KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutus

Eetu Mahonen

PROTOTYYPIN KEHITYS UNREAL ENGINEN BLUEPRINTEILLÄ

Opinnäytetyö
Joulukuu 2018

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Joulukuu 2018
Tietojenkäsittelyn koulutus

Tikkarinne 9
80200 JOENSUU
+ 358 13 260 600

Tekijä(t)
Eetu Mahonen

Nimeke
Prototyypin kehitys Unreal Enginen Blueprinteillä

Tiivistelmä

Tässä opinnäytetyössä käsitellään videopelin prototyypin kehitystä Unreal Engine -pelinkehitysalustan versiolla 4.19.2. Opinnäytetyössä kehitystä tarkastellaan visuaalisen ohjelmoinnin näkökulmasta. Työ keskittyy Unreal Engine-pelinkehitysalustan tarjoamaan visuaaliseen ohjelmointiin nimeltä Blueprint Visual Scripting. Työssä myös sivutaan muita kehitykseen liittyneitä asioita.

Opinnäytetyön tarkoitus on havainnollistaa, miten prototyypin oleellisimmat osat on toteutettu. Tavoitteena oli luoda toimiva runko, joka mahdollistaa jatkokehityksen, sekä oppia ohjelmistoista ja niiden prosesseista.

Lopputuloksena syntyi prototyyppi, jota käytetään tulevan henkilökohtaisen projektin runkona.

Kieli	Sivuja 45
suomi	Liitteet 0

Asiasanat

visuaalinen ohjelmointi, pelikehitys, pelisuunnittelu, unreal engine

Karelia AMMATTIKORKEAKOULU

THESIS
December 2018
Business Information Technology

Tikkarinne 9
80200 JOENSUU
+358 13 260 600

Author(s)
Eetu Mahonen

Title
Developing a Prototype Using Blueprints of Unreal Engine

Abstract

The thesis focuses on a prototype made in Unreal Engine development platform version 4.19.2. In the thesis, the development of the prototype is examined from the point of view of visual programming. Thesis focuses at visual programming in Unreal Engine development platform. Thesis also covers some other issues and subjects related to development.

The thesis intends to illustrate, how the most important parts of the prototype have been implemented. The goal was to create a functional framework/prototype that allows further development, as well as learning about software.

The result of the development, was a prototype that will be used as the backbone of a personal project in the future.

Kieli

Finnish

Pages 45

Keywords

visual scripting, game development, game design, unreal engine

Sisältö

1 Johdanto

Tässä opinnäytetyössä tarkastellaan, mitä kaikkea pelin kehitys tyhjältä pöydältä vaatii. Tavoitteena oli luoda prototyyppi videopelistä, jonka aihealueeksi valittiin kauhu. Tekninen toteutus pyrittiin tekemään niin pitkälle kuin mahdollista hyödyntäen ainoastaan UE:n Blueprinttejä.

Visuaalinen ohjelmointi tai Visual Scripting ei tietyllä mittakaavalla ole kovin uusi työkalu, mutta sen käyttö puolestaan on yleistynyt viime vuosina, minkä takia aihetta on syytä tarkastella. Tässä opinnäytetyössä keskitytään Unreal Enginen tarjoamaan visuaalisen ohjelmoinnin työkaluun nimeltä Blueprints Visual Scripting tai yleisimmin tunnetulta nimeltä Blueprints.

Tieto-osio käsittelee toteutukseen käytettyjä työkaluja ja ohjelmistoja. Pääpaino kyseisessä osuudessa on Blueprintien toiminta, mitä ne konkreettisesti ovat, kuinka niitä käytetään ja mitä tulee ottaa huomioon niiden käytössä. Edellä mainittujen lisäksi osiossa käydään läpi muita kehityksessä huomioon otettavia asioita, kuten visuaaliset aspektit ja valitun genren toimintamallit.

Opinnäytetyö on toiminnallinen. Toiminnallisena osuutena oli pyrkimys kehittää prototyyppi videopelistä. Raportti toimii kuvauksena, miten toiminnallinen osuus toteutettiin. Raportti pohjautuu kirjoittajan omiin kokemuksiin, joita prototyypin parissa työskentely on tuonut. Kehitys käydään läpi siinä järjestyksessä kuin kirjoittaja on asiat suorittanut tai niin, että kokonaisuus on mahdollisimman helppo ymmärtää. Aluksi käsitellään, mitä kaikkea tulee ottaa huomioon valmistelussa ennen teknisen toteutuksen aloittamista. Edellä mainittuun kuuluvat oleellisesti storyboardin teko, käytettävien ohjelmistojen valinta, toteutuksen aikataulu sekä pelin genre.
Lopputulos esitetään lyhyillä selostuksilla ja kuvilla. Yhteenvedossa muodostetaan kokonaiskuva asetettujen tavoitteiden ja toteutuksen onnistumisen välillä**.**

2 Visuaalinen ohjelmointi

Visuaalinen ohjelmointi tarkoittaa käytännössä visuaalisesti toteutettavaa ohjelmointia. Kyse on ohjelmiston rakentamisesta visuaalisesti graafisten elementtien avulla. Rakentamista voidaan verrata palapelin kasaamiseen, jolloin graafiset osat vastaavat yksittäisiä palikoita, jotka yhdistetään kokonaisuudeksi. Jokainen palikka vastaa inputia, toimintoa, tapahtumaa tai muuta ohjelman toimintaan vaikuttavaa tekijää.

Visuaalisen ohjelmoinnin ja perinteisen (kirjallisen) ohjelmoinnin toiminnalliset erot ovat suhteellisen suuret. Visuaalinen ohjelmointi on paljon rajatumpaa kuin perinteinen ohjelmointi. Yksi isoimmista eroista on syntaksien ja kirjastojen käyttäminen. Visuaalisessa ohjelmoinnissa rakennuspalikat on usein sidottu toisiinsa joko pääkirjaston tai muun vastaavan kautta. Perinteisessä ohjelmoinnissa käytettävää kieltä voidaan muuttaa tai vaihtoehtoisesti viitata kirjastoihin tai ulkoisiin funktioihin. Vaikka visuaalinen ohjelmointi on helpommin ymmärrettävää, ei visuaalinen ohjelmointi voi kilpailla perinteisen ohjelmoinnin tekstillisen informaation kanssa. (Dehouck 2015.)

3 Unreal Engine ja Blueprintit

Unreal Engine on Epic Gamesin kehittämä pelimoottori. Guiness World Records valitsi vuonna 2014 Unrealin neljännen iteraation menestyksekkäimmäksi pelimoottoriksi.

3.1 Unreal Enginen lyhyt historia

Unreal Enginen ensimmäinen versio syntyi, kun Epic MegaGames, Inc:n perustaja Tim Sweeney inspiroitui ensimmäisistä suureen suosioon nousseista FPS-peleistä, kuten Wolfenstein 3D, Doom ja Quake. Sweeney halusi luoda

vakavasti otettavan kilpailijan id Software LLC:n luomalle Quake sarjalle. Lopputuloksena Epic MegaGames julkaisi vuonna 1998 pelin nimeltä Unreal, jonka takia kyseisen pelin tekemiseen käytetty moottori sai kekseliäästi nimekseen Unreal Engine. Unreal-pelin julkaisun yhteydessä yritys alkoi myös lisensoimaan moottoriaan muille pelikehittäjille. Yritys vaihtoi nimensä vuonna 1999 jonka jälkeen se tunnettin nimellä Epic Games. (Bleszinski 2010.)

Unreal Engine 2 julkaistiin vuonna 2002. Julkaisussa nähtiin myös USA:n armeijan rekrytointiin kehittämä videopeli, America's Army. (Kennedy 2002. McLeroy, 2008.)

Unreal Engine 3 näki päivänvalon vuonna 2004. Korjausten ja teknisten päivitysten lisäksi, moottorin suosio kasvoi entisestään. Vuoteen 2009 asti pelimoottorilla tehtyjen pelien julkaiseminen ja myyminen edellytti lisenssiä. Tämä kuitenkin poistui vuonna 2009, kun Epic Games julkaisi ilmaisen version Unreal Enginen kehitystyökaluista, mitkä tunnettiin nimellä Unreal Development Kit tai lyhennettynä UDK. (IGN 2009.)

Unreal Engine 4:stä (joka oli kehityksessä jo vuodesta 2003) saatiin ensimmäiset tiedot jo vuonna 2005. Epic Games julkaisi lopullisen tuotteen maaliskuussa vuonna 2014, joka oli kohdistettu kehittäjäyhteisölle. Saman vuoden syyskuussa yritys ilmoitti moottorin olevan ilmainen kouluille ja opiskelijoille. Lopputuloksena yritys päätyi julkaisemaan moottorin julkiseen käyttöön vuonna 2015. Pelimoottorin käyttäminen on täysin ilmaista. Ainoastaan julkaistusta tuotteesta tulee maksaa 5 %:n rojaltit Epic Gamesille. Unreal Enginen uusin versio kirjoitushetkellä on 4.19. (Epic Games 2018a, Epic Games 2018b.)

3.2 Blueprintit

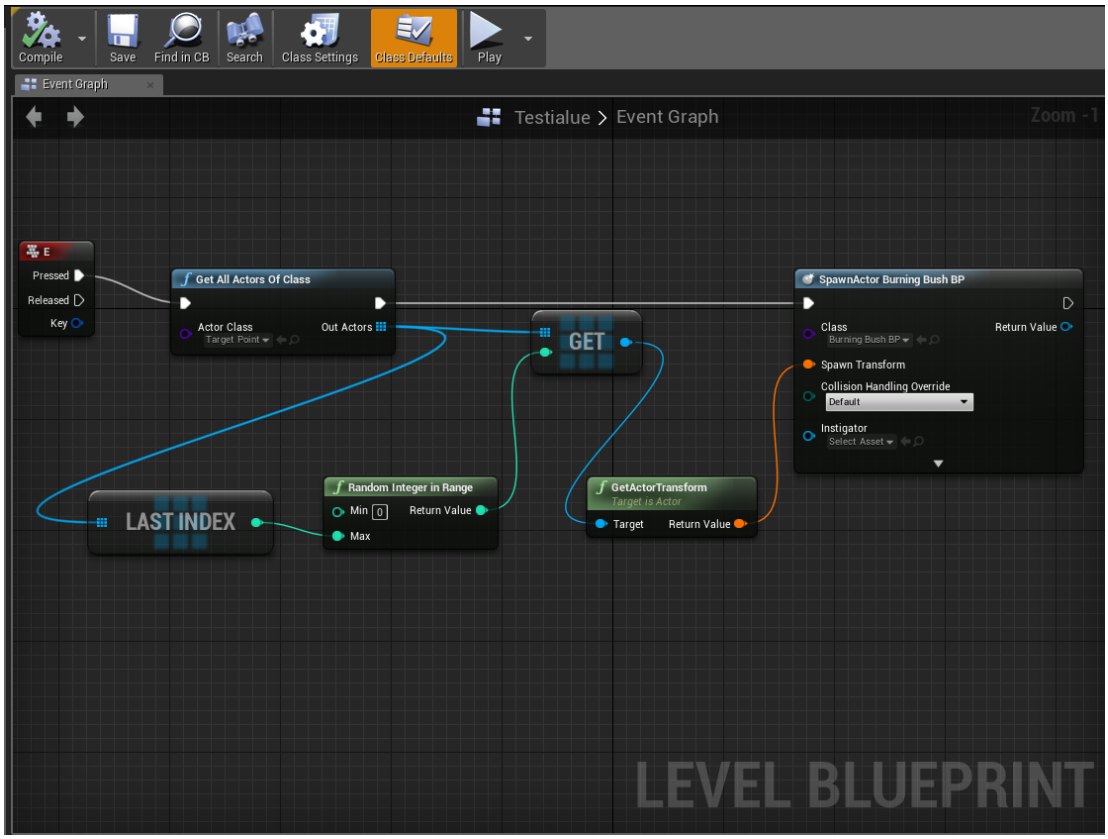
Blueprintit ovat Unreal Enginen versio visuaalisesta ohjelmoinnista. Perusidea Blueprintien toiminnassa on valmiiksi toiminnalisuutta sisältävien nodejen yhdistäminen toimivaksi kokonaisuudeksi. Blueprintit noudattavat

ohjelmointiparadigmaa nimeltä olio-ohjelmointi tai englanniksi object-oriented programming.

3.2.1 Blueprintien tyypit ja käyttötarkoitukset

Blueprinttejä on viisi (5) erilaista. Näistä kaksi käytetyintä ovat Level Blueprint ja Blueprint Class.

Level Blueprint on nimensä mukaisesti koko yksittäisen kartan, kentän tai alueen kattava graafi. Kyseisessä Blueprintissä on myös mahdollista vaikuttaa eriteltyihin osa-alueisiin, kuten esimerkiksi Actoreihin (kuva 1). (Epic Games 2018c.)

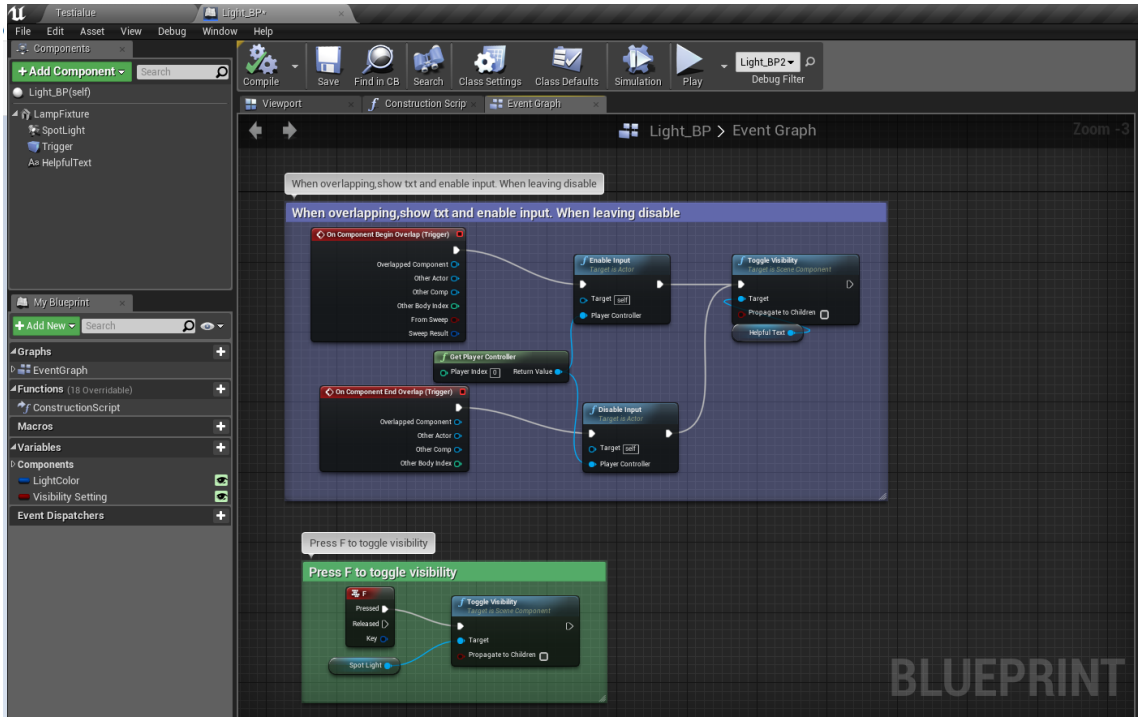


Kuva 1. Actoriin vaikuttavaa toiminnallisuutta Level Blueprintissä.

Yleisesti Level Blueprintiin pyritäänkin kirjoittamaan kaikki se toiminnallisuus, jota ei ole erikseen eritelty Blueprint Classissa ja/tai, jonka halutaan vaikuttavan kokonaisen kentän tai kartan alueella. Level Blueprinteistä puhuttaessa niitä

kutsutaan yksinkertaisesti Level Blueprinteiksi. Mikäli peli sisältää useamman tason, erittely voi tapahtua sanomalla level1 Blueprint. (Epic Games 2018d.)

Toinen käytetyimmistä Blueprinteistä on nimeltään Blueprint Class, joka lyhennetään yleensä vain Blueprintiksi. Tämä johtuu siitä, että kyseistä Blueprintiä käytetään pääsääntöisesti luomaan yksittäisiä olioita, jotka vastaavat tietystä osa-alueesta (kuva 2). Projektia tehdessä Blueprint Classeja voi olla parhaimmillaan jopa satoja, jolloin niihin viitataan esim. nojatuolin Blueprinttinä eikä Blueprint Class-nojatuolina. Opinäytetyössä näihin viitataankin jatkossa asian ”x Blueprintinä”.



Kuva 2. Esimerkki seinävalon Blueprint Classista.

Blueprint Classeja voi olla useita erilaisia. Nämä voidaan jakaa kahteen kategoriaan, Parent ja Child. Parent toimii yleensä kokonaisuutena, josta toiminnallisuutta peritään, esimerkiksi ”Tontut BP”, joka sisältäisi kaikki pelin tonttujen jakamat ominaisuudet tai yhteiset tekijät. Child puolestaan sisältää toiminnallisuuden, jotka ovat ominaisia vain tietyille tontuille tai tietylle tontulle. Child BP voisi olla nimeltään vaikkapa ”HyppiväTonttu_BP”, joka sisältäisi tarvittavat funktiot jatkuvaan hyppimiseen ja kaikki muut tiedot kuten esimerkiksi elinvoima olisi peritty Parent BP:stä.

Vähemmän käytetyt Blueprintit ovat nimeltään Data-Only Blueprint, Blueprint Interface ja Blueprint Macro Library. Näistä ensimmäinen on tavallaan Blueprint Class. Erona on, että Data-Only Blueprint sisältää vain koodin, muuttujat ja komponentit, jotka on peritty parentilta. Data-Only Blueprintiin ei voi lisätä uusia elementtejä, kuten Blueprint Classissa, mutta kyseinen Blueprint mahdollistaa perittyjen ominaisuuksien muokkaamisen. (Epic Games 2018f.)

Blueprint Interface puolestaan on funktioiden kokoelma. Epic Games kuvailee Blueprintiä seuraavasti. ” The use of Blueprint Interfaces allows for a common method of interacting with multiple disparate types of Objects that all share some specific functionality.” (Epic Games 2018g.)

Viimeisen Blueprintin nimi on Blueprint Macro Library, joka on nimensä mukaan kirjasto, josta macroja voidaan suorittaa tarpeen tai tilanteen mukaan.

> Blueprint Macros, or Macros, are essentially the same as collapsed graphs of nodes. They have an entry point and exit point designated by tunnel nodes. Each tunnel can have any number of execution or data pins which are visible on the macro node when used in other Blueprints and graphs. (Epic Games 2018h.)

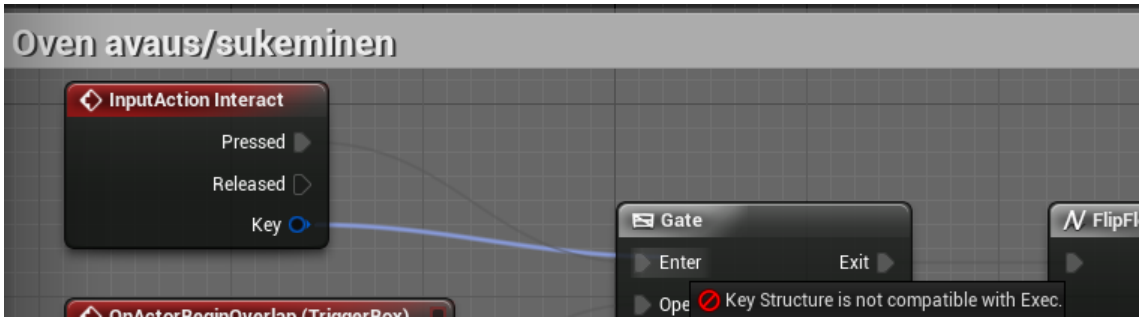
3.2.2 Blueprintien toimivuus käytännössä

Kuten aiemmin mainittu Blueprintien käyttö on visuaalista ohjelmointia. Tämä toimii niin, että kokonaisuus rakennetaan valmiista palikoista, jotka vastaavat tiettyä kirjoitettua koodia. Yksittäisiä rakennuspalikoita kutsutaan nodeiksi, joita yhdistelmällä rakennetaan toimiva kokonaisuus.

Mitä nodet ovat? Nodet ovat objekteja, kuten esimerkiksi funktiot ja variablet. Nodeja käytetään Blueprintissä loppujen lopuksi samalla tavalla kuin legopalikoita. Näitä yhdistämällä loogisesti tai halutulla tavalla saadaan aikaan suoritettava toiminto. Kuvassa 2 on nähtävissä kaksi erillistä toimintoa. Ylempi osio sisältää toiminnon ohjetekstin ilmestymiselle, kun pelaaja kävelee lampulle määritettyyn triggeriin. Kuvan alempi toiminto sytyttää ja sammuttaa lampun pelaajan painaessa näppäintä ”F”.

Nodet sisältävät pinnejä, joihin voi liittää esimerkiksi variableja. Noden input ja output on aina asetettu siten, että input on noden vasemmassa reunassa ja

output oikeassa. Nodet suoritetaan aina vasemmalta oikealle siinä järjestyksessä, kuin ne on liitetty toisiinsa. Liittäminen näkyy visuaalisesti (kuva 2) näkyvänä viivana nodejen välissä. Mikäli tietyn objektin liittäminen toiseen ei ole mahdollista, ilmoittaa UE tästä antamalla viestin nodejen yhteensopimattomuudesta (kuva 3). (Epic Games 2018e.)



Kuva 3. Yhteensopivuusvirhe.

Blueprintit vai perinteinen koodaus, tämä on täysin kiinni tekijästä. Toiset vannovat Blueprintien helppouden nimeen, kun taas toiset pitävät menetelmää hitaana. Yleisesti suositus onkin, että ohjelmointikielet opeteltaisiin niin ettei ohjelmointiin tarvitsisi käyttää Blueprintejä. Lähtökohtaisesti visuaalinen ohjelmointi, kuten UE:n Blueprintit ovat kuitenkin lähinnä suunnittelijoiden työkalu.

4 Tavoitteet, suunnittelu ja valinnat

Osiossa käydään läpi asetetut tavoitteet ja suunnittelu. Aihealueena toiminnallinen osio oli itselleni täysin uutta, joten raportissa käydyt asiat opittiin työstämisen aikana.

4.1 Tavoitteet

Osiossa on listattu opinnäytteen keskeiset tavoitteet, joihin kuuluvat toiminnallisuudet sekä rungon valmistuminen jatkokehitystä varten. Pienempiä tavoitteita olivat ohjelmistojen oppiminen, prototyypin haasteellisuus, tarina sekä pelikehityksen kokonaisuuden hahmottaminen.

Tavoitteena oli luoda kauhupelin prototyyppi, joka toimisi runkona tulevaisuudessa. Halusin prototyypin olevan toiminnallisuuksiltaan mahdollisimman valmis jatkokehitystä varten. Pelaajan tuli pystyä juoksemaan, keskustelemaan NPC:n kanssa, juoksemaan, käyttämään taskulamppua, avaamaan ovia ja kuolemaan. Lisäksi prototyypin tuli sisältää jonkinlaiset menut pelin aloitukselle ja kuolemalle.

Prototyypin tuli myös olla sen verran haastava, ettei sitä pysty läpäisemään alle kymmenessä minuutissa. Prototyypin ei ollut tarkoitus sisältää selkeää tarinaa tai ohjata pelaajaa oikeisiin paikkoihin suoranaisesti eli pelaajaa ei pidetä kädestä.

Toiminnallisuudet tuli olla tehty kokonaisuudessaan Blueprinteillä ja UE:n sisältämillä työkaluilla. Ohjelman toiminnallisuus siis toteutettiin täysin visuaalisella ohjelmoinnilla perinteisen kirjallisen sijaan. Visualeihin keskittyvät asiat pyrittiin tekemään Blenderillä ja Photoshopilla.

Tavoitteena oli nähdä, mitä kaikkea pelin tai prototyypin tekeminen vaatii ja oppia käyttämään valittuja työkaluja.

4.2 Tekemisen suunnittelu ja työkalujen valinta

Projektin suunnittelu alkoi ohjelmien valitsemisella ja ympäristön hahmottelulla. Versionhallintaan harkitsin ohjelmia nimeltä TortoiseSVN sekä GitHub. Versionhallintaan valittiin TortoiseSVN, joka oli tuttu harjoittelun puolesta ja valmiiksi asennettuna. Tämä lähtökohtaisesti siksi, että kokemusta Tortoisen käytöstä löytyi jonkin verran. Serveriksi päätyi VisualSVN maksuttomuuden ja

luotettavan toimintansa takia ja koska kyseistä serveriä oli suositeltu Tortoisen puolelta. Pelimoottorina toimii Unreal Enginen versio 4.19.2 joka valittiin Blueprintien olemassaolon sekä kehitysalustan oppimisen takia. Unreal Engine oli tuttu harjoittelusta, mutta halusin syventyä ohjelmaan enemmän, josta syystä Unityä ei valittu. Objektien tekemiseen valittiin Blender sekä Photoshop. Blender valittiin ohjelman maksuttomuuden puolesta koska ensimmäisen valintani Maya, on maksullinen. Lisäksi Blender on yleisesti pelikehityksessä käytetty ohjelma, jonka oppiminen edes jollain asteella on suotavaa. Photoshop valittiin koska se oli saatavilla ja kokemusta käytöstä löytyi jonkin verran, sekä koska Photoshop on yleisesti hyvin suosittu ohjelma. Vaihtoehto Photoshopille olisi ollut PaintShop Pro, jota ei käytetty koska Photoshop oli ohjelmana tutumpi.

Suunnittelu ja hahmottelu tehtiin piirtämällä kuvia erilaisista skenaarioista, joita prototyypissä oli mahdollista olla. Piirrettyjen kuvien tai ns. kuvakäsikirjoituksen tarkoitus oli toimia suuntaviivana projektin edetessä mutta piirtopöydän puuttuessa koin parhaaksi soveltaa asioita lennosta ja jättää tarinallisen osuuden pois.

Ennen teknisen toteutuksen aloittamista asennettiin valitut ohjelmistot ja hahmoteltiin tarinaa. Edellä mainitun lisäksi oli valittava olisiko prototyyppi ensimmäisen- vai kolmannen persoonan kuvakulmasta. Päädyin ensimmäiseen persoonaan graafisen työskentelyyn vaadittavien laitteiden puutteen takia.

4.3 Genre, aihealue ja sisältö

Prototyypin genreksi valikoitui kauhu. Genren sisältämät toimintatavat nojaavat pelien näkökannasta vahvasti tarinankerrontaan, valaistukseen ja äänimaailmaan. Mainitsin kuitenkin, ettei prototyypissä erityisesti panosteta näihin aihealueisiin vaan keskitytään tekniseen toteutukseen. Kauhu on mielestäni genrenä sellainen, ettei se täysin ole riippuvainen omista toimintatavoistaan/kliseistään. Toisin kuin esimerkiksi fantasia, joka on riippuvainen elementeistä, jotka kyseiseen genreen yleensä liitetään, ei

kauhulla ole selkeää määritystä mitä lasketaan kauhuksi ja mitä ei. Edellä mainitusta syystä kauhuksi voidaankin määritellä kaikki mitkä tuottavat mm. ahdistusta, järkytystä, pelkoa, epämukavuutta tai vastenmielisyyttä.

Aihealueen valinta ei kuitenkaan ollut helppoa genren laajan määrittelyn takia. Ihmiset eivät luonnostaan halua olla tekemisissä kyseenalaisten asioiden tai tabujen kanssa. Vaikka tämä ei suoranaisesti aiheuttanut itselleni rajoitteita, tuli aihealueet valita tulevaisuutta ajatellen. Tavoitteena on kehittää prototyypistä julkaistava peli seuraavien vuosien aikana, joten ikärajoitukset sekä ihmisten reaktiot tiettyihin aiheisiin tulee ottaa huomioon. Aihealueiksi valitsin ihmisyyden, syyllisyyden ja menetyksen käsittelyn.

Sisältöä hahmotellessa tuli pohtia miten aihealueet saadaan tuotua esille. Yleisesti voidaan sanoa, ettei ihmisyyttä voida määritellä suoraan, toisen painajainen on toisen taivas. Myös menetyksen ja syyllisyyden käsittely on hyvin yksilöllistä. Lopputuloksena aihealueita käsitellään geneerisesti sekä omien kokemuksieni mukaan. Sisällön hahmottelun ja pohtimisen jälkeen oli aika aloittaa itse toteutus.

5 Toteutus

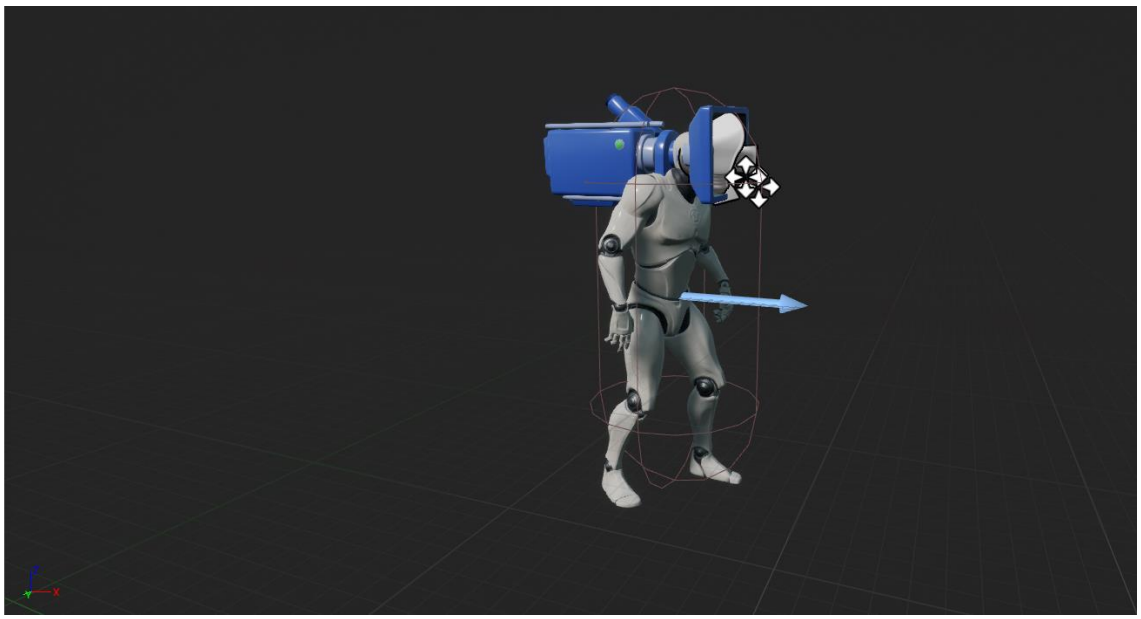
Tässä luvussa käsitellään teknistä toteutusta. Ensimmäiseksi luotiin uusi tyhjä kenttä UE:n tarjoamista vaihtoehdoista. Projektille annettiin nimeksi NomNomSimulaattori erään myöhemmin leikatun kohtauksen takia. Päädyin valitsemaan kolmannen persoonan pohjan, jonka muokkasin ensimmäiseen persoonaan.

5.1 Hahmot

Hahmoja luotiin teknisesti kolme kappaletta: pelaaja, vihollinen (pursuer) ja NPC, jonka kanssa pystytään käymään keskusteluja. Vihollista ja NPC:tä voidaan monistaa, koska alkuperäinen versio omistaa kaiken toiminnallisuuden, ei kopioiden toimintoja käydä erikseen läpi.

5.1.1 Pelaaja

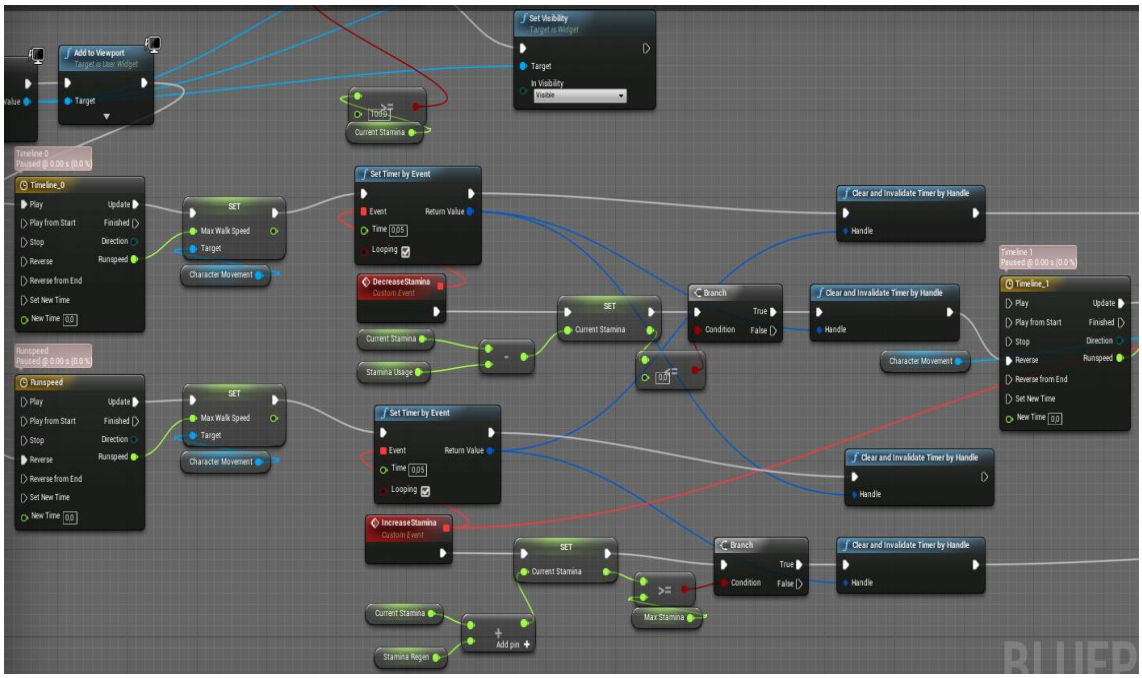
Kolmannen persoonan muuttaminen ensimmäiseen persoonaan toteutettiin avaamalla pelaajahahmon viewport ja siirtämällä kamera manuaalisesti x-, y- ja z-akseleita käyttäen hahmon pään sisään. Kameran ollessa sijoitettuna siinä pisteessä kuin halusin sen olevan, annettiin kameralle ”Parent socket”. Kameralle annettu ”Parent socket” toimii kiinnityspisteenä, joka tässä tapauksessa oli hahmon pää. Kiinnityspiste sitoo kameran annettuun meshin osaan, lopputulos on nähtävissä kuvassa 4. Runkoon kiinnittämisen jälkeen kameran asetuksia säädettiin siten, että kamera vastaanottaa rotaatiot hahmolle annettujen inputtien mukaisesti. Kameran siirtämisen jälkeen voitiin aloittaa hahmon toiminnallisuuksien lisääminen ja muokkaaminen.



Kuva 4. Kamera kiinnitetty pelattavan hahmon päähän.

Template-hahmot sisältävät oletuksena liikkumiseen vaadittavat toiminnallisuudet. Ensimmäiseksi hyppääminen poistettiin käytöstä ja hahmon muokkaaminen aloitettiin luomalla taskulamppu. Taskulampun luominen toteutettiin siten, että hahmon päässä olevaan kameraan lisättiin spottivalo sekä tietyllä alueella vaikuttava valo. Valojen näkyvyys piilotettiin ja projektin asetuksiin lisättiin painike taskulampun käyttöön. Hahmon Blueprintiin lisättiin toiminnallisuus, jossa pelaajan painaessa näppäintä ”F” valot vaihdetaan näkyviksi. Valojen tehoa ja asetuksia säädettiin useita kertoja projektin edetessä valoa korostavan sumun takia.

Seuraavana vuorossa oli kävely ja juoksu (kuva 5). Normaalisti template-hahmo juoksee jatkuvasti mutta animaatio kävelylle on olemassa, vaikkakin ”piilotettuna”. Kävelyn ja juoksun erottaminen tapahtui niin, että hahmon liikkumisnopeutta säädettiin hahmon Blueprintissä. Halusin hahmon myös sisältävän staminaa kuvaavan palkin sekä siirtymän kävelyn ja juoksun välillä. Asetuksiin lisättiin nappi juoksulle, joka on tässä tapauksessa vasen shift. Nappia painaessa näytölle luodaan palkki, joka näyttää staminan määrän, mikä vähenee juostessa. Palkin luomisen jälkeen suoritetaan kiihdytys kävelynopeudesta juoksunopeuteen käyttämällä aikajanaa. Stamina toimii siten, että annetusta arvosta vähennetään annettu määrä niin pitkään, kunnes kokonaisarvo saavuttaa nollan tai pelaaja lopettaa napin painamisen. Nollan saavuttaminen tai napin painamisen lopettaminen palauttaa hahmon aikajanan kautta takaisin kävelytilaan ja staminan arvoon aletaan lisäämään ennalta määrättyä lukua, kunnes annettu maksimiarvo on saavutettu.



Kuva 5. Juoksemisen toiminnallisuus.

Seuraavaksi vuorossa oli hahmon kuoleminen. Yksinkertaisesti hahmolle annettiin elinvoiman arvo, josta vähennetään vastaanotettu vahinko. Mikäli elinvoima putoaa nollaan, poistetaan pelaajan input, kursori laitetaan näkyviin ja ruudulle luodaan ruutu, joka ilmoittaa pelaajan kuolleen. Ruudussa on valittavana mahdollisuudet aloittaa alusta tai lopettaa pelaaminen.

Pelattavan hahmon askelten äänet lisättiin animaatiot sisältävään Blueprintiin. Kävelyn animaatioon lisättiin ääntä toistavat notifikaatiot alkavaksi kohdalta, kun jalka osuu maahan (kuva 6). Äänten lisäämiseen on paljon hyödyllisempiäkin vaihtoehtoja mm. materiaalin tunnistus, jolloin ääni vaihtuu sen mukaan millä materiaalilla hahmo kävelee. Prototyypissä kuitenkin päädyin toteuttamaan askelten äänet näin, ajan säästämiseksi.

Kuva 6. Askelten äänet asetettuna animaatioon.

5.1.2 Pursuer

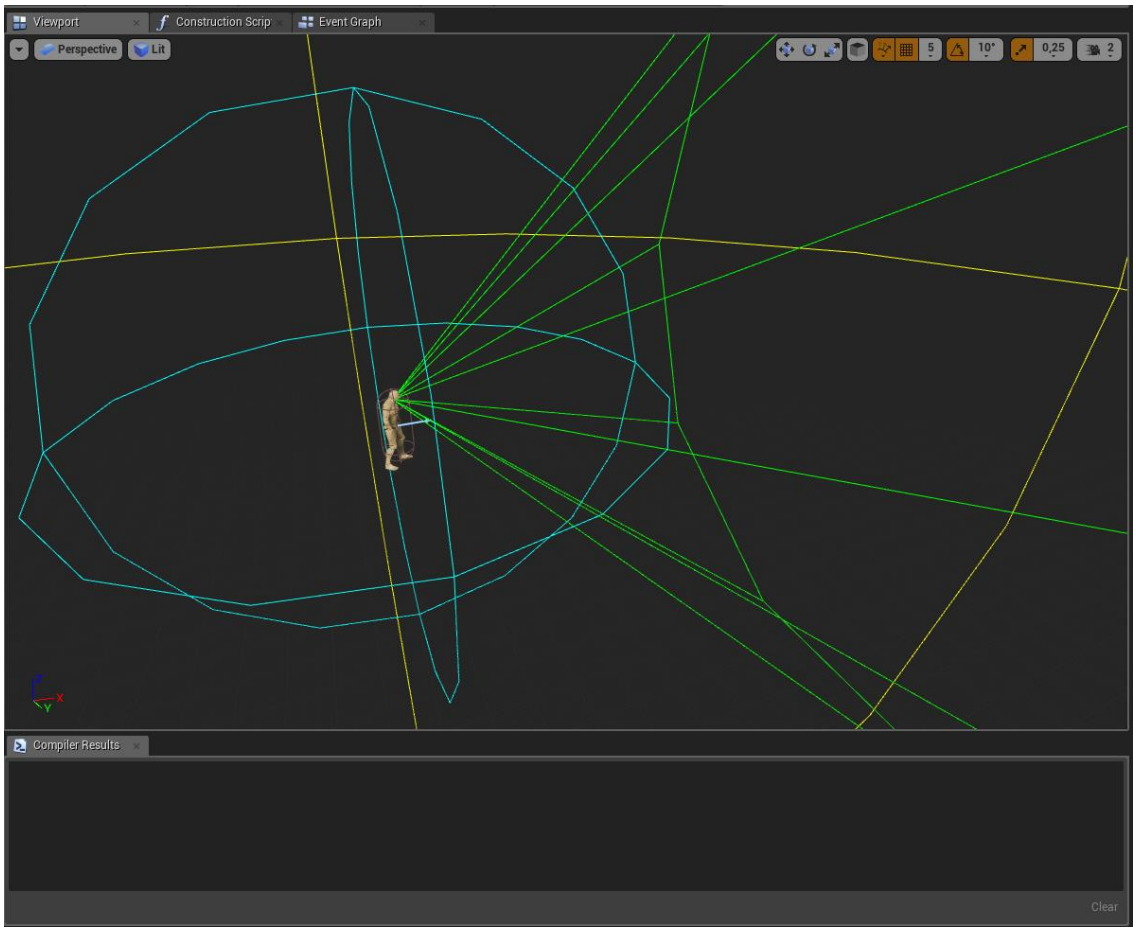
Pursuer on prototyypissä esiintyvä vihollishahmo, jonka päätehtävänä on tappaa pelaaja. Hahmon luonti aloitettiin tekemällä pelattavasta hahmosta kopio, josta poistettiin kaikki toiminnallisuus. Halusin luoda hahmolle kolme erillistä toimintoa: roam, chase ja patrol. Toimintojen vaihtaminen nopeasti oli tärkeää, joten hahmon toiminnallisuuksien lisääminen aloitettiin luomalla Enumeration Blueprint, minne lisättiin nimet halutuille toiminnoille. Edellä mainitun jälkeen hahmon omaan Blueprintiin lisättiin node (event begingplay), joka aloittaa toimintansa välittömästi prototyypin käynnistyttyä. Noden perään laitettiin kytkin, johon liitettiin kutsut jokaisen halutun toiminnan koodiin. Toiminnan vaihtamiseksi nopeasti lisättiin hahmolle vielä julkinen muuttuja, joka sisältää halutut toiminnot. Julkinen muuttuja mahdollistaa toimintojen vaihtamisen suoraan kentästä. Toiminnan vaihtaminen toimii yksinkertaisesti siten, että hahmo valitaan kentässä ja muuttuja tuo hahmon asetuksiin menun, josta toiminnallisuuden voi vaihtaa.

Pohjatyön jälkeen oli aika tehdä jokaisen halutun toiminnon toiminnallisuus. Roam tai satunnainen vaeltelu ennalta määrätyllä alueella toimii siten, että kenttään lisätään navigaatioon tarkoitettu volyymi. Volyymin sisältä otetaan satunnainen piste määrätyltä etäisyydeltä hahmon sijainnista ja käsketään hahmo siirtymään kyseiseen pisteeseen.

Chase eli jahtaaminen toimii aiemmin mainitun navigaatio volyymin sisällä niin, että pelaajan sijainti annetaan suoraan ja komennetaan hahmo liikkumaan kyseiseen pisteeseen.

Patrol eli partiointi oli aiempiin verrattuna monimutkaisempi toteuttaa. Toiminto vaatii erikseen määritellyt pisteet, joiden välillä hahmo kulkee. Hahmolle luodaan integer muuttuja sekä taulukko (array) jonka tarkoituksena on pitää kirjaa annetuista pisteistä. Seuraavaksi hahmolle annetaan komento liikkua taulukosta haettuun pisteeseen. Pisteeseen siirtymisen jälkeen tarkistetaan, onko integer isompi tai yhtä suuri kuin taulukon kokonaisarvo, josta vähennetään arvo yksi, joka on siis se piste mihin on liikuttu. Mikäli arvo on yhtä suuri tai isompi asetetaan integerin arvoksi 0 ja toiminto aloitetaan alusta. Mikäli edellä mainittu ei ole tosi, lisätään integeriin 1 ja suoritetaan toiminto alusta.

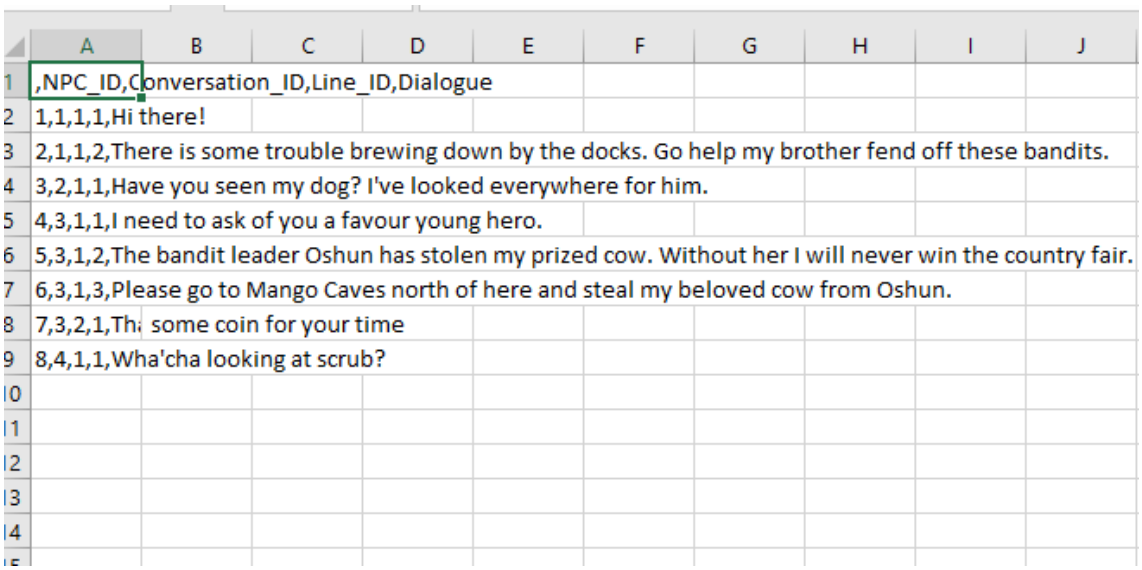
Seuraavaksi tehtiin toiminto, jossa pursuer suorittaa itsemurhahyökkäyksen nähdessään pelaajan. Unrealissa on sisäänrakennettuna hahmojen kuulo- ja näköalue nimeltä PawnSensing (kuva 7). Edellä mainittua hyödyntäen voidaan hahmo komentaa juoksemaan nähdyn objektin viereen ja tuhoamaan itsensä, tehden samalla vahinkoa määriteltyyn objektiin. Toimintoon käytettiin vain ”näkökykyä” eikä reagointia ääniin mikä sekin olisi mahdollista. Tämä on toteutettu siten, että pursuerin ”nähdessä” pelaajan, vaihdetaan aiemmin annettu patrol tai roam toiminto chaseksi. Pursuer jahtaa pelaajaa niin pitkään, kun pelaaja pysyy näkökentässä. Saavuttaessaan annetun etäisyyden pelaajasta, pursuer tuhoaa itsensä. Pelaajan on mahdollista onnistua pakenemaan, jolloin Chase vaihdetaan takaisin siihen toimintoon, joka oli aktiivisena ennen Chasen käynnistymistä. Pakeneminen toteutuu silloin, jos pelaaja onnistuu katkaisemaan näköyhteyden tai liikkumaan pursuerille asetetun liikkumisalueen ulkopuolelle.



Kuva 7. Pursuerin näkö- ja kuuloalue.

5.1.3 Keskusteleva NPC

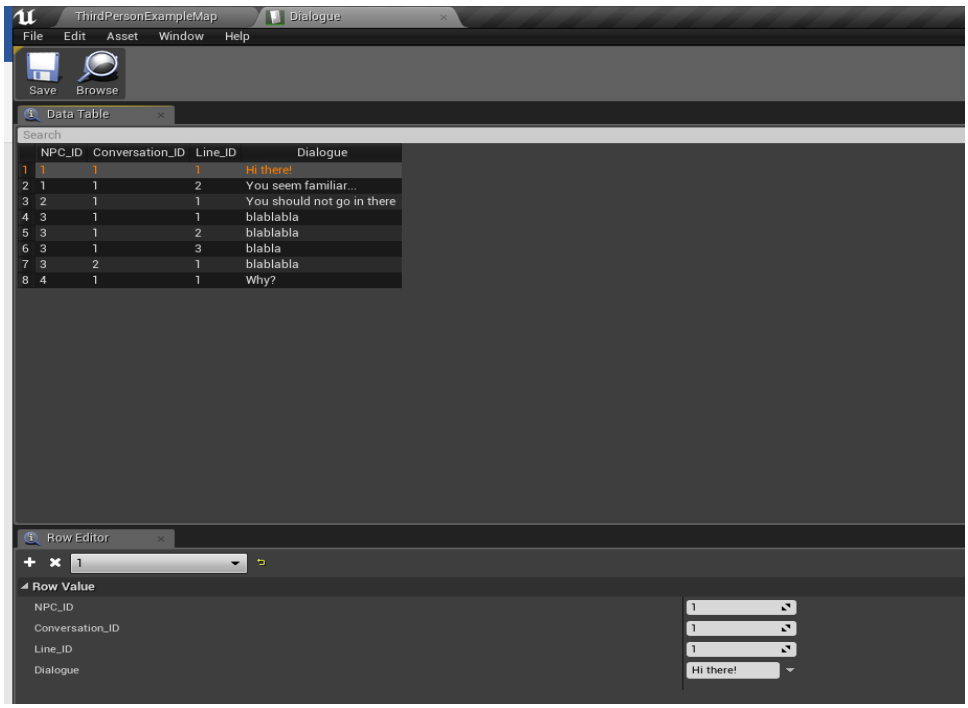
NPC tai Non Playable Character on mahdollista tehdä usealla tavalla. Päädyin tekemään toiminnallisuuden niin, että NPC:n kopioiminen olisi mahdollista. Toteutuksen ja jatkokehityksen kannalta valinta on hyvä ratkaisu, koska se mahdollistaa keskustelujen helpon muokkaamisen ilman Blueprintien muutoksia. Lyhykäisyydessään tämä toimii siten, että Excelillä luodaan kuvassa 8 näkyvä .csv-tiedosto. Tiedosto itsessään koostuu seuraavista tiedoista: NPC_ID, Conversation_ID, Line_ID ja Dialogi. Edellä mainituille tiedoille annetaan arvot sen mukaan, mitä halutaan tehdä.



	A	B	C	D	E	F	G	H	I	J
1	,NPC_ID,Conversation_ID,Line_ID,Dialogue									
2	1,1,1,1,Hi there!									
3	2,1,1,2,There is some trouble brewing down by the docks. Go help my brother fend off these bandits.									
4	3,2,1,1,Have you seen my dog? I've looked everywhere for him.									
5	4,3,1,1,I need to ask of you a favour young hero.									
6	5,3,1,2,The bandit leader Oshun has stolen my prized cow. Without her I will never win the country fair.									
7	6,3,1,3,Please go to Mango Caves north of here and steal my beloved cow from Oshun.									
8	7,3,2,1,Th some coin for your time									
9	8,4,1,1,Wha'cha looking at scrub?									
10										
11										
12										
13										
14										

Kuva 8. Excelin tiedosto, jota hyödynnetään keskusteluissa.

UE:n puolella luodaan uusi structure eli rakenne keskustelulle. Rakenteen tulee sisältää samat arvot, niissä muodoissa, kun ne ovat excelissä, esim. NPC_ID on integer ja Dialogi puolestaan Text. Seuraavaksi luodaan uusi Data Table johon .csv yhdistetään. Yhdistämisen jälkeen Data Table näyttää tältä (kuva 9).

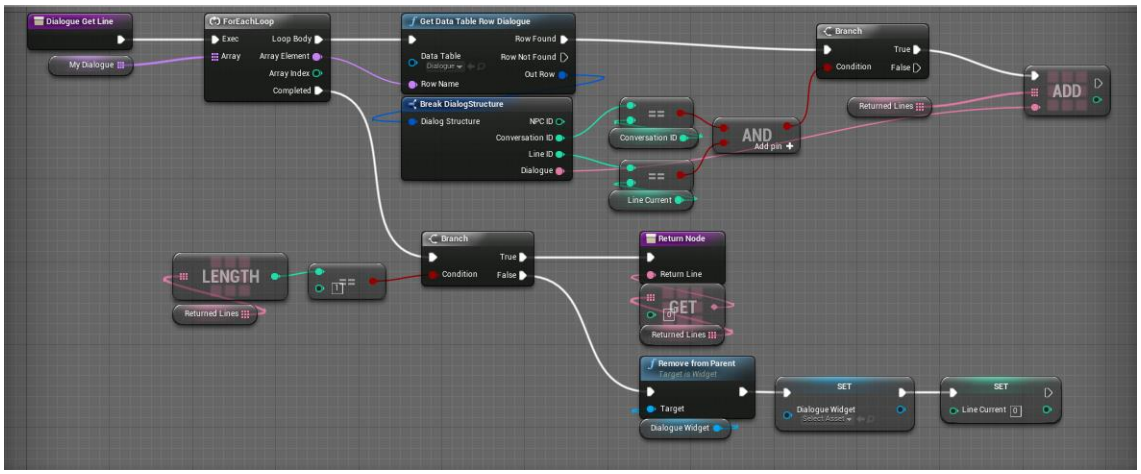


Kuva 9. Data Table jossa voidaan muokata dialogia.

Yhdistäminen suoritetaan NPC-hahmon Blueprintissä valitsemalla node ”event beginplay” joka siis aloittaa toimintansa ohjelman käynnistyttyä. Haetaan Data Tablesta dialogien rivien nimet, suoritetaan looppi jossa haetaan ja jäsennetään dialogi vastaamaan hahmolle annettua ID:tä, nämä tallennetaan taulukkoon. Seuraavaksi luodaan widget eli ruudulle ilmestyvä objekti, joka tässä tapauksessa on tekstilaatikko. Widgetille annetaan muuttuja, joka toimii dialogitekstin vastaanottajana. Viimeiseksi luodaan uusi Blueprint interface joka on vain luku-tiedosto. Edellä mainittuun tiedostoon annetaan yksinkertainen funktio puhumiselle.

Pohjatyön jälkeen voidaan aloittaa itse toiminnallisuuden luominen. NPC-hahmolle luodaan funktiot dialogin luonnille sekä dialogin vuorosanojen hakemiselle. Dialogin luonti (kuva 10) toimii tässä prototyypissä siten, että ensiksi tarkastetaan dialogia varten luodun widgetin olemassaolo eli onko se ruudulla vai ei. Mikäli widget on ruudulla, haetaan widgetissä olevaa dialogille tarkoitettua muuttujaa ja liitetään se widgettiin. Mikäli widget ei ole ruudulla tehdään edellä mainitun lisäksi widgetin luonti sekä tuominen ruudulle.

Seuraavaksi lisätään dialogin vuorosanojen haku. Tämä suoritetaan omana funktionaan jossa loopin sisällä tarkastetaan keskustelun ID sekä mitä riviä pelaajalle näytetään. Näytetyt rivit lisätään taulukkoon, joka tulostaa ruudulla olevaan widgettiin keskustelun seuraavan rivin, mikäli sellainen on olemassa.



Kuva 10. Dialogin toiminnallisuus.

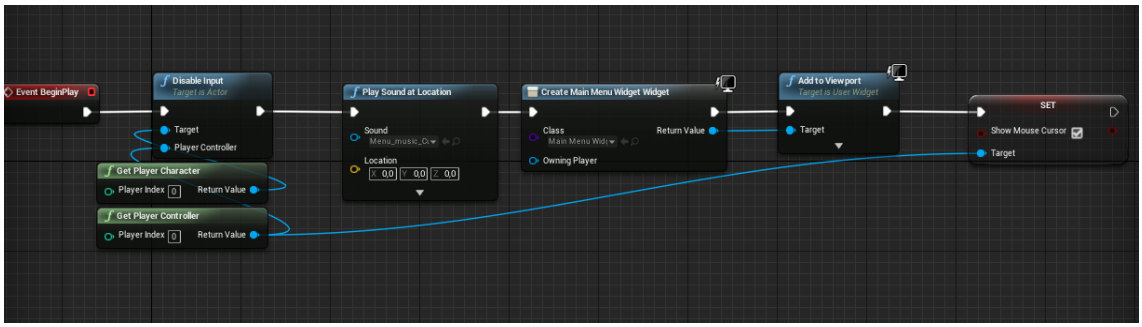
Keskustelut ovat nyt helposti muokattavissa ja ID:tä vaihtamalla voidaan valita mitä kukin NPC sanoo. NPC:n toiminnallisuuteen lisättiin vielä komennot

kääntyä pelaajaa kohti keskustelun alkaessa sekä dialogilaatikon (widgetin) poistamisen ruudulta, jos pelaaja päättää poistua kesken keskustelun.

5.2 Kentät ja valikot

Kentän rakentaminen aloitettiin avaamalla UE:n tarjoama pohja. Pohjaa käytetään pakkaamattoman version menuna. Tämä suoritetaan luomalla uusi gamemode jolla templatessa olemassa oleva korvataan. Korvauksen jälkeen, kun prototyyppi käynnistetään tässä kentässä, ruudulle luodaan alkuvalikko ja pelaajalle annetaan mahdollisuus käynnistää tai sulkea peli.

Alkuvalikko luodaan aiemmin raportissa mainitulla menetelmällä, jossa luodaan widget-Blueprint mihin lisätään halutut visuaaliset elementit ja toiminnot painikkeille. Widgetin luonnin jälkeen luodaan erillinen Blueprint joka sisältää komennot tuoda menu ruudulle (kuva 11).



Kuva 11. Päävalikon luominen.

Menun valmistumisen jälkeen, prototyyppiin luotiin täysin uusi kenttä, joka siis toimii lopullisena pelialueena. Kenttään lisättiin halutun kokoinen alue maata, jota kuitenkin on mahdollista kasvattaa tai pienentää tarpeen mukaan. Maan lisäämisen jälkeen aloitettiin kentän rakentaminen UE:n tarjoamilla muodoilla. Muotoihin kuuluvat perinteiset geometriset muodot.

Tavoitteena oli rakentaa yksi suhteellisen laaja rakennus, jonka sisällä prototyyppi tapahtuisi. Rakennusten sisälle sijoittuvissa peleissä on etuna, ettei

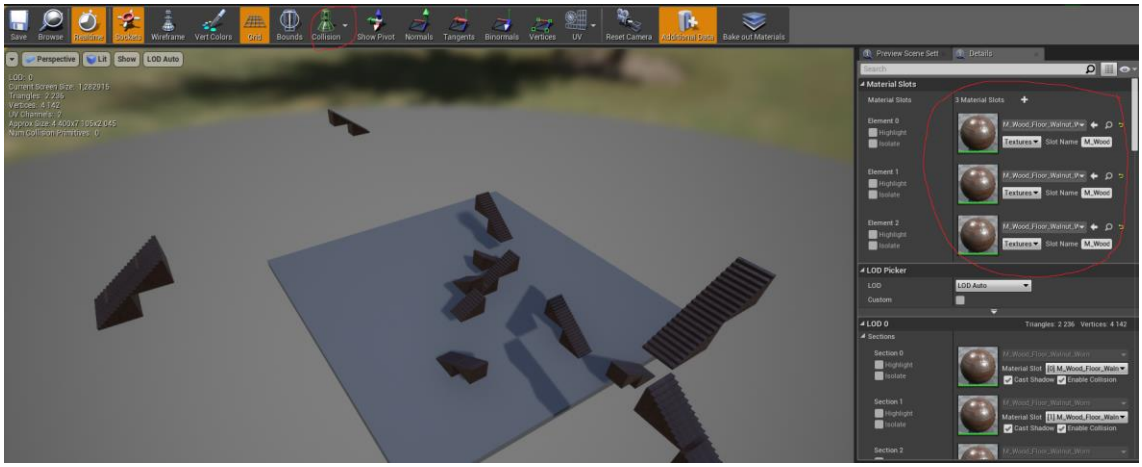
rakennuksen ulkopuoli (alue, jota pelaaja ei näe) tarvitse olla valmis tai siloteltu, kuten kuvasta 12 on nähtävissä.

Kuva 12. Rakennus kuvattuna ulkopuolelta.

Rakennuksen hahmottelu aloitettiin lattiasta. Lattia tehtiin ensin, jotta saatiin kuva siitä miten laaja ja minkä muotoinen rakennuksesta tulee. Rakennuksen muodon ja koon ollessa selvät oli aika asetella seinät. Lattioihin ja seiniin käytettiin samaa paneelia (muotoa), seinän tekstuuriksi valittiin leikattu kivi ja lattiaan puu. Lattian ja seinän eriävät tekstuurit ovat lähinnä ulkoasullinen muutos eikä sillä ole toiminnallista merkitystä. Seiniä asetellessa on viimeinen mahdollisuus vaikuttaa rakennuksen ulkomuotoon ja luonnollisesti huoneiden lukumäärään. Halusin rakennukseen vähintään kaksi kerrosta, joten seiniä asetellessa tuli huomioida myös portaiden sijainnit. Toiseen kerrokseen vievät portaat päätettiin sijoittaa geneerisesti ensimmäiseen huoneeseen ulko-oven jälkeen.

Portaat olivat aluksi kierreportaat, mutta skaalaaminen aiheutti useita ongelmia kollisioverkon ja hahmon liikkumisen välillä, että päädyin käyttämään suoria portaita. Portaikot luotiin käyttämällä brusheja joista luotiin staattisia meshejä. Meshin luonnin jälkeen portaisiin tulee asettaa kollisio verkko, jotta pelaaja pystyy käyttämään portaita. Mikäli kollisiota ei ole asetettu, hahmot pystyvät kävelemään objektin sisään. Kollisiot asetetaan avaamalla luotu mesh ja valitsemalla haluttu kollisio menusta. Meshin valikoista (kuva 13) saa myös

asetettua halutut tekstuurit nopeasti drag & dropin sijaan, tämä on suositeltavaa, jotta tekstuureja ei tarvitse asettaa pinta kerrallaan.

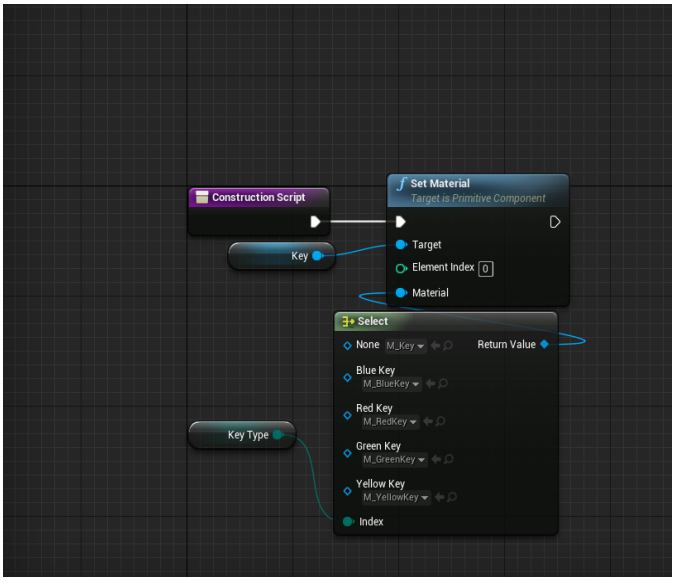


Kuva 13. Kentän portaikkojen tekstuurit ja kollisio.

Pelattavan alueen muotojen ollessa selvät ja portaat paikallaan, aloitettiin kentän toiminnallisuuksien lisääminen. Ensiksi tyhjiin oviaukkoihin lisättiin ovet, joita pelaaja voi aukaista ja sulkea, mikäli omistaa oikean avaimen. Ovet luodaan tekemällä uusi Blueprint jonka viewportiin lisätään karmit, itse ovi ja laatikko kollisio. Ajan säästämiseksi ovien tuli olla sellaisia, että samaa ovea pystyttiin kopioimaan. Kopioinnin mahdollistaminen tehtiin luomalla kentän omaan Blueprintiin toiminto, joka hakee kaikki kentässä olevat ovet ja tarkastaa minkä oven kanssa ollaan vuorovaikutuksessa. Tällä tavalla kaikki kopioidut ovet eivät avaudu samaan aikaan, kun pelaaja on vuorovaikutuksessa oven kanssa. Oven avautuminen vaatii animaation, joka toistetaan, kun pelaaja avaa tai sulkee oven. Kenttään luotiin uusi animaatio(matinee), johon asetettiin aika mikä avaamiseen menee sekä oven rotaation alku- ja loppupiste. Matinee toistaa rotaation alkupisteestä loppupisteeseen annetussa ajassa.

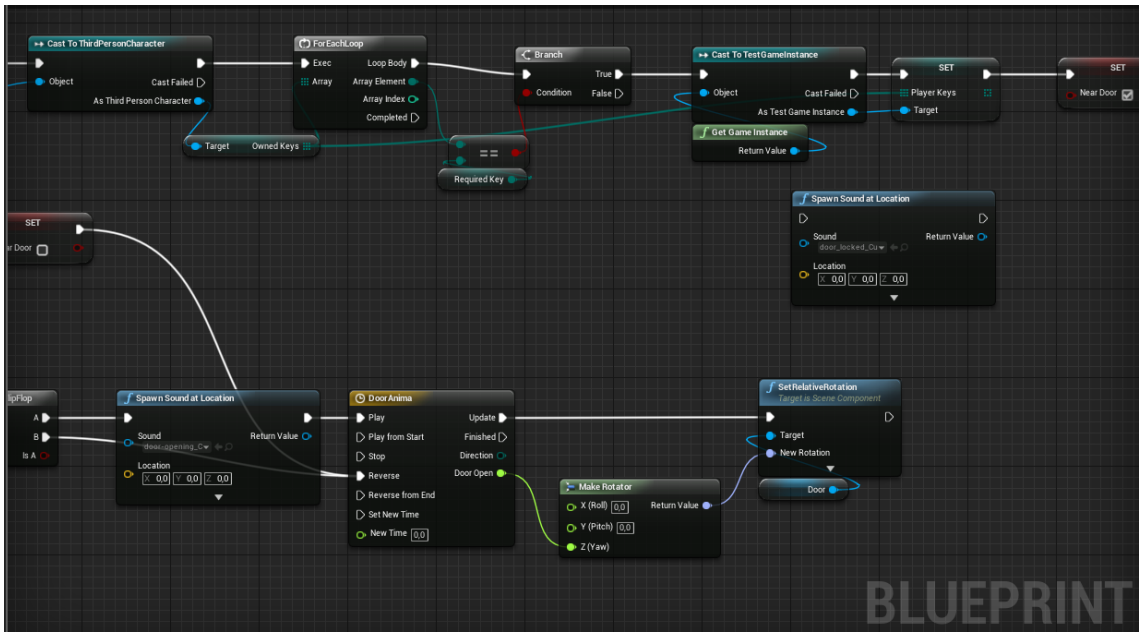
Tässä vaiheessa luotiin uusi lista (enumeration) johon tallennettiin halutut avaimet. Avainten listauksen jälkeen luotiin uusi luokka, joka nimetään yksinkertaisesti avaimeksi. Prototyypissä avaimen virkaa toimittaa pallo, joka sisältää kollision, jotta avain voidaan ”kerätä”.

Avaimen funktioihin asetettiin materiaalin vaihto (kuva 14), joka haetaan aiemmin tehdystä listasta. Tämä mahdollistaa avaimen tyypin muuttamisen suoraan kentästä. Projektiin luotiin uusi instanssi, johon lisättiin muuttujaksi pelaajan omistamat avaimet. Instanssin luomisen jälkeen palattiin takaisin avaimen Blueprintiin. Pelaajan osuessa avaimen kollisioon lisätään avain pelaajan omistamiin avaimiin ja tuhotaan kentässä oleva avain. Ruudulle lisätään widget joka ilmoittaa pelaajalle, että avain on kerätty.



Kuva 14. Avaimen tyypit.

Palataan takaisin oven Blueprintiin (kuva 15) jossa luotiin ovelle uusi muuttuja, joka toimii tarkastuksena voiko pelaaja avata ovea, tälle annettiin nimeksi ”near door”. Pelaajan astuessa oven kollisiolaatikkoon tarkastetaan, onko pelaajan hallussa oikeaa avainta, mikäli avain löytyy, asetetaan muuttuja ”near door” todeksi. Avain tarkastetaan muuttujalla, jonka arvoa verrataan listaan tallennettuihin avaimiin.



Kuva 15. Oikean avaimen tarkistus ja muuttujan tilan vaihtaminen.

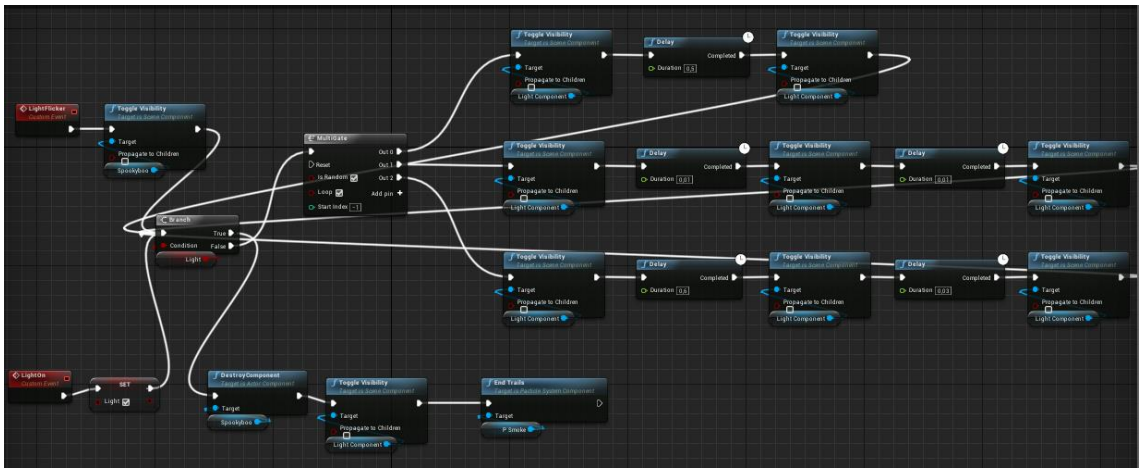
Luotiin uusi tapahtuma, jossa tarkastetaan ”near door” muuttujan tila. Muuttujan ollessa tosi luodaan ääni ja aloitetaan avaamisen animaatio sekä asetetaan relatiivinen rotaatio, jotta ovi aukeaa. Lopuksi lisättiin uusi toiminnallisuus, jossa pelaajan poistuessa oven kollisiolaatikosta asetetaan muuttuja (near door) epätodeksi ja suoritetaan animaatio käänteisesti, jolloin ovi sulkeutuu. Prototyyppi sisältää myös kenttien välisen teleportaation niin, että hahmon keräämät avaimet säilyvät. Tämä toteutettiin yksinkertaisesti säilömällä ja hakemalla avaimet luodusta instanssista ennen ja jälkeen teleportaation.

Ovien ollessa paikoillaan oli aika lisätä kenttään sisältöä. NPC -hahmot ja viholliset sijoitettiin halutuille paikoille. Avaimet sijoiteltiin siten, että se rajoittaa hiukan pelaajan vapautta liikkua ja edetä. Ensimmäisen oven jälkeen lisättiin kollisiolaatikko, joka poistaa oven ja vangitsee pelaajan rakennukseen, naurun säestämänä.

5.3 Tapahtumat

Kentässä olevien tapahtumien aloitus on sijoitettu kentän omaan Blueprintiin. Pääosin tapahtumat alkavat, kun pelaaja osuu kenttään sijoitettuun kollisiolaatikkoon ja loppuvat tai muuttuvat pelaajan joko poistuessa laatikosta tai osuttua seuraavaan laatikkoon. Itse tapahtumat on luotu erillisiin Blueprinteihin, mikäli tapahtuma vaati useampia nodeja. Tämä pitää kentän oman Blueprintin hitusen siistimpänä ja nopeuttaa työskentelyä, kun muokattavaa osiota ei tarvitse etsiä useiden komentojen joukosta.

Yksi prototyypissä käytetyistä tapahtumista (kuva 16) on hahmon ilmestyminen käytävään. Tämä on toteutettu luomalla uusi Blueprint johon lisättiin komponenteiksi spottivalo, partikkeliefekti sekä itse hahmo. Valo ja partikkeliefekti ovat näkyviä mutta hahmo asetettiin näkymättömäksi. Blueprintin puolelle lisättiin node joka muuttaa hahmon näkyväksi valon vilkkumisen tahtiin.



Kuva 16. Valon välkkyminen.

Valon vilkkuminen toteutettiin ajastimilla ja valon tilaa säätävillä nodeilla, näitä tehtiin kolme erilaista versiota, joiden ajastimissa on eri ajat. Nämä vaihtoehdot kiinnitettiin porttiin, joka sattumanvaraisesti valitsee toteutettavan version. Valitun version suoritettuaan, portti valitsee uuden. Tätä toistetaan niin pitkään, kunnes tapahtuma saa käskyn lopettaa. Tapahtumien käynnistyminen on toteutettu kentän Blueprintissä jossa kutsutaan tehtyä tapahtumaa, kun pelaaja astuu tietylle alueelle. Tapahtuma lopetetaan pelaajan osuessa kollisiolaatikkoon, jolloin toiminto tuhotaan, valon näkyvyys vaihdetaan pysyvästi näkyväksi ja partikkeliefektin toisto lopetetaan. Edellä mainitun lisäksi kentän Blueprintissä haetaan kenttään ennalta asetetun pisteen sijainti

maailmasta, johon luodaan pursuer. Lopuksi tuhotaan olemassa olleet kollisiot, jotta tapahtumat eivät toistu aina, kun pelaaja saapuu alueelle.

Muissa prototyypissä olevissa tapahtumissa objekteja lisätään tai poistetaan maailmasta, kun pelaaja osuu kollisioon. Lisäksi prototyyppi sisältää tapahtuman missä pelaaja teleportataan pisteestä x aiemmin valittuun pisteeseen y.

5.4 Visuaaliset asetukset

Skyspheren oma valonlähde muutettiin kaiken kattavasta valosta suunnattuun valonlähteeseen. Suunnattu valonlähde on asetettu niin, että valonsäteet pääsevät sisään rakennuksen itäisistä ikkunoista, tämä on nähtävissä kuvassa 17.

Kuva 17. Ikkunoista tulevat valonsäteet.

Valonlähteen väriä säädettiin oranssihtavaksi sekä varjojen luonti asetettiin käyttöön. Taivaan väritys on sidottu auringon sijaintiin taivaalla ja auringon sijaintia muutettiin vastaamaan iltapäivää oikeanlaisen värityksen saamiseksi.

Kenttään myös lisättiin sumuefekti, joka vahvistaa valon näkyvyyttä ja rajoittaa näköetäisyyttä ilman valoa. Sumun päätarkoitus onkin toimia valoa johtavana tekijänä, jolloin tapahtumiin sidotut valot korostuvat.

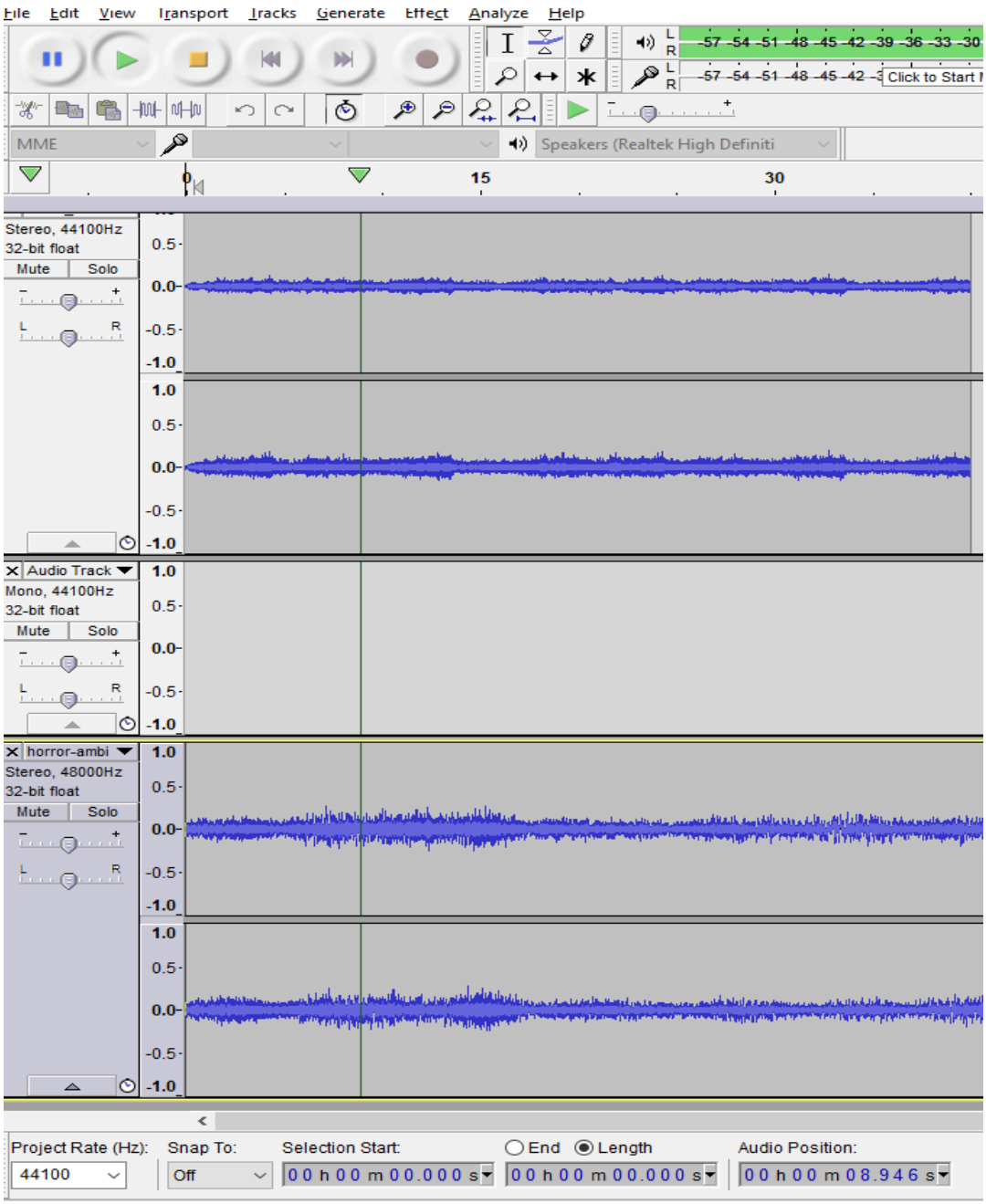
6 Enginen ulkopuoliset elementit

Luvussa käydään pikaisesti läpi Unreal Enginen ulkopuolisia asioita, joita prototyypissä on käytetty.

6.1 Äänet

Äänet etsittiin sivuilta, joissa tarjottavat äänet ovat ns. rojalti vapaita. Käyttäjät lisäävät itse äänittämiään ääniä sivustolle, josta kuka tahansa voi käyttää kyseisiä ääniä lähes vapaasti. Äänien lisääjät voivat esimerkiksi vaatia nimeään näkyviin sen työn lopputeksteihin, jossa ääniä käytetään. Kaikki käyttäjät eivät kuitenkaan vaadi mitään vaan tuottavat sisältöä auttaakseen muita.

Äänien etsimisen jälkeen suoritettiin äänten muokkaus (kuva 18), joka prototyypin tapauksessa tarkoitti usein kahden ääniraidan yhdistämistä, taajuuksien säätämistä ja kompressoimista.

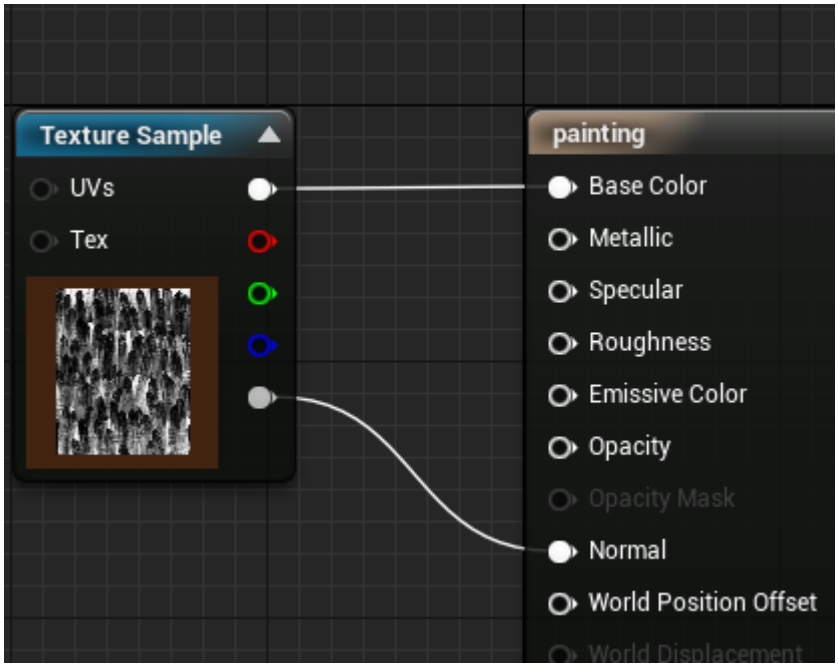


Kuva 18. Kahden ääniraidan yhdistäminen.

Muokkaamisen jälkeen äänet tallennettiin muotoon .wav ja importoitiin projektiin. Projektiin lisäämisen jälkeen äänistä tuli luoda erikseen cue, jotta äänet saatiin toimimaan halutusti.

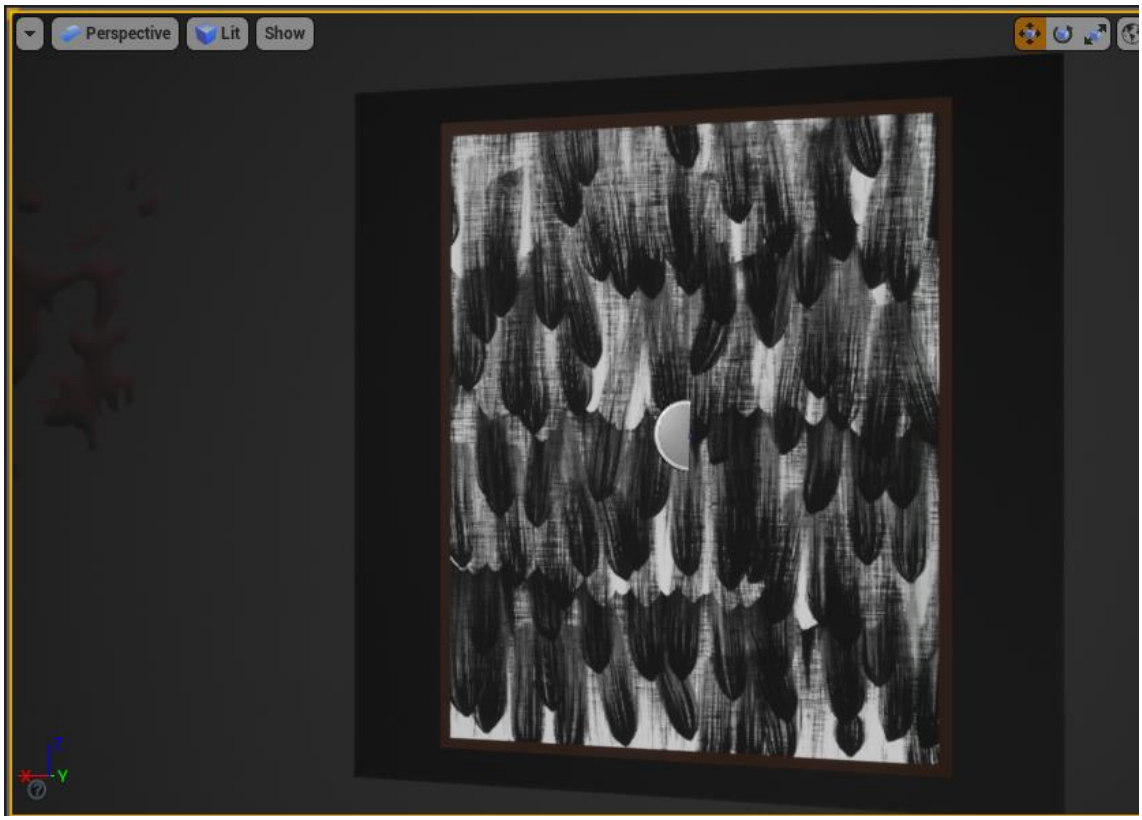
6.2 Decalit

Decalit eli ”siirtokuvat” luotiin photoshoplla. Käytännössä ohjelmalla luotiin kuva, joka importoitiin projektiin. Kuvan tulee sisältää alphakerros tai alphakerroksen tulee olla omana tiedostonaan, jotta kuva toimii oikein.



Kuva 19. Decalin kytkennät.

Tiedostojen importoinnin jälkeen tiedostosta luotiin materiaali, jonka kytkennät näkyvät kuvassa 19. Lähes aina alphakerros liitetään normaaliin pinniin. Materiaalin asetusten ollessa halutut, kenttään asetettiin decal, jonka materiaaliksi kuva annettiin. Lopputuloksena on siirrettävä kuva, jolla voi luoda yksityiskohtia maailmaan (kuva 20).



Kuva 20. Decal asetettu kenttään.

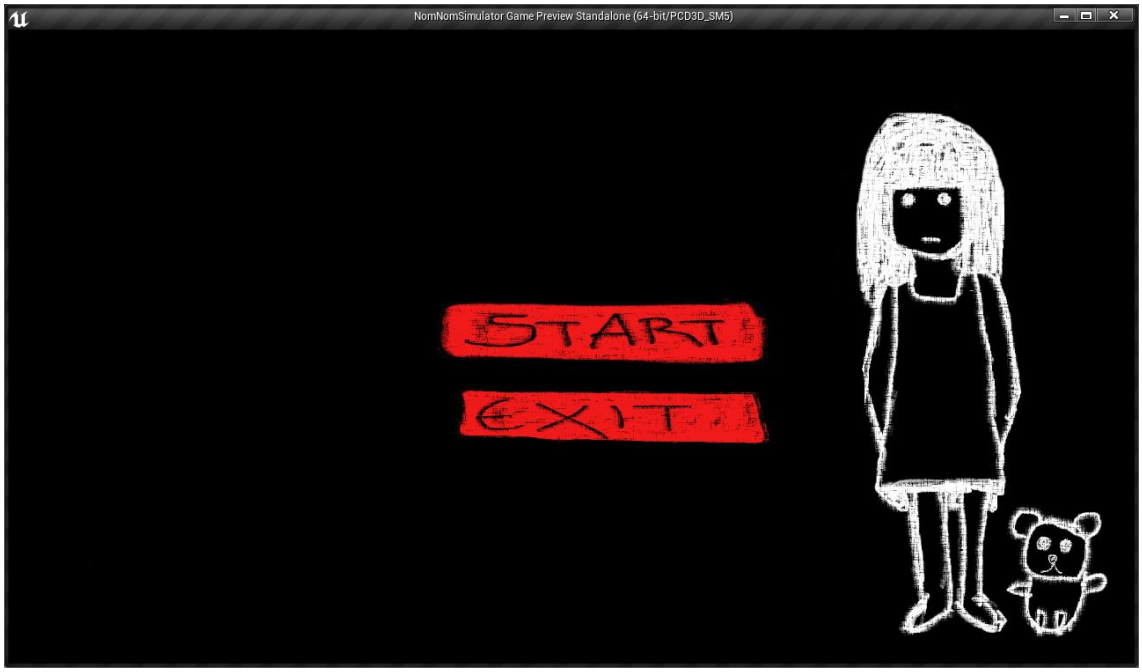
Mikäli decaliin ei halua mustaa taustaa on alphakerros luotava siten, että vain näkyviin haluttava osa on korostettuna.

7 Lopputulos

Luvussa esitellään kehityksen lopputulos. Prototyypin ison alueen takia, tuloksissa esitetään vain oleelliset tapahtumat ja mekaniikat.

Prototyyppiä ei ole erikseen pakattu suoraan käynnistyvään muotoon. Prototyypin asetukset on asetettu siten**,** että pelin voi käynnistää vakiona avautuvasta ikkunasta. Käynnistämisen jälkeen ruutuun avautuu päävalikko, jossa on mahdollista joko käynnistää tai sulkea peli. Päävalikon (kuva 21)

taustalla soi musiikki ja painikkeet vaihtavat väriä, kun hiiri viedään painikkeen päälle.

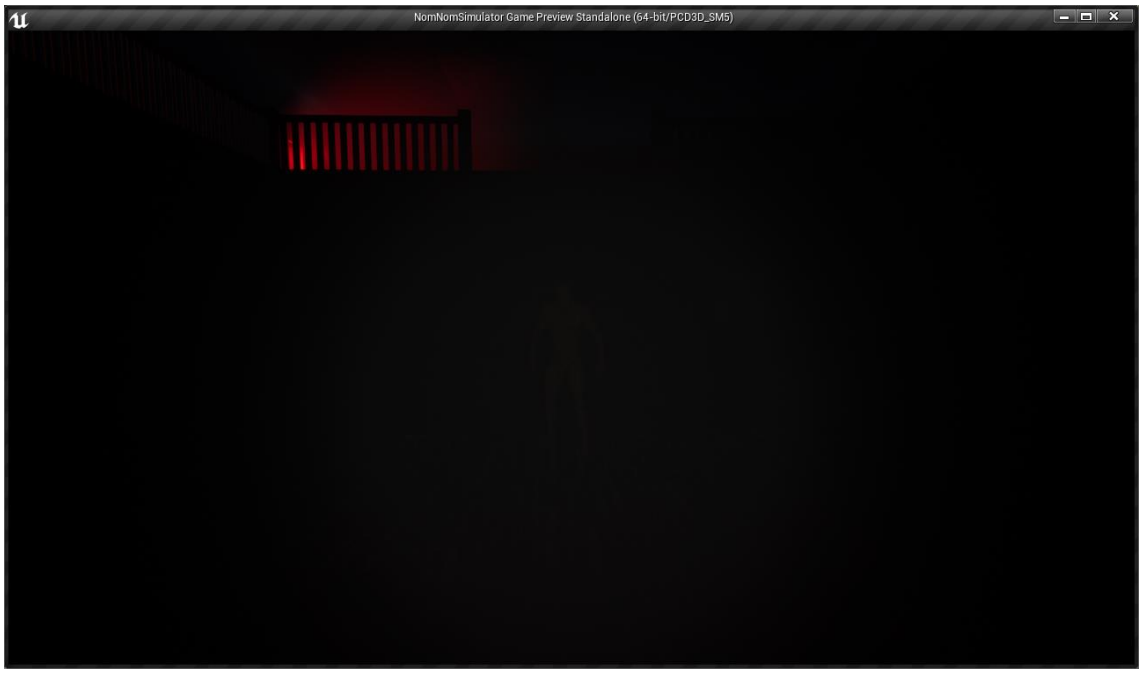


Kuva 21. Prototyypin päävalikko.

Pelaajan valitessa pelin aloitus, ladataan pelattava kenttä. Erillistä latausruutua ei prototyypissä ole, joten pienehkön ajan jälkeen pelialue aukeaa välittömästi. Pelaajaa tervehditään kylteillä, jotka sisältävät ohjeita (kuva 22). Näppäinten lisäksi pelaajalle kerrotaan pallojen toimivan avaimina, sekä ohjeistetaan avainten kerääminen.

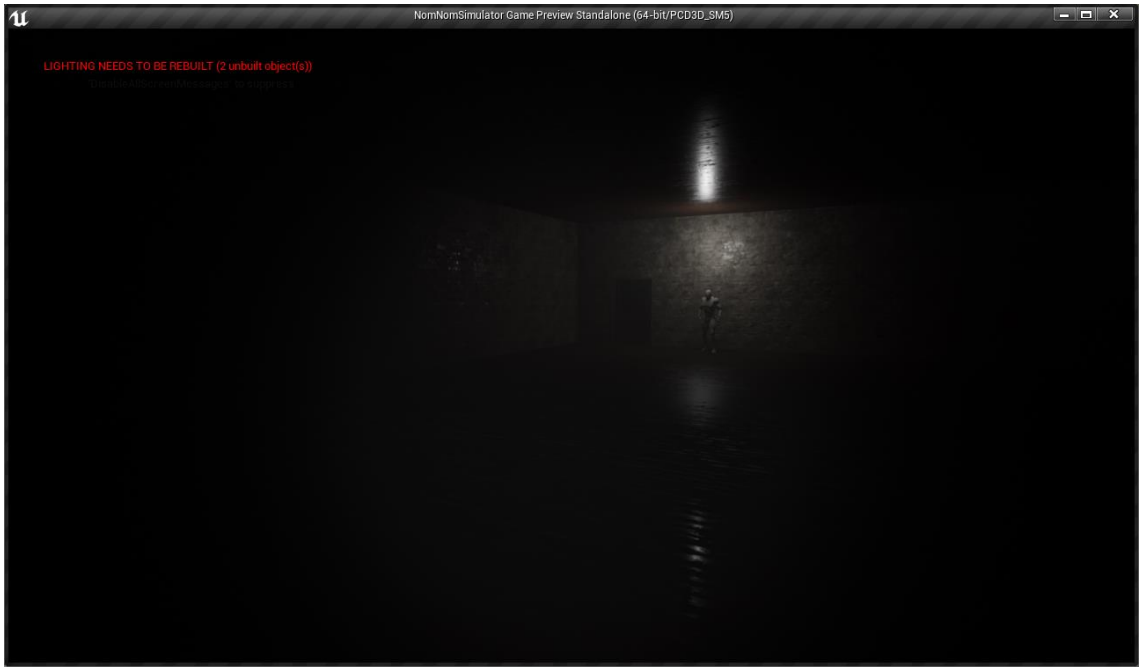
Kuva 22. Ohjekyltit ja tie ovelle.

Pelaaja ohjataan kylttien ja tien avulla todellisen pelialueen ovelle. Rakennuksen ulkopuolinen vaeltaminen ei ole mahdollista sillä pelaaja tapetaan välittömästi, mikäli tien läheisyydestä poistutaan. Pelaajan astuttua ovesta sisälle (kuva 23) aloitetaan taustamusiikin toistaminen. Ensimmäinen havaittava asia on portaiden päässä oleva valo sekä hahmo. Tästä eteenpäin pelaaja saa liikkua rakennuksessa täysin vapaasti. Hahmoa lähestyttäessä hahmo häviää ilmaan huokauksen saattelemana. Pelaajan liikkuessa tietyn matkan päähän ovesta, oven sijainnista kuuluu naurua ja ovi korvataan seinällä.



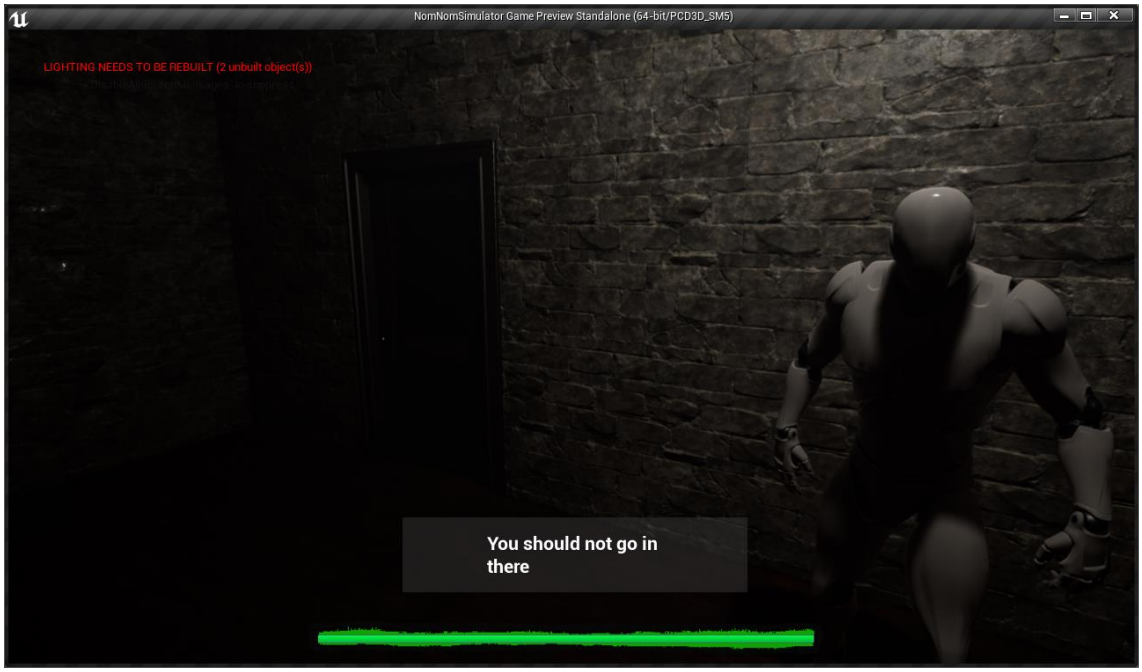
Kuva 23. Ensimmäinen näkymä rakennukseen astumisen jälkeen.

Todennäköisesti pelaaja reagoi nauruun kääntymällä ympäri ja huomaa oven kadonneen. Oven katoamisen lisäksi ympärilleen katsominen paljastaa oven vasemmalla puolella olevan pöydän sekä oven oikealla puolella olevan hahmon, joka näkyy kuvassa 24.



Kuva 24. Näkymä oven oikealta puolelta.

Oikealla puolella oleva hahmo ei katoa lähestyttäessä. Hahmolle voi jutella, jolloin hahmo kääntyy pelaajaa kohti ja kertoo ettei ovesta ole suotavaa mennä (kuva 25). Mikäli ovesta päättää mennä sisälle, aloitetaan lapsien vaikerointi ja tehosteäänet. Ovi aukeaa käytävään, joka sisältää kaksi ikkunaa. Ikkunat on sijoitettu siten, että ne päästävät valonsäteet sisään ja heikentävät taskulampun tehoa. Käytävän päässä näkyy savua, vilkkuva valo ja hahmo, joka välkkyy näkymättömästä näkyväksi lampun tahdissa.

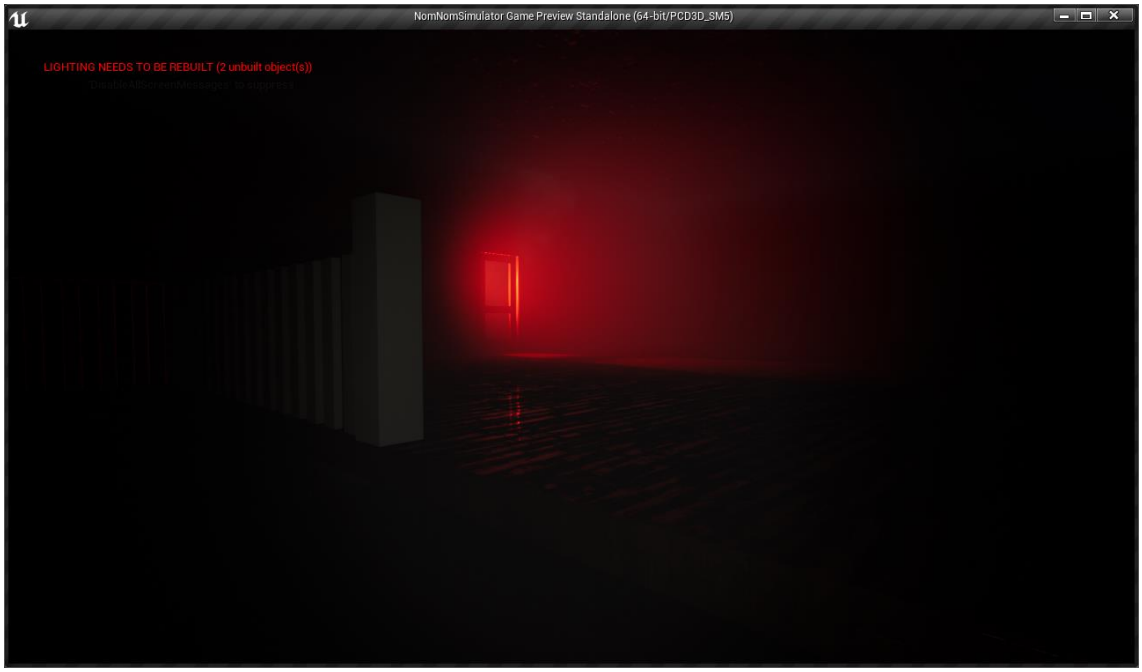


Kuva 25. NPC varoittaa pelaajaa.

Mikäli pelaaja päättää edetä käytävää pitkin hahmon luokse, hahmo sekä savu haihtuvat ilmaan paljastaen oven. Pelaajalla ei vielä tässä vaiheessa ole mahdollista omistaa avainta kyseiseen oveen, joten se pysyy suljettuna. Lähestyttäessä ovea pelaajan selän taakse luodaan raportissa aiemmin mainittu pursuer. Pursuer tappaa pelaajan, jolloin ruutuun ilmestyy valikko (kuva 26), jossa peli on mahdollista lopettaa tai käynnistää uudelleen.

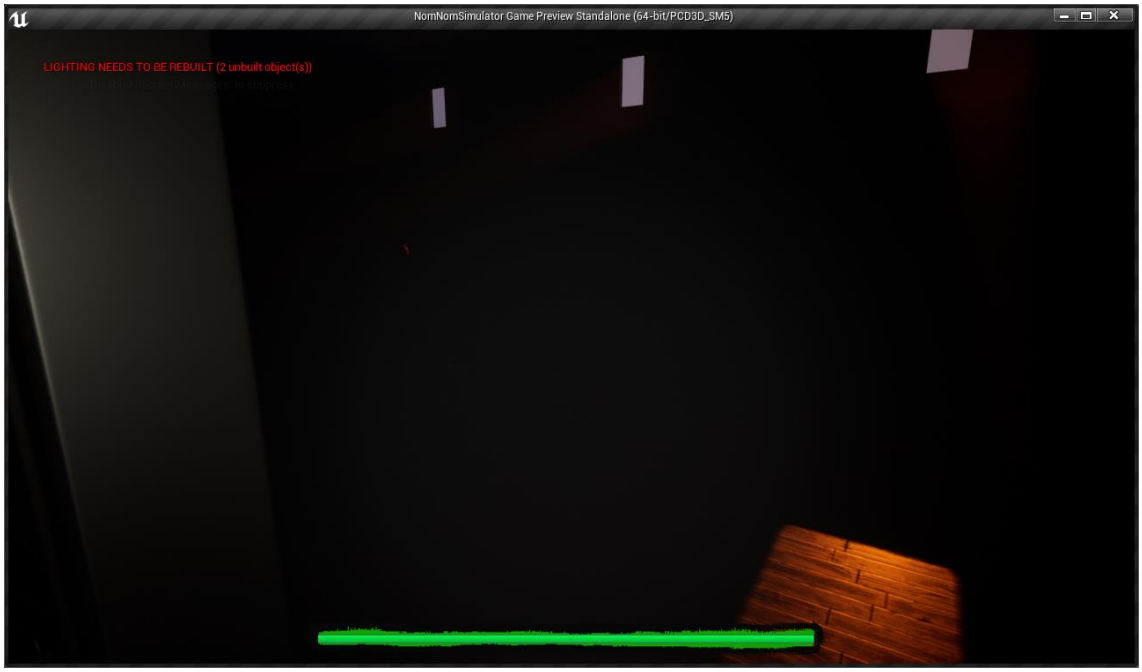
Kuva 26. Kuolemisen jälkeinen valikko.

Mikäli pelaaja joko heti ensimmäisellä kerralla tai kuolemasta viisastuneena päättää kävellä pääoven jälkeisiä portaita ylös, aloitetaan naisen tuskanhuudot yhdistettynä oven paukuttamiseen. Tuskanhuutojen lopuksi toistetaan murinaa, jolloin oven paukuttaminen lakkaa. Äänet on asetettu samoin kuin pääoven katoamisessa, joten pelaaja pystyy kuulemaan äänen lähtevän tietystä pisteestä. Kuvassa 27 näkyy äänien lähtöpisteeksi määritelty ovi.



Kuva 27. Ovi, joka on asetettu äänien lähtöpisteeksi.

Rakennuksen toinen kerros on myös täysin tutkittavissa, vaikka sisältöä parvekkeen, toisen NPC:n ja punaisen huoneen lisäksi ei oikein olekaan. Punaiseen huoneeseen, josta huudot kuuluivat ei myöskään pääse ilman avainta. Pelaaja joutuukin etsimään avaimet rakennuksesta. Pelaajan löytäessä oikean avaimen oikeaan oveen voidaan maailmassa edetä. Yksi avain on asetettu huoneen nurkkaan (kuva 28), jonka sisällä juoksentelee pursuer. Pursuer aloittaa pelaajan jahtaamisen, mikäli pelaaja kävelee hahmon näkökenttään. Karkuun juokseminen avaimen hakemisen yhteydessä on vaihtoehto, mutta nykyisellä vaikeusasteella se on lähes mahdotonta.



Kuva 28. Avain näkyy hämärästi huoneen perällä.

Ennemmin tai myöhemmin pelaaja onnistuu keräämään ensimmäisen avaimensa. Tämä mahdollistaa etenemisen pääoven vasemmalla puolella olevasta ovesta, joka johtaa rakennuksen alla olevaan sokkeloon (kuva 29). Sokkelo sisältää useita mahdollisia reittejä, joista luonnollisesti vain yksi ei ole umpikuja. Alue on rakennettu siten, että eksyminen on äärimmäisen todennäköistä ja useat reitit risteytyvät keskenään.

Kuva 29. Sokkelo kuvattuna ulkopuolelta maan alta.

Pelaajan läpäistessä sokkelon**,** portaikon päässä odottaa jälleen katoava hahmo, avain sekä objekti. Sokkelosta löytyvä avain kuuluu punaiselle huoneelle. Pelaajalla on nyt mahdollisuus joko peruuttaa takaisin tai lähestyä objektia. Objekti (kuva 30) teleporttaa pelaajan huoneeseen, joka sijaitsee sen käytävän takana, jossa vilkkuva hahmo oli.

Kuva 30. Teleportin objekti.

Huoneesta löytyy prototyypin viimeinen avain, joka avaa oven käytävään. Mikäli käytävään (kuva 31) astuu tältä puolelta, vilkkuva hahmo, pursuer tai äänet eivät ole nähtävissä tai kuultavissa**.**

Kuva 31. Käytävä, jossa pursuer tappaa pelaajan.

Käytävään menemisen sijaan pelaajalla on myös mahdollisuus jatkaa ns. maaliin, joka sijaitsee käytävän oven oikealla puolella. Portaiden päässä on uusi objekti joka teleporttaa pelaajan kokonaan uuteen kenttään. Uusi kenttä ei toistaiseksi sisällä muuta kuin avaimen ja teleportin takaisin pelattavaan kenttään. Kerätyt avaimet säilyvät instanssien vaihdosta huolimatta, mutta kentästä löytyvälle avaimelle ei ole vastaavaa ovea.

Kuva 32. Pelaajan viimeinen etappi.

Pelaaja on nyt kuvan 32 limbossa, josta ei voi paeta kuin sulkemalla peli. Kehityksen tässä vaiheessa, tämä lasketaan prototyypin läpäisyksi.

8 Yhteenveto

Valinta siitä mitä halusin tehdä, oli hyvin helppoa. Vaikeus kuitenkin tuottaa sellainen prototyyppi oli tiedossa alusta alkaen. Kauhu nojaa hyvin vahvasti visuaaliseen tarinankerrontaan sekä äänimaailmaan. Näitä en kuitenkaan nykyisillä välineillä tai osaamisella pystynyt tekemään, sillä tasolla kuin haluaisin niiden olevan. Tyydyin siis tekemään teknisen puolen niin valmiiksi, kuin pystyin, jotta tulevaisuudessa voin jatkokehittää prototyypistä valmiin pelin.

Henkilökohtaisella tasolla koin olevani epäonnistunut jo ennen toteutuksen aloitusta. Pelistudioiden tekemiin prototyyppeihin tutustuessani kuitenkin huomasin, että prototyypit ovat yleensä äärimmäisen karkeita lopputuloksia, joissa yleensä testataan yhtä tai kahta toiminnallisuutta. Prototyypeissä olevat kentät eivät myöskään yleensä sisällä tekstuureita, yksityiskohtia, tarinaa tai ääniä. Tästä syystä tekemäni prototyyppi muistuttaa ehkä enemmän alphaa

kuin prototyyppiä, mutta vain hiukan. Lopputulos on työmääräänsä ja aiempaan kokemukseeni pelikehityksestä verraten hyvä, kokemusta kehityksestä ei harjoittelun lisäksi oikeastaan ollut.

Toiminnallisuudet toimivat kuten pitää, muutamia bugeja lukuun ottamatta. Ongelmia aiheutti erityisesti vielä korjaamatta oleva satunnainen bugi juoksuun liittyen. Ensimmäistä kertaa juoksulle tarkoitettua näppäintä painettaessa, prototyypin käynnistyksen jälkeen, juokseminen välillä lukkiutuu päälle, kunnes pelaaja painaa nappia uudestaan. Juoksemisen äänet myös toistuivat väärin, minkä takia ne on toistaiseksi poistettu käytöstä.

Visuaaliset ongelmat olivat runsaampia ja uskon niiden johtuvan kokemuksen puutteesta. Valonsäteiden tunkeutuminen seinien läpi, tekstuurien klippaus sekä varjojen ongelmat tulivat tutuiksi. Varjot ja tekstuurit sain korjattua mutta valonsäteiden pääseminen seinien läpi on yhä mysteeri. Kentän rakentaminen myös tuotti päänvaivaa mutta siitä selvittiin yleisesti hyvin.

Valitsin aihealueen siksi, että pelikehitys kiinnostaa itseäni ja halusin oppia aiheeseen liittyviä asioita. Halusin lisäksi haastaa itseni, joten suhteellisen tuntemattoman ison työn valitseminen tuntui sopivalta. Lähes kaikki mitä projektissa tein oli itselleni täysin tuntematonta ja jouduin opettelemaan asiat lennosta. Vaikka pidän lopputulosta hyvänä, kun huomioon otetaan se, etten aihealueesta kovin paljoa tiennyt, en henkilökohtaisesti ole tyytyväinen. Olisiko haluamani lopputuloksen saavuttaminen edes ollut mahdollista? Todennäköisesti ei. Projekti kuitenkin opetti paljon, mitä pidän suuremmassa arvossa kuin lopputulosta. Projekti eteni hyvin ja nautin tekemisestä suuresti. Prototyypin kehittäminen on opettanut paljon käytetyistä ohjelmista, eri tavoista ratkaista ongelmia ja pitkäjänteisyyttä. Ohjelmien käyttäminen ja erilaiset toiminnot avautuivat entisestään. Osaaminen ei ehkä ole vielä ammattilaisen tasolla, mutta suhteellisen lähellä, mitä ohjelmien käyttöön tulee. Tulevaisuutta ajatellen ohjelmointia tulee opetella lisää sekä rohkeasti kokeilla erilaisia toimintatapoja toiminnoille. Toteutustapojen muuttaminen parantaa myös suorituskykyä, esimerkiksi valojen vilkkumiseen tehty ajastinviritelmä ei ole tehty järkevästi.

Saavutettiinko tavoitteet? Mielestäni kyllä. Kaikki tavoitteet saavutettiin eikä prototyyppi jäänyt pelkästään kasaksi keskeneräisiä toiminnallisuuksia. Kaikki toimivat yhteen kuten pitää ja ehdin vielä tavoitteiden päälle rakentaa peliin vihollisen, teleportaation avaimilla sekä tapahtumia. Prototyypissä pystytään aukomaan ovia, jotka vaativat avaimet, ovilla on animaatio sekä ääni avautumiseen ja sulkeutumiseen. Pelattava hahmo pystyy kävelemään sekä juoksemaan, vaikka kuten aiemmin mainittu jonkinasteinen bugi sinne jäikin. Keskustelut ja NPC:t toimivat kuten pitääkin. Keskustelujen tekstipalkissa on jonkinlainen rajoite ja mikäli keskustelua rämpyttää voi teksti skaalautua hetken väärin. Tämä ei kuitenkaan mielestäni vielä tässä vaiheessa haittaa koska väärin skaalautuminen kestää sekunnin kymmenyksiä. Taskulamppu toimii, vaikka kirkkautta tuleekin jatkokehittäessä säätää korkeammalle. Menut sekä aloitukselle ja kuolemalle toimivat kuten pitääkin, jatkoa ajatellen päämenuun lisätään mahdollisuudet säätää mm. resoluutiota. Tarinapuoli ja aihealueen käsittely jäi hiukan vajaaksi mutta eipä tavoitteena ollutkaan saada niitä täysin mukaan. Kaikki toiminnallisuudet on toteutettu Blueprinteillä kuten tavoitteena oli.

Raportin kirjoittaminen oli itselleni haastavin osa koko projektissa. Ajan käytön kanssa ei tullut ongelmia tarkan aikatalutuksen ansiosta. Noudatin aikataulua, joka asetti jokaiselle viikolle tietyn määrän tehtävää, tietenkin muutoksia tapahtui ja jotkin asiat veivät aikaa odotettua enemmän.

Lähteet

Bleszinski, C. 2010. History of the Unreal Engine. IGN http://www.ign.com/articles/2010/02/23/history-of-the-unreal-engine?page=1. 24.4.2018

Dehouck, R. 2015. The maturity of visual programming http://www.craft.ai/blog/the-maturity-of-visual-programming/. 20.11.2018

Epic Games. 2018a. Unreal Engine 4.19 Release Notes. Epic Games https://docs.unrealengine.com/en-US/Builds/4_19. 24.4.2018

Epic Games. 2018b. Epic Games https://www.unrealengine.com/en-US/what-is-unreal-engine-4. 24.4.2018

Epic Games. 2018c. Types of Blueprints. Epic Games https://docs.unrealengine.com/en-US/Engine/Blueprints/UserGuide/Types. 25.4.2018

Epic Games. 2018d. Level Blueprint. Epic Games https://docs.unrealengine.com/en-US/Engine/Blueprints/UserGuide/Types/LevelBlueprint. 25.4.2018

Epic Games. 2018e. Connecting Nodes. Epic Games https://docs.unrealengine.com/en-US/Engine/Blueprints/BP_HowTo/ConnectingNodes. 1.5.2018

Epic Games. 2018f. Data-Only Blueprint. Epic Games https://docs.unrealengine.com/en-US/Engine/Blueprints/UserGuide/Types/ClassBlueprint#Data-OnlyBlueprint. 6.11.2018

Epic Games. 2018g. Blueprint Interface. Epic Games https://docs.unrealengine.com/en-US/Engine/Blueprints/UserGuide/Types/Interface. 6.11.2018

Epic Games. 2018h. Blueprint Macro Library. Epic Games https://docs.unrealengine.com/en-US/Engine/Blueprints/UserGuide/Types/MacroLibrary. 6.11.2018

IGN. 2009. Epic Games announces Unreal Development Kit, powered by Unreal engine 3. IGN https://www.ign.com/articles/2009/11/05/epic-games-announces-unreal-development-kit-powered-by-unreal-engine-3. 6.11.2018

Kennedy, B. 2002. Uncle Sam Wants You (To Play This Game). The New York Times https://www.nytimes.com/2002/07/11/technology/uncle-sam-wants-you-to-play-this-game.html. 6.11.2018

McLeroy, C. 2008. Improving “America’s Army”. Soldiers magazine https://web.archive.org/web/20171119173617/https://www.army.mil/article/11935/improving_americas_army. 6.11.2018