsien lyöntejä, mutta lyöntiyhdistelmät nollaantuvat aina kolmannen lyönnin jälkeen. Pelaajalla on myös elämä ja kestävyys mittari. Elämä mittarilla määritetään, kuinka paljon kuntopisteitä pelaajalla on. Jos kuntopisteet laskevat nollaan tai miinuksen puolelle, pelaaja menettää tajuntansa ja häviää taistelun. Tämän tapahtuessa pelaaja siirretään pois taisteluareenalta takaisin pelimaailmaan ja hän menettää 90 % elämäpisteistään. Kestävyysmittarilla taasen määritetään asioita kuten kuinka paljon pelaaja voi lyödä tai torjua vastustajan liikkeitä. Pelaaja menettää pienen määrän kestävyyspisteitä, kun hän tekee lyönnin tai torjuu vastustajan lyönnin onnistuneesti. Pelaaja menettää suuremman määrän kestävyyspisteitä myös, jos vastustaja saa osuttua pelaajaan ilman että iskua torjutaan. Taistelussa on myös monimutkainen laskujärjestelmä vahingon laskemiselle. Käytännössä pelaajalla on voima ja nopeus tasot, jotka ovat taistelun aikana vastavaikutuksessa vastustajan voima ja nopeus tasojen kanssa. Jos pelaajan voima ja nopeus tasot ylittävät vastustajan, on mahdollista 90 % ajasta tehdä täysinäistä vahinkoa lyönneillä. Täysinäinen vahinko lasketaan kertomalla pelaajan voimataso kahdella. Kerrotaan myös pelaajan ketteryystaso luvulla 0.1, jonka jälkeen nämä lasketaan

yhteen. Lopputulos pitää vielä lopuksi kääntää negatiiviseksi arvoksi, jotta vahinko voidaan laskea oikein. Muuten lyönneillä tehdään vain osittaista vahinkoa. Osittainen vahinko lasketaan yksinkertaisesti hakemalla sattumanvarainen luku tuloksesta kahden float -tyyppisen luvun avulla. Laskujen tekemiseen voidaan käyttää Unreal Enginen tarjoamaa matemaattinen lause solmua, johon voidaan yksinkertaisesti kirjoittaa haluttu funktio, ja se luo automaattisesti mukautetun solmun laskufunktiota varten (Epic Games p). Mukana taistelussa on myös efektit, jotka indikoivat milloin eri asioita tapahtuu taistelussa, kuten asuman ottaminen, kestävyyspisteiden loppuminen tai torjunnan onnistunut tapahtuminen. Hahmot välkkyvät hetkellisesti punaisen värisinä, kun lyönti osuu vahingoittaen osuttua hahmoa. Hahmot välkkyvät keltaisena, kun kestävyys pisteet ovat loppu ja hahmot välkkyvät sinisenä, kun vastustajan lyönti torjutaan onnistuneesti. Kuvassa 10. on pelin taistelussa tapahtuma, kun vastustajan lyönti vahingoittaa pelaajaa. Kuvassa näkyy, kuinka pelaaja välkkyy punaisena ottaessa vahinkoa, ja kuinka pelaajahahmo horjahtaa hieman lyönnin iskusta. Vasemmassa alakulmassa näkyvät pelaajan elämäpisteet ja kestävyysmittari allekkain. Ottaessa vahinkoa vahingon määrä näkyy myös ruudulla leijuvana punaisena numerona.

Kuva 10. Vastustaja vahingoittaa pelaajaa lyönnillä (The Last Drop 2022)

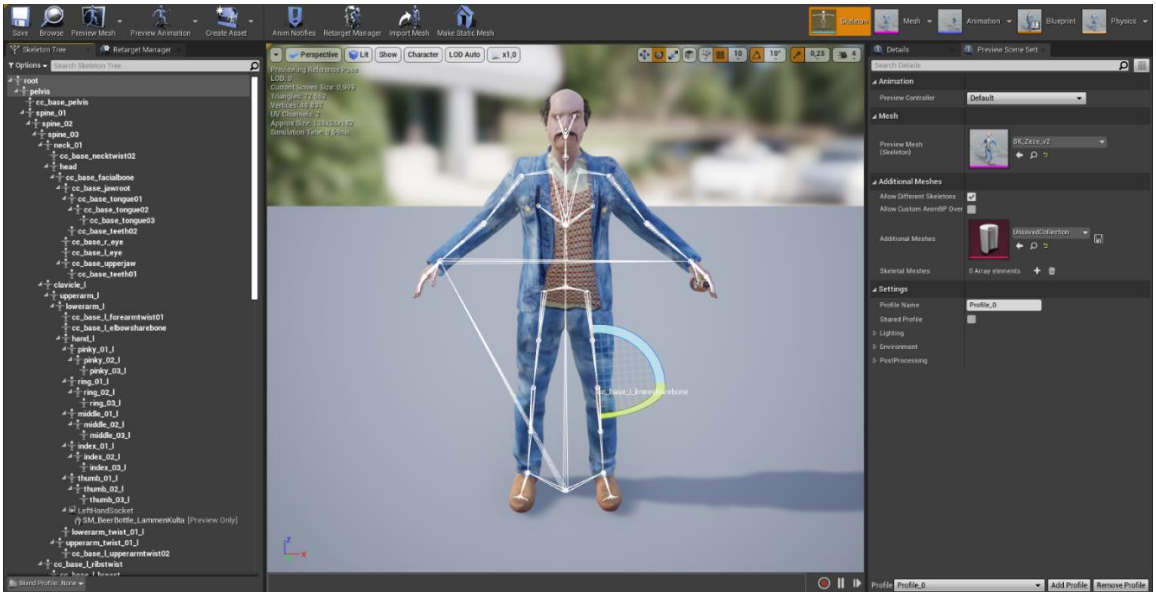
Tällaisen toiminnan tuottamiseksi pitää muokata pelin hahmojen fysiikkaresursseja ja kirjoittaa blueprint -koodia fysiikoiden aktivoimiseksi. Sillä tässä tapauksessa fysiikat halutaan aktivoida hahmoissa vain, kun lyönti rekisteröidään, pitää fysiikat asettaa oletuksena pois päältä ja aktivoida vasta kun hitboxin ja vastustaja hahmon välillä tapahtuu päällekkäistapahtuma. Tämän takia järjestelmä on hyvinkin monimutkikas.

5.2 Hahmojen luuranko

Jokaisella pelihahmolle, joille halutaan asettaa joko animaatioita tai simuloituja pelifysiikoita pitää olla oma luuranko. Käytännössä luuranko on hierarkia ”luita”, jotka ohjaavat miten tietyt osat pelihahmon kehoa liikkuvat. Näillä luilla pelikehittäjät yrittävät jäljitellä oikeaa ihmisen biologista luurankoa, jotta hahmon liikkeet olisivat mahdollisimman todentuntuisia. Luurankoja käytetään pääosin animaatioiden luomiseen, mutta tässä tapauksessa ne ovat osa myös pelin fysiikkasimulaatioita. Fysiikkojen aktivoituessa ne käytännössä siis aktivoidaan pelihahmon luissa, eikä itse 3D-mallissa. Työnkulku tässä on siis, että ensin pelifysiikat simuloidaan tietyissä osissa pelihahmon luurankoa, jonka jälkeen luuranko viestii sille asetetulle 3D-mallille, miten mallin pitää muuttaa muotoa. (Epic Games q.) Peliprojektin luurangot ovat automaattisesti luotuja Adoben Mixamo palvelun avulla. Mixamo tarjoaa työkalun automaattisen luurangon luomiselle. Vaikka se ei ole yhtä hyvä kuin käsin luodut luurangot, se on pätevä pienille peliprojekteille, joilla ei ole paljoa työvoimaa tai aikaa.

Unreal Engine tarjoaa kehittäjille visuaalisen tavan luurangon muokkaamiselle luurankoeditori työkalulla. Editorin avulla voidaan muokata pelihahmoille asetettuja luurankoresursseja, jotka on joko luotu etukäteen toisilla ohjelmilla. Pelihahmon 3D-malliin on etukäteen yleensä luotu luuranko valmiiksi, ja sille generoidaan automaattisesti luurankoresurssi, kun kyseinen 3D-malli tuodaan Unreal Enginen sisälle. (Epic Games r.)

Kuvassa 11. on näkymä Unreal Enginen luurankoeditorista, jossa voidaan muokata luurangon eri ominaisuuksia.



Kuva 11. Luurankoeditori (The Last Drop 2022)

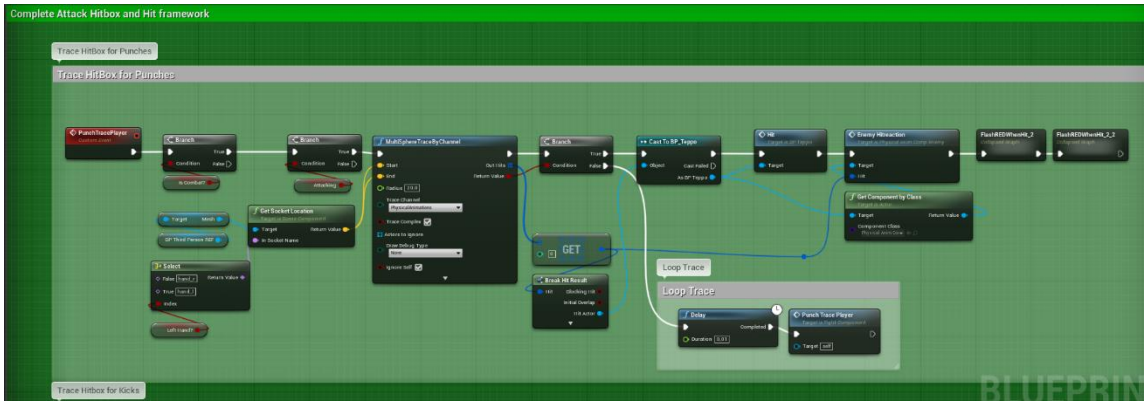
Luurankoeditorissa voidaan tehdä useitakin eri muokkauksia luurankoon, kuten luoda pistokohtia kiinni luurangon yksittäisiin luihin. Näitä pistokohtia voidaan käyttää kiinnittämään 3D-meshejä itse pelihahmon luihin. Näitä voidaan käyttää esimerkiksi aseiden kiinnittämiseen pelihahmon käteen. Kuvassa nro pelin hahmon käteen on liitetty olutpullo pistokohdan avulla. Yhtenä tärkeimmistä luurankoeditorin ominaisuuksista on myös luurankopuu. Luurankopuu on hierarkkinen luettelo pelihahmon kaikista luista. Sen avulla voidaan tarkastella luiden nimiä ja miten luut periytyvät toisilleen. Ylimpänä luuna on tässä tapauksessa pelvis, eli lantioluu. Se toimii myös niin sanotusti juuriluuna (Epic Games r). Sillä juuriluu on asetettuna pelihahmon lantioluuhun, root motion animaatioiden käyttö piti sulkea pois. Root motion animaatioilla tarkoitetaan pelianimaatioita, joissa pelihahmon jalkojen välissä maan tasalla asetettu juuriluu ohjaa hahmon liikettä animaation aikana. Näin voidaan luoda realistista liikettä esimerkiksi lyöntianimaatioissa, joissa pelihahmo ottaa myös askeleen eteenpäin. Edespäin liikkumista ei siis normaalisti tapahdu animaatioissa ilman root motion animaatioiden käyttöä, vaan se pitää muissa tapauksissa käsin ohjelmoida ominaisuudeksi animaatioiden tapahtuessa. Tämän rajoituksen takia valittiin taisteluun animaatiot, joissa pelihahmon jalat eivät liiku. Jos root motion animaatiot yritetään laukaista, kun pelihahmon luurangon juuriluuna toimii lantioluu, animaatiot hajoavat täysin käyttökelvottomiksi. Kuvassa nro on esimerkki pelihahmon animaatioiden hajoamisesta, kun root motion animaatiot on asetettu päälle väärän juuriluun kanssa. Tämän korjaaminen vaatisi suurempaa luurangon muokkaamista kolmannen osapuolen ohjelmilla, joten projektissa ei lopuksi käytetä root motion animaatioita niiden hyödyistä riippumatta. (Epic Games s.)

5.3 Blueprint osuus

Taistelujärjestelmän blueprint osuudessa ohjelmoidaan itse lyöntimekaniikat, hitboxien ilmestyminen, fysiikkasimulaation aktivoiminen ja myös fysiikkojen palautus takaisin lepotilaan. Lyönnit toimivat yhdistelmämäisesti, tarkoittaen sitä, että lyönneillä on kolme asetettua animaatiota toistumaan, jos lyöntipainiketta painetaan useasti lyhyen ajan sisällä. Jos pelaaja ei paina lyöntipainiketta uudestaan tai liian hitaasti, lyöntiyhdistelmä palautuu alkuun.

Hitboxien luomiselle on monia erilaisia tapoja, mutta halutun toiminnallisuuden saavuttamiseksi on oikeastaan vain yksi oikea tapa tehdä ne. Hitboxit voidaan käytännössä asettaa suoraan pelaajan ja vihollisen 3D-mallin päälle, mutta tämä tuo esille tiettyjä ongelmia. Ensinnäkin, sillä hahmojen 3D-mallia halutaan liikuttaa fysiikkasimulaatiolla, tulee siis myös lyönnin osua pelaajan malliin eikä näkymättömään kapselikomponenttiin. Syynä tähän on, että hahmon luuranko hoitaa fysiikkojen toimivuuden 3D-mallissa, ja jos lyönti rekisteröityy vain kapselikomponenttiin, se ei tunnista osuttua luuta, sillä pelaajan kapselikomponentilla ei ole omaa luurankoa. Täten hitboxien luonti itse malliin on turhaa. 3D-mallilla on omat

fysiikkatörmäyslaatikot luotuna itse malliin, joka toimii käytännössä samoin tässä ominaisuudessa kuin kapseli hitboxit. Erona on kuitenkin se, että koska törmäyslaatikot ovat suoraan kiinnitettynä itse 3D-malliin kiinni, ne voivat rekisteröidä osuttuja luurangon osia päällekkäistapahtuman tapahtuessa. Kuvassa 12. on blueprint koodi hitboxien luomiselle lyöntien aikana.



Kuva 12. Blueprint hitbox frameworkille

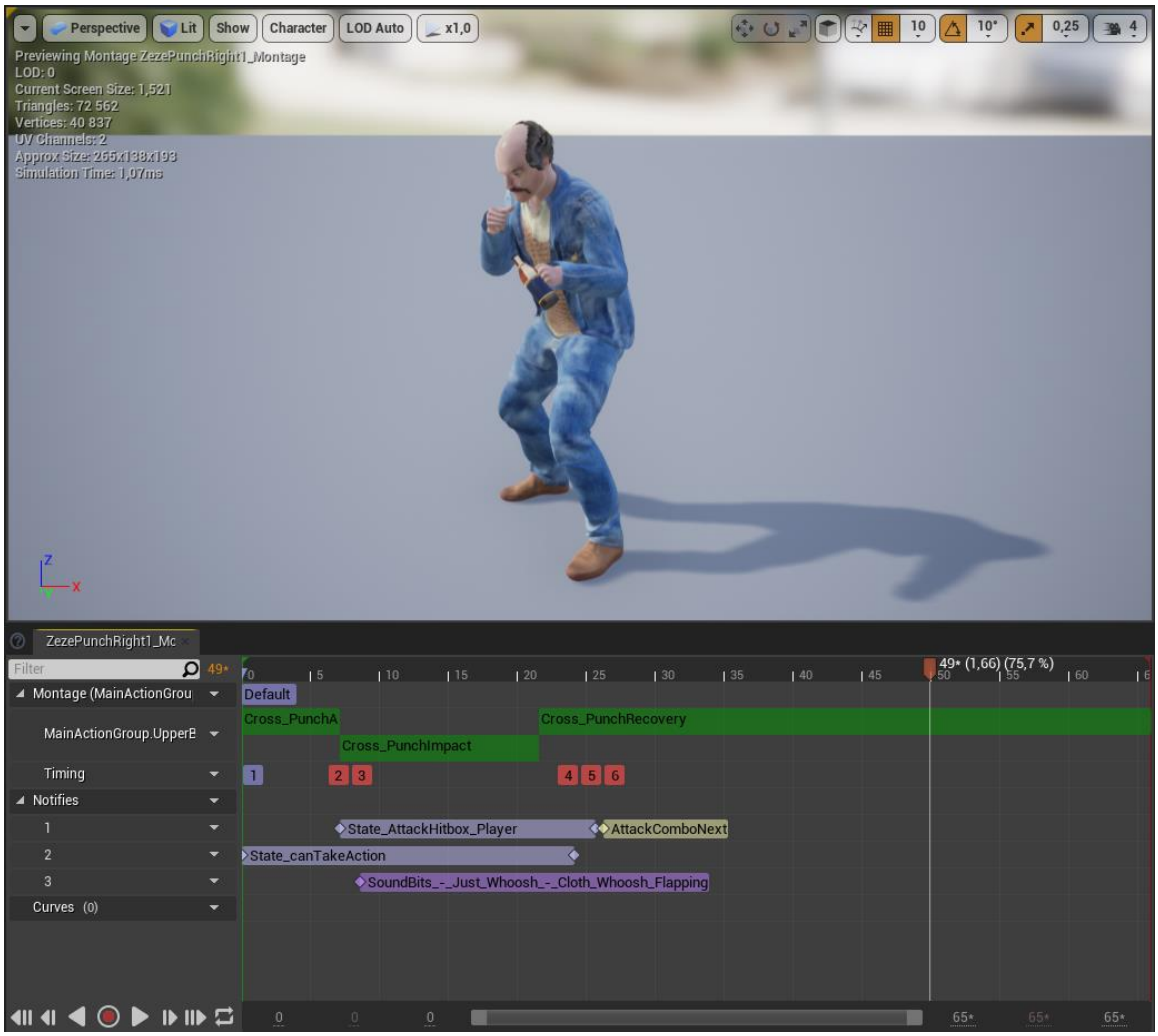
Sillä hitboxeja ei aseteta suoraan malliin, ne pitää luoda ajon aikana lyönnin tapahtuessa. Tähän toimii hyvin mukautetun tapahtuman luominen. MultiSphereByChannel solmulla voidaan synnyttää pallon muotoinen törmäyslaatikko, joka toimii tässä tapauksessa lyönnin hitboxina. Solmulla on monia eri asetuksia pallon luomiseen, kuten pallon säde, säiekanava ja törmäyslaatikon aloitus ja lopetus paikat. Koska kyseessä on lyönti törmäyslaatikko, pallon halutaan syntyvän pelaajahahmon nyrkin kohdalle. Tätä varten voidaan käyttää Select -solmua, jolla on epätosi ja tosi tapahtumat. Boolean ”Left Hand?” muuttujaa käyttämällä voidaan määrittää, kumpaan käteen törmäyspallo syntyy lyönnin tapahtuessa. Tämä boolean on asetettuna todeksi vasemman käden lyöntien kehyksessä ja epätodeksi oikean käden lyönneissä.

Toisena tärkeänä asetuksena on myös radius, jolla voidaan määrittää synnytetyn pallo törmäysalueen säteen koko. Tällaisessa taistelumekaniikassa on hyvä asettaa hitboxin koko jonkin verran suuremmaksi kuin pelaajahahmon oman nyrkin koko oikeasti on, sillä pelaajalle halutaan luoda pieni etulyöntiasema. Myös toisinpäin vastustajan hitbox asetetaan pelimallin nyrkkiä isommaksi, mutta ei yhtä suureksi kuin mikä pelaaja hahmolla on. Syynä ei ole pelkästään pienen etulyönnin luominen, vaan myös taistelu mekaniikkojen tuntumisen parantaminen. Jos hitboxit ovat liian pienet pelaajalla, voi pelaajalle tulle tuntemus, että lyönnin olisi pitänyt osua, vaikka se meni ohitse ja myös toisinpäin, että vastustajan lyönnin olisi pitänyt mennä ohitse, vaikka se oikeasti osui. Hitboxit pitää myös pystyä synnyttämään juuri oikeaan aikaan, ettei lyönnit osu liian aikaisin tai liian myöhään lyöntianimaation

aikana. Normaalisti aktivoituna hitbox synnytettäisiin heti lyöntianimaation alkaessa, ja se katoaisi animaation päättyessä. Tähän ratkaisuksi löytyy animation state, eli animaatiotila.

Lyöntianimaatioina käytetään Adoben verkkosivulta nimeltä Mixamo haettuja animaatioita. Mixamo tarjoaa rojaltivapaita animaatioita, joita saa vapaasti käyttää yksityisissä kuin myös kaupallisissa projekteissa ilman attribuutiota, joten sivusto sopii hyvin opinnäytetyön tarkoituksiin (Adobe 2021).

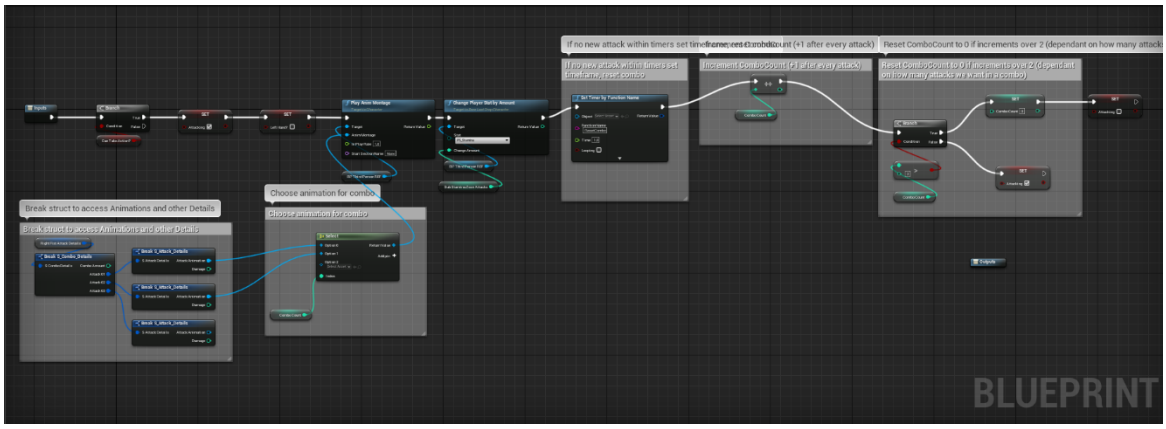
Pelihahmojen lyöntianimaatiot on jaettu kolmeen eri osaan. Nämä ovat odotus, isku ja palautuminen. Nämä kolme osaa kuvastavat lyöntianimaation eri vaiheita, ja niiden jakaminen näin helpottaa kehittäjän työtä synnyttää hitbox oikeaan aikaan ja myös parantaa animaatioiden muokattavuutta, sillä niiden joitain osia voidaan tarvita nopeuttaa tai hidastaa pelattavuuden tuntuvuuden hyväksi. Esimerkiksi animaation alku, jossa pelihahmo tuo kättään taaksepäin aloittaakseen lyönnin, eli odotus, asetetaan tapahtumaan hieman hitaammin kuin normaali animaation toistonopeus. Animaation keskikohta, eli isku on kohta, jolloin lyönnin tulisi osua vastustajaan. Tämä asetetaan toistumaan huomattavasti nopeammin kuin normaali animaation toistonopeus. Tähän syynä on se, että lyönnin iskun halutaan tapahtuvan nopeasti, ettei lyönti vaikuta liian hitaalta. Tähän osioon asetetaan myös animaatiotila, jossa laukaistaan hitbox koodi. Animaatiotila laukaistaan, kun animaation toisto pääsee tiettyyn kohtaa animaatiota ja se päätetään, kun tämä kohta on toistettu kerran. Näin saadaan hitbox synnytettyä juuri oikeaan aikaan lyönnin kohdalla. Kolmantena animaation osuutena on palautuminen. Tämän on animaation kohta, jossa pelaaja palautuu lyönnistä takaisin perustilaan. Palautumisen osuus on asetettu toistumaan hitaammin kuin normaalisti, sillä jos pelaaja ei osu lyöntiään vastustajaan, siitä halutaan rankaista pelaaja hieman hitaammalla palautumisanimaatiolla. Kuvassa 13. on Unreal Enginen animaatioeditori, jossa pelaajahahmon lyönnit ovat jaettuna kolmeen osaan, ja animaatiotila hitboxien luomiselle on asetettuna keskelle animaatioita.



Kuva 13. Animaatioeditori (The Last Drop 2022)

Tässä editorissa on myös animaatiotila pelaajan liikkumisen estämiselle animaation aikana. Sillä animaatiossa hahmon jalat eivät liiku, olisi luonnottoman näköistä, jos pelaaja liikkuisi lyöntianimaation aikana. Tällä myös estetään muidenkin peliominaisuuksien laukaiseminen lyöntianimaation aikana. AttackComboNext on Animation notify, eli animaatioilmoitus. Toisin kuin animaatiotila, se voidaan laukaista vain kerran. Sillä laukaustaan koodia, jolla mahdollistetaan, että missä kohtaa animaatiota voidaan siirtyä lyöntien seuraavaan lyönnin yhdistelmään (Epic Games t).

Yhtenä tärkeänä syynä miksi MultiSphereByChannel solmua käytetään tässä, on sen säekanava asetus kuin myös Out hits asetus. Säekanavan asetuksen avulla voidaan määrittää törmäyskanava, joka vaikuttaa siihen mihin synnytetty törmäyspallo voi osua. Unreal Enginen projektiasetuksissa voidaan luoda mukautettuja säekanavia. Lyöntejä varten on luotu PhysicalAnimations niminen säekanava, jota käytetään pelkästään lyöntien rekisteröimisessä. Tätä varten pitää siis asettaa hahmot tekemään päällekkäisyystapahtumat tämän säekanavan kautta. Kuvassa 14. on blueprint koodi lyöntien yhdistelmien luomiselle.

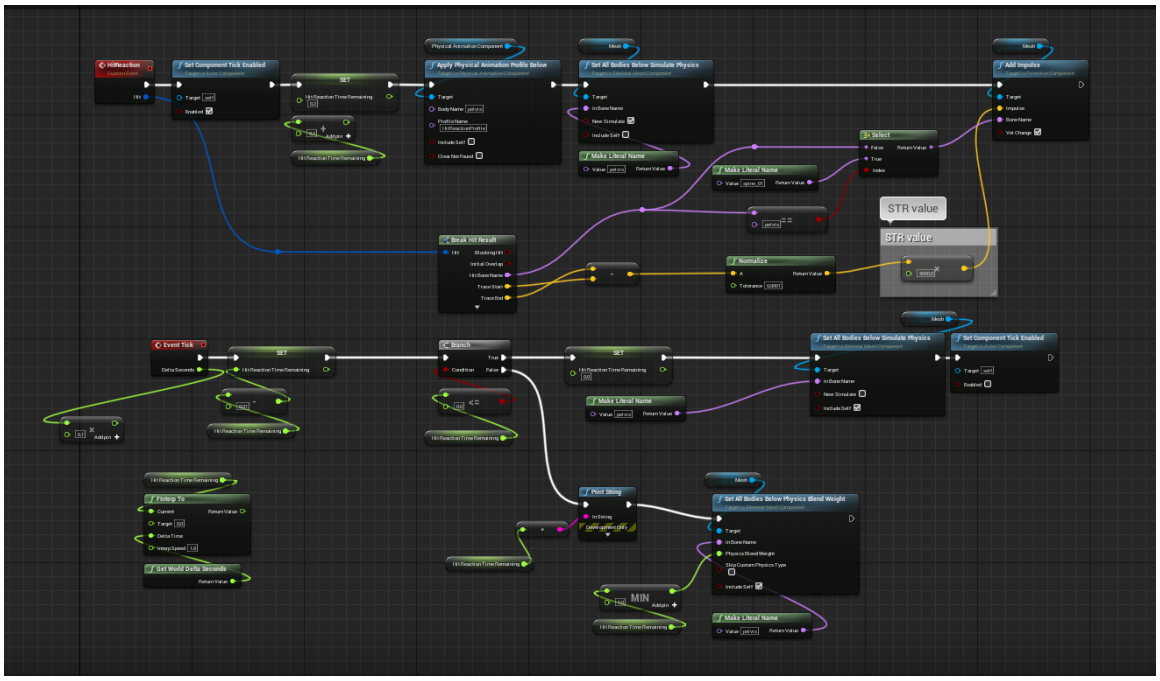


Kuva 14. Lyöntien yhdistelmien kehys

Pelaajan lyöntianimaatiot ovat tallennettuna struktuuriin nimeltä S_Combo_Details, josta niitä voidaan kutsua muuttujan avulla. Selector -solmulla voidaan valita halutut animaatiot struktuurista, joita halutaan käyttää lyönneissä. Lyöntiyhdistelmien määrä määritellään kokonaisluku -muuttujilla ja lyöntiyhdistelmien resetointi yksinkertaisesti Branch -solmulla, joka toimii samalla tavalla kuin if-lause normaalissa ohjelmoinnissa (Epic Games u). Sille siis annetaan ensin ehto, joka joko täyttyy tai ei täyty. Tässä tapauksessa ehtona on, että onko kokonaislukumuuttuja ComboCounter suurempi kuin kaksi. Jos ehto on tosi, ComboCounter asetetaan olemaan nolla, eli lyöntiyhdistelmät resetoidaan ja hyökkääminen keskeytetään. Jos ehto on epätosi hyökkäyksiä, voidaan jatkaa, kunnes ehdon vaatimukset täyttyvät. Lyöntiyhdistelmät resetoidaan nollaan myös, jos pelaaja ei tee uutta hyökkäystä pienen ajan sisällä. ComboCounterin ollessa nolla lyöntiyhdistelmä hyökkäykset aloitetaan alusta.

5.3.1 Fysiikka osuus

Fysiikat aktivoidaan myös blueprinttien avulla. Osumisreaktion halutaan vaikuttavan vain hahmojen ylävartaloon, tulee todeta, että fysiikat saavat vaikuttaa vain lantioluusta ylöspäin. Tämä kuitenkin aiheuttaa myös ongelmia kyseisissä hahmomalleissa, sillä lantioluu toimii myös hahmojen juuriluuna. Tämä tarkoittaa käytännössä sitä, että jos lyöntiosuma osuu lantioluuhun luuhun, hahmon jalat nousevat ilmaan ja hahmo näyttää leijuvan paikoillaan. Tämä ei tietenkään ole realistista, joten se pitää korjata. Siihen onneksi löytyy yksinkertainen korjaus, kun Select -solmun avulla voidaan määrittää tosi ja epätosi tapahtumat. Näin voidaan asettaa, että jos osuttu luu on lantioluu, osutaan sittenkin spine_01 nimiseen luuhun, joka vastaa lantioluun yllä olevaa selkärankaluuta. On myös tarpeellista lopettaa fysiikka simulaatio ja palauttaa hahmon malli alkuperäiseen asentoon. Tämä tehdään muuttujien avulla. Kuvassa 15. on blueprint koodi lyöntireaktioiden fysiikkojen aktivoimisesta ja simuloimisesta.

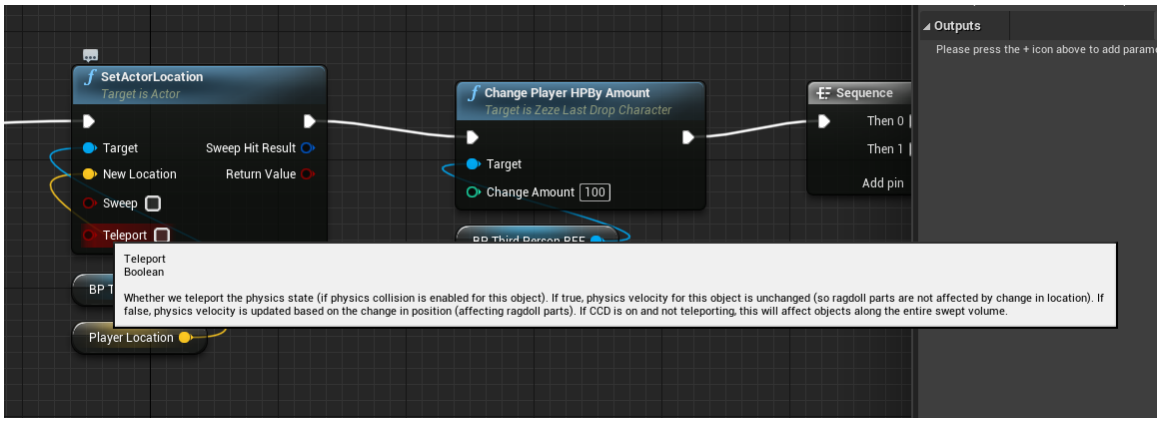


Kuva 15. Blueprint fysiikkakoodista

Yhtenä ongelmana fysiikkojen kanssa ilmestyy tapa, miten pelaaja hahmo siirretään avoimesta maailmasta taisteluareenalle. Tapa miten pelaaja siirretään, on käytännössä pelaajahahmon position koordinaattien muuttaminen ajon aikana. Tämä tarkoittaa käytännössä sitä, että hahmoa liikutetaan ajon aikana hyvin nopeasti paikasta A paikkaan B. Taistelun loppuessa pelaaja hahmo liikutetaan samalla tavalla takaisin avoimeen maailmaan, joka aiheuttaa valtavan liikkeen takia fysiikkasimulaatiossa häiriön. Tämä häiriö ilmestyy pelaajahahmon ylävartalon venymisenä ja valtavana nykimisenä areenan vaihdon tapahtuessa. Toimeksiantajan toivomuksesta pelaaja hahmoa ei saa tuhota kesken pelin ajon, joten sen uudelleen synnyttäminen taisteluareenalle ei ole tähän tilanteeseen sopiva ratkaisu. Tähän kuitenkin löytyi nopea ja yksinkertainen ratkaisu itse solmussa, jota käytetään pelaaja hahmon siirtämiseen maailmassa. Solmu, jota käytetään siirtämään pelaaja hahmoa, on SetActorLocation ja siitä löytyy asetus teleporttaamiselle, joka muuttaa solmun toimivuutta sillä tavalla, että kun asetus on asetettu todeksi pelaajan fysiikan tilaa ei muuteta teleportin aikana. Tämä korjaa fysiikkasimulaatiossa tapahtuvan venymisen ja nykimisen teleportin tapahtuessa. Apply Physical Animation Profile Below -solmulla asetetaan koodin ajamisen ajan voimassa oleva fysiikka animaatio profiili, joka käynnistyy koodin lauetessa. Siinä myös nimetään luu, josta ylöspäin halutaan simuloida fysiikkoja, joka on tässä tapauksessa lantioluu. SetAllBodiesSimulatePhysics -solmulla aktivoidaan fysiikka simulaatio laittamalla ”New simulate” asetus todeksi. AddImpulse -solmulla luodaan impulssi osuttuun kehonosaan, joka fysiikka simulaation aikana heilauttaa pelihahmoa impulssin vastakkaiseen

suuntaan. Impulssin arvon tarvitsee olla hyvinkin suuri, jotta saadaan aikaan realistisen oloinen reaktio lyönteihin.

EventTick -tapahtumassa käsitellään float -muuttujilla pelihahmojen palautuminen takaisin normaaliin tilaan impulssin jälkeen. Fysiikka simulaatio pitää siis palauttaa takaisin normaaliin tilaan ja sitten asettaa pois päältä, jotta se ei vaikuta muihin animaatioihin. Normaalisti tämän tapahtuman käyttäminen on huonoa käytäntöä, sillä se on tapahtuma, joka laukaistaan jokainen kuva sekunti (Epic Games v). Tämä tarkoittaa sitä, että jos sitä käytetään paljon raskaissa mekaniikoissa, se voi aiheuttaa suorituskykyongelmia ja täten huonontaa pelin pelattavuutta. Tämä vaikuttaa myös suoraan pelin fysiikoihin, sillä Unreal Enginen fysiikat ovat suoraan sidonnaisia pelin suorituskykyyn ja pelin kuvataajuuteen. Alhainen kuvataajuus aiheuttaa täten häiriöitä pelin fysiikoissa, jotka ilmenevät esimerkiksi fysiikkasimulaation aikana pelihahmon simuloitujen alueiden nykimisenä, ja fysiikkojen palautuminen on hyvinkin hidasta ja venynyttä. Näiden ongelmien takia EventTickiä käyttäessä se pitää aktivoida ja deaktivoida erikseen. Solmulla SetComponentTickEnabled voidaan asettaa tapahtumat laukaisemaan koodia tai se voidaan pysäyttää. EventTick laukeaa tässä tilanteessa, kun pelihahmon lyönti rekisteröityy osuneeksi vastustajaan. Koodi laukaisee fysiikka simulaation käyntiin ja sen jälkeen luo impulssin osuttuun osaan vartaloa, joka on lantioluuta ylempänä. EventTickillä jatkuvasti tarkkaillaan hahmon fysiikoita ja se suljetaan, kun hahmo on palannut takaisin oletusasentoon, jolloin myös suljetaan fysiikka simulaatio. Kuvassa 16. blueprint koodi pelaajahahmon siirtämiselle avoimesta pelimaailmasta taisteluareenalle.



Kuva 16. SetActorLocation -solmu siirtää pelaajaa pelimaailman sisällä

Ongelmana fysiikka materiaalien luonnissa oli, että pelaajan sisällä olevat törmäyskuplat törmäsivät toisiinsa fysiikka simulaation tapahtuessa. Tämä on normaalisti haluttua käytöstä, mutta kun kaksi vierekkäistä törmäyskuplaa, jotka ovat hieman toistensa sisällä luovat törmäystapahtuman alkaa tapahtumaan rajuja ongelmia fysiikka simulaatiossa. Nämä häiriöt ilmestyvät 3D-mallin rajuna venymisenä ja rajuna nykimisenä. Tähän ongelmaan on

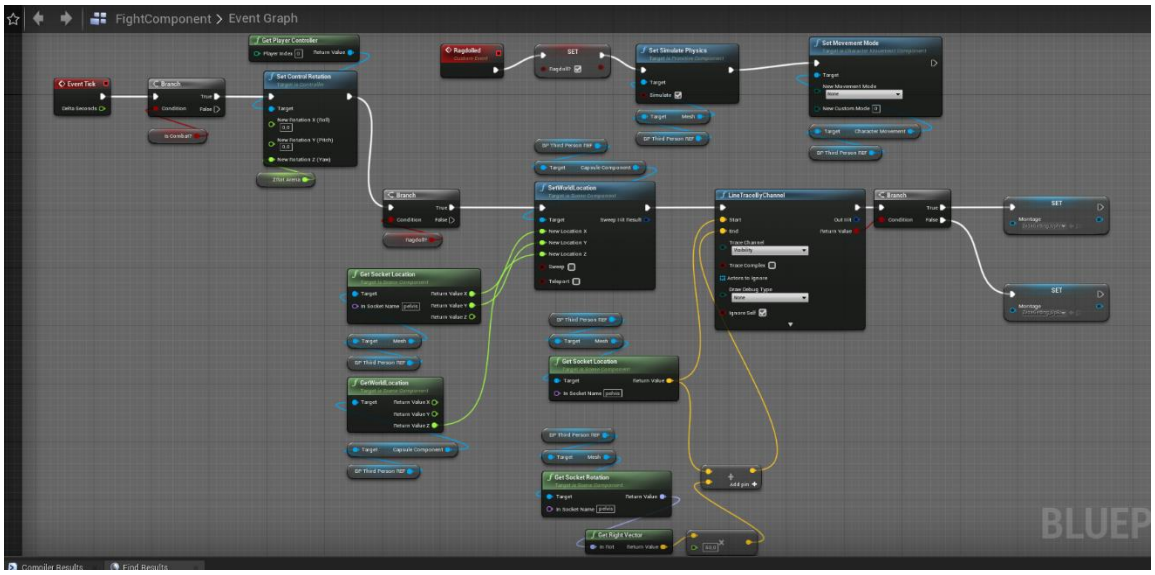
onneksi yksinkertainen ratkaisu. Hahmon sisäiset törmäyskuplat pitää asettaa niin, että vierekkäiset kuplat, jotka ovat vähänkään päällekkäin eivät luo törmäystapahtumia keskenään. Niiden kuitenkin pitää pystyä luomaan törmäystapahtumat muiden kuplien kanssa, muuten hahmon 3D-mallin osat läpäisevät itsensä fysiikka simulaation aikana. Toisena tärkeänä asiana on rajoitukset kuplien välillä. Törmäyskuplat on kiinnitetty toisiinsa rajoituksilla, jotka ohjaavat sitä, miten hahmon eri osat voivat liikkua. Näillä rajoituksilla voidaan rajoittaa esimerkiksi kyynärpään, kaulan tai vaikka lantion liikettä pelihahmoissa. Nämä ovat realistisuuden takia tärkeää asettaa oikein, sillä muuten simulaation tapahtuessa pelihahmojen raajat voivat taipua luonnottomiin suuntiin (Epic Games i). Näiden asetusten jälkeen hahmon fysiikka simulaatio toimii halutulla tavalla.

5.3.2 Ragdoll kaatuminen

Toisena vaatimuksena toimeksiantajalta oli ragdoll mekaniikka taisteluun. Tämä mekaniikka käytännössä rankaisee pelaajaa siitä, jos hän ottaa liian paljon vahinkoa tai ei väistä tai torju vastustajan lyöntejä. Tämä mekaniikka toimii sillä tavalla, että kun pelaaja ottaa tietyn määrän vahinkoa, niin pelaajahahmo kaatuu fysiikka simulaation avulla maahan. Maassa pelaaja on puolustuskyvytön ja joutuu odottamaan muutaman sekunnin ennen kuin pelaaja hahmo nousee takaisin ylös. Molempien ragdoll fysiikka simulaation kuin myös siitä palautumisen ja ylös nousemisen tulee olla sulava ja realistisen näköinen, ettei pelaajan immersio katkea pelin aikana. Ragdollilla pelikehityksessä tarkoitetaan fysiikkasimulaatiota, jossa pelihahmon kehossa aktivoidaan fysiikkasimulaatio, joka johtaa hahmon kehon menemään veltoksi simulaation alueilla (Epic Games f). Tämä on useissa peleissä suosittu tapa saada pelihahmot kaatumaan tiettyjen tilanteiden tapahtuessa, kuten pelaajan ottaessa suuren iskun itseensä tai pelaajan kuollessa. Fysiikkasimulaatio myös tässä tilanteessa toimii pätevänä korvikkeena käsin tehdyille animaatioille, joiden tekeminen veisi suuren määrän kehitysaikaa. Täten fysiikkasimulaatiota käytetään tässäkin tapauksessa animaatioiden sijaan niin ajan säästämiseksi kuin myös pelimekaniikkojen vaatimusten kannalta (Tyler 2022 b).

Jotta voidaan huomata koska pelaajahahmo on kaatuneena maahan pitää käyttää Event-Tick -solmua. Itse ragdollille on asetettu oma mukautettu tapahtuma, jossa yksinkertaisesti aktivoidaan fysiikka simulaatio pelihahmon koko kehossa. Tämä tapahtuma kutsutaan myöhemmin lyönnin tapahtuessa, ja aktivoidaan jos lyönnin aiheuttama vahinko on tarpeeksi suuri. Jotta ragdollista voidaan nousta ylös, pitää tietää miten päin pelihahmo on kaatunut, jotta voidaan asettaa luonnollisen mukainen ylös nouseminen animaatio pelihahmolle. Tämän mekaniikan saavuttamiseksi käytetään LineTraceByChannel -solmua. Tämä solmu ampuu näkymättömän säteen laskettuun suuntaan ja tunnistaa siihen osuvan geometrian.

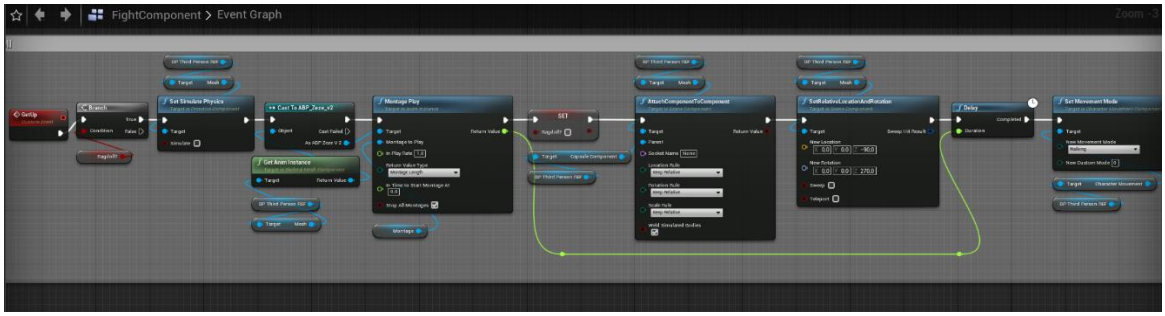
Pelihahmosta ammutaan kaksi sädettä, joista toinen ammutaan hahmon pelaajahahmon edestä ja toinen pelaajahahmon takaa ja ne laukaistaan hahmon lantioluun korkeudelta. Näillä säteillä pystytään huomaamaan miten päin pelihahmo on kaatuneena ragdollissa jonka jälkeen voidaan laukaista sopiva luonnollisen oloinen animaatio ylös nousemiselle (Epic Games x). Nämä animaatiot ovat muuttujia, joita kutsutaan myöhemmin toisessa mukautetussa tapahtumassa. Yhtenä ongelmana ilmeni myös se, että animaatiot työnsivät pelihahmon pelimaailman alle. Tämä korjataan käyttämällä SetWorldLocation -solmua, jolla siirretään pelihahmon kapseli komponenttia ylöspäin pelimaailmassa. Oikea sijainti johon pelihahmo pitää siirtää löydetään siirtämällä hahmoa sen lantioluun perusteella. Pelihahmon liikkuminen pitää myös ottaa pois käytöstä ragdollin aikana, sillä muuten hahmo pystyisi liikkumaan maata pitkin epäluonnollisen mukaisesti. Kuvassa nro on Blueprint -solmujen koodi EvenTick koodille, joka hoitaa pelihahmon ylös nousemisen. Kuvassa 17. on myös mukautettu tapahtuma ”Ragdolled”, joka hoitaa pelihahmon fysiikka simulaation aktivoimisen ragdollia varten.



Kuva 17. EventTick toiminta ragdollille ja siitä ylös nousemisen animaatioille

Tähän luotiin myös toinen mukautettu tapahtuma ylös nousemisen palauttamiselle. Kun pelihahmo nousee ylös ragdollista fysiikka simulaatio pitää ottaa pois käytöstä. Aiemmin luotuja animaatiomuuttujia kutsutaan toistumaan, jonka jälkeen pelihahmon kapselikomponentti liitetään takaisin kiinni pelihahmo 3D-malliin. Tässä on syynä se, että kapselikomponentti on kiinni pelihahmon juuriluussa, joka tässä tapauksessa on lantioluu. Hahmon kaatuessa ragdolliin hahmon kapselikomponentti putoaa osittain pelimaailman läpi, ja se aiheuttaa ongelmia ragdollista ylös nousemisessa. Kapseli komponentin uudelleen liittäminen pelihahmoon korjaa tämän, ja animaatiot toistuvat halutulla tavalla pelimaailman maassa.

Pitää myös aktivoida pelihahmon liikkuminen takaisin päälle pelihahmon noustua ylös. Kuvassa 18. on mukautettu tapahtuma ragdollista ylösnousemisen viimeistelylle.



Kuva 18. Blueprint solmut ylös nousemiselle ragdollista

Ongelmana ilmestyy myös se, että pelihahmon kaatuessa ragdolliin hahmon yläkeho ei simuloi fysiikkoja oikein. Hahmon yläkeho pysyy samassa animaatiossa kuin fysiikka simulaation alkaessa, eikä putoa veltoksi, miten sen kuuluisi. Pelaaja voi myös vielä ragdollissa käyttää hyökkäyksiä ja torjua iskuja, joka ei ole haluttua toimintaa. Tähän löytyi yksinkertainen ratkaisu, jossa boolean -tyyppisellä muuttujalla voidaan estää nämä toiminnot ragdollin ollessa aktiivisena. (Epic Games s.) Kuvassa 19. pelihahmo on nousemassa ylös ragdoll tilasta taistelun aikana.

Kuva 19. Vasemmalla pelihahmo nousemassa ylös ragdollista (The Last Drop 2022)

Ragdollin jäykkään käyttäytymiseen ei löytynyt ratkaisua. Tähän voi olla syynä se, että peliprojektissa työskentelee useampi ohjelmoija samanaikaisesti, ja muutoksia pelihahmojen fysiikoihin on voitu tehdä jonkun toisen kehittäjän toimesta, sillä pelifysiikoita käytetään muuallakin pelissä. Halutut pelimekaniikat taistelussa kuitenkin ovat toiminnassa.

Pelihahmo kaatuu ragdollina maahan, kun vastustajan lyönti tekee tarpeeksi vahinkoa, ja ragdollista pääsee nousemaan ylös sujuvasti.

6 Yhteenveto ja pohdinta

Opinnäytetyön tarkoituksena oli tutkia Unreal Engine -pelimoottorin tarjoamia pelifysiikka-ominaisuuksia ja implementoida niitä pelimekaniikkoihin optimaalisimmilla tavoilla. Tähän prosessiin sisältyi yleisesti Unreal Enginen fysiikkamoottorin ominaisuuksien ja sen tarjoaminen työkalujen tutkiminen kuin myös käytännössä taistelumekaniikkojen luonti videopeliin, jossa käytetään pelifysiikkoja hyödyksi. Katsoen tuloksia taistelumekaniikoista ja niihin liitettyjen pelifysiikkojen toimivuudesta voidaan sanoa, että on suuriltaosin löydetty hyvinkin optimaalisia tapoja luoda fysiikkasimulaatiolla reaktioita pelin taisteluun. Nämä fysiikkapohjaiset reaktiot näyttävät luonnollisimmilta kuin normaalit animaatiot ja ne aktivoituvat myös järkevästi pelin ajon aikana ilman että ne vaikuttavat liian suuresti pelin kokonaiseen suorituskykyyn. Parannettavaa kyllä olisi vielä pelin ragdoll -mekaniikoissa, sillä ne toimivat vielä liian jäykästi. Myös pelin taistelussa käytetyt animaatiot ovat kömpelöitä, mikä johti siihen, että ne piti jakaa kolmeen eri osaan, jotta niistä saataisiin paremman tuntuiset ja reagoivammat. Myös itse pelimekaniikkojen käytännöllisyys on vielä kyseenalaistettavaa, sillä peliä ei olla vielä pelitestattu paljoa.

Yleisesti videopelifysiikat ovat suuressa suosiossa pelikehityksessä ja niitä käytetään useissa videopeleissä edes jonkin verran ja samankaltaisia pelifysiikoita käytetään monissakin videopeleissä. Myös Unreal Engine on hyvin suosittu pelimoottori pelinkehityksessä, sillä sen tarjoamat työkalut vähentävät pelikehittäjien työmäärää huomattavasti. Niin Blueprint -ohjelmoiminen kuin myös Unreal Enginen monet editorit, kuten fysiikkaeditori helpottavat työnkulkua, sillä kaikkea ei tarvitse ohjelmoida alusta asti. Näiden työkalujen ja itse Unreal Engine -pelimoottorin ymmärtäminen on pelinkehitysalalla hyvinkin tärkeää. Jatkokehityksenä pelin ragdollista voisi tehdä realistisemman ja myös siitä takaisin ylös nousemisesta voisi tehdä vieläkin sulavamman. Myös pelin taistelumekaniikoita voisi testata enemmän, jotta voidaan selvittää mitkä fysiikkamekaniikat sopivat peliin ja mitkä eivät sovi.

Lähteet

Adobe. 2021. Mixamo Common questions. Viitattu 13.11.2022. Saatavissa

https://helpx.adobe.com/creative-cloud/faq/mixamo-faq.html

Epic Games. a. Unreal Engine commercial game deployment guidelines. Viitattu 05.10.2022. Saatavissa https://www.unrealengine.com/en-US/release

Epic Games. b. Unreal Engine 5.0 Release Notes. Viitattu 11.10.2022. Saatavissa https://docs.unrealengine.com/5.0/en-US/unreal-engine-5.0-release-notes/

Epic Games. c. Unreal Engine 5 Documentation. Physical Materials. Viitattu 31.10.2022. Saatavissa https://docs.unrealengine.com/5.0/en-US/physical-materials-in-unreal-engine/

Epic Games. d. Unreal Engine 5 Documentation. Physics Bodies. Viitattu 23.11.2022. Saatavissa https://docs.unrealengine.com/5.1/en-US/physics-bodies-in-unreal-engine/

Epic Games. e. Unreal Engine 5 Documentation. Destruction Overview. Viitattu 20.11.2022. Saatavissa https://docs.unrealengine.com/5.1/en-US/destruction-overview/

Epic Games. f. Unreal Engine 5 Documentation. Physics-based animation. Viitattu 21.11.2022. Saatavissa https://docs.unrealengine.com/5.1/en-US/physics-driven-animation-in-unreal-engine/

Epic Games. g. Unreal Engine 5 Documentation. Collision Overview. Viitattu 26.10.2022. Saatavissa https://docs.unrealengine.com/5.0/en-US/collision-in-unreal-engine---overview/

Epic Games. h. Unreal Engine 5 Documentation. Physics Asset Editor Interface. Viitattu 4.11.2022. Saatavissa https://docs.unrealengine.com/5.0/en-US/physics-asset-editor-interface-in-unreal-engine/

Epic Games. i. Unreal Engine 5 Documentation. Creating a Physics Constraint Profile. Viitattu 4.11.2022. Saatavissa https://docs.unrealengine.com/5.0/en-US/creating-a-physics-constraint-profile-in-unreal-engine/

Epic Games. j. Unreal Engine 5 Documentation. Physical Materials Reference. Viitattu 20.11.2022. Saatavissa https://docs.unrealengine.com/5.0/en-US/physical-materials-reference-for-unreal-engine/

Epic Games. k. Unreal Engine 5 Documentation. Introduction to Blueprints. Viitattu 20.10.2022. Saatavissa https://docs.unrealengine.com/5.0/en-US/introduction-to-blueprints-visual-scripting-in-unreal-engine/

Epic Games. l. Unreal Engine 4 Documentation. Introduction to Blueprints. Viitattu 23.11.2022. Saatavissa https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/GettingStarted/

Epic Games. m. Unreal Engine 5 Documentation. Level Blueprint. Viitattu 22.10.2022. Saatavissa https://docs.unrealengine.com/5.0/en-US/level-blueprint-in-unreal-engine/

Epic Games. n. Unreal Engine 4 Documentation. Blueprint Class. Viitattu 1.11.2022. Saatavissa https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Types/ClassBlueprint/

Epic Games. o. Unreal Engine 5 Documentation. Components. Viitattu 26.10.2022. Saatavissa https://docs.unrealengine.com/5.0/en-US/components-in-unreal-engine/

Epic Games. p. Unreal Engine 5 Documentation. Math Expression Node. Viitattu 21.11.2022. Saatavissa https://docs.unrealengine.com/5.1/en-US/math-expression-node-in-unreal-engine/

Epic Games. q. Unreal Engine 5 Documentation. Skeletons. Viitattu 19.11.2022. Saatavissa https://docs.unrealengine.com/5.0/en-US/skeletons-in-unreal-engine/

Epic Games. r. Unreal Engine 5 Documentation. Skeleton Editor. Viitattu 19.11.2022. Saatavissa https://docs.unrealengine.com/5.0/en-US/skeleton-editor-in-unreal-engine/

Epic Games. s. Unreal Engine 5 Documentation. Root Motion. Viitattu 19.11.2022. Saatavissa https://docs.unrealengine.com/5.1/en-US/root-motion-in-unreal-engine/

Epic Games. t. Unreal Engine 5 Documentation. Animation Notifies. Viitattu 21.11.2022. Saatavissa https://docs.unrealengine.com/5.1/en-US/animation-notifies-in-unreal-engine/

Epic Games. u. Unreal Engine 5 Documentation. Blueprint Variables. Viitattu 21.11.2022. Saatavissa https://docs.unrealengine.com/5.1/en-US/blueprint-variables-in-unreal-engine/

Epic Games. v. Unreal Engine 5 Documentation. Event Tick. Viitattu 21.11.2022. Saatavissa https://docs.unrealengine.com/5.1/en-US/BlueprintAPI/AddEvent/EventTick/

Epic Games. x. Unreal Engine 5 Documentation. Traces Overview. Viitattu 21.11.2022. Saatavissa https://docs.unrealengine.com/5.1/en-US/traces-in-unreal-engine---overview/

Farris, J. 2020. Forging new paths for filmmakers on “The Mandalorian”. Epic Games. Viitattu 09.10.2022. Saatavissa https://www.unrealengine.com/en-US/blog/forging-new-paths-for-filmmakers-on-the-mandalorian

Ivyer, S. 2021. Core mechanics of fighting games. Dev. Viitattu 21.11.2022. Saatavissa

https://dev.to/goldenxp/core-mechanics-of-fighting-games-24ej

Nvidia Developer. PhysX. Viitattu 12.10.2022. Saatavissa https://developer.nvidia.com/physx-sdk

The Last Drop. 2022. Pixtell Oy. Julkaisematon aineisto. Viitattu 23.11.2022

Tyler, D. a. 2022. How to choose the best video game engine. GameDesigning. Viitattu 21.11.2022. Saatavissa https://www.gamedesigning.org/career/video-game-engines/

Tyler, D. b. 2022. The Role of Physics in Video Game Design. GameDesigning. Viitattu 21.11.2022. Saatavissa https://www.gamedesigning.org/learn/game-physics/