Karelia-ammattikorkeakoulu
Tradenomi, tietojenkäsittely (AMK)

Unreal Engine 4 ja Gameplay Ability System

Tommi Kekomäki

Opinnäytetyö, 09/2022

www.karelia.fi

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Syyskuu 2022
Tietojenkäsittely

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Tekijä(t)
Tommi Kekomäki

Nimike
Unreal Engine 4 ja Gameplay Ability System

Opinnäytetyön aiheena on roolipelikykyjen ja yleisien roolipelielementtien luominen peliprojektissa. Tavoitteena on luoda jaettava peliprojekti, jossa esitellään kykyjen luomista roolipelissä, huomioiden myös moninpelimahdollisuudet. Työn tavoitteena on lisäksi projektin vaiheiden dokumentointi ohjeiksi.

Projekti toteutettiin Unreal Engine 4 -pelimoottorilla ja kykyjen luomiseen käytettiin Gameplay Ability System -liitännäistä. Projektin kaikki vaiheet on dokumentoitu GitHub-palveluun.

Opinnäytetyön tuloksena syntyi ladattava demoprojekti ja sen dokumentaatio GitHub-palvelussa. Demo-projektista on tehty myös toinen versio, jota esitellään YouTube-videolla.

Kieli
suomi

Sivuja 27
Liitteet 2
Liitesivumäärä 2

Asiasanat
Unreal Engine, Gameplay Ability System, roolipeli

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
September 2022
Bachelor of Business Administration Programme in Karelia

Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600 (switchboard)

Author (s)
Tommi Kekomäki

Title
Unreal Engine 4 and Gameplay Ability System

The topic of the thesis is the creation of role-playing abilities and general role-playing elements on a game project. The goal is to create a downloadable game project that showcases and displays the creation of abilities in a role-playing game, also keeping multiplayer possibilities in mind. The process will be documented as a guide for the project creation.

The project is implemented by using Unreal Engine 4 and Gameplay ability system plugin for abilities. All phases of the project are documented on GitHub.

As a result, there is a downloadable demo-project and a documentation for all the steps included in creation of the demo. In addition, there is another project created from the demo, which is shown on a YouTube-video.

Language
Finnish

Pages 27
Appendices 2
Pages of Appendices 2

Keywords
Unreal Engine, Gameplay Ability System, Role-playing game

Sisältö

1 Johdanto

Karamian (2018) mukaan roolipelit saivat alkunsa lautapeleistä. Lautapeliversioista roolipelit siirtyivät tietokoneille, mutta useat lautapelien roolipelielementit ovat vielä mukana suurimmassa osassa tietokoneroolipelejä. Roolipelielementeillä tarkoitetaan ominaisuuksia, toimintoja ja tapoja, joilla peli on tuotettu. Yleisin roolipeleihin kuuluva elementti on Adamsin (2014) mukaan maailma, jossa pelaaja pystyy seikkailemaan ja suorittamaan tärkeää tehtävää. Pelihahmolla seikkaillaan maailmassa, jossa hahmo kehittyy ja saavuttaa uusia kykyjä pelin edetessä. Roolipelit kiinnostavat minua peligenreistä eniten ja siksi olen paneutunut niihin opinnäytetyössäni.

Opinnäytetyön tavoite on luoda Unreal Engine -pelimoottorilla roolipeliprojekti ja ladattava demo, jossa esitellään roolipelielementtejä. Projektissa ei luoda valmista peliä, vaan toteutetaan roolipeleihin tarkoitettuja toimintoja. Projektiin kuuluu esimerkiksi seuraavia roolipelielementtejä: pelihahmolla vapaasti maailmassa liikkuminen, esineet, käyttöliittymä, taistelu, pelihahmon attribuutit ja pelihahmon kyvyt. Projektissa ei ole tarinaa, vaan siihen on toteutettu vain pelin toiminnallisuudet.

Roolipelielementtien toteuttamiseen hyödynnetään Unreal Enginen Gameplay Ability System -liitännäistä (GAS). GAS on tarkoitettu pelihahmojen kykyjen ja attribuuttien toteuttamiseen. Kyvyillä pelihahmo taistelee vihollisia vastaan ja yrittää saada niiden elämäpisteattribuutin arvoa nollaan. Kykyjen ja attribuuttien luominen pelihahmoille, taistelun toteutus hyödyntäen kykyjä, kykyjen visualisoiminen käyttöliittymässä ja muiden roolipelimekaniikkojen toteutus hyödyntäen GAS:ia on projektin päätarkoitus. Tavoitteena on oppia ja ymmärtää GAS:n toimintaperiaate ja luoda sillä kykyjä sekä hyödyntää sitä yleisien roolipelielementtien kanssa.

Opinnäytetyön tuotoksena syntynyt roolipelidemo ei ole ladattavissa tekijänoikeuksien ja GitHub-palvelun tiedostojen kokorajoitteiden vuoksi. Roolipelidemosta on YouTube-palvelussa video (Liite 2), jossa näytetään roolipelielementtien toteutuksia, hyödyntäen Epic Gamesin malleja ja visuaalisia efektejä.

Jaettava demo on ladattu GitHub-palveluun ja siitä on luotu dokumentaatio. Dokumentaatiosta löytyy 117 kuvaa projektin malleista ja kykyjen sekä toimintojen luomisesta, kymmeniä koodilaatikoita, joista löytyy satoja rivejä koodia sekä selitykset GAS:n ominaisuuksista ja projektissa käytetyistä funktioista. (Liite 1.)

Tässä raportissa käydään läpi projektin sisältö kokonaisuudessaan ja selostetaan alustavasti GAS-liitännäinen ja roolipelielementit, joita hyödynnetään projektissa. Raportissa kerrotaan myös, mitä dokumentaatio sisältää sekä roolipelidemon ja ladattavan demon idea. Viimeiseksi arvioidaan projektin onnistumista ja sen haasteita.

2 GAS-liitännäinen ja roolipelielementit

Tässä luvussa tarkastellaan GAS:ia ja roolipelielementtejä. Aluksi käydään läpi GAS:n teoriaa, historiaa ja liitännäisen käyttämistä projektissa. Projektissa luodaan roolipelidemo, minkä vuoksi on hyvä tietää, mikä määrittelee pelistä roolipelin. Luvussa käydään läpi myös projektissa hyödynnettävät roolipelielementit.

2.1 GAS-liitännäinen

Gameplay Ability System (GAS) on Epic Gamesin luoma liitännäinen Unreal Engine -pelimoottorille. Opinnäytetyö perustuu liitännäisen käyttämiseen, opiskeluun ja sen hyödyntämiseen Unreal Enginellä. GAS luotiin alun perin Epic Gamesin Paragon-peliprojektille hahmojen iskujen ja kykyjen luomiseen. Paragon ei kuitenkaan päässyt julkaisupisteeseen asti, eikä siitä siksi ole virallista

Epic Gamesin kirjoittamaa julkaisua. Paragonissa käytetyt työkalut ja mallit jäivät Unreal Engine 4:n ja Unreal Engine 5:n käyttöön täysin ilmaiseksi (kuva 1). (Epic Games 2021a.)

Kuva 1. GAS-liitännäinen käytössä Unreal Engine 5-demossa (Epic Games 2021b).

Epic Gamesin (2021) mukaan GAS on joustava liitännäinen Unreal Enginelle, jolla pystyy rakentamaan RPG (roolipeli)- tai MOBA (taisteluareenamoninpeli) -nimikkeessä esiintyviä kykyjä, iskuja ja ominaisuuksia. Kyvyt, iskut ja ominaisuudet ovat pelissä käytettäviä mekaniikkoja, jotka pelihahmo pystyy aktivoimaan. Mekaniikat liittyvät yleensä taisteluominaisuuksiin, mutta GAS-liitännäisellä voi toteuttaa myös esimerkiksi oven avaamisen ja hyppäämisen.

Ohjelmoija voi rakentaa toimintoja tai passiivisia kykyjä pelin hahmojen käytettäväksi sekä tehosteita, jotka voivat lisätä tai kuluttaa erilaisia ominaisuuksia. Ohjelmoija voi myös säätää helposti, milloin, miten ja missä kykyjä voi käyttää ja miten ne käyttäytyvät. GAS voi auttaa suunnittelemaan, toteuttamaan ja yhdistämään tehokkaasti pelin sisäisiä kykyjä yksinkertaisesta hyppäämisestä niin monimutkaisiin kykyihin kuin nykyaikaisissa RPG- tai MOBA-nimikkeissä on nähty. (Epic Games 2021a.)

2.1.1 GAS-liitännäisen hyödyt

GAS:lla on monia vahvuuksia. Yksi sen vahvuuksista on joustavuus. Unreal Engineä käyttävä kehittäjä saattaa olettaa, että ohjelmointi rajoittuu liitännäisen vuoksi. Useassa liitännäisessä rajoitteita voi ilmetä, kun projektissa tulee käyttöön uusia työkaluja tai liitännäisiä, jotka eivät sovellu aiempien liitännäisten kanssa. Kaikki peliprojektit eroavat toisistaan ja projektien edetessä suunnitelmat pelin toteutuksesta kehittyvät ja muuttuvat. Jos peliprojekti on rakennettu kokonaan tukeutuen liitännäiseen, täytyy huomioida, että silloin saattaa joutua luopumaan muista työkaluista, joista olisi projektissa hyötyä. GAS:n käyttäminen ei peruuta muiden työkalujen käyttöä vaan toimii lisätyökaluna niiden rinnalla. Muiden työkalujen hyödyntäminen GAS:n kanssa voi vaatia ohjelmointia tapauksissa, jossa GAS:n funktiot vaikuttavat toiseen työkaluun, eikä työkalussa ole huomioitu GAS:n käyttämistä. GAS on suunniteltu moninpeleihin ja ohjelmoijan ei tarvitse itse miettiä moninpelifunktioita kykyjen luomisessa.

GAS pitää sisällään paljon funktioita ja toimintoja, jotka säästävät ohjelmoijan aikaa. Ohjelmoija pystyy hallitsemaan kykyjen, attribuuttien ja efektien käyttäytymistä sekä niiden vaikutuksia ilman vaivaa suunnitella oma systeemi kykyjen luomiseen. Valmiiden ominaisuuksien vuoksi ohjelmoija voi keskittyä kykyjen luomiseen.

2.1.2 GAS-liitännäisen käyttäminen projektissa

GAS-liitännäinen on laaja kokonaisuus. Siihen liittyy paljon termejä ja toimintoja, eikä kaikkea pysty oppimaan muutaman kuukauden aikana. Tavoitteena GAS:n suhteen minulla oli hallita liitännäisen käyttäminen ja soveltaa sitä kykyjen tuottamiseen demoprojektissa. Kun kyky on luotu, pitää sen käyttäytymiselle tehdä säännöt, miten se saa toimia ja pystyy toimimaan. Kykyjen käyttäminen halutulla tavalla piti opetella useiden kokeilujen avulla.

Kaikilla pelihahmoilla on pelissä käytössä attribuutteja, esimerkiksi elämäpisteet. Tavoitteenani oli pystyä vaikuttamaan attribuuttien arvoihin luokalla, joka laskee, kuinka paljon kyvyt voivat tuottaa vahinkoa. Tällä tavalla pelaaja pystyy kamppailemaan vihollisten kanssa ja säätämään vihollisten vaikeutta attribuuttien muuttamisella.

Jos haluaa luoda valmiin pelin, täytyy GAS:ia soveltaa. Soveltamisella tarkoitetaan esimerkiksi kykyjen ja attribuuttien visualisoimista käyttöliittymässä ja kykyjen aktivoimista eri tavoilla.

2.2 Roolipelielementit

Kun valitsee peliprojektin tyyliksi roolipelin, täytyy ohjelmoijan miettiä mikä tekee pelistä roolipelin ja mitä toiminnallisuuksia se vaatii. Lähdin toteuttamaan roolipelille pohjaa, josta ohjelmoija voi itse lähteä kokoamaan roolipeliä dokumentaation avulla. Pelimekaniikat ja elementit, jotka valitsin tarvittavaksi projektiin, on valittu kahden e-kirjan mukaisesti. Teoksissa Fundamentals of Role-Playing Game Design (Adams 2014) ja Building an RPG with Unity 2018 - Second Edition (Ferro & Sapio 2018) kerrotaan roolipeleistä ja niiden elementeistä Opinnäytetyön demoprojektissa ei ole toteutettu kaikkia elementtejä, koska ne ovat ohjelmoijan peliprojektista riippuvia toteutuksia ja demoprojekti on tarkoitettu vain roolipelitoteutuksen aloittamiseen.

Roolipelit perustuvat maailmaan, jossa on hahmo tai hahmoja, joita pelaaja kontrolloi. Maailmassa seikkaillaan ja edetään pelitarinassa suorittamalla erilaisia ongelmanratkaisutehtäviä, autetaan muita pelihahmoja ja metsästetään hirviöitä. Pelihahmot pystyvät hyödyntämään niille olennaisia kykyjä ja attribuutteja ongelmien ja tehtävien selvittämiseen. Pelin edetessä hahmo voi saada kykyjä ja attribuutteja enemmän sekä nostaa niiden tasoa, mikä lisää personallisuutta pelihahmolle.

Roolipeleille olennaista on myös hahmon vaikuttaminen maailmaan ja siellä tapahtuviin asioihin. Tämä pitää sisällään maailman muokkaamisen ja siellä olevien asioiden ja olioiden kanssa toimimista. (Ferro & Sapio 2018.)

GAS-liitännäisestä on esimerkkiprojekteja ja dokumentaatioita internetissä, mutta mitkään löytämistäni dokumentaatiosta eivät ole moninpeliroolipelejä. Opinnäytetyön dokumentaatio keskittyy kokonaan roolipeleihin ja ottaa huomioon niiden käytön moninpeleissä. GAS:n idea toki on sama kuin muissa netistä löytyvissä esimerkeissä, mutta soveltaminen on erilaista ja sopii erityisesti roolipelien tuottamiseen. Dokumentaation projekti on toteutettu ”hack and slash (toimintapelityyppi, jossa keskitytään voittamaan vihollinen lähitaistelussa)” -tyylillä. Pelihahmoilla on käytössä elämäpiste- ja mana-attribuutti, jotka pyritään saamaan nollaan vastustajan kukistamiseksi.

2.2.1 Tehtävät ja seikkailu

Roolipelien idea on kyky seikkailla pelihahmolla maailmassa, joka on pelaajalle täysin arvaamaton. Mitä enemmän tutkittavia yksityiskohtia maailmassa on, sitä mielenkiintoisemmalta maailma pelaajalle vaikuttaa. (Ferro & Sapio 2018.)

Roolipelit kertovat yleensä pelaajalle tarinaa. Tarinassa pelaajan tavoite on suorittaa jokin tärkeä tehtävä. Tehtävä on jaettu useaan osaan, joissa on useita lyhyempiä tehtäviä. Lyhyet tehtävät nostavat hahmon kykyjä ja voimia, joiden avulla pelihahmo pystyy suorittamaan vaikeampia tehtäviä ja valmistuu lopullisen tavoitteen suorittamiseen. Viimeisenä haasteena nykyaikaisissa roolipeleissä pelaajan on yleensä kukistettava vaikea vastus tai haaste, minkä jälkeen pelin tarina saadaan viimeisteltyä. (Adams 2014.)

Demoprojektissa ei pelihahmolla ole tehtäviä tai tarinaa. Maailmaan voi sijoittaa vihollisia ja esineitä mutta tarinaa ei demossa ole, koska projektista ei luoda valmista peliä. Ohjelmoija voi luoda haluamansa maailman demoprojektiin ja sijoittaa sinne dokumentaation esimerkeissä luotuja vihollisia ja esineitä. Viholliset ja kyvyt ovat säädettäviä ja niistä pystyy tekemään niin vaikeita kun halutaan, dokumentaation esimerkkien mukaisesti. Ohjelmoija itse luo haluamansa tarinan ja päättää miten maailmassa toimitaan.

2.2.2 Pelaajan varasto ja esineet

Pelin edetessä pelihahmo löytää uusia esineitä ja asioita maailmassa. Esineet vaikuttavat esimerkiksi pelin tarinan kulkuun, pelihahmon attribuutteihin ja pelihahmon kykyihin. Varaston idea on säilöä pelaajan keräämät ja ansaitut tavarat. Pelaaja voi halutessaan esimerkiksi laittaa tavaroita päälleen, syödä tai juoda ne tai käyttää tavaraa tehtävän suorittamiseen. (Ferro & Sapio 2018.)

Roolipelidemossa ei pelaajalla ole varsinaista varastoa. Varaston tekeminen on pitkä ja hankala prosessi, jos siihen ei ole perehtynyt ja se pitäisi käsitellä kokonaan omana aiheenaan. Tavarat vaikuttavat usein pelihahmon kykyihin ja attribuutteihin, joten lisäsin peliin miekan ja kolikon. Pelaaja pystyy nostamaan miekan maasta, mikä antaa pelihahmolle kyvyn lyödä miekalla. Jos pelaaja nostaa toisen miekan maasta, vanha miekka tuhoutuu. Pelihahmolla on siis yksi tila miekalle käytössään. Kolikko tippuu, jos pelaaja tappaa vastustajan. Kolikko nostaa pelihahmon valittua attribuuttia ja attribuutin nostamisen jälkeen kolikko tuhoutuu.

2.2.3 Pelihahmon kyvyt ja attribuutit

Pelihahmo käyttää kykyjä ja attribuutteja hyödyksi pelin edetessä. Kyvyt ja attribuutit antavat hahmolle persoonallisuutta ja pelaajalle valtaa muokata hahmoa mieleiseksi. Monesti roolipeleissä pelihahmolle voi valita luokkia, joilla on erilaisia kykyjä ja attribuutteja. (Ferro & Sapio 2018.)

Kyvyt liittyvät roolipeleissä yleensä taikuuteen. Taikoja on monenlaisia ja niiden tavoite on tehdä vahinkoa viholliseen, suojautua tai liikuttaa hahmoa tai pelimaailman objekteja. Ohjelmoijan on suunniteltava kykyjen voimakkuudet ja se, miten niitä voi käyttää. Pelihahmojen taikuutta rajataan attribuuttien määrällä ja sillä, miten nopeasti niitä pystyy käyttämään. Mana-attribuutti on yleinen attribuutti peleissä, joissa on taikuutta. Pelihahmon taiat kuluttavat mana-attribuutin arvoa ja jos se on liian alhainen, taikaa ei voi käyttää. Attribuuttien arvoja yleensä nostetaan käyttämällä erilaisia esineitä. (Adams 2014.)

Demoprojektissa kaikki iskut kuluttavat mana-attribuuttia, paitsi miekalla lyönti ja iskujen väistö. Ne eivät ole demossa taikakykyjä, joten ne eivät kuluta mana-attribuuttia. Ohjelmoija voisi halutessaan määritellä uuden kestävyysattribuutin, jota kuluttavat kyvyt, jotka ei ole taikakykyjä.

Demoprojektissa pelihahmolle luokkia on yksi. Ohjelmoija voi halutessaan kehittää demoprojektia ja lisätä siihen uusia luokkia, jotka hyödyntävät eri kykyjä ja attribuutteja. Luokan toteutus näytetään dokumentaatiossa ja samalla tavalla ohjelmoija voi toteuttaa uusia luokkia. Ohjelmoijan on suunniteltava itse luokan valitseminen ja luokan kehittyminen pelin edetessä, koska luokkien toteutus on erilainen jokaisessa projektissa.

2.2.4 Taistelu

Roolipeleihin yleensä kuuluu viholliset ja taistelu niiden kanssa. Roolipelejä, jossa taisteluita ei ole, on olemassa, ja niissä taistelut on korvattu esimerkiksi ongelmanratkaisutehtävillä. Pelaaja pääsee näissä tilanteissa käyttämään pelissä opittuja taitojaan hyödyksi ja selviytyy niiden avulla pelin esteistä. (Ferro & Sapio 2018.)

Demoprojektissa voi vapaasti kävellä maailmassa, jossa on vihollisia, jotka hyökkäävät pelaajan havaitessaan. Taistelu suoritetaan GAS-liitännäisellä luoduilla kyvyillä käyttäen miekkailua ja taikuutta. Pelaaja pystyy väistelemään iskuja ja pyrkii kukistamaan viholliset, jotka menehtyessään tiputtavat esineen.

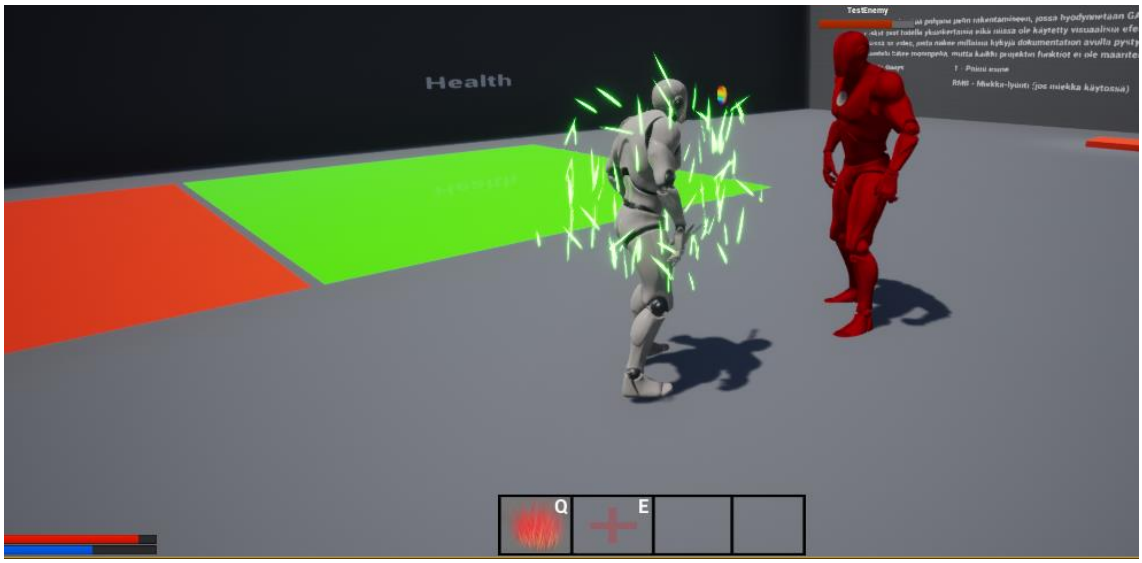
2.2.5 Käyttöliittymä

Pelaajalle täytyy näyttää pelissä paljon asioita, jotka viestivät pelaajalle mitä tapahtuu. Monimutkaisemmissa peleissä pelaajalle kerääntyy paljon dataa ja data pitää olla pelaajan saatavilla mahdollisimman selkeästi ja helposti. Kerättyä dataa ovat esimerkiksi attribuuttien määrät, kerätyt esineet, tehtävien kulku, tuotettu vahinkomäärä ja se, mitä tavaraa pelaajalla on päällä tai käytössä. (Ferro & Sapio 2018.)

Kerätyn datan näyttäminen pelaajalle voi olla myös haitallista. Jotkut pelaajat haluavat nähdä kaiken pelimekaniikkoihin liittyvän datan pelitaitojen maksimoi-

miseksi, toiset pelaajat taas ajattelevat niiden pilaavan fantasiamaailmaan eläytymistä. Ohjelmoijan on tärkeä siksi miettiä, mikä on olennaista pelaajalle näytettävää dataa, jota pelaaja varmasti tarvitsee. (Adams 2014.)

Demoprojektissa käyttöliittymässä näkyy pelaajan attribuuttien arvot ja kykyjen kuvakkeet. Kuvakkeet muuttuvat harmaaksi tilanteissa, joissa niitä ei voi käyttää. Kuvakkeet kertovat pelaajalle selkeästi mitä kykyjä hänellä on käytössä ja voiko sitä käyttää. Vihollisien päällä leijuu myös palkki, joka kuvaa niiden elämäpisteattribuutin arvoa. Kun viholliseen tuottaa vahinkoa kyvyllä, tulee vahinkomäärästä leijuva ilmoitus käyttöliittymään ja vihollisen elämäpistepalkki muuttuu harmaammaksi vahinkomäärän mukaisesti (kuva 2).



Kuva 2. Ladattavan demoprojektin käyttöliittymä.

3 Työkalut

3.1 Unreal Engine

Unreal Engine 4 on Epic Gamesin suunnittelema pelimoottori, jolla ohjelmoija voi tuottaa erilaisia projekteja, esimerkiksi pelejä, videomateriaalia ja työelämässä käytettävää simulaatiota. Unreal Enginessä käytettävä ohjelmointikieli on C++. Ohjelmoija pystyy kirjoittamaan koodia tai hyödyntämään visuaalista ohjelmointimenetelmää nimeltä Blueprint. Blueprint-menetelmä on yleensä aloittelijalle helpompi oppia kuin koodin kirjoitus. (Epic Games 2021c.)

C++-koodia kirjoittamalla ohjelmoija pystyy itse tekemään Blueprint-menetelmässä käytettäviä funktiota ja luomaan C++-luokista uusia Blueprint-luokkia. Tästä syystä Unreal Enginen käyttäjän olisi suositeltavaa osata kummatkin tavat ohjelmoida. (Epic Games 2021c.)

3.2 Projektissa hyödynnetyt lähteet

Projektin toteutustavat perustuvat kahteen lähteeseen, Dan Tranekin GAS-dokumentaatioon (Tranek 2021) ja Epic Gamesin dokumentaatioon (Epic Games 2021a).

3.3 GitHub-palvelu

Projektin versiohallintaan on hyödynnetty GitHub-palvelua. Ladattava demo on myös saatavilla GitHub-palvelun etusivulla. Demosta luotu dokumentaatio löytyy GitHubista osiossa Wiki. (Liite 1.)

3.4 Epic Games -kauppa

Roolipelidemossa on hyödynnetty paljon Epic Games -kaupan malleja, animaatioita ja visuaalisia efektejä. Kaikki mallit ovat ladattavissa ilmaiseksi Epic Games -kaupassa. Animaatiot ja visuaaliset efektit ovat maksullisia, mutta mitään niistä ei tarvitse käyttää projektin toimintojen toteuttamiseen. Ladattavassa demossa hyödynnetään vain valmiita Unreal Engine -projektin malleja.

3.5 YouTube-palvelu

Projekti tarvitsi tavan esitellä roolipeliprojektia, koska se ei ole ladattavissa. YouTube-palvelu oli itselle tutuin palvelu ladata videoita katsottavaksi. Roolipelin video on kuvattu OBS-sovelluksella, eikä sitä ole editoitu (Liite 2.).

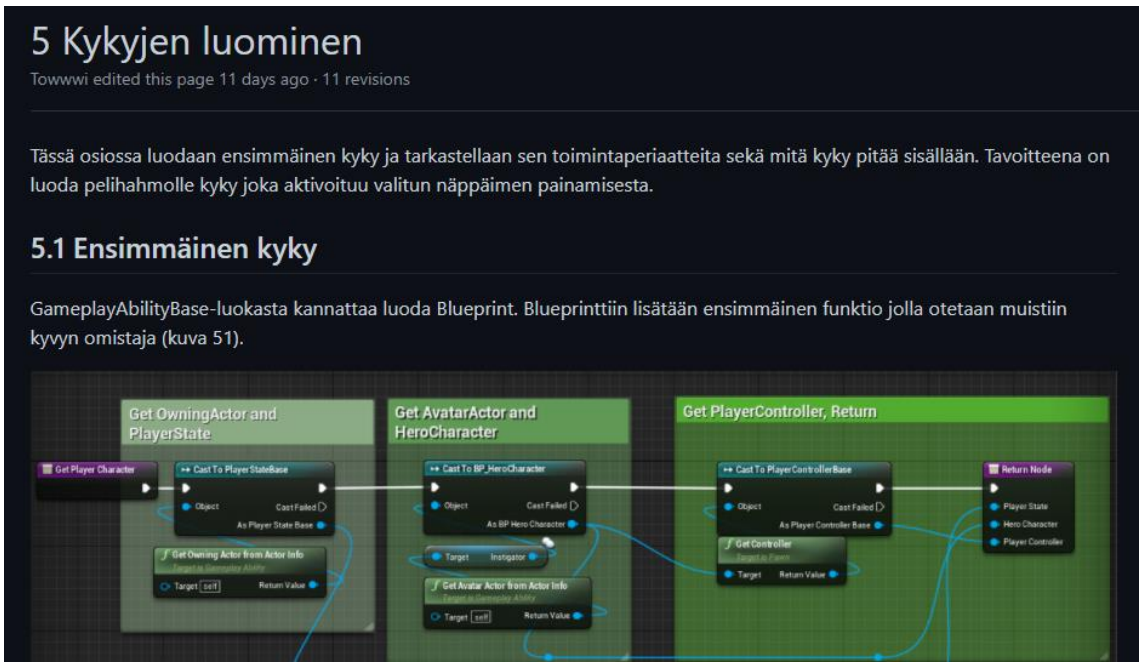
4 Työn tuotokset: dokumentaatio ja demo

4.1 Dokumentaatio

Dokumentaation tavoite on selittää lukijalle GAS-liitännäisen idea, kertoa sen hyödyistä ja opettaa sen käyttö kuvia ja koodilaatikkoja käyttäen. Dokumentaation käyttämiseen lukijalla pitää olla perusymmärrys Unreal Engine 4:n käytöstä ja C++-ohjelmoinnista. Funktiot ja toiminnot on kuvattu lyhyesti ja niistä on näytetty esimerkit.

GAS:n opetteluun dokumentaatio tarjoaa ohjeet lukijalle ”kädestä pitäen”. Dokumentaatiossa näytetään toteutettavat toiminnot ja kaikki mitä ne edellyttävät. Lukija pystyy ohjeita seuraamalla luomaan saman kuin dokumentaatiossa. Tavoite on olla aloittelijalle sopivampi kuin Epic Gamesin ja Dan Tranekin dokumentaatiot, jotka ovat teoreettisempia ja perustuvat itseopiskeluun ja asioiden kokeilemiseen.

Dokumentaatio on suomenkielinen kokonaisuus GAS:n käyttämisestä. Dokumentaatiota voi seurata järjestyksessä ja tehdä samalla omaa projektia tai ladata demoprojektin ja tarkastella sitä dokumentaation kanssa. Dokumentaatiossa kaikki Blueprint-toteutukset on näytetty kuvilla. C++-koodi näytetään aluksi kuvilla ja myöhemmin koodilaatikoilla (kuva 3). Itse opin paremmin ohjelmointia, kun kirjoitan koodin itse, enkä kopioi suoraan dokumentaatiosta. Tämän vuoksi koodi on esitetty aluksi kuvina ja sen jälkeen koodilaatikoina, joista voi suoraan kopioida koodin omaan käyttöön, ilman vaivaa kirjoittaa sitä itse.



Kuva 3. Projektin dokumentaatio.

Dokumentaatiossa on monta lukua, joissa käydään läpi GAS:n teoriaa, käyttöä ja projektissa käytettyjä funktiota. Luvut n nimetty mahdollisimman kuvaamalla tavalla ja ne on suunniteltu tärkeysjärjestykseen, jos lukijan tavoitteena on oppia GAS:n käyttö. GAS:n käytön jälkeen opetetaan sen soveltamista peliprojektiin. Seuraavissa alaluvuissa esitetään dokumentaation sisältö.

4.1.1 Gameplay Ability System

Dokumentaatiossa käydään läpi GAS-liitännäistä kokonaisuutena. Tavoite on kuvata, mikä GAS on. Luvussa myös selitetään, miksi GAS:ia kannattaa käyttää sekä selitetään, millaisia kykyjä pelissä voisi olla. Tavoitteena on saada lukija ymmärtämään mihin GAS:ia voi käyttää. (Liite 1).

4.1.2 GAS-termistö

GAS-liitännäiseen liittyy useita termejä, jotka käyttäjän täytyy ymmärtää. Dokumentaatiossa käydään yleiset termit läpi ja miten ne liittyvät GAS:n käyttöön. Termit on valittu oman arvioni mukaisesti siitä, miten paljon niitä käytetään ohjelmoidessa GAS:lla. Luvussa käsiteltyjä asioita käytetään kaikissa projektissa esiintyvissä kyvyssä, mitkä toteutetaan GAS-liitännäisellä. Jos ohjelmoija ei ymmärrä termejä, ohjelmointityö voi vaikeutua ja dokumentaation seuraaminen on vaikeaa. (Liite 1).

4.1.3 Ability System Component -luokka ja attribuuttien asettaminen

Dokumentaatiossa kerrotaan, miten GAS-liitännäinen aktivoidaan ja aloitetaan ohjelmointi. GAS:n käyttämisen aloittamiseen tarvitaan aina tietyt funktiot. Ilman funktiota pelihahmot eivät pysty käyttämään kykyjä tai ottamaan vastaan attribuutteja, esimerkiksi pelihahmon elämäpisteitä.

Dokumentaation tässä osioissa luodaan pelihahmolle Actor-komponentti, AbilitySystem-komponentti, jonka avulla se pystyy käyttämään kykyjä. Pelihahmolle myös luodaan sille hahmokohtaiset attribuutit ja testataan niiden arvojen muuttumista peliprojektin sisällä. Attribuutit ovat numeroarvoja, jotka sijaitsevat AttributeSet-luokassa. Attribuutteja voi käyttää esimerkiksi pelihahmojen elämäpisteissä, vahingon tuottamisen arvona tai valuuttana. Jos pelissä on pelimekaniikkoihin liittyvä numeroarvo, kannattaa harkita sen luomista GAS-attribuuttina, varsinkin jos attribuutin arvon pitää näkyä moninpelissa kaikille pelaajille. GAS-attribuutit on valmiiksi replikoitu näkyviksi moninpeleissa. (Liite 1).

4.1.4 Kykyjen aktivointi ja antaminen

Peleissä kyvyillä on oltava joku tapa aktivoitua, jotta niiden aktivoiminen on mahdollista. Tässä projektissa kyvyt aktivoidaan painamalla näppäimistön näppäimiä. Dokumentaatiossa selostetaan, miten kykyjen aktivoiminen liitetään näppäimistöön ja annetaan pelaajalle ensimmäinen kyky, jonka se voi aktivoida valitulla näppäimellä. (Liite 1).

4.1.5 Kykyjen luominen

Dokumentaatiossa selitetään tarkemmin kykyjen luominen ja ohjeistetaan ensimmäisen kyvyn toteuttaminen. Kun ensimmäinen kyky on luotu, käsitellään ja ohjeistetaan seuraavan haastavamman kyvyn toteutus. Haastavampi kyky tuottaa vahinkoa vastustajaan, joten luvussa luodaan myös luokka vahinkolaskelmalle. Vahinkolaskelma tarkistaa hahmojen attribuutteja ja laskee niiden mukaan tuotettavan vahinkomäärän. Tavoitteena on luoda kyky, jossa pelihahmo toistaa animaation, luo lentävän projektiilin, joka tuottaa vahinkoa viholliseen osuessaan. (Liite 1).

4.1.6 Roolipelidemo

Dokumentaatiossa käydään läpi roolipelidemon idea ja kerrotaan roolipeliele-menteistä, joita projektissa toteutetaan. Roolipelidemo on kopioitu projekti, jossa hyödynnetään animaatioita ja visuaalisia efektejä. Tavoite on saada tietoa tulevista dokumentaation toiminnoista ja saada käsitys siitä, millainen roolipeli-demosta tulee, koska seuraavassa luvussa ei enää opetella vain GAS:n toimin-toja, vaan luodaan myös roolipelitoimintoja, jotka hyödyntävät GAS:ia. (Liite 1).

4.1.7 Roolipelidemon toteutus

Dokumentaation laajin osio käsittelee roolipelidemon toteutusta. Tavoite on to-teuttaa kaikki luvussa 2.3.6 Roolipeli-demo mainitut toiminnot. Lukijan on tarkoi-tus oppia hyödyntämään GAS:ia käytännössä ja saada luotua pelattava demo, jossa pystyy taistelemaan vihollisten kanssa ja poimimaan esineitä. Lukijan on tarkoitus oppia GAS:n ominaisuuksien käyttäminen ja soveltaminen pelin tuotta-miseen. (Liite 1).

4.1.8 Virheenkorjaus

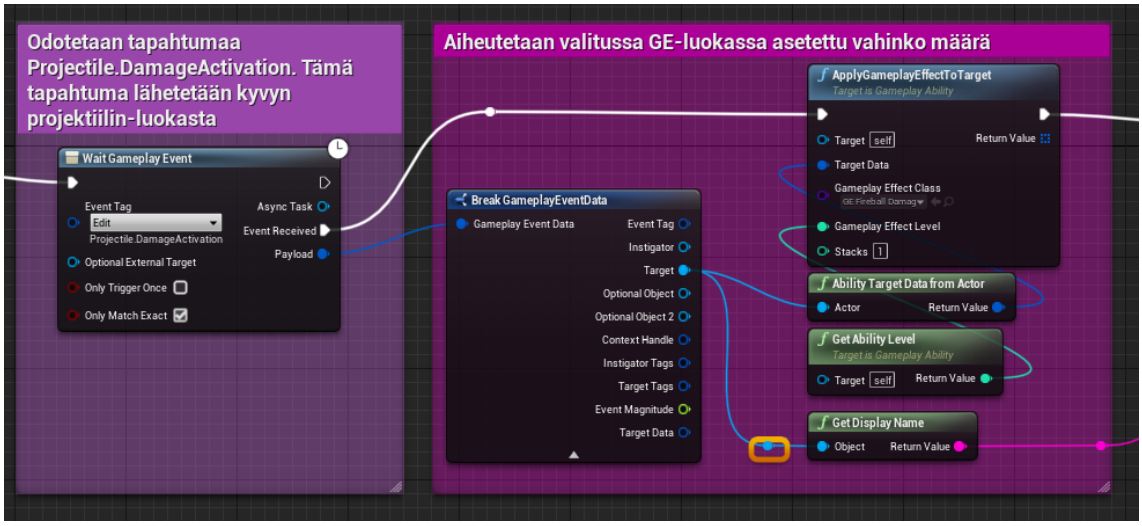
GAS:n opetteluprosessissa ohjelmoijalle saattaa tulla ongelmia ja ne voi olla vaikea havaita ilman aputyökaluja. Aputyökaluina toimii kaksi virheenkorjaus-ruutua, jotka voi kytkeä päälle Unreal Enginessä. Dokumentaatiossa ohjeiste-taan aputyökalujen aktivoiminen ja näytetään kuvilla niiden sisältö. (Liite 1).

4.2 Demo

Dokumentaatiossa toteutettava demoprojekti on ladattavissa dokumentaation sivustolla. Demo on vapaasti käytettävissä Unreal Engine -projekteihin. Demoprojektin tarkoitus on näyttää lukijalle, miten hyödyntää GAS:ia roolipe-leissä. Roolipeli-genre on valittu oman mieltymykseni mukaisesti. Tavoitteena on näyttää ja opettaa lukijalle yleiset roolipelielementtien toteutukset ja näyttää niiden rakentaminen, hyödyntäen GAS-liitännäistä. Projektissa huomioidaan projektin jatkaminen moninpelitoteutukseksi.

Useissa kyvyissä pelihahmo toistaa animaation ja tekijänoikeuksien vuoksi animaatioita ja visuaalisia efektejä ei voinut ladata nettiin. Tämän vuoksi ladattavassa projektissa käytetään vain yksinkertaisia malleja ja Unreal Enginen valmiita animaatioita, joita saa käyttää vapaasti kaikissa Unreal Engine -projekteissa. Ladattavaa demoa tutkimalla lukijan on tarkoitus saada selkeä käsitys kykyjen toiminnoista ja luomisesta.

Suurin osa demon Blueprinteistä on myös kommentoitu suomeksi (kuva 4). Kommenttien toistamisen välttämiseksi useasti toistuvassa koodissa kommentti on saatettu jättää pois, jos se on kommentoitu jo aiemmin. Funktioiden ja muiden koodirakenteiden nimet on kommenteissa aina kuvattu englanniksi niiden oikeilla nimillä selkeyden vuoksi.



Kuva 4. Blueprint-luokissa löytyy selitykset toiminnoille.

4.3 Roolipeli-demo

Roolipelidemo on kopioitu projekti dokumentaatiossa toteutetusta demoprojektista. Roolipelidemossa käytetään malleja ja visuaalisia efektejä, joita ei tekijänoikeuksien vuoksi voi ladata GitHub-palveluun jaettavaksi ja siksi sitä esitellään vain videolla (kuva 5). GitHub-palveluun ladatussa demossa on hyödynnetty vain Unreal Enginen ilmaisia valmiita malleja ja tämän takia sitä voi käyttää kaikissa Unreal Engine -projekteissa.

Roolipelidemoa ei mallien tekijänoikeuksien ja tiedostojen suuren koon vuoksi pysty lataamaan GitHubiin ilman maksullista versiota GitHub-palvelusta. YouTube-palveluun on ladattu video roolipelidemosta, jossa näytetään, millaisen Unreal Engine-projektin dokumentaation avulla pystyy luomaan hyödyntämällä, Epic Gamesin malleja ja visuaalisia efektejä (Liite 2). Videolla myös selitetään kykyjen toiminnallisuuksista. Videon katsomiseen on suositeltavaa olla peruskäsitys GAS:n termeistä.

Kuva 5. Roolipeli-demo, jossa hyödynnetään ladattuja malleja ja animaatioita.

5 Pohdinta

Tässä luvussa arvioin projektin onnistumista. Projekti onnistui kokonaisuudessaan hyvin ja dokumentaatiosta ja demosta tuli onnistunut tapa näyttää GAS:n toimintaa roolipeleissä. Se on ainut suomenkielinen dokumentaatio GAS-liitännäisestä ja pystyy auttamaan lukijaa GAS:n opettelussa ja roolipeliprojektin tuottamisessa. Toiset netistä löytyvät GAS-dokumentaatiot ovat tiedoltaan laajoja mutta liitännäisen käyttö pitää opetella täysin itse. Tässä dokumentaatiossa näytetään esimerkit kaikista toiminnoista selityksillä, mikä pyrkii ehkäisemään virheitä.

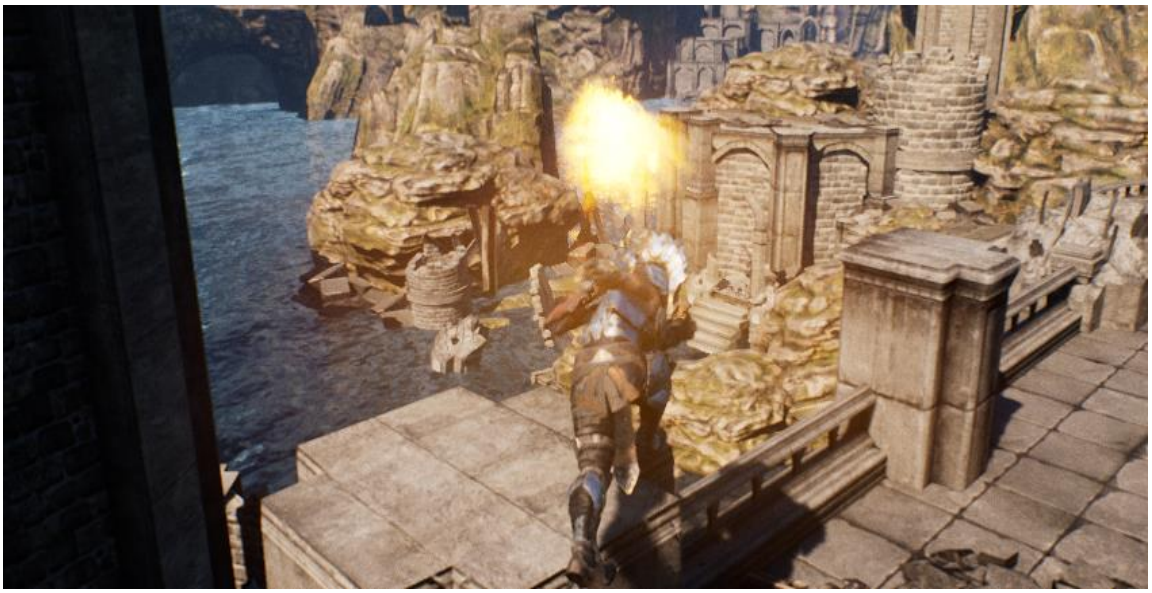
5.1 Opitut asiat GAS-liitännäisestä ja Unreal Enginestä

Projektin aikana opin todella paljon Unreal Enginen eri aiheista. Tavoitteeni oli vain oppia GAS:n käyttö, mutta todellisuudessa opittavaa tulikin paljon enemmän, liittyen roolipelitoimintoihin.

5.1.1 GAS

Luvussa 2.1.2 kerrotaan GAS:n tavoitteista projektissa. Päätavoitteeni suurimpana kokonaisuutena oli GAS:n käyttäminen. Pystyn nyt hyödyntämään sitä kaikissa tulevissa Unreal Engine -projekteissani.

Opin luomaan näyttäviä kykyjä, jotka tuottavat halutun määrän vahinkoa vastustajaan (kuva 6). Vahingon tuottamisesta minulla oli selvä idea jo projektin alussa. Pelihahmo lyö tai heittää vastustajaa projektiilillä, joka törmätessään tarkastaa, mihin projektiili osuu. Vahinkomäärä lasketaan erillisessä laskelmaluokassa, joka tarkastaa hahmon attribuuttien arvot ja sen perusteella laskee vahinkomäärän. Haasteena oli heitettävän projektiilin luominen ja vahingon välittäminen oikeaan viholliseen. Ladattavassa demoprojektissa pelaaja pystyy heittämään projektiilin viholliseen sekä vihollinen pystyy heittämään projektiilin pelaajaan. Projektiili tuottaa halutun määrän vahinkoa kohteeseen.



Kuva 6. Pelihahmo käyttää tulipallokykyä.

Pelihahmolla myös on kyky väistää vihollisen kykyjä, lyödä miekalla ja lisätä elämäpisteitä. Demoprojektista löytyvät kyvyt pystyvät aktivoimaan näppäimistöllä ja miekalla lyöminen hiirennäppäimellä. Käytettävät näppäimet on määrätty editorissa. Kyvyt ja attribuutit on visualisoitu käyttöliittymässä, joten pelaaja pystyy näkemään omat attribuutit sekä vihollisten elämäpisteet.

Pelikehittäjänä GAS on minulle kokonaan uusi opittu työkalu, jota voin hyödyntää työnhakemisessa pelialalla. Moninpelit kiinnostavat minua enemmän kuin yksinpelit ja GAS:lla pystyn luomaan nyt kaikkiin tuleviin projekteihini attribuutit ja pelihahmojen kyvyt liitännäistä käyttäen ja ne toimivat moninpelissa. Uskon käytäväni GAS:ia ainakin kaikissa omissa Unreal Engine -projekteissani.

5.1.2 Viholliset

Roolipeli, jonka idea on voittaa vastustaja, on mahdotonta toteuttaa ilman ainuttakaan vastustajaa, joten siihen oli pakko lisätä myös vihollisia. Vihollisilla piti olla kyky liikkua, käyttää kykyjä taistelemiseen, vahingoittaa pelihahmoa, vahingoittua itse ja kuolla. Vihollisen tekoäly ei projektissa ollut aiheena, joten tein taistelusta mahdollisimman yksinkertaisen. En aiemmin ollut tehnyt lähitaistelutekoälyä, joten kaikki oli aiheeseen liittyen lähes uutta. Vihollisten kanssa pystyy nyt taistelemaan.

Pelikehittäjänä opitut asiat tekoälystä jäivät aika vajaaksi. Osaan luoda tekoälyn, joka ei sovellu valmiiseen peliin. Valmiissa pelissä vaadittaisiin monimutkaisempi tekoäly, joka tekisi taistelusta mielenkiintoisempaa, mutta projektin tekoäly riittää kykyjen näyttämiseen. Opin kuitenkin luomaan funktiot yksinkertaisien taistelutilanteiden luomiseen pelissä, hyödyntäen GAS:n kykyjä. Seuraavaksi tekoälyn käyttäytymistä ja funktioita pitää muokata mielenkiintoisemmaksi ja ennalta-arvaamattomaksi.

5.1.3 Roolipeli-ominaisuudet

Oma ideani projektissa oli ensiksi opetella GAS:n käyttö ja sen jälkeen hyödyntää sitä roolipelissä. Myös dokumentaatio on toteutettu tällä tavalla. Demoon tuli paljon toiminnallisuuksia, jotka ovat tuttuja roolipeleissä, joissa taisteleminen on olennainen osa peliä.

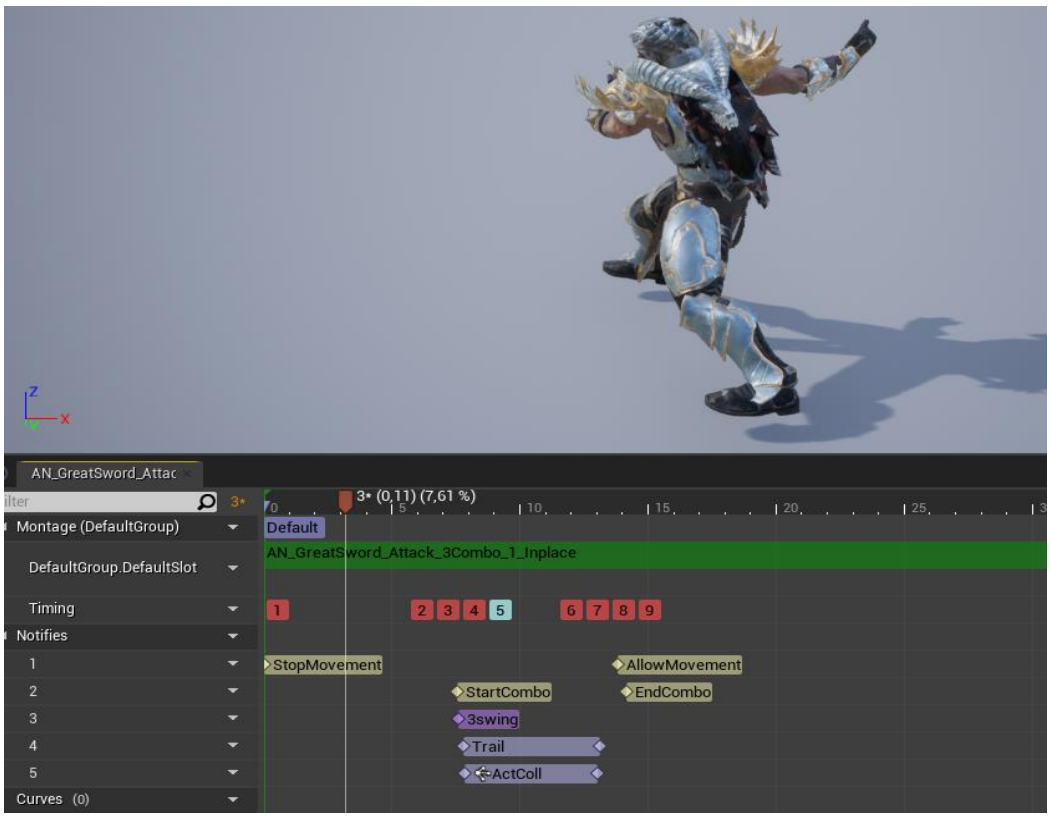
5.1.4 Moninpeli

Moninpeli-toiminnot ovat laaja ja kokonainen oma aiheensa, enkä siksi halunnut ottaa sitä käsiteltäväksi aiheeksi. GAS on kuitenkin suunniteltu moninpeleille, joten en jättänyt sitä myöskään kokonaan huomioimatta. Kaikki pelin kyvyt ja viholliset toimivat moninpelissä, ja opin projektin ohella hieman moninpelitermistöä ja sen ideaa. Se miten funktiot ajetaan moninpeleissa serverillä, on täysin projektikohtaista ja vaatii jokaisessa luokassa päätöksen, miten sen replikaatio toimii pelissä. Tämän takia päätin jättää moninpelifunktiot mahdollisimman vähäiseksi.

Opin kuitenkin riittävästi tietoa moninpelitoiminnoista, että pystyn jatkossa luomaan moninpelin. En pysty vielä kuitenkaan sanomaan ymmärtäväni moninpelitoimintoja täysin, enkä osaisi luoda toimintoja optimaalisella tavalla.

5.1.5 Animaatiot

Kykyjen aktivoimisessa yleensä pelihahmo toistaa jonkin animaation, esimerkiksi miekalla lyönti. Animaatioissa Unreal Enginessä voi myös kutsua funktioita ja toteuttaa paljon hyödyllisiä ominaisuuksia. Tämäkin oli minulle aika uutta asiaa ja loin useita funktioita, joita hyödynsin kyvyissä. Funktiot kutsuttiin animaatioista ja tällä tavalla sain tietyn funktion toteutumaan juuri haluttuna hetkenä animaatiossa (kuva 7).



Kuva 7. Miekkalyöntikyvyn animaatio kutsuu yhdeksän funktiota animaation aikana.

Tavoitteeni oli hyödyntää samaa luurankoa kaikissa animaatioissa. Tämän takia opettelin myös hahmojen luiden uudelleenkohdentamisen. Uudelleenkohdentamisella tarkoitan saman luurangon jakamista useammalla hahmolla.

Kyky aktivoida funktioita tietyssä kohdassa animaatiota on hyödyllinen kyky kaikissa tilanteissa, joissa pelihahmo tekee animaation ja ohjelmoija haluaa jotain tapahtuvan samalla hetkellä. Tämä varmistaa toiminnon toistuvan aina ja vain jos animaatio toistetaan haluttuun kohtaan asti.
Uudelleenkohdentaminen säästää ohjelmoijalta aikaa, kun projektiin tulee uusia malleja. Ohjelmoijan ei tarvitse muokata mallien luurankoja, vaan pystyy kohdentamaan ne uudestaan sopivaksi.

5.2 Projektin haasteet

Projektissa tuli useasti haasteita, jotka usein liittyivät ohjelmointiin. Haasteet estivät projektia etenemästä ja hidastivat työskentelyä. Ajattelin projektin sujuvan nopeasti ilman suurempia ongelmia. Arvioin myös väärin ajan, miten pitkään ongelmien selvittämiseen voi mennä aikaa.

5.2.1 Dokumentaation haasteet

Dokumentointi itsessään olikin paljon työläämpää, kuin olin kuvitellut. Ohjelmointiosuus ja siihen liittyvä ongelmien selvitys vei todella paljon aikaa. Ohjelmoinnin jälkeen kävin vielä kaiken tekemäni uudestaan läpi ja selitin ne kuvaavalla tavalla. Tein myös kaikesta koodista koodilaatikot tai otin kuvat dokumentaatioon. Tämä toimintatapa kaksinkertaisti käytetyn aikani vain ohjelmointiin verrattuna. Mahdollisesti kyllä opin tällä tavalla aiheista paremmin, kun jouduin selittämään ne vielä uudestaan. Dokumentaatioon olisi mahdollisesti voinut vielä lisätä esimerkkejä erilaisista kyvyistä, mutta ajan käytön takia niitä ei tullut niin paljon kuin ajattelin. Idea kaikissa kykyjen luomisessa on sama, joten tämä ei ollut välttämätöntä.

5.2.2 Demon haasteet

Ohjelmallisesti demo oli hankala toteuttaa. Aluksi en meinannut päästä ollenkaan alkuun GAS-liitännäisen käyttöön. Opin nopeasti kuitenkin liitännäisen perusidean. Ohjelmointikielen kanssa minulla ei ollut vaikeuksia. Ongelmat liittyivät GAS:n toiminnallisuuksiin ja niiden hyödyntämiseen. GAS täytyi ymmärtää kokonaisuudessaan hyvin, että sitä pystyi hyödyntämään peliominaisuuksien luomiseen.

5.3 Ajankäyttö

Ajankäyttö muuttui suunnitellusta paljon. Aluksi idea oli saada projekti valmiiksi nopeasti, kahden kuukauden aikana. Nopeasti selvisi, että dokumentointi tuo paljon lisätyötä ja projekti vie todellisuudessa monta kuukautta niillä työtunneilla, mitä projektiin pystyn käyttämään. Projektilla ei ollut pakollista päivää, milloin se pitää valmistua, joten projektin aikataulu muuttui usealla kuukaudella. Suurimmaksi osaksi aikaa meni ohjelmointiin ja siihen liittyvään ongelmien selvittämiseen. Valmista koodia oli paljon, mutta kirjoitin myös yli tuhat riviä omaa

koodia sekä kymmeniä blueprinttejä. Aihe oli minulle tuntematon, joten ongelman selvittäminen tuotti haasteita. Koodista löytyvien ongelmien takia projekti ei välttämättä edennyt ollenkaan, kunnes ongelmat oli saatu korjattua.

Minun täytyi myös suunnitella, millaisen demon haluan toteuttaa. Käytettyjä roolipelielementtejä ei voinut suoraan toteuttaa tietyistä malleista vaan hyödynsin oppimiani ohjelmointitaitoja ja kävin läpi vanhoja Unreal Engine -projekteja, joista löysin hyödynnettäviä funktiota ja toimintatapoja.

Lähteet

Adams, E. 2014. Fundamentals of Role-Playing Game Design. https://learning.oreilly.com/home/. 04.05.2022.

Epic Games. 2021a. Gameplay Ability. https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/GameplayAbilitySystem/GameplayAbility/. 25.09.2021.

Epic Games. 2021b. Valley of the Ancient Sample. https://docs.unrealengine.com/5.0/en-US/ContentAndSamples/ValleyOfTheAncient/. 30.03.2022.

Epic Games. 2021c. Programming and Scripting. https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/. 22.09.2022.

Ferro, S. & Sapio, L. 2018. Building an RPG with Unity 2018 - Second Edition. https://subscription.packtpub.com/book/game-development/9781788623469/1/ch01lvl1sec11/characteristics-of-an-rpg. 19.04.2022.

Karamian, V. 2018. Building an RPG with Unity 2018. https://learning.oreilly.com/library/view/building-an-rpg/9781788623469/d7bcf98c-967e-495c-95a7-dcaccfa646bb.xhtml. 05.05.2022.

Prinke, M. 2021. A Guided Tour of Gameplay Abilities. Inside Unreal [video]. https://www.youtube.com/watch?v=YvXvWa6vbAA&t=49s. 06.10.2021.

SabreDartStudios. 2018. Intro to Gameplay Abilities in Unreal Engine 4 [video]. https://www.youtube.com/watch?v=Ev2P6BTUxN0. 06.10.2021.

Tranek, D. 2021. GAS Documentation. https://github.com/tranek/GASDocumentation. 01.10.2021.

Dokumentaatio

https://github.com/Towwwi/GAS/wiki.

Roolipelidemovideo

https://www.youtube.com/watch?v=Jd4xKmdOlp8.