Jesse Sainio

Roolipeliliitännäisen toteutus visuaalisella ohjelmointityökalulla

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

21.8.2017

Metropolia

Tekijä Otsikko Sivumäärä Aika	Jesse Sainio Roolipeliliitännäisen toteutus visuaalisella ohjelmointityökalulla 39 sivua 27.3.2018
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	tieto- ja viestintätekniikka
Ammatillinen pääaine	Pelisovellukset
Ohjaajat	Lehtori Antti Laiho

Insinöörityön tavoitteena oli toteuttaa liitännäinen, mikä mahdollistaa erilaisten toimintaroolipelien prototyypin nopean valmistuksen. Liitännäinen sisältää useimmista roolipeleistä löytyvät peruselementit, kuten heijastusnäytön, josta näkee pelaajan terveyspisteet, taikapisteet ja kestävyyspisteet, taitopuun hahmonkehitystä varten sekä esineiden keräämiseen ja käyttämiseen liittyvät toiminnot.

Liitännäisen kaikki ominaisuudet oli tarkoitus toteuttaa niin, että niitä on helppo käyttää. Liitännäisen kaikki osat rakennettiin niin, että ne toimivat yhdessä sekä erikseen. Ominaisuuksista tehtiin vain yksinkertaiset luurankomallit, joita käyttäjä voi laajentaa sekä muokata haluamallaan tavalla.

Insinöörityön lopputulos oli pääpiirteittäin sen mukainen, kuin se suunniteltiin. Joitakin ominaisuuksia ei ehditty saada valmiiksi, ja muut jäivät kaipaamaan vielä optimointia. Kokonaan tavoitteet eivät toteutuneet, ja liitännäinen jäi vielä kesken.

Avainsanat	Unreal Engine, pelimoottori, sinikopio, liitännäinen, roolipeli, visuaalinen ohjelmointi

Author Title Number of Pages Date	Jesse Sainio Implementing a Role-playing plugin using visual scripting method. 39 pages 27 March 2018
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Game Applications
Instructors	Antti Laiho, Senior Lecturer

Aim of the thesis was to create a plugin that allows fast production of different action role-playing game prototypes. Plugin consists of basic elements found in most role-playing games such as heads up display showing players health, stamina and magic points, skill tree for character development and ability to collect and use different items.

All features were designed so that they would be easy to use. All parts of the plugin were designed to function on their own and as a part of each other. Only basic skeleton models of each feature were produced so that users could easily modify and expand the plugin the way that they desire.

Overall, the finished project matched pretty well the desired outcome. Some features weren't finished in time and others need clean up and optimization. As a whole the desired goals weren't met, and the plugin was left unfinished.

Keywords	Unreal Engine, game engine, blueprint, plugin, role-playing game, visual scripting

Sisällys

Lyhenteet

RPG	Role-playing game. Pelialalla yleisesti käytetty lyhenne roolipelikategoriasta.
JRPG	Japanese role-playing game. Käytetään kuvaamaan Japanissa tehtyjä ja samankaltaisia roolipelejä.
D&D	Dungeon&Dragons. Fantasia-aiheinen roolilautapeli.
FPS	First Person Shooter. Ammuntapeli-genre jossa kuvakulma on ensimmäisen persoonan näkökulmassa.
TPS	Third Person Shooter. Ammuntapeli-genre jossa kuvakulma on kolmannen persoonan näkökulmassa.
NPC	Non player character. Tietokoneen ohjaama hahmo pelimaailmassa.

1 Johdanto

Opinnäytetyön tarkoituksena oli tutustua tarkemmin Unreal Engine 4 -pelimoottoriin ja etenkin pelimoottorin visuaaliseen koodausmenetelmään. Tämä visuaalinen koodausmenetelmä on nimeltään sinikopio (eng. Blueprint).

Osana joitakin kouluprojekteja käytettiin Unreal Engine -pelimoottoria. Näissä projekteissa oli kuitenkin haasteena se, että kukaan ryhmän jäsenistä ei ollut käyttänyt pelimoottoria aikaisemmin. Vuoden aikana projekti saatiin jotenkin pelattavaan kuntoon, mutta jos haluttiin käyttää pelimoottoria jatkossa, täytyi siihen tutustua paremmin.

Idea työhön tuli, kun pelattiin Assassin's Creed Origins ja The Legend of Zelda: Breath of the Wild -pelejä. Haluttiin nähdä, miten vastaavanlainen peli toteutetaan. Koska ei ollut realistista lähteä rakentamaan koko peliä, päätettiin keskittyä pelattavaan hahmoon sekä siihen liittyviin ominaisuuksiin.

Työn tavoitteena oli toteuttaa liitännäinen, mikä mahdollistaa erilaisten toimintaroolipelien prototyypin nopean valmistuksen. Liitännäinen sisältää useimmista roolipeleistä löytyvät peruselementit, kuten heijastusnäytön, josta näkee pelaajan terveyspisteet, taikapisteet ja kestävyyspisteet, taitopuun hahmonkehitystä varten sekä esineiden keräämiseen ja käyttämiseen liittyvät toiminnot. Kaikki ominaisuudet oli tarkoitus toteuttaa niin, että niitä on helppo laajentaa ja muokata halutulla tavalla.

Työ käy läpi, miten liitännäistä käytetään sekä miten se on toteutettu. Työstä käy ilmi, miten liitännäisen eri osia käytetään ja miten ne toimivat yhdessä sekä erikseen. Toteutus käydään läpi pääpiirteittäin, sillä syvemmin liitännäisen toteutukseen syventyminen ei ole tämän työn puitteissa järkevää. Työssä käydään läpi myös esimerkkejä siitä, miten liitännäistä voi laajentaa sekä jatkokehittää.

2 Roolipelien perusrakenne

Kun lähdin suunnittelemaan liitännäisen sisältöä, minun piti ensin määrittää, mikä tekee pelistä RPG (Role-playing game) -pelin. Google tarjoaa monta lähdettä, joiden mukaan RPG-peli on peli, jossa pelaaja astuu fiktiivisen hahmon saappaisiin ja yrittää toteuttaa jotakin tavoitetta tiettyjen sääntöjen puitteissa [1]. Nykypäivänä tuo määritelmä sopii tosin lähes mihin tahansa peliin. Nykyään monet pelit ottavat vaikutteista useasta genrestä eivätkä ole enää pelkästään tietyn genren pelejä. Miten siis määritellä RPG-peli? Kenties helpoin tapa kunnolla määritellä, mitkä elementit tekevät RPG-pelin, on katsoa historiaan ja ensimmäisiin RPG-peleihin. Kun lähdetään siitä, mikä aikoinaan määritteli RPG-genren, päästään käsiksi genren olemukseen.

Ennen digitaalisia RPG-pelejä kaikki lähti aikoinaan liikkeelle Dungeons & Dragons (D&D) -pelistä, joka oli ensimmäisiä RPG-lautapelejä. Se loi useat RPG-genren käsitteet, kuten hahmon luokka, hahmon kehitys sekä tarina, joka kuljettaa pelaajaa eteenpäin. [2.] Hyvin pian D&Dn jälkeen tulivat ensimmäiset RPG-videopelit, kuten Ultima ja Wizardry, jotka olivat ensimmäisiä merkittäviä RPG-videopelejä. Näiden kahden pohjalta luotiin nykyäänkin tunnetut ja elossa olevat videopelisarjat Dragon Quest ja Final Fantasy. [3; 4.] Molemmat pelit luokitellaan JRPG (Japanese role-playing game) -genreen, joka eroaa länsimäisestä RGP-genrestä huomattavasti.

JRPG-pelit ovat yleensä enemmän tarinavetoisia ja keskittyvät hahmojen välisiin suhteisiin ja dialogiin, kun taas länsimaalaiset RPG-pelit keskittyvät enemmän toimintaan. Toinen iso ero näiden kahden välillä on se, että länsimaiset RPG-pelit omaksuvat monia ominaisuuksia muista genreistä, kuten Fallout- ja Mass Effect -sarjat. Fallout-pelien taistelu toimii samoin kuin FPS (First Person Shooter) peleissä. Mass Effect -sarja käyttää taisteluissa TPS (Third Person Shooter) -mekaniikkoja. Molemmat pelit luokitellaan kuitenkin RPG-peleiksi, vaikka ne lainaavat mekaniikkoja muista genreistä. JRPG pelit puolestaan pysyttelevät niin hyvässä kuin pahassakin enimmäkseen oman genrensä sisällä. Tästä syystä JRPG-pelit sopivat paremmin määrittelemään RPG-genreä sen puhtaimmassa muodossa. [5.]

Koska Final Fantasy ja Dragon Quest pelisarjat olivat ensimmäisten RPG-pelien joukossa ja ne jatkuvat vielä nykyään, päätettiin määritellä RPG-pelien pääelementit niiden pohjalta. Jo molempien sarjojen ensimmäisistä peleistä löytyivät RGP-genren peruselementit, kuten tehtävien kuljettama tarina, pelaajan kehitys, NPC (Non player character) -hahmojen kanssa puhuminen, erilaiset kerättävät esineet, mahdollisuus ostaa ja myydä esineitä sekä taistelu vihollisia vastaan. Ensimmäisestä Final Fantasy -pelistä löytyi myös Dragon Questin vasta myöhemmin omaksumat hahmojen luokat, kuten soturi tai velho. Uusien pelien myötä molempiin sarjoihin alkoi ilmestyä ominaisuuksia, kuten sivutehtäviä, uusien kykyjen oppiminen ja pelaajan ominaisuudet esim. voima, nopeus ja ketteryys. [6; 7.]

Uusien pelien myötä lista vain pitenee. Viimeisin Final Fantasy -sarjan peli, Final Fantasy XV, tuo sarjaan monia ominaisuuksia, jotka löytyvät myös monista muiden genrejen peleistä [8]. Viimeisten vuosien aikana myös JRPG-pelit ovat alkaneet yhdistellä elementtejä muista genreistä. Oikeastaan nykypäivänä eri genrejen rajat ovat niin häälyväisiä ja kiisteltyjä, että niiden tarkkailu ja analysointi ei tämän työn puitteissa ole mahdollista. Tästä syystä keskitytään RPG-genren alkuun ja jätetään vähemmälle huomiolle genren nykytilanteen.

Mikä siis on RPG-genren ydin? Mitä pelaaja teki ensimmäisissä Final Fantasy- ja Dragon Quest -peleissä? Suurin osa pelaajan ajasta meni hierontaan (eng Grinding) saadakseen pelattavan hahmon tarpeeksi vahvaksi, jotta pääsisi eteenpäin. Kaikki alkaa ja palaa hahmon kehitykseen. Niinkin yksinkertainen on RPG-pelien ydin. Kaikki pelaajan ohjaaman hahmon tai hahmojen kehitykseen liittyvät seikat toki monimutkaistavat asiaa. Kuitenkin kaiken ytimessä on hahmon kehitys pelaajan tekemien päätösten mukaan. [9.] Tämän lisäksi pelistä on toki hyvä löytyä muutakin, ei pelkästään RPG-peleille tyypillisiä ominaisuuksia, kuten tarina, taistelut ja seikkailu. Lista jatkuu, mutta rajallisen ajan sekä resurssien kannalta täytyy työhön sisällytettävät ominaisuudet rajata jotenkin.

Mitä siis tarvitaan hahmon kehitykseen kaikkein yksinkertaisimmillaan? Hahmon taso on se mistä kaikki lähtee, mutta nykypäivänä hahmon tason nousuun liittyy paljon muutakin. Hahmon tason noustessa voidaan lisätä myös hahmon terveys-, taika- ja kestävyyspisteitä sekä hahmon ominaisuuksia kuten voimaa, ketteryyttä ja nopeutta. Lisäksi usein pelaaja voi avata hahmon tason noustessa uusia kykyjä. Näistä on hyvä lähteä liikkeelle. Hahmon tason nostamiseen tarvitaan jokin syy tason nousulle, kuten tehtävien suorittaminen tai vihollisten kukistaminen. Vihollisten kukistamiseen tarvitaan todennäköisesti

jokin ase sekä suoja vihollisen hyökkäyksiä vastaan. Aseen voi saada esimerkiksi ostamalla sen kaupasta tai ottamalla kukistetulta viholliselta. Vihollinen todennäköiset tekee myös pelaajalle vahinkoa, joten terveyspisteitä olisi myös hyvä saada nostettua jotenkin. Jotta tehtävät olisivat mielekkäitä, on hyvä olla myös NPC-hahmoja, joiden kanssa voi puhua. Nyt näyttää olevan kasassa sopiva joukko perusominaisuuksia, joista on hyvä lähteä liikkeelle.

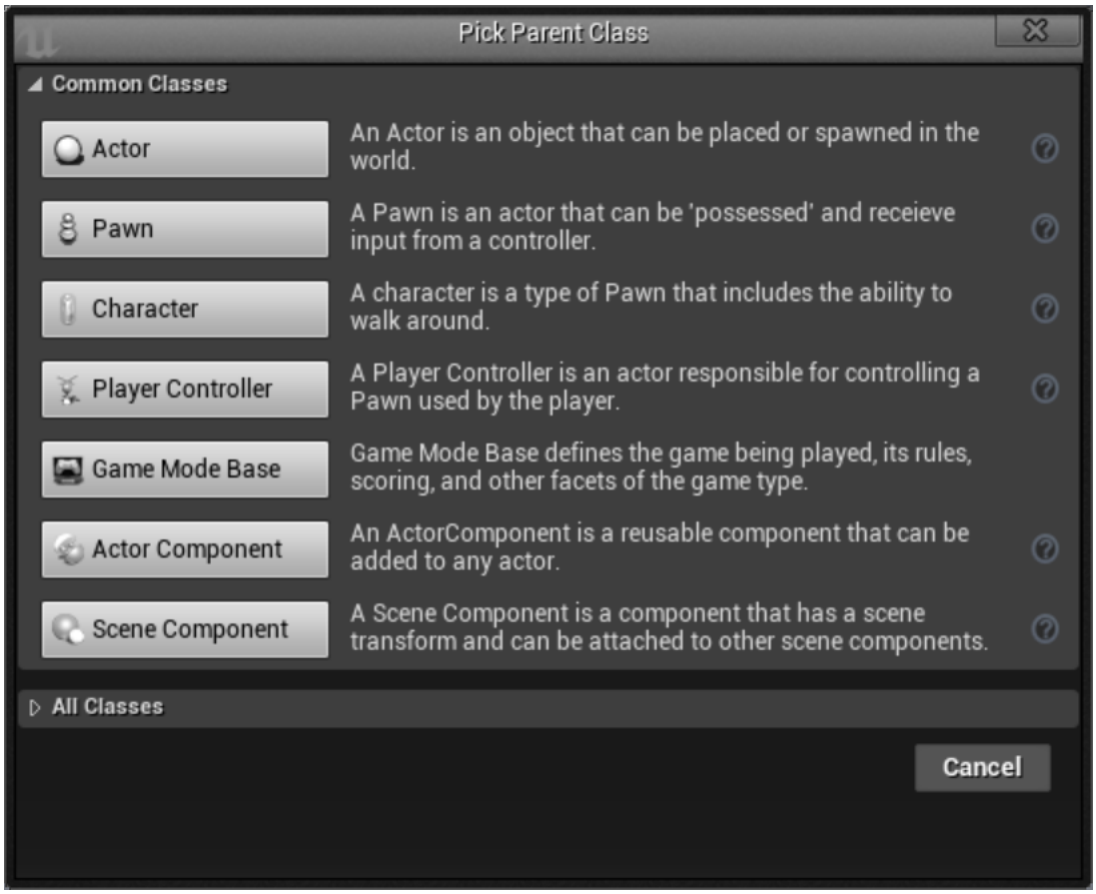
- hahmon terveys-, taika- ja kestävyyspisteet, taso ja kyvyt
- kerättäviä esineitä, kuten aseita suojia ja ruokaa
- kauppa, josta voi ostaa kyseisiä esineitä
- dialogisysteemi, jotta voidaan puhua muille hahmoille
- taistelusysteemi
- suoritettavia tehtäviä.

Näistä kuudesta ominaisuudesta saadaan kasattua yksinkertainen RPG-prototyyppi. Todettiin, että näiden ominaisuuksien toteuttaminen olisi todennäköisesti sopivan laaja projekti tämän työn aiheeksi.

3 Liitännäisen toiminta

Jotta ymmärretään helpommin, miten liitännäinen toimii, on hyvä ensin käydä läpi pelimoottoriin ja etenkin sinikopioihin liittyviä käsitteitä. Näihin lukeutuvat sinikopio, näyttelijä, pelinappula, hahmo, pelaajaohjain, tekoälyohjain sekä komponentti.

Sinikopio on Unreal Engine -pelimoottorissa käytetty solmupohjainen visuaalinen koodausmenetelmä. Sinikopioilla voidaan toteuttaa käytännössä kaikki, mitä voidaan tehdä koodaamalla C++:alla. Sinikopioita käytetään yhdistämällä erilaisia solmuja, funktioita ja muuttujia toisiinsa. [10.] Sinikopiot jaetaan niiden käyttötarkoituksen mukaan eri luokkiin (eng. Blueprint Classes), kuten kuvasta 1 näkyy, jotka perivät ominaisuuksia luokka hierarkian mukaisesti. Sinikopioita käytettäessä on hyvä olla tietoinen, mitä luokkaa milloinkin tulee käyttää. Sinikopion luokka määrää monia sen ominaisuuksia ja rajaa sen käyttöä joissakin tilanteissa. [11.]



Kuva 1. Sinikopioiden perusluokat.

Näyttelijä (eng. Actor) on kaikkein yleisin perusluokka. Näyttelijäluokan sinikopiot ovat yleisimmin objekteja, jotka asetetaan pelimaailmaan ja joilla on jokin ominaisuus, kuten ovi, joka aukeaa, kun pelaaja menee oven lähelle. Luokan sinikopiot voivat myös olla erittäin monimutkaisia käyttötarkoituksesta riippuen.

Pelinappula (eng. Pawn) ja hahmo (eng. Character) luokan sinikopioita käytetään silloin, kun halutaan pelaajan voivan kontrolloida jotakin objektia. Hahmo-luokka eroaa pelinappulaluokasta siten, että se sisältää valmiiksi hahmon liikuttamiseen tarvittavia funktioita kuten esimerkiksi juoksemisen ja hyppimisen.

Pelaaja-ohjain (eng. Player controller) luokka ottaa vastaan käskyjä pelaajalta, näppäimistön, hiiren tai ohjaimen välityksellä ja välittää käskyt pelattavalle hahmolle. Tekoälyohjain (eng. Ai controller) -luokka toimii käytännössä samoin kuin pelaajaohjain, mutta pelaajan sijaan se saa käskyt esimerkiksi käytöspuulta (eng. Behavior Tree).

Komponentti (eng. Component) -luokan sinikopioilla pystytään laajentamaan muiden luokkien ominaisuuksia lisäämällä kyseiseen luokkaan näyttelijäkomponentti (eng. Actor Component). Komponenttiluokan sinikopiot eivät toimi yksin vaan ne täytyy aina lisätä johonkin muuhun luokkaan.

Työssä käytetään termiä liitännäinen (eng. Plugin) kuvaamaan sen teknistä toteutusta. Pelimoottori määrittelee kuitenkin liitännäisen osana pelimoottoria, joka voidaan kytkeä päälle ja pois päältä käyttäjän halutessa [12]. Tästä syystä liitännäinen ei käytännössä ole oikea termi kuvaamaan projektin toteutusta. Koska toteutukselle ei löydy suoraa termiä, käytetään liitännäistä, jotta tekstiä on helpompi lukea ja ymmärtää. Lukijan on kuitenkin hyvä pitää mielessä, että kyseessä ei ole pelimoottorin määrittelemä liitännäinen.

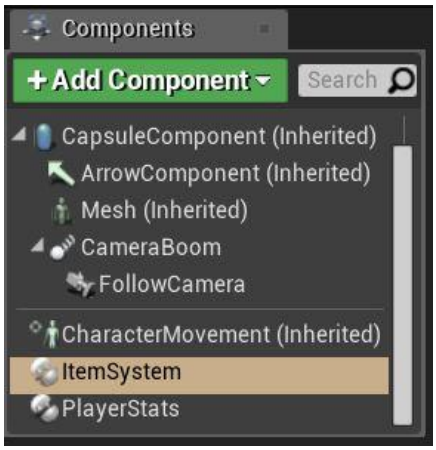
Liitännäinen koostuu kuudesta eri osasta. PlayerStats-komponentti vastaa kaikista pelaajan tilaan liittyvistä ominaisuuksista, kuten terveyspisteistä, taikapisteistä, kestävyyspisteistä, heijastusnäytöstä ja paljon muusta. ItemSyste -komponentti vastaa pelaajan inventaarion hallinnasta sekä vuorovaikutuksesta hahmojen ja esineiden kanssa. Komponentti vastaa myös tavaroiden käytöstä. CombatSystem-komponentti huolehtii taisteluun liittyvien asioiden hoitamisesta. Komponentin toteutus on vielä alussa ja tällä hetkellä komponentissa on vain kaksi osaa: aseiden ottaminen käyttöön sekä itse hyökkääminen. WeatherManager on vastuussa pelimaailman lämpötilasta sekä vuorokaudenajasta. Manageri pitää kirjaa myös pelissä kuluneesta ajasta. DialogueSystem on

vastuussa pelaajan ja NPC-hahmojen välisistä keskusteluista. QuestSystem antaa pelaajalle suoritettavia tehtäviä ja pitää kirjaa pelaajan hyväksymistä, suorittamista ja epäonnistuneista tehtävistä pelin aikana.

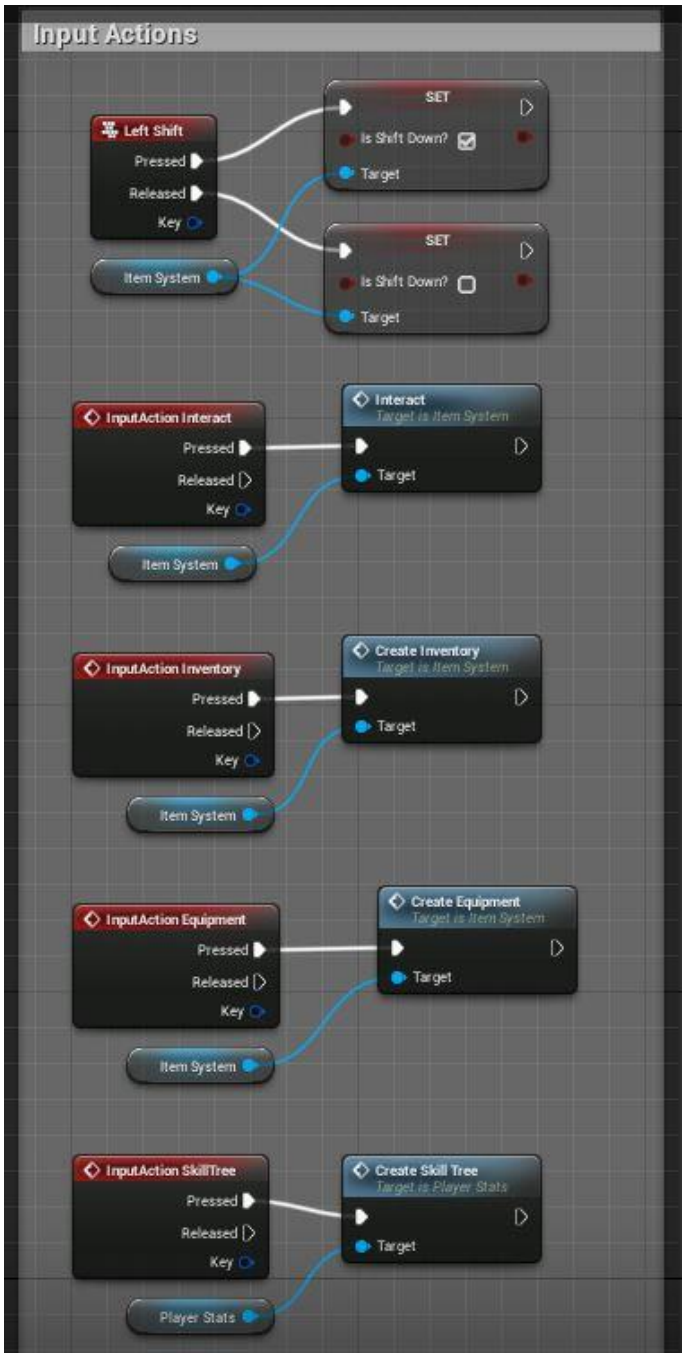
Liitännäisen eri osien käyttöönotto vaihtelee komponentista riippuen. PlayerStats-, ItemSystem- ja CombatSystem- komponentit lisätään pelattavaan hahmoon. WeatherManager lisätään haluttuun tasoon. DialogueSystem puolestaan käyttää pohjanaan NPC-hahmoja. QuestSystem taas käyttää kaikkia edellä mainittuja tapoja toimiakseen.

3.1 PlayerStats

PlayerStats-komponentti on helppo ottaa käyttöön lisäämällä se pelattavan hahmon sinikopioon kuvan 1 mukaisesti. Koska komponentit eivät voi ottaa vastaan käskyjä pelaajalta, kuten hahmoluokan sinikopio, täytyy asettaa tarvittavat komennot kutsumaan haluttuja funktioita, kuten kuvasta 2 näkyy.



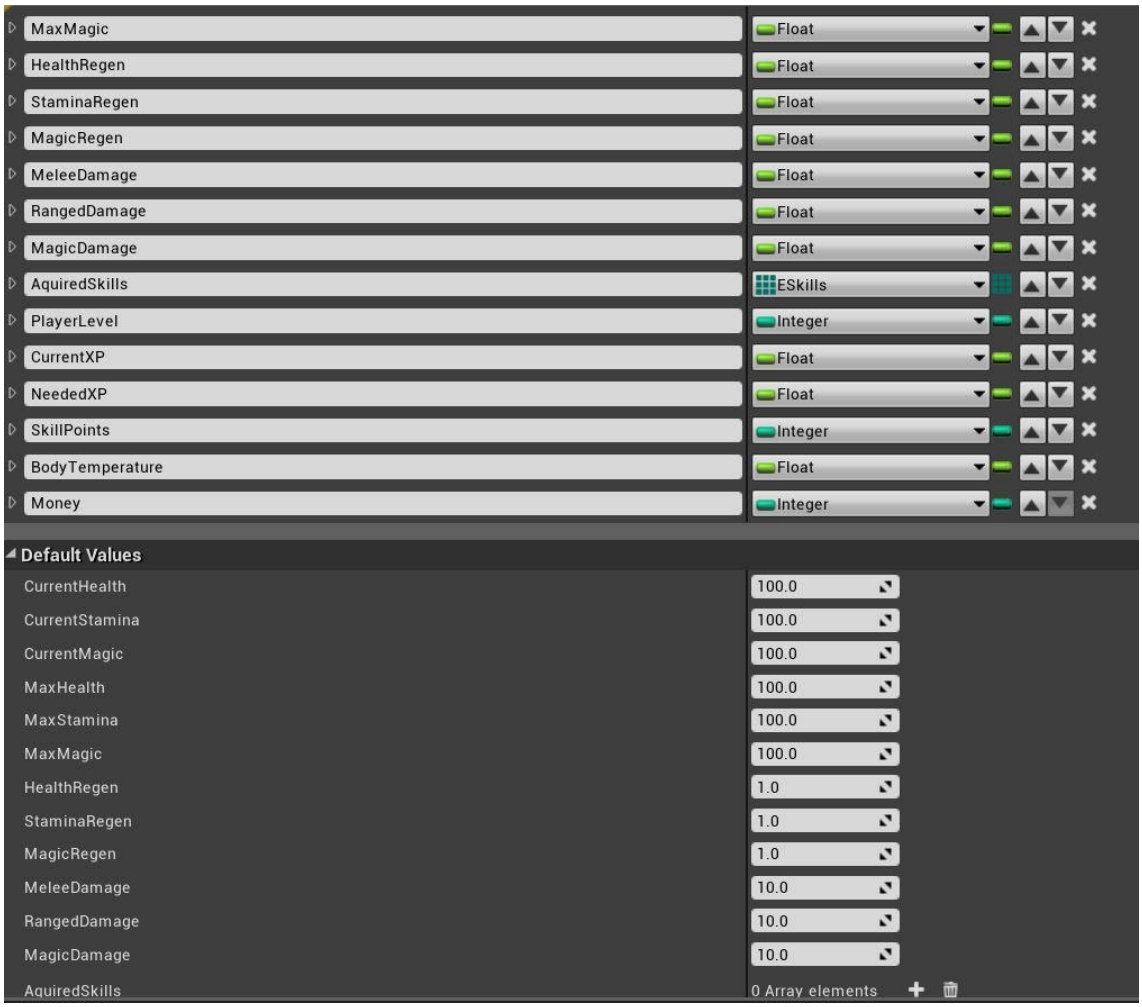
Kuva 1. Komponenttien lisäys hahmoon.



kuva 2. Pelaajan syöttämien käskyjen välitys komponenteille.

PlayerStats-komponentti sisältää yksinkertaiset toteutukset mm. pelaajan terveys-, taika-, kestävyyspisteiden lisäämiseen ja vähentämiseen, kokemuspisteiden sekä pelaajan tason lisäämiseen sekä taitopuun, josta pelaaja voi avata hahmolleen uusia kykyjä. Komponentilla ohjataan myös WeatherManager-systeemin vaikutuksia pelaajaan, mikäli niin halutaan.

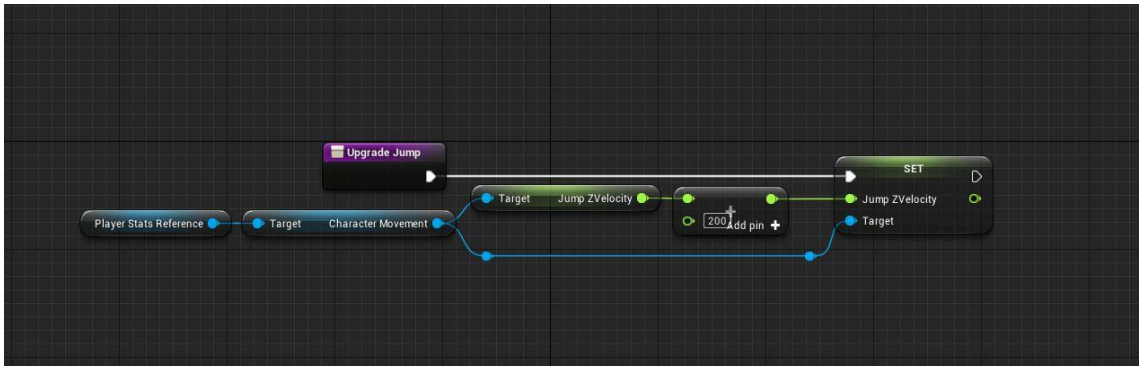
Pelaajan ominaisuudet kuten terveys-, taika-, kestävyyspisteet on kerätty kuvan 3 mukaisesti rakenne (eng. Structure) -luokan sinikopioon, jota muokkaamalla voidaan muokata pelaajalle haluttuja ominaisuuksia. Pelaajan eri pisteitä lisätään ja vähennetään komponentista löytyvien funktioiden avulla, joita kutsutaan halutuissa tilanteissa.



Kuva 3. Pelaajan sisältämät ominaisuudet.

Taitopuu avaa taitopisteitä vastaan uusia kykyjä pelaajan käyttöön, kuten kuvasta 4 näkyy. Puuta voidaan muokata lisäämällä ja poistamalla haluttuja painikkeita. Painikkeille määritetään halutut toiminnot, kuten kuvasta 5 nähdään.

Kuva 4. Pelaajan taitopuu.

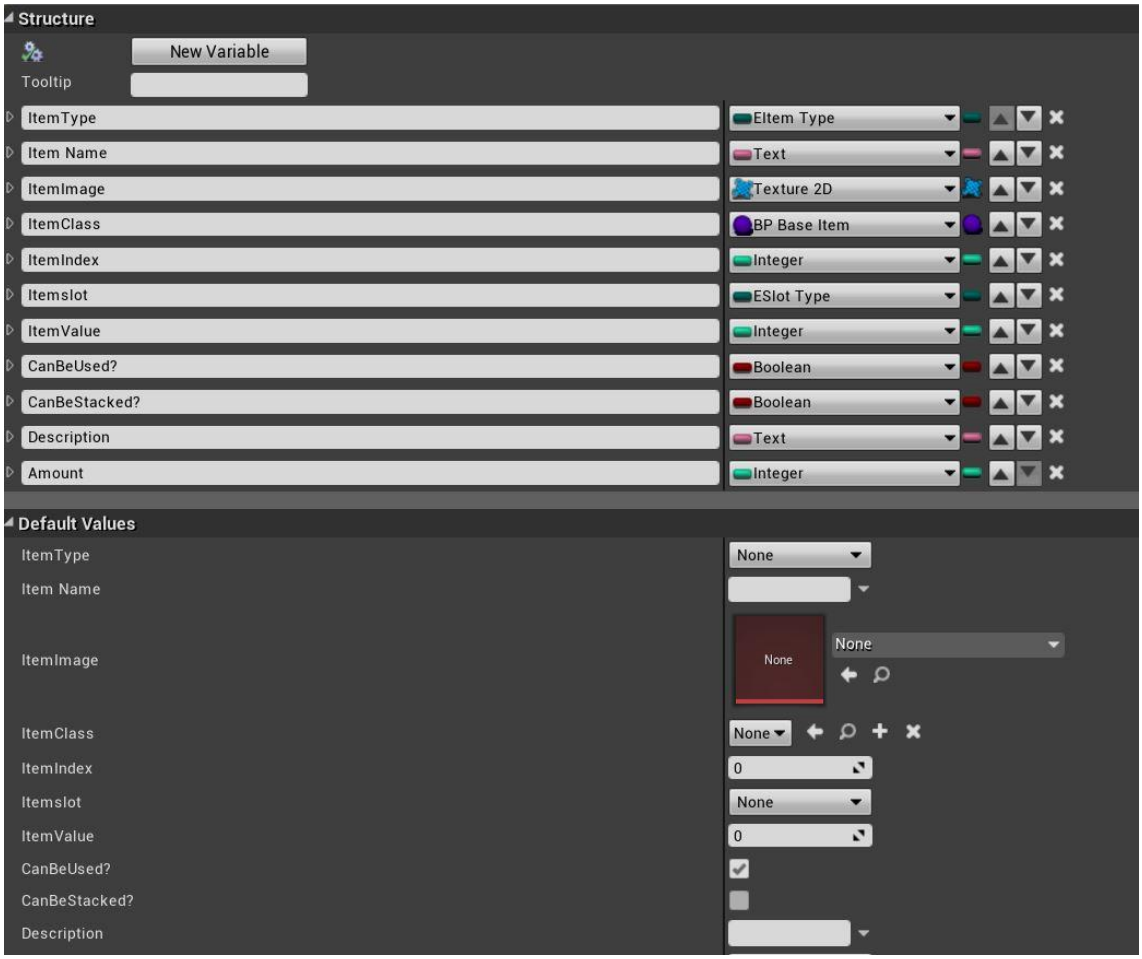


Kuva 5. Taitopuun ImprovedJump-kyvyn toteutus.

3.2 ItemSystem

ItemSystem-komponentti otetaan käyttöön samalla tavalla kuin PlayerStats-komponentti lisäämällä se hahmoon ja määrittämällä tarvittavat komennot. ItemSystem-komponentti sisältää toteutukset esineiden poimimiseen, pelaajan hallussa olevien tavaroiden hallintaan sekä tavaroiden ostamiseen.

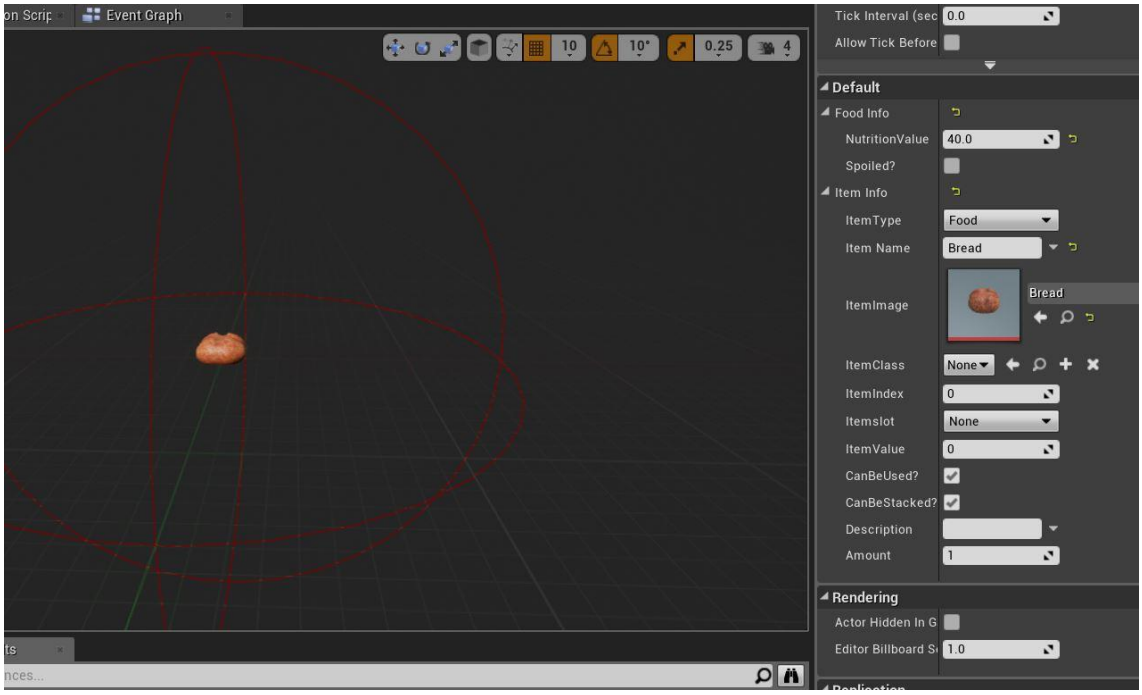
Esineiden poimimiseen tarvitaan kaksi asiaa: ItemSystem komponentti ja poimittavat esineet. Poimittavia esineitä on helppo luoda tekemällä BaseItem-sinikopiosta lapsia ja määrittämällä niille halutut ominaisuudet. BaseItem-sinikopio sisältää rakenneluokan muuttujan nimeltään ItemInfo. Tämä rakennesinikopio sisältää esineiille määritettyjä ominaisuuksia kuten tyyppi, nimi, luokka ja arvo, kuten kuvasta 6 nähdään.



Kuva 6. Esineille määritettävät ominaisuudet.

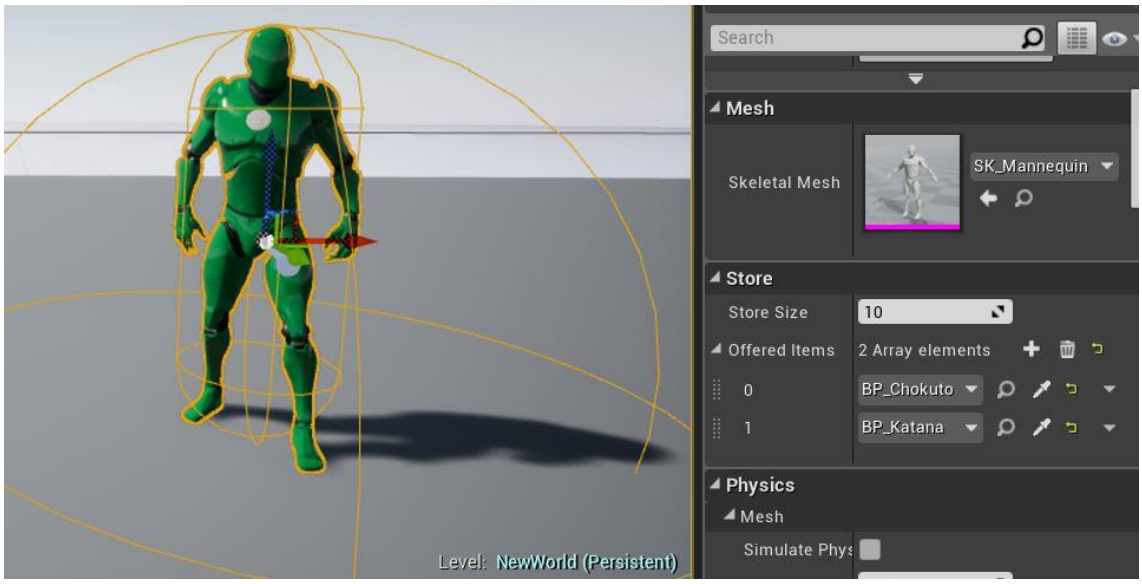
Esineet voivat niiden tyypistä riippuen sisältää lisää myös vain niille tarkoitettuja ominaisuuksia, kuten aseilla olevat WeaponInfo-sinikopion sisältämät ominaisuudet, joita ei löydy esimerkiksi syötäviltä esineiltä. WeaponInfo toimii samoin kuin aikaisemmin esitetty ItemInfo ja sitä voidaan muokata sellaiseksi kuin halutaan. Erilaisia esinetyyppejä voidaan luoda lisää helposti sitä mukaan, kun niitä tarvitaan tekemällä uusi rakennesinikopio ja määrittelemällä sinne halutut ominaisuudet.

Kun halutaan tehdä poimittava esine, kuten esimerkiksi leipä, jonka voi syödä, valitaan BaseItem-sinikopiosta tehty lapsi nimeltään BaseConsumable. Nimensä mukaan tästä sinikopiosta tehdyt lapset on tarkoitettu syötäviksi esineiksi. Tehdään siitä uusi lapsi ja annetaan sille halutut ominaisuudet sekä StaticMesh, joka näkyy pelaajalle pelimaailmassa, kuten kuvassa 7 näkyy. Kun tämä leipä sijoitetaan pelimaailmaan, pelaaja voi sen poimia.



Kuva 7. Poimittavan leivän ominaisuudet.

ItemSystem-komponenttiin kuuluva kauppa on vielä kesken. Tällä hetkellä pelaaja pystyy ostamaan uusia esineitä, mutta ei pysty vielä myymään omistamiaan esineitä. Myyjän lisääminen peliin tapahtuu tekemällä BaseMerchant-sinikopiosta lapsen, ja antamalla sille halutut esineet myyntiin, kuten kuvasta 8 näkyy. Myynnissä olevat esineet ovat samoja esineitä kuin maailmasta poimittavat esineet.



Kuva 8. Myytävien esineiden asettaminen.

Pelaajan omistamat esineet näkyvät pelaajan inventaariossa. Valitsemalla esineen Inventaarioikkunasta pelaaja pystyy käyttämään esineitä. Esineen tyyppi määrittää, mitä tapahtuu pelaajan käyttäessä esinettä. Esimerkiksi ruokaa syömällä saa palautettua elämäpisteitä. Aseita ja suojia voi puolestaan ottaa pelaajan käyttöön, jolloin ne lisätään pelaajan varusteisiin, kuten kuvasta 11 näkyy. Pelaaja voi myös halutessaan heittää pois omistamiaan esineitä, jolloin ne tiputetaan takaisin pelimaailmaan.

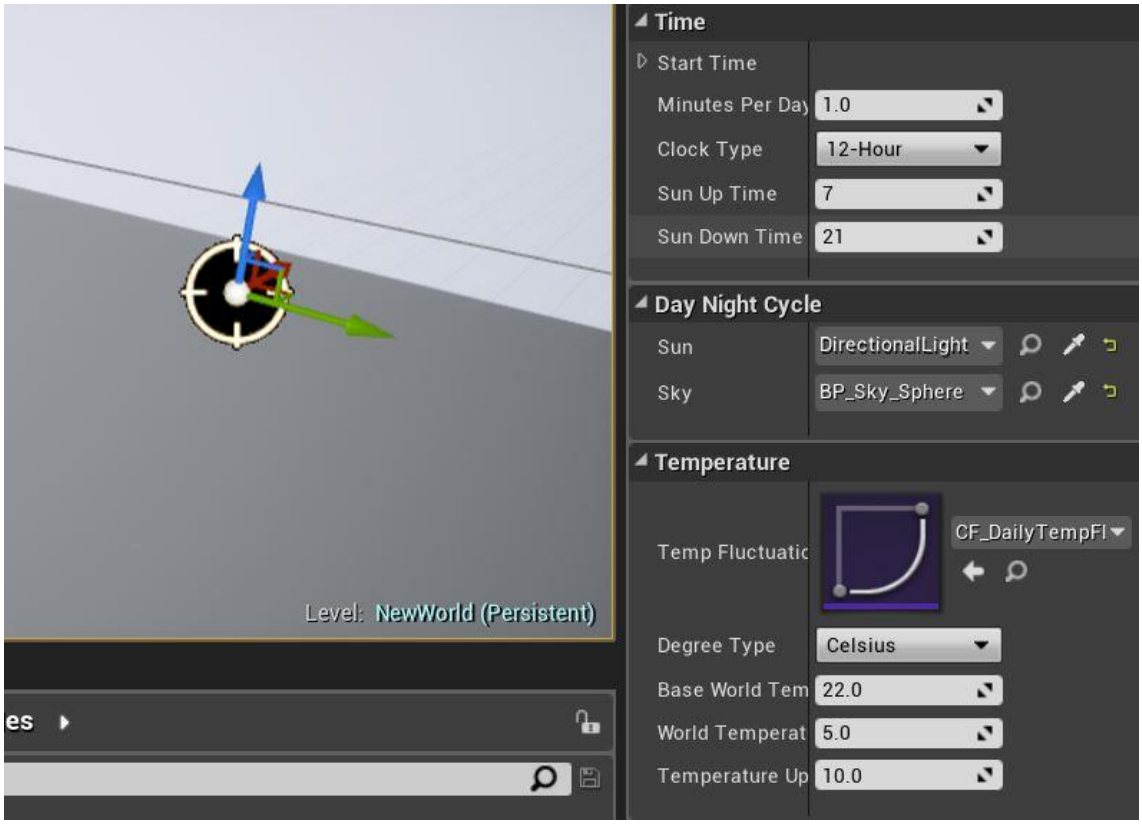
Kuva 11. Pelaajan Inventaario- ja varusteet-ikkunat.

3.3 CombatSystem

CombatSystem komponentti huolehtii taisteluun liittyvien asioiden hoitamisesta, kuten taisteluun liittyvän logiikan ja animaatioiden pyörittämisestä sekä vastustajalle vahingon tekoon liittyvistä laskelmista. Komponentin toteutus on vielä alussa ja tällä hetkellä komponentissa on vain kaksi osaa aseiden ottaminen käyttöön sekä itse hyökkääminen. Komponentin toteutus ja toiminta ovat suurelta osin kiinni hyökkäykseen liittyvistä animaatioista ja niiden puute on yksi syy, miksi komponentin toteutus jäi kesken.

3.4 WeatherManager

WeatherManager on vastuussa pelimaailman kellonajan sekä lämpötilan laskemisesta. Ne voidaan näyttää myös pelaajalle, mikäli niin halutaan esimerkiksi heijastusnäytössä. WeatherManager otetaan käyttöön yksinkertaisesti asettamalla sinikopio pelimaailmaan ja säätämällä sen asetukset halutunlaisiksi, kuten kuvassa 12 näkyy.



Kuva 12. WeatherManagerin asetukset.

Pelin aika on täysin käyttäjän säädettävissä. Käyttäjä voi säätää aloitusajan sekä päivän pituuden ja auringon nousu- ja laskuajan. Jotta auringon paikka ja taivas päivittyisivät, ne täytyy määrittää käyttämään maailmassa olevaa valonlähdettä sekä taivaskuplaa. Pelimaailman lämpötila voidaan asettaa haluttuun arvoon, ja sille voidaan antaa tietyt rajat, joiden sisällä se voi vaihdella. Lämpötila voidaan asettaa myös muuttumaan vuorokaudenajan mukana.

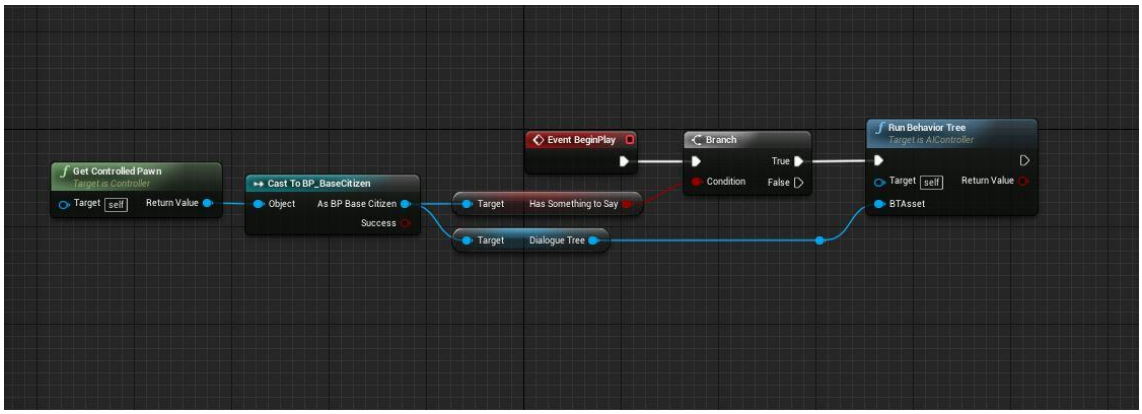
Lämpötilaa voidaan muokata halutuissa paikoissa WeatherModifier-sinikopion avulla. Se sisältää kaksi eri asetusta. Ensimmäinen muuttaa maailman lämpötilaa jonkin verran ja toinen kumoaa maailman lämpötilan sekä asettaa sen haluttuun arvoon. Esimerkkinä on nuotio, joka lämmittää ilmaa sen ympärillä jonkin verran, sekä talo, jonka sisällä on lämmin eikä lämpötila vaihtele vuorokaudenajan mukana.

3.5 DialogueSystem

Koska dialogi on keskeinen osa RPG-genren pelejä, sen toteutukseen kuluu huomattavasti aikaa. Tästä syystä tulin siihen lopputulokseen, että koko systeemin rakentaminen nollasta veisi liian kauan. Vastauksena ongelmaan löysin valmiin dialogisysteemiliitännäisen, joka on MIT vapaan lähdekoodin lisenssin alla [13; 14]. Tämä tarkoittaa, että liitännäistä voi käyttää ja muokata vapaasti, halutulla tavalla.

Systeemi laajentaa pelimoottorin AI hahmoille tarkoitettua käytöspuusinikopiota. Systeemiä käytetään lisäämällä NPC-hahmolle DialogueController-sinikopio ja käytöspuu, johon on lisätty halutut dialogit.

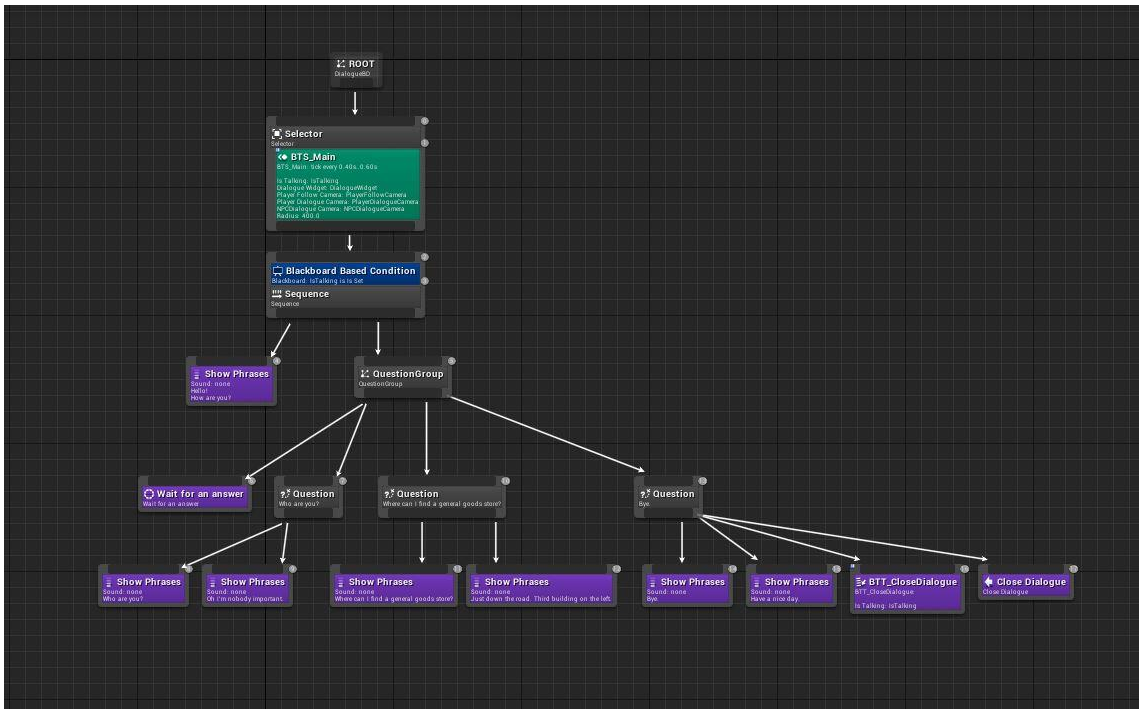
DialogueCntroller on käytännössä pelimoottorista löytyvä tekoälyohjainluokan sinikopio. Luokan sinikopioita käytetään NPC-hahmojen ohjaukseen. Tässä tapauksessa sinikopio huolehtii, että NPC-hahmon sisältämä dialogi käytöspuu lähtee käyntiin, kuten kuvasta 13 näkyy.



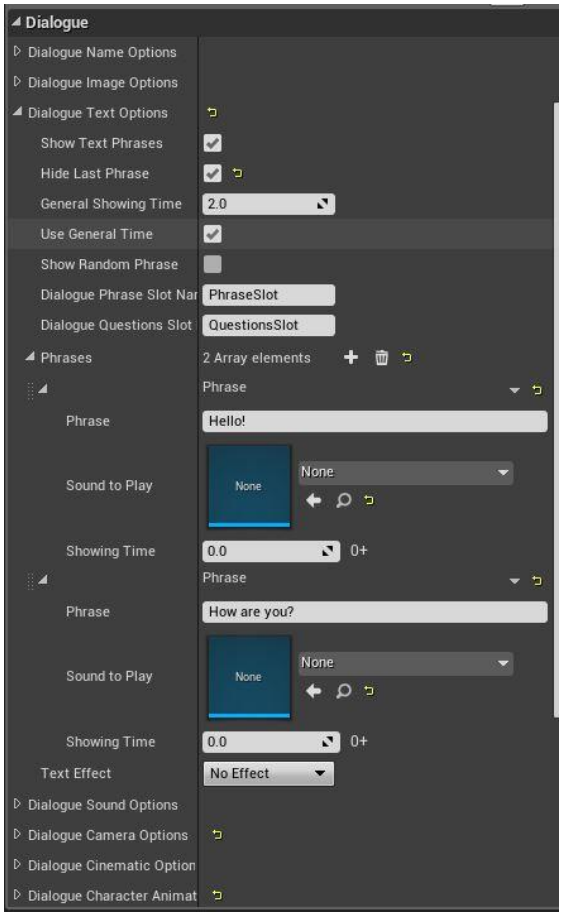
Kuva 13. DialogueController-sinikopion BeginPlay-tapahtuma.

Jotta dialogit sisältävä käytöspuu saadaan toimimaan kunnolla, tarvitaan liitutaulu (eng. Blackboard) -luokan sinikopio, joka sisältää muuttujat, joita käytöspuu käyttää. Jotta liitutaulu sinikopion muuttujat saadaan käyttöön, tarvitaan käytöspuuyhteys (eng. BTService) -luokan sinikopio. Tämä sinikopio on vastuussa liitutaulusinikopion sisältämien muuttujien arvojen asettamisesta ja muuttamisesta.

Käytöspuu sinikopio vastaa dialogin sisällöstä ja sen eteenpäin viemisestä. Käyttäjä määrittää halutut dialogit sekä asetukset dialogin esittämiselle sekä eteenpäin viemiselle, kuten kuvasta 14 nähdään. Aluksi asetetaan käytöspuu käyttämään liitutaulua ja käytöspuu yhteyssinikopioita. Dialogin lisääminen tapahtuu lisäämällä ShowPhrases-solmuja. Kuten kuvasta 15 nähdään, solmut sisältävät monia asetuksia, kuten puhujan nimi ja kuva, miten kauan dialogia näytetään, kameran käytös, käytettävät animaatiot, soitettavat ääniefektit yms. jotka käyttäjä voi määrittää. Dialogin etenemistä kontrolloidaan erilaisilla solmuilla, kuten kuvasta 14 nähdään.

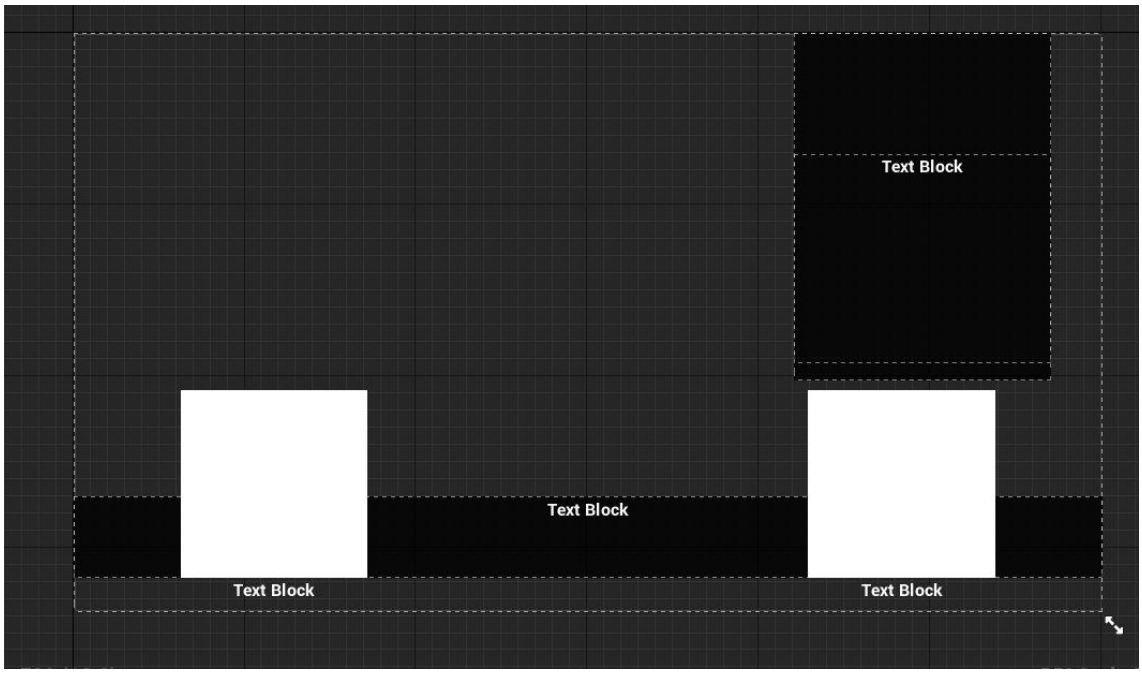


Kuva 14. Dialogi käytöspuu.



Kuva 15. Näytettävän dialogin asetukset.

Jotta dialogi saadaan näkyviin pelaajalle, tarvitaan kuvan 16 mukainen DialogueWidget-sinikopio. Sinikopion tekstejä ja kuvia ohjataan käytöspuun ShowPhrases-solmun asetuksilla.



Kuva 16. DialogueWidget-sinikopion ulkoasu.

Systeemin laajentaminen ja muokkaaminen on vielä kesken. Tällä hetkellä dialogin aloittamiseen käytetään ItemSystem-komponentin sisältämää interaktiologiikkaa.

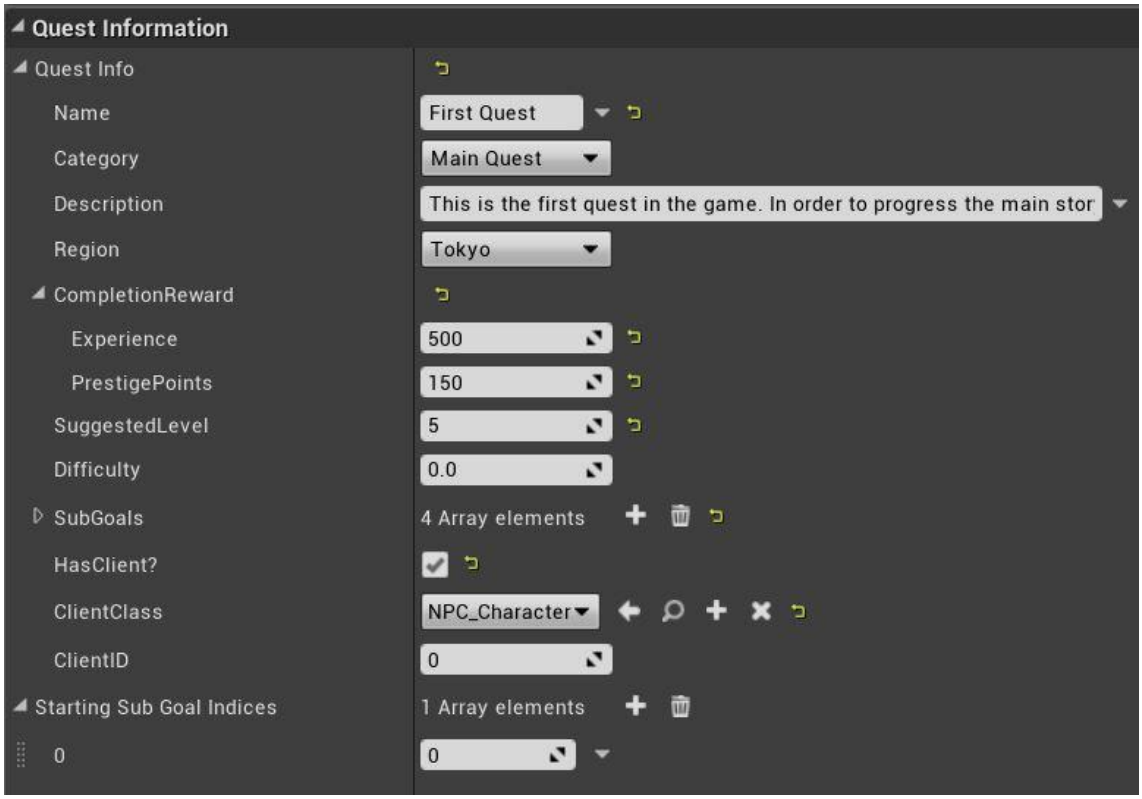
3.6 QuestSystem

QuestSystem on yksi liitännäisen suurimpia osia. Ajan puutteen takia sen toteutus on vielä kesken. Tällä hetkellä systeemi toimii ainoastaan täysin itsenäisesti eikä sitä ole vielä ehditty integroimaan liitännäisen muihin osiin. Systeemi koostuu kolmesta osasta, joita ovat QuestManager, MasterQuest ja QuestCharacter. QuestManager vastaa tehtävien antamisesta sekä päivittämisestä. MasterQuest-sinikopio vastaa tehtävien sisällöstä, ja pelaajalle annettavat tehtävät ovat sen lapsia. QuestCharacter on pelattava hahmo, jonka avulla pelaaja kommunikoi systeemin muiden osien välillä.

QuestSystem otetaan käyttöön lisäämällä pelattavan hahmon BeginPlay-tapahtumaan SpawnActor-solmu. Solmun kohteeksi valitaan QuestManager-sinikopio. Tämä solmu

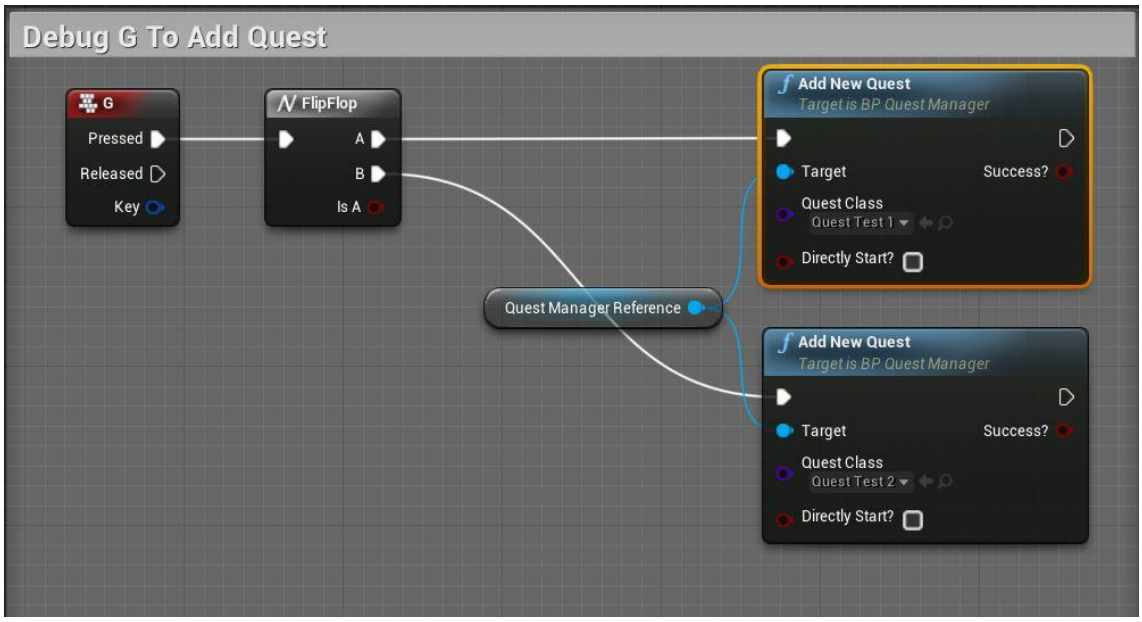
luo pelin alkaessa pelimaailmaan QuestManager-sinikopion, joka sen jälkeen huolehtii tehtäviin liittyvän logiikan pyörittämisestä. Jotta QuestManager voi antaa pelaajalle tehtäviä suoritettavaksi, ne pitää ensin tehdä.

Uusia tehtäviä tehdään tekemällä MasterQuest-sinikopiosta lapsi ja asettamalla sen asetukset, kuten kuvasta 17 nähdään. Tehtävien asetukset löytyvät QuestInfo-nimisestä rakennesinikopiosta, jota muokkaamalla asutusten sisältöä voi muokata halutunlaiseksi. Tällä hetkellä systeemissä on kaksi erityyppistä tehtävää, päätehtäviä ja sivutehtäviä. Tehtävien tavoitteet vaihtelevat metsästys-, etsintä- ja puhetehtävien välillä. Tehtävän tyypistä riippuen tehtävälle voi antaa erilaisia onnistumis- ja epäonnistumisehtoja, esimerkiksi metsästystehtävälle annetaan vihollisen tyyppi ja metsästettävä lukumäärä.

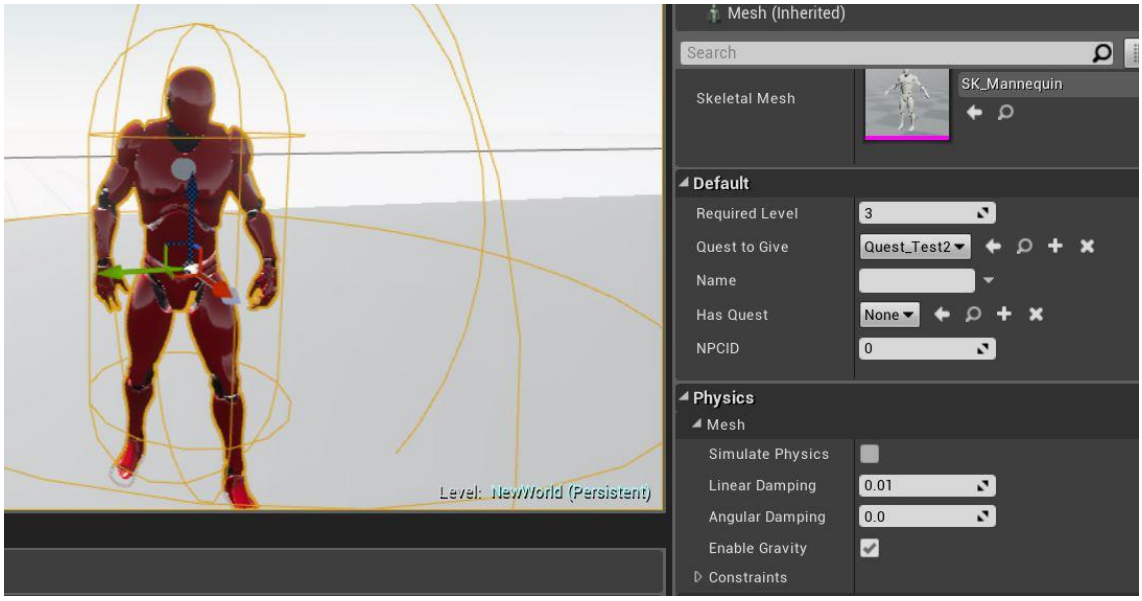


Kuva 17. Tehtävän asetukset.

Tehtävien antaminen pelaajalle toimii kutsumalla QuestManagerin sisältämää AddNewQuest-funktiota, kuten kuvasta 18 nähdään. Funktiota voidaan esimerkiksi kutsua, kun pelaaja puhuu tietylle NPC-hahmolle. Kuten kuvasta 19 nähdään, tehtävien antamiselle voidaan asettaa myös ehtoja, kuten että pelaajan tason tulee olla yli tietyn arvon.

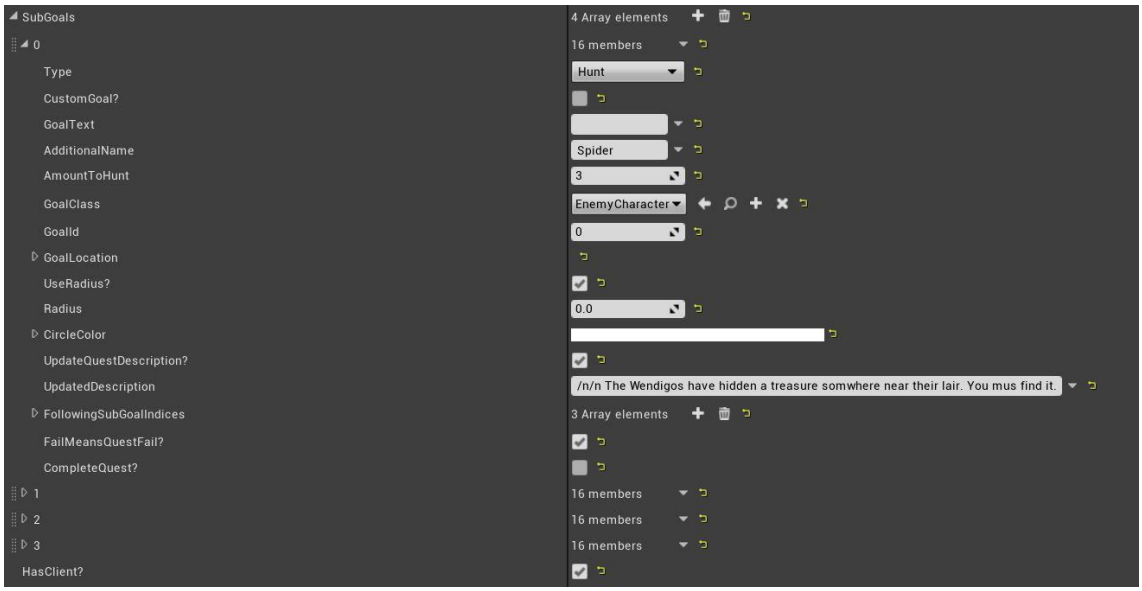


Kuva 18. Uuden tehtävän antaminen pelaajalle AddNewQuest-funktiolla.



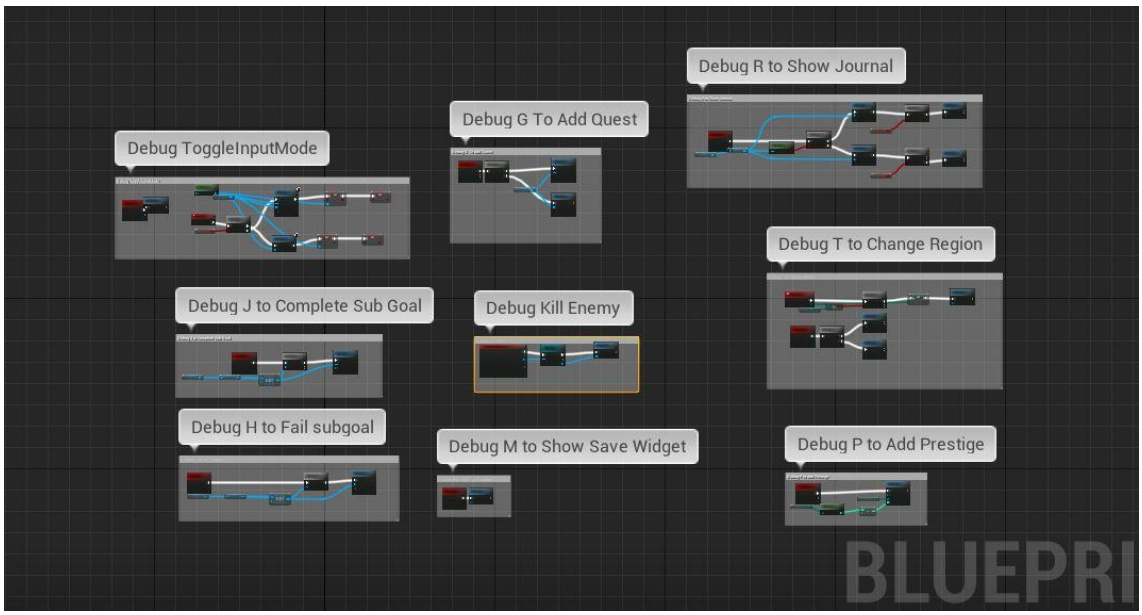
Kuva 19. Tehtävän saamisen ehdot.

Tehtävien sisältämien tavoitteiden läpäisemisen ehdot voidaan asettaa tehtävän asetuksissa. Samalla voidaan asettaa, päästäänkö tehtävä läpi suorittamalla kyseinen tavoita, ja epäonnistuuko tehtävä, jos kyseinen tavoite epäonnistuu. Tehtäville voi asettaa niin monta tavoitetta kuin haluaa. Jokainen tavoite asetetaan erikseen tehtävän asetuksista, kuten kuvasta 20 nähdään.



kuva 20. Tehtävien sisältämien tavoitteiden asettaminen.

Tällä hetkellä systeemi on vielä kesken, ja kaikki sen toiminnot toimivat QuestCharacter-hahmoon asetettujen debug-näppäinten avulla tai törmäyspallo (eng. Sphere collision) -komponentin avulla, kuten kuvasta 21 nähdään. Käytettäessä systeemiä käyttäjä voi itse määritellä, mistä ja milloin mitäkin ominaisuutta käytetään.



Kuva 21. Pelaajahahmon sisältämät debug-tapahtumat.

Kuten aikaisemmin mainittiin, liitännäisen kaikki osat ovat hyvin alkeellisia, niin sanottuja luurankototeutuksia. Niillä on toteutettu vain muutamia perustoimintoja näyttämään, miten niitä käytetään ja mitä mikäkin osa tekee. Tarkoitus on, että käyttäjä muokkaa ja laajentaa itse liitännäisen eri osia sopimaan omiin käyttötarkoituksiinsa.

4 Liitännäisen toteutus

4.1 Suunnittelu

Tutustuessa eri opetusvideoihin alkoi muodostua kuva ongelmasta kaikissa opetusvideoissa. Kaikki toteutukset tehtiin tietylle hahmolle, eli ne olivat sidottuja yhteen Character-luokan sinikopioon. Jos pelissä olisi useampi hahmo tai tarkoitus olisi tehdä useampi prototyyppi, joudutaan hahmo luomaan joka kerta uudestaan. Hahmo pystytään toki helposti kopioimaan uudeksi hahmoksi, mutta tällöin kopioituvat kaikki ominaisuudet. Jos uusi hahmo ei tarvitse kaikkia samoja ominaisuuksia, uusi hahmo pitää käydä läpi ja poistaa kaikki turhat funktiot, muuttujat ja muut osat, joita ei tarvita.

Tutustuessa aikaisemmin Advanced Locomotion System -liitännäiseen, tutustuttiin sinikopio luokkaan nimeltä näyttelijäkomponentti [15]. Tätä sinikopiota käyttämällä on helppo jakaa samoja ominaisuuksia useiden eri sinikopioiden välillä. Tästä saatiin idea käyttää näyttelijäkomponenttisinikopiota saamaan samat ominaisuudet helposti käytettäviksi juuri sinne, missä niitä tarvitsin.

Edelleen ongelmana olivat niin sanotut turhat ominaisuudet. Kaikki hahmot eivät välttämättä tarvitse kaikkia suunnittelemiani ominaisuuksia. Esimerkiksi NPC-hahmojen ei tarvitse osata poimia esineitä, mikä taas on erittäin oleellinen ominaisuus pelaajalle. Päätettiin ratkaista ongelma jakamalla ominaisuudet eri kategorioihin. Näin saatiin kolme kategoriaa: PlayerStats, ItemSystem ja CombatSystem, joista jokainen vastaa tietyn alueen toteutuksesta. Nyt pystyttiin helposti jakamaan eri hahmoille niiden tarvittavat ominaisuudet joutumatta muokkaamaan hahmoja sen enempää.

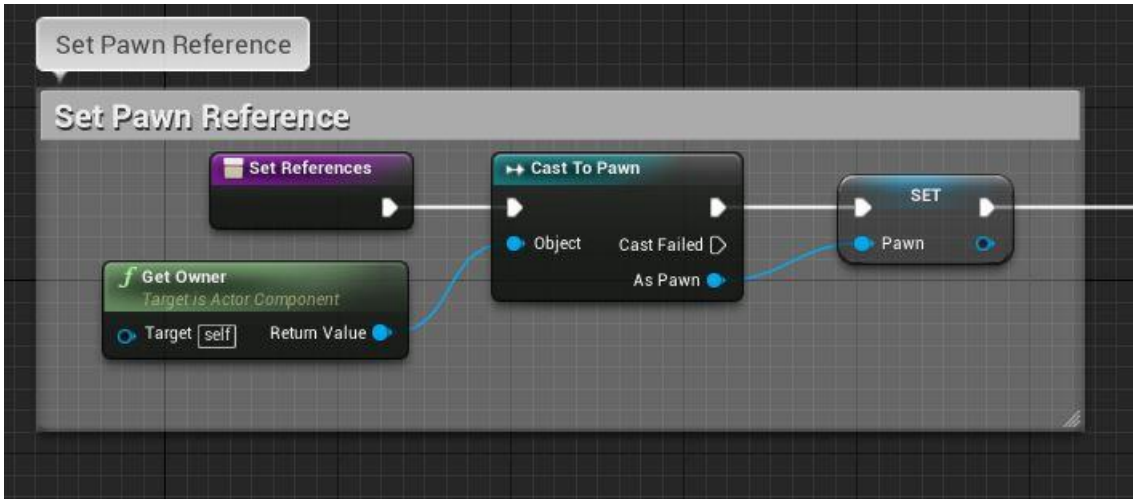
4.2 Player Stats

PlayerStats-komponentti vastaa kaikista pelaajan tilaan liittyvistä ominaisuuksista, kuten terveyspisteistä, taikapisteistä, kestävyyspisteistä, heijastusnäytöstä ja paljon muusta.

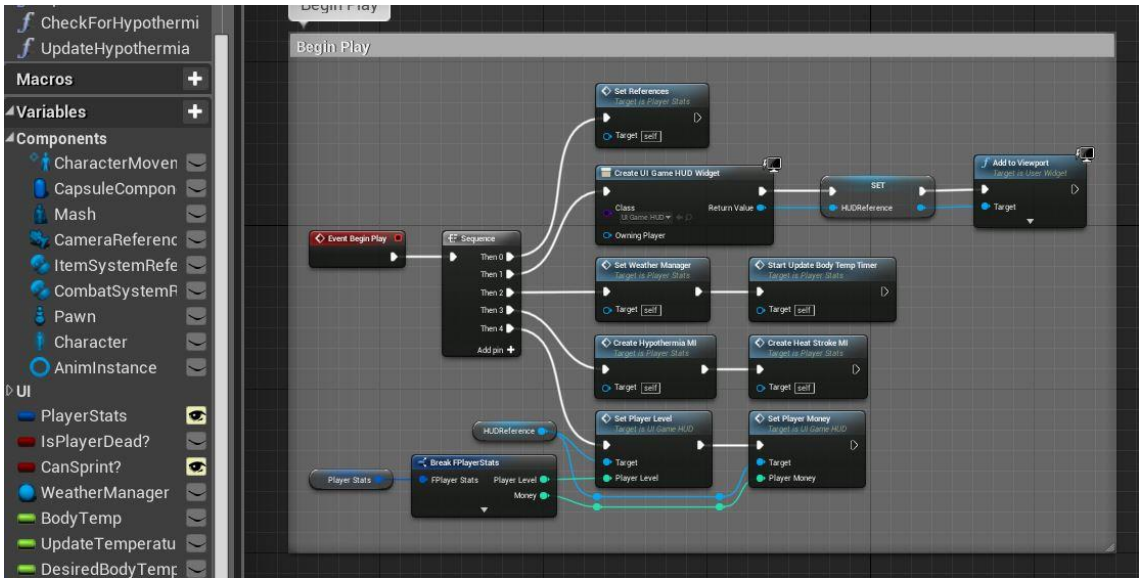
Komponentin tärkein osa on SetReferences-funktio. Tämä funktio vastaa siitä, että komponentin käytettävissä ovat kaikki sen tarvitsemat viittaukset. Tärkein näistä on kuvassa 22 esitetty pelinappulaviittaus. Viittaus vastaa siitä, että komponentti tietää, mille objektille se kuuluu eli kuka sen ”isäntä” on. Pelinappulaviittauksen avulla päästään käsiksi isännän sisältämiin ominaisuuksiin ja niitä voidaan muokata halutulla tavalla.

Muita tarvittavia viittauksia ovat hahmo, hahmon liike (eng. CharacterMovement), kapseli (eng. Capsule), SkeletalMesh, animaatio-ohjain (eng. Anim Instance) sekä kamera. Myös viittaukset liitännäisen muihin osiin ItemSystem ja CombatSystem asetetaan tässä funktiossa. Kaikki viitteet tallennetaan muuttujiksi, jotta niitä on helppo käyttää tarvittaessa. Kuten kuvasta 23 näkyy, SetReferences-funktiota kutsutaan ja tarvittavat muuttujat tallennetaan ensimmäisenä pelin alkaessa.

Muilta osin komponentin toteutus vastaa hyvin pitkälti samaa kaavaa, jota ominaisuuksien toteutus vastaisi, mikäli ne tehtäisiin perinteisesti tiettyyn hahmoon sitoen. Erona on se, että isännän ominaisuuksia muokatessa käytetään viitemuuttujia niiden suoran muokkauksen sijaan. Ominaisuudet on toteutettu hyvin pitkälti Titanic Gamesin RPG Tutorial Series [16.] -ohjeiden mukaisesti. Yhtenä erona lähdemateriaalin kanssa on se, että pelaajaan liittyvät muuttujat on tallennettu rakenneluokan sinikopioon, josta niitä haetaan tarvittaessa eikä suoraan PlayerStats-komponenttiin.

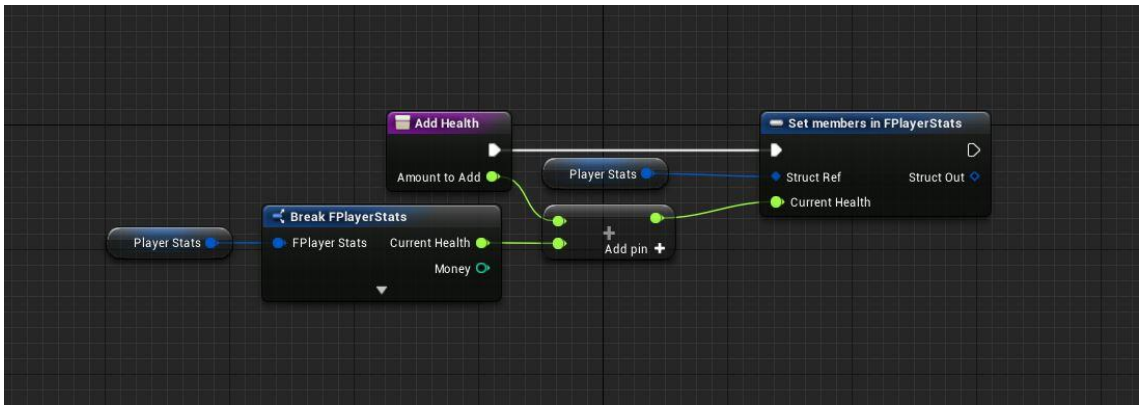


Kuva 22. Pawn-viitteen asettaminen.



Kuva 23. PlayerStats-komponentin BeginPlay-tapahtuman toteutus sekä tallennetut viitemuuttujat.

Pelaajan terveys-, taika-, kestävyys- ja kokemuspisteitä sekä pelaajan tasoa muokataan hyvin yksinkertaisilla funktioilla, jotka ottavat sisään halutun arvon ja lisäävät tai vähentävät sen pelaajan pisteistä. Kuvassa 24 lisätään haluttu määrä pelaajan elämäpisteisiin. Muut edellä mainitut funktiot toimivat hyvin samalla tavalla. Näitä funktioita voidaan sitten kutsua tarvittaessa.

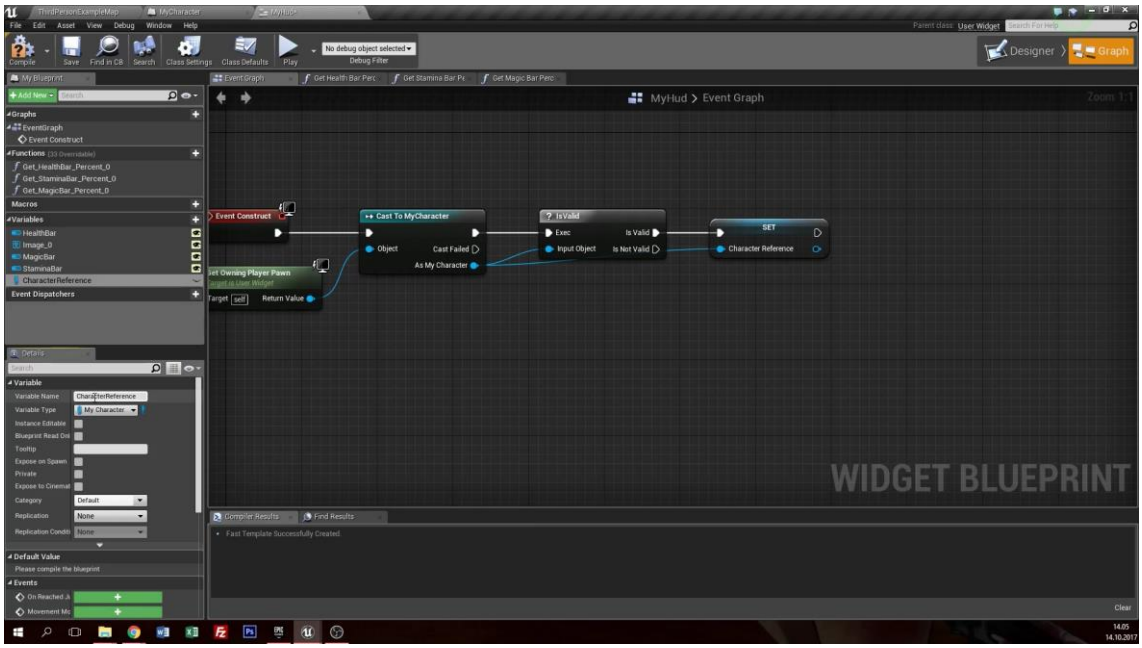


Kuva 24. AddHealth-funktion toteutus.

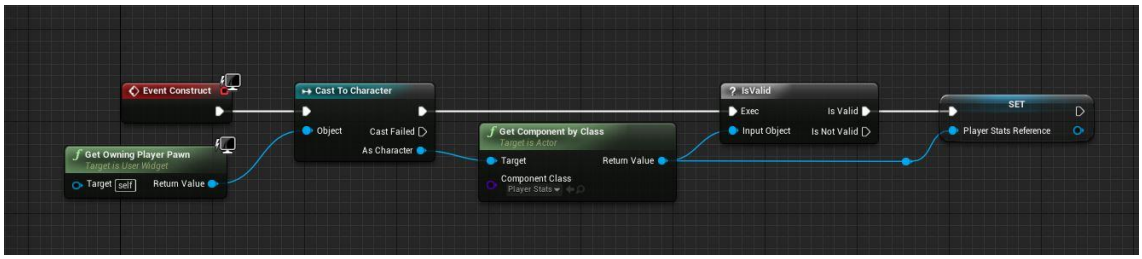
PlayerStats-komponentti vastaa myös WeatherManager-systeemin vaikutuksista pelaajan tilaan. Eri funktiot säätelevät pelaajan ruumiinlämpöä ja vastaavat esimerkiksi hypotermian tai lämpöhalvauksen aiheuttamista oireista. WeatherManager-systeemi ja sen

vaikutukset pelaajaan on toteutettu pitkälti Titanic Gamesin Survival Game Tutorial Series [17.] ohjeiden mukaan. WeatherManager-systeemiä ja sen ominaisuuksia tarkastellaan myöhemmin.

PlayerStats-komponentin sisältämän heijastusnäytön toteutuksessa on hyvä myös huomioida muutama kohta. Normaalisti haetaan heijastusnäyttöä tehdessä hahmo, jolle heijastusnäyttö kuuluu. Tämä hahmo tallennetaan viitemuuttujaksi, jota voidaan käyttää myöhemmin kuvan 25 mukaisesti. Komponentin kanssa perusidea on sama, mutta hahmon sijaan viitteeksi tallennetaan PlayerStats-komponentti. Kuvasta 26 näkyy, miten hahmosta haetaan sen sisältämä komponentti ja tarkistetaan, että se varmasti löytyy.



Kuva 25. Hahmon tallentaminen viite muuttujaksi heijastusnäytössä.

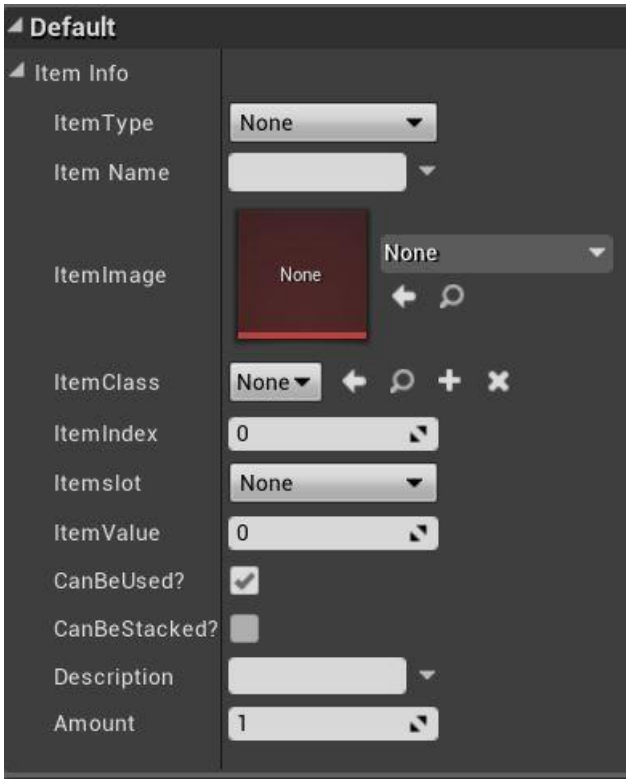


Kuva 26. PlayerStats-komponentin tallentaminen viite muuttujaksi heijastusnäytössä.

4.3 Item System

ItemSystem-komponentti vastaa pelaajan inventaarion hallinnasta sekä vuorovaikutuksesta hahmojen ja esineiden kanssa. Komponentti vastaa myös tavaroiden käytöstä. Komponentti käyttää samoja viitemuuttujia kuin PlayerStats-komponentti ja muuttujat asetetaan samoin kuin PlayerStats-komponentissa.

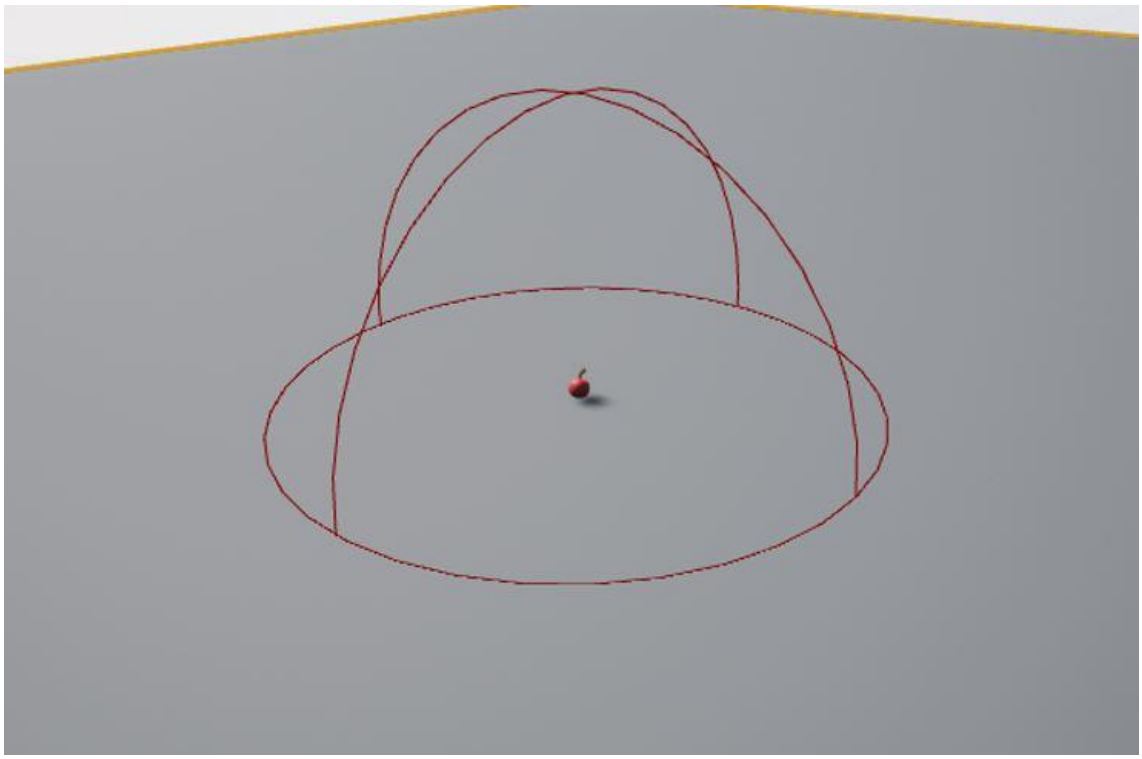
Tavaroiden poimiminen ja hahmojen kanssa puhuminen on toteutettu Titanic Gamesin RPG Tutorial Series [16.] -ohjeiden mukaisesti. Poimittavilla esineillä on niille määritettyjä ominaisuuksia niiden tyypistä riippuen. Kuten kuvasta 27 näkyy, näitä ominaisuuksia ovat mm. nimi, tyyppi ja arvo. Aseilla ja suojuksilla on tämän lisäksi niille ominaisia arvoja kuten hyökkäysvoima, kantomatka, hyökkäysnopeus ja suojauksen luokitus.



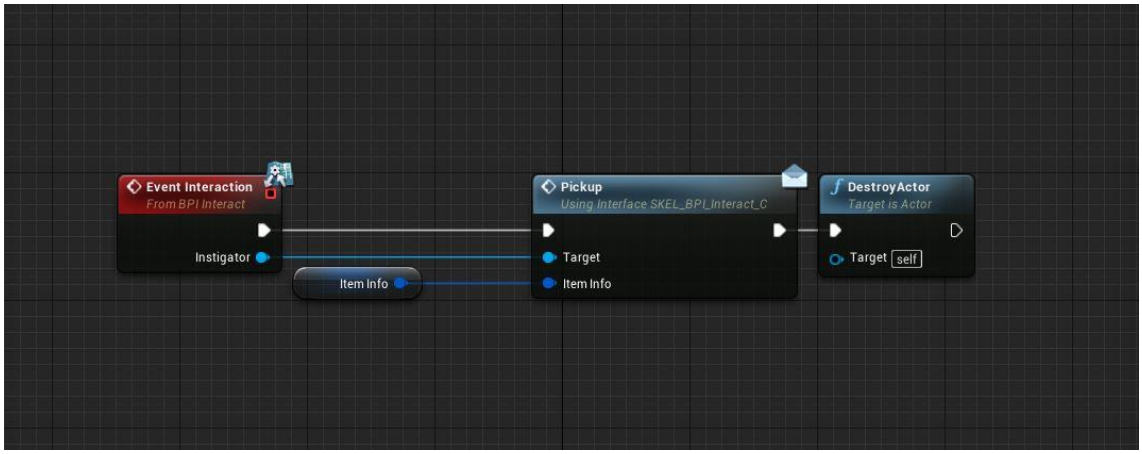
Kuva 27. Esineiden ominaisuuksien asetukset.

Kaikilla poimittavilla esineillä on kuvassa 28 näkyvä törmäyspallokomponentti, joka rekisteröi sen sisään astuvat hahmot. Mikäli hahmo on pelaaja, esine lähettää itsestään signaalin ItemSystem-komponentille, joka rekisteröi sen. Kun pelaaja haluaa poimia esi-

neen, lähetetään esineelle interaktioviesti. Esine reagoi viestiin sille annettujen ominaisuuksien mukaisesti. Yleisimmin lähettämällä tiedot itsestään takaisin ItemSystem-komponentille ja sen jälkeen tuhoamalla itsensä, kuten kuvasta 29 näkyy.



Kuva 28. Esineiden interaktioalue.



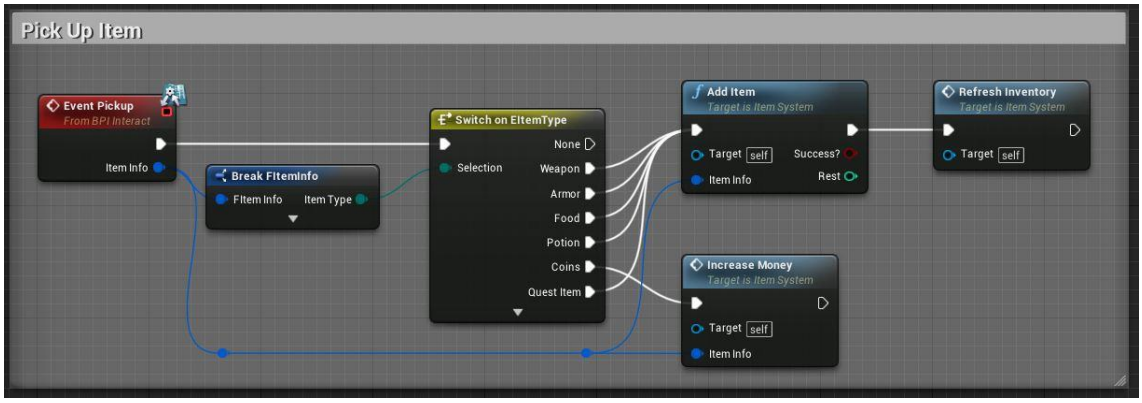
Kuva 29. Esineiden interaktiotapahtuma.

Hahmojen kanssa puhuminen toimii tällä hetkellä samalla periaatteella. Hahmosta riippuen sille lähetettävä interaktioviesti aiheuttaa esineiden tavoin jonkin edeltä määritetyn

toiminnon, kuten kaupan avaamisen. Erona esineisiin on, että interaktion päätyttyä hahmot eivät tuhoa itseään, vaan niiden kanssa voi puhua useaan kertaan.

Inventaario huolehtii pelaajan hallussa olevien tavaroiden luetteloinnista ja näyttämisestä pelaajalle sitä pyydettäessä. Pelaajan poimimat sekä ostamat esineet listataan inventaarioon. Inventaario pitää huolen myös esineiden käyttämisestä.

Kun pelaaja poimii esineen, tiedot esineestä lähetetään ItemSystem-komponentille. Esineen tyypistä riippuen komponentti suorittaa tietyn tehtävän. Käytettävät esineet lisätään inventaariolistalle ja arvoesineet lisätään automaattisesti pelaajan lompakkoon rahana kuvan 30 mukaisesti.

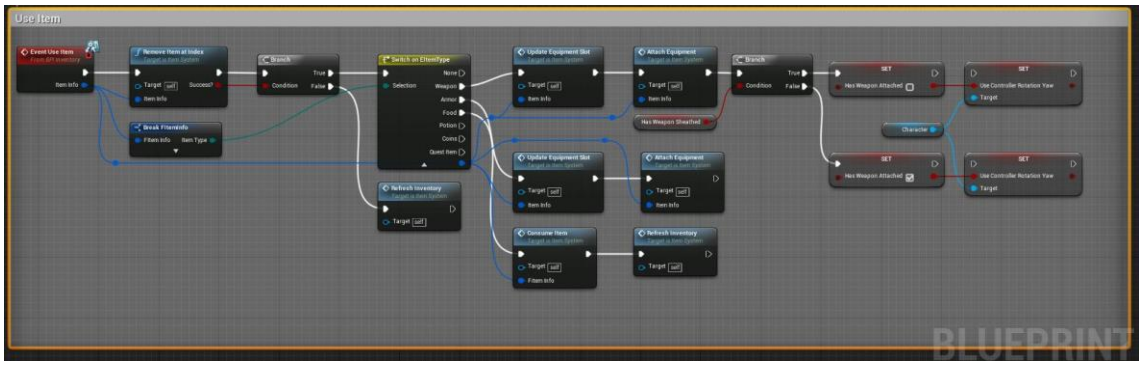


Kuva 30. Esineiden poimintatapahtuma.

Inventaarion toteutus on sekoitus Titanic Gamesin RPG Tutorial Seriesin [16.] ja UnrealGaimeDevin Tutorial Series Inventory Systemin [18] ohjeita. Kun pelaaja avaa inventaarioikkunan, komponentti käy läpi listan pelaajan hallussa olevista esineistä ja lisää niistä ikonit inventaarioikkunaan.

Kun pelaaja käyttää omistamiaan esineitä, komponentti suorittaa eri toiminnon riippuen esineelle asetetusta tyypistä, kuten kuvasta 31 nähdään. Tällä hetkellä käytössä on kolme eri toimintoa. Aseet ja suojavarusteet siirretään inventaariolistasta varustelistaan, jonka jälkeen ne luodaan maailmaan ja kiinnitetään hahmoon. Aseiden kohdalla tarkistetaan vielä, laitetaanko ase pelaajalle suoraan käteen vai ”tuppeen”. Ruoka poistetaan inventaariosta ja sen sisältämä NutritionValue lisätään pelaajan elämäpisteisiin.

Aseiden ja suojavarusteiden poistaminen käytöstä tekee käytännössä saman asian, mutta toisin päin. Esineet siirretään varusteet listalta inventaariolistaan ja ne tuhotaan maailmasta.



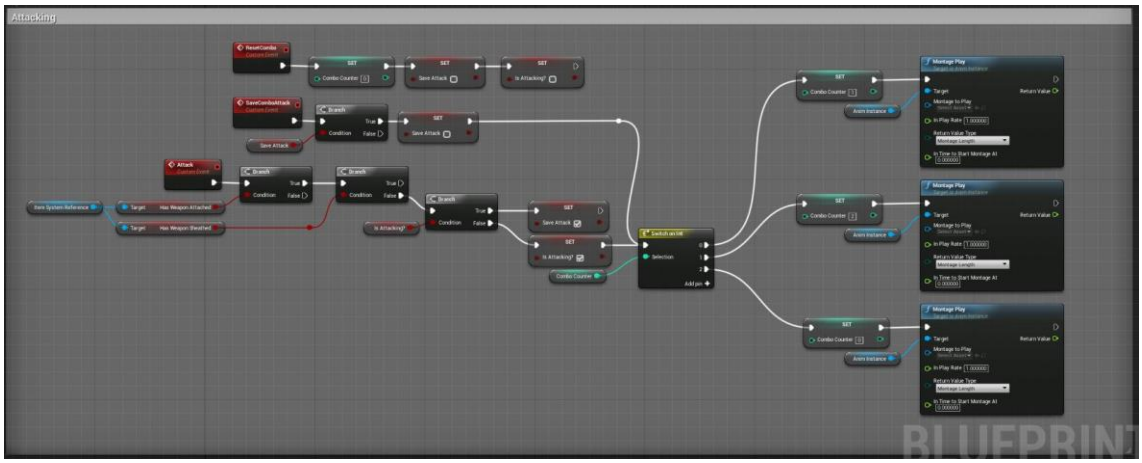
Kuva 31. Esineiden käyttämisen toteutus

4.4 Combat System

CombatSystem-komponentti huolehtii taisteluun liittyvien asioiden hoitamisesta. Komponentin toteutus on vielä alussa ja tällä hetkellä komponentissa on vain kaksi osaa: aseiden ottaminen käyttöön sekä itse hyökkääminen.

Ottaessa aseen käyttöön komponentti tarkistaa, onko pelaajalla ase varustettuna vai eikö ole. Jos ase löytyy, se siirretään pelaajan käteen valmiiksi hyökkäämistä varten. Aseen poistaminen käytöstä toimii päinvastaisesti

Kuten kuvasta 32 nähdään, hyökkäys toimii siten, että komponentti tarkistaa ensin, onko pelaajalla ase kädessä. Mikäli näin on, seuraavaksi komponentti tarkistaa, onko pelaaja valmiiksi hyökkäämässä. Tällä varmistutaan, ettei pelaaja voi hyökätä useaan kertaan samanaikaisesti. Jos pelaaja ei ole hyökkäämässä, komponentti suorittaa hyökkäyksen ja pyörittää hyökkäysanimaation. Samalla asetetaan arvo ComboCounter-muuttujalle, joka mahdollistaa halutun mittaisten hyökkäyssarjojen tekemisen.

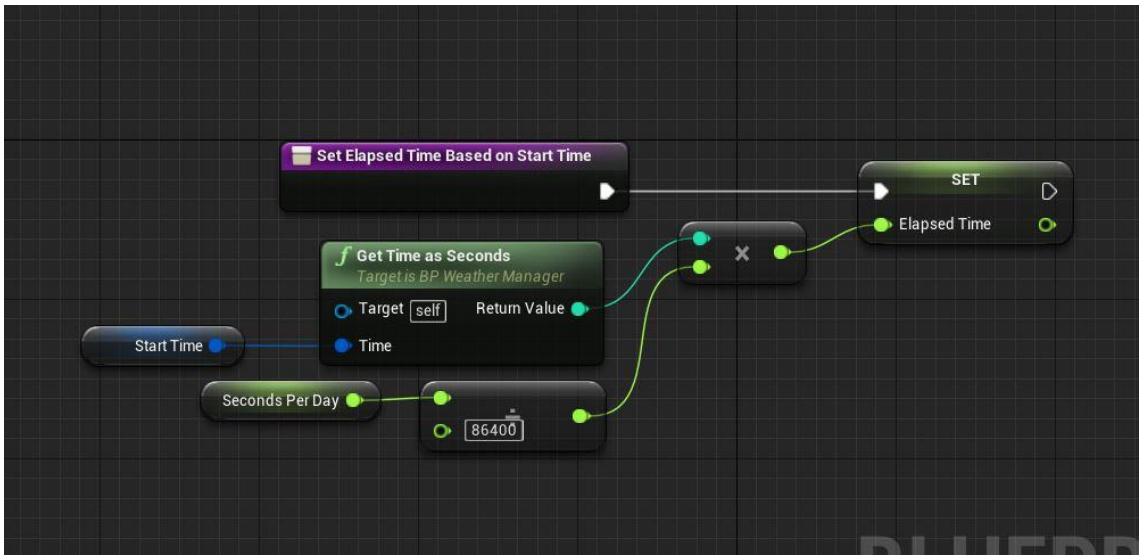


Kuva 32. Hyökkäystapahtuma.

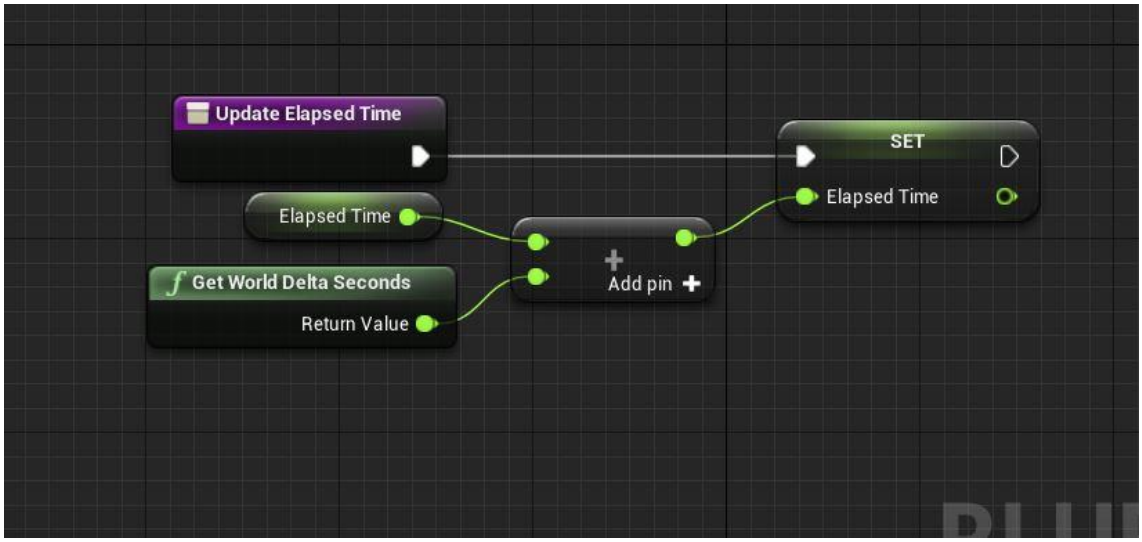
4.5 Weather Manager

WeatherManager on vastuussa pelimaailman lämpötilasta sekä vuorokaudenajasta. Manageri pitää kirjaa myös pelissä kuluneesta ajasta. Systeemiin kuuluu myös Weather Modifier, jolla voidaan säätää haluttua lämpötilaa. Esimerkiksi talon sisällä tai nuotion lähellä voi olla lämpimämpi kuin ulkona. Manageri on toteutettu seuraamalla Titanic Gamesin Survival Game Tutorial Series [17.] -ohjeita.

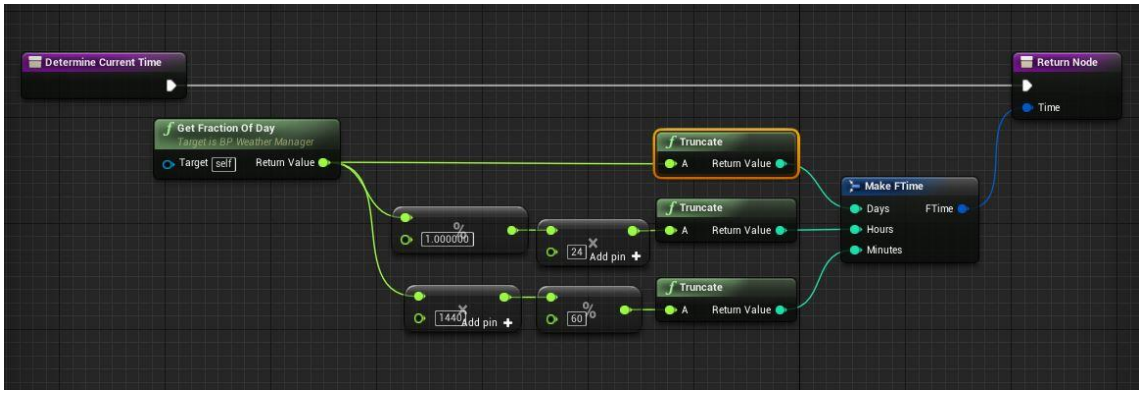
WeatherManager seuraa ajankulkua pelin sisällä. Kellonaika sekä pelissä kulunut aika saadaan selville muutaman matemaattisen funktion avulla. Pelin alussa kulunut aika asetetaan kuvan 33 mukaisesti käyttäjän asettamasta alkamisajasta ja asetetusta päivän pituudesta ja sitä päivitetään pelin mukana käyttämällä deltasekunteja kuvan 34 mukaisesti. Kellonaika saadaan kuvan 35 mukaisesti käyttämällä GetFractionOfDay-funktiota, joka jakaa pelin alusta kuluneen ajan päivän pituudella.



Kuva 33. Ajan asettaminen pelin alussa.

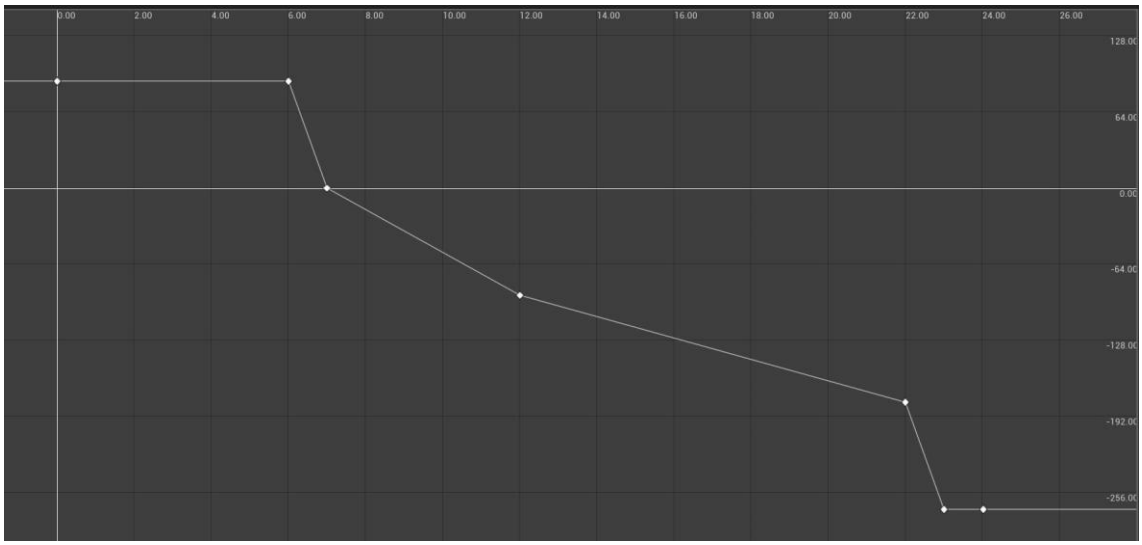


Kuva 34. Kuluneen ajan päivittäminen.



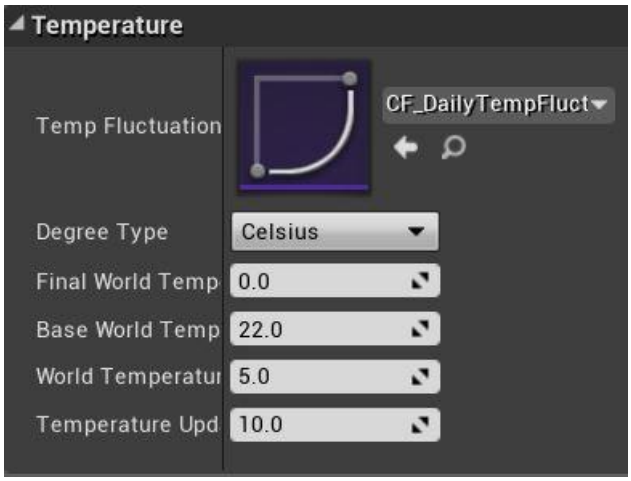
Kuva 35. Kellonajan määritys.

Auringon nousua ja laskua säädellään DayNightCycle käyrän avulla. Se määrittelee xy-asteikossa auringon korkeuden minäkin kellonaikana kuvan 36 mukaisesti, jollain voidaan helposti käyrää säätämällä muokata päivän ja yön kestoa.



Kuva 36. DayNightCycle-käyrä, jossa kellonajat ovat x-akselilla ja auringon paikka y-akselilla.

WeatherManager säätää myös pelimaailman lämpötilaa. Lämpötila voidaan asettaa haluttuun arvoon, ja sille voidaan antaa tietyt rajat, joiden sisällä se voi vaihdella kuvan 37 mukaisesti. Lämpötila voidaan asettaa myös muuttumaan vuorokaudenajan mukana DailyTempFluctuation-käyrän avulla. Käyrä toimii hyvin samalla tavalla kuin aikaisemmin esitelty DayNightCycle-käyrä.



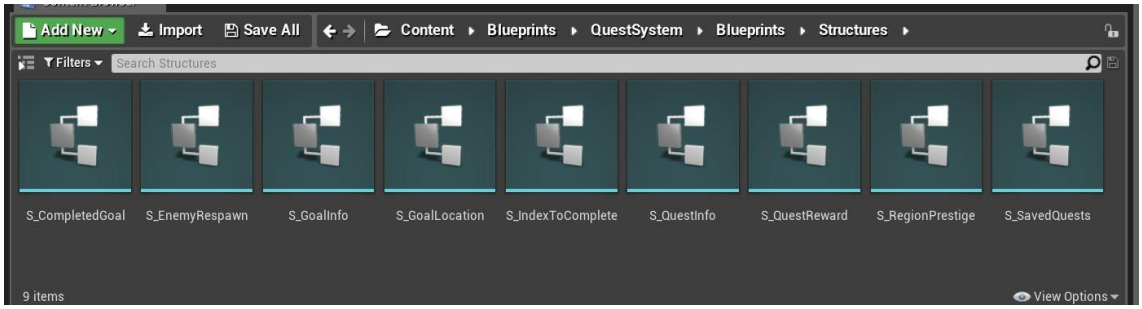
Kuva 37. Pelimaailman lämpötilan asetukset.

4.6 Dialogue System

DialogueSystem-komponentin toteutusta ei käydä tarkemmin läpi muutamasta syystä. Ensimmäiseksi koska se on toteutettu C++-koodilla, ja tämän työn tarkoitus on keskittyä sinikopioihin. Toiseksi siihen ei ole perehdytty tarpeeksi hyvin, jotta voitaisiin selostaa kunnolla, miten se on toteutettu. Näistä syistä johtuen komponentin tarkempi analyysi joutuu valitettavasti odottamaan parempaa aikaa.

4.7 Quest System

QuestSystem-komponentti on toistaiseksi täysi kopio UnrealGaimeDevin Quest System [19.] -tutoriaalista. Se on vastuussa kaikista tehtävistä, joita pelaajan on tarkoitus suorittaa pelin aikana. Komponentin QuestManager sinikopio pitää kirjaa pelaajan hyväksymistä, suorittamista ja epäonnistuneista tehtävistä. Komponentin toiminta perustuu suurimmaksi osaksi luettelo- (eng. Enumeration) ja rakenneluokan sinikopioihin. Nämä sinikopiot sisältävät kaiken datan, jonka avulla tehdään, muokataan ja suoritetaan tehtäviä. Kuvassa 38 nähdään esimerkki komponentin sisältämistä rakennesinikopioista.

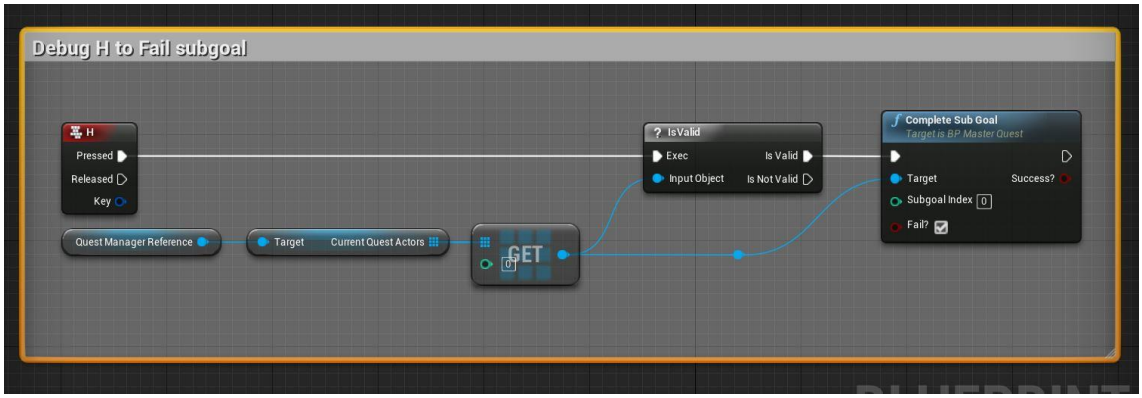


Kuva 38. QuestSystem-komponentin tarvitsemat rakennesinikopiot.

Kun pelaaja saa uuden tehtävän AddNewQuest-funktion avulla, QuestManager tarkistaa varmuuden vuoksi, onko pelaaja saanut sen aikaisemmin, ja mikäli ei ole, tallettaa sen CurrentQuestActors-muuttujaryhmään. Kun uusi tehtävä on tallennettu, se luodaan maailmaan, jotta päästään käsiksi sen sisältämiin tietoihin. Tiedot lisätään heijastusnäyttöön sekä QuestJournalWidget-sinikopioon. Funktio tarkistaa viimeiseksi, onko tehtävän asetuksiin merkitty, että se aloitetaan automaattisesti pelaajan saatua sen - ja jos niin on, aktivoi funktio kyseisen tehtävän.

Tehtäville annetaan niiden asetuksissa ehdot, joiden mukaan tehtävät voi läpäistä. Tällä hetkellä tehtävät voi läpäistä vain suorittamalla niiden sisältämän tavoitteen, johon on merkitty todeksi CompleteQuest?-muuttuja. Tavoitteiden suorittamista valvoo QuestManager. Esimerkiksi etsi-tyypin tehtävän voi suorittaa löytämällä tietyn esineen. Komponentin sisältämillä esineiltä löytyy OnPickUp-tapahtuma, jota kutsutaan tällä hetkellä, kun pelaaja menee esineen lähelle. OnPickUp-tapahtuma puolestaan kutsuu QuestManagerin sisältämää OnObjectFound-tapahtumaa. Tämä tapahtuma käy läpi pelaajan hyväksymät tehtävät ja tarkistaa, kuuluuko kyseinen esine jonkin tehtävän löydettäväksi esineeksi. Mikäli se kuuluu, tapahtuma kutsuu MasterQuest-sinikopion CompleteSubGoal-funktiota, joka merkitsee tavoitteen suoritetuksi ja tarkistaa, onko tavoitteen CompleteQuest?-muuttuja tosi. Mikäli näin on, merkitään myös koko tehtävä suoritetuksi, ja kaikki sen sisältämät muut suorittamattomat tavoitteet epäonnistuneiksi.

Tehtävien epäonnistuminen toimii MasterQuest-sinikopion sisältämän CompleteSubGoal-funktion avulla. Funktio sisältää Fail?-muuttujan, joka asetetaan funktiota kutsuttaessa joko todeksi tai epätodeksi. Jos muuttuja on tosi, kun funktiota kutsutaan, se asettaa suoritetun tavoitteen epäonnistuneeksi, kuten kuvasta 39 nähdään. Jos tavoitteen asetuksissa on laitettu todeksi FailMeansQuestFail?-muuttuja, epäonnistuu silloin myös koko tehtävä.



Kuva 39. Debug-tapahtuma tavoitteen epäonnistumiselle.

Laajennuksen eri osien toteutukset käytiin läpi pintapuolisesti ja niistä esiteltiin vain tärkeimmät asiat. Toteutusten syvempi läpikäynti olisi vaatinut lukijalta paljon syvempää ymmärrystä pelimoottorin ja sinikopioiden toiminnasta.

5 Laajennuksen analyysi

Kaiken kaikkiaan liitännäisen eri osat toimivat sekä yksin että yhdessä halutulla tavalla. Toki liitännäisen toteuttaa vain muutamia hyvin perustason ominaisuuksia, mutta jokainen liitännäisen osa on tehty siten, että niitä on helppo muokata ja laajentaa halutunlaiseksi. Liitännäinen on tarkoituksella suhteellisen yksinkertainen. Tämä johtuu siitä, että se on tarkoitettu luurangoksi, jolla voidaan nopeasti rakentaa prototyyppejä. Tarkoitus on kuitenkin hioa kokonaisuus sellaiseksi, että sitä laajentamalla se toimii myös valmiissa peleissä.

Työn alussa listattiin taulukossa 1 näkyvät ominaisuudet, joihin tässä työssä keskitytään. Samasta taulukosta nähdään myös, mikä komponentti vastaa minkäkin ominaisuuden toteutuksesta.

Taulukko 1. Liitännäisen osat.

Hahmon terveys-, taika- ja kestävyyspisteteet, taso ja kyvyt.	PlayerStats-komponentti.
Kerättäviä esineitä.	ItemSystem-komponentti.
Kauppa.	ItemSystem-komponentti.
Dialogisysteemi.	DialogueSystem-komponentti.
Taistelusysteemi.	CombatSystem-komponentti.
Suoritettavia tehtäviä.	QuestSystem-komponentti.

Taistelusysteemiä lukuun ottamatta kaikista suunnitelluista ominaisuuksista saatiin aikaiseksi jonkinlainen versio. Osaa ehdittiin kehittää ja hioa pidemmälle kuin muita. Jatkossa on tarkoitus käydä kaikkia osia läpi ja parantaa sekä optimoida niistä mahdollisimman helppokäyttöiset.

Tällä hetkellä ItemSystem-komponentti hoitaa kaiken interaktion pelaajan ja maailman välillä. Aluksi se oli hyvä ratkaisu, sillä ainoa asia, jonka kanssa pelaaja oli kanssakäymisessä, olivat tavarat, joita pelaaja voi poimia. Liitännäisen saadessa lisää sisältöä alkoi kuitenkin käydä selväksi, että tämä oli huono valinta. Nyt ItemSystem hoitaa osittain täysin sille kuulemattomia tehtäviä, kuten interaktiot hahmojen kanssa. Jotkin interaktiot hahmojen kanssa, kuten kauppa, kuuluvat osittain ItemSystem-komponentille, mutta esimerkiksi dialogien pyörittäminen ei kuulu sille.

Ratkaisu tähän on siirtää kaikenlainen interaktio pelaajan ja maailman välillä omaan komponenttiinsa ja lähettää sieltä tilanteen mukaan tehtävät oikeisiin paikkoihin. Samalla pystytään muokkaamaan NPC-hahmojen kanssa käytävä interaktiologiikka, joka on tällä hetkellä suora kopio esineiden kanssa käytettävästä logiikasta, sopimaan paremmin tarvittaviin tilanteisiin.

CombatSystem-komponentista on tarkoitus tehdä valmis kokonaisuus, jolla saadaan tehtyä peliin taistelu systeemi, joka muistuttaa pelien Asassin’s Creed Origins ja The Legend of Zelda: Breath of the Wild -taistelusysteemiä [20; 21]. Systeemi koostuu kolmesta osasta, hyökkäämisestä, väistämisestä ja viholliseen lukittautumisesta. Hyökkäys koostuu kevyestä ja raskaasta hyökkäyksestä. Väistäminen tapahtuu pelaajan ohjauksen mukaan, joko eteen, taakse, oikealle tai vasemmalle vihollisen hyökätessä, ja sen käyttäytyminen riippuu osittain siitä, onko pelaaja lukittautunut viholliseen vai ei. Pelaaja voi lukita kameran viholliseen, jolloin pelaajan liikkuminen muuttuu niin, ettei pelaaja voi vapaasti kääntyä minne haluaa, vaan pelaajan rintamasuunta on koko ajan viholliseen päin.

WeatherManager-komponenttia on tarkoitus laajentaa kattamaan myös pelimaailman sää eikä pelkästään vuorokauden aika. Myös vuodenajat on tarkoitus saada mukaan komponenttiin. Toteutus vaatii pelimoottorista löytyvän taivaskupla (eng. SkySphere) sinikopion lähempää tutkimista. Tällä hetkellä kyseinen sinikopio muuttaa taivaan väriä ja auringon paikkaa WeatherManagerin asetusten mukaisesti kellonajan mukana. En ole

ehtinyt tutustumaan taivaskupla sinikopioon vielä niin hyvin kuin olisin halunnut. Toteutuksen suurin kysymys on, onnistuuko sään muutokset kyseisellä sinikopiolla vai tarvitaanko niitä varten kustomoitu versio sekä jotakin uutta grafiikkaa.

Koska käytän parhaillaan valmista liitännäistä dialogisysteemissä, täytyy minun perehtyä siihen syvemmin ja selvittää miten se oikeastaan toimii kannen alla. Koska liitännäinen käyttää toteutuksessaan C++ koodia, en ole vielä päässyt käsiksi sen toteutukseen. Tarkoitus on selvittää, kannattaako minun toteuttaa kyseinen liitännäinen itse vai yrittää laajentaa jo olemassa olevaa systeemiä. Koska liitännäinen on vapaan lähdekoodin alaisuudessa, voin vapaasti muokata sitä ja integroida sen laajennukseeni.

Todennäköisesti eniten muokkausta vaatii QuestSystem. Tarkoitus on muokata systeemiä sopimaan liitännäisen muihin osiin ja integroida se kunnolla liitännäiseen. Systeemiä täytyy myös optimoida, sillä sen toteutus on monessa kohtaa tällä hetkellä kyseenalaista, ja voidaan todennäköisesti toteuttaa paremmin. Miten se konkreettisesti toteutetaan, on vielä täysin suunnittelematta. Koska kyseessä on suhteellisen iso ja RPG pelien kannalta tärkeä komponentti sen hiominen ja integroiminen laajennukseen tulee todennäköisesti viemään aikaa, sillä se täytyy toteuttaa huolella.

6 Yhteenveto

Insinöörityön tavoitteena oli tutustua paremmin Unreal Engine 4 -pelimoottorin visuaaliseen koodausmenetelmään. Tavoite päätettiin toteuttaa tekemällä liitännäinen, joka mahdollistaa yksinkertaisten RPG pelien prototyyppien helpon rakentamisen.

Työssä käytiin läpi, miten liitännäisen eri osia käytetään ja laajennetaan, sekä pintapuolisesti miten liitännäisen eri osat ovat toteutettu. Arvioitiin myös, mitä parannettavaa liitännäisessä on ja mitä tehdään seuraavaksi.

Työssä päästiin melko hyvin asetettuihin tuloksiin. Saatiin aikaiseksi liitännäinen, joka toteuttaa suurimman osan asetetuista tavoitteista. Dialogi ja tehtäväkomponenttien teko jäi ajan puutteen takia vielä kesken, mutta niistä toimivat kuitenkin yksinkertaiset versiot. Kaikki liitännäisen osat vaativat vielä optimointia, jotta käyttäjät voivat helpommin käyttää ja muokata liitännäistä haluamallaan tavalla. Monet sinikopiot sisältävät vielä usean eri toteutuksen samalle asialle, ja ne täytyy siivota ja poistaa turhat osat.

Haasteena työn toteutuksessa ole se, että sinikopiot käyttävät monia samoja käytäntöjä, kuin C++-koodi. Koska ohjelmointi C++-koodilla on vielä suhteellisen vierasta, jouduin selvittämään, miten tietyt asiat toteutetaan C++-koodilla ja miten sitä sovelletaan sinikopioihin. Toinen haaste varsinkin taisteluun liittyvien osien toteutuksessa on animaatioiden puute. Tarkoitus on kuitenkin toteuttaa taisteluun liittyvät ominaisuudet teknisesti niin pitkälle kuin mahdollista, ennen kuin saadaan tarvittavat animaatiot.

Lähteet

1 Role-Playing Game. Verkkoaineisto. Techopedia. <https://www.techopedia.com/definition/27052/role-playing-game-rpg>. Luettu 20.03.2018.

2 What is D&D? Verkkoaineisto. Dungeons and Dragons. <http://dnd.wizards.com/dungeons-and-dragons/what-is-dd>. Luettu 20.03.2018.

3 Video Game History - 40 Years of The RPG Genre. Verkkoaineisto. GameSkinny. <https://www.gameskinny.com/b2ezo/video-game-history-40-years-of-the-rpg-genre>. Luettu 20.03.2018.

4 RPG evolution. Verkkoaineisto. <http://iml.jou.ufl.edu/projects/Spring05/Hill/video.html>. Luettu 20.03.2018.

5 Western RPGs vs Japanese RPGs. Video. Extra Credits. <https://www.youtube.com/watch?v=l_rvM6hubs8>. Katsottu 20.03.2018.

6 Final Fantasy series. Verkkoaineisto. Final Fantasy Wiki. <http://finalfantasy.wikia.com/wiki/Final_Fantasy_series>. Luettu 20.03.2018.

7 Dragon Quest series. Verkkoainesito. Dragon Quest Wiki. <http://dragonquest.wikia.com/wiki/Dragon_Quest_(game_series)>. Luettu 20.03.2018.

8 Final Fantasy XV. Verkkoaineisto. Final Fantasy Wiki. <http://finalfantasy.wikia.com/wiki/Final_Fantasy_XV>. Luettu 20.03.2018.

9 What Makes an RPG an RPG: a universal definition. Verkkoainesito. Sinister Design. <http://sinisterdesign.net/what-makes-an-rpg-an-rpg-a-universal-definition/>. Luettu 20.03.2018.

10 Introduction to Blueprints. Verkkoaineisto. Epic Games. < https://docs.unrealengine.com/latest/INT/Engine/Blueprints/GettingStarted/index.html>. Luettu 14.10.2017.

11 Blueprint Class. Verkkoainesito. Epic Games. <https://docs.unrealengine.com/en-US/Engine/Blueprints/UserGuide/Types/ClassBlueprint>. Luettu 13.03.2018.

12 Plugins. Verkkoainesito. Epic Games. <https://docs.unrealengine.com/en-us/Programming/Plugins>. Luettu 06.04.2018

13 Dialogue System. Nettisivu. MavrinSoft. <https://mavrinsoft.com/dialogue/>. Luettu 10.03.2018.

14 MIT License. Verkkoaineisto. tl;drLegal. <https://tldrlegal.com/license/mit-license#summary>. Luettu 10.03.2018.

15 Advanced Locomotion System. Sovelluslaajennus. LongmireLocomotion. <https://www.unrealengine.com/marketplace/advanced-locomotion-system-v1>. Ostettu 29.07.2017.

16 RGP Tutorial Series. Video. Titanic Games. <https://www.youtube.com/watch?v=Tvxs4vAEtGQ&list=PLtpN-aPTkjdL6jauQ8ZbiB9xe9ay_zfWNE>. Katsottu 06.08.2017-10.08.2017.

17 Survival Game Tutorial Series. Video. Titanic Games. <https://www.youtube.com/watch?v=2020wWwM1Ng&list=PLtpN-aPTkjdL5G4ytDfvakn6VrtATqPJtS>. Katsottu 13.08.2017.

18 Tutorial Series Inventory System. Video. UnrealGaimeDev. <https://www.youtube.com/watch?v=81wnJaixBnQ&list=PLmKKTERcjTP-KEPl0nk48Tpmj-iWmzqo_Q>. Katsottu 30.08.2017-03.09.2017.

19 Tutorial Series Quest System. Video. UnrealGaimeDev. <https://www.youtube.com/playlist?list=PLmKKTERcjTPIkMH9iK3gTUyl976_TW1hj>. Katsottu 09.03.2017-11.03.2017.

20 Assassin's Creed Origins: How Combat Has Changed. Verkkoaineisto. Ubisoft. <https://assassinscreed.ubisoft.com/game/en-us/news/152-293393-16/assassins-creed-origins-how-combat-has-changed>. Luettu 20.03.2018.

21 Dave Tach. 03.2017. The Legend of Zelda: Breath of the Wild beginner’s guide. Verkkoaineisto. Polygon. <https://www.polygon.com/breath-of-the-wild-guide-walkthrough/2017/3/2/14753874/cooking-fighting-weapons-loot-inventory-tips-tricks>. Luettu 20.03.201.