KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma / ohjelmistotekniikka

Kim Kuusisto

PELIN TUOTTAMINEN UNREAL DEVELOPMENT KIT -YMPÄRISTÖSSÄ

Opinnäytetyö 2014

TIIVISTELMÄ

KYMENLAAKSON AMMATTIKORKEAKOULU
Ohjelmistotekniikka

KUUSISTO, KIM	Pelin tuottaminen Unreal Development Kit -ympäristössä
Opinnäytetyö	52 sivua
Työn ohjaaja	Laboratorioinsinööri Marko Oras
Toukokuu 2014	
Avainsanat	UDK, UE, Pelimoottori

Peliteollisuus on kasvanut viimeisten vuosien varrella huomattavaksi osaksi viihdeteollisuutta ja se näkyy Kymenlaaksossa yritysten määrän nousuna. Monet yritykset käyttävät projekteihinsa Unreal Development Kit pelinkehitysympäristöä. UDK:n ajankohtaisuus tulee nousemaan uuden konsolisukupolven ja Epic Gamesin kehittämän Unreal Engine 4:n myötä. Opinnäytetyön tekohetkellä ilmainen kehitysympäristö toimi Unreal Engine 3:lla.

Opinnäytetyön tarkoituksena oli tutustua UDK:n tarjoamiin kehitysmahdollisuuksiin ja opetella käyttö siihen asteeseen, että sillä kyettäisiin luomaan esimerkkipeli pienessä kehitystiimissä. Tarkoituksena oli ohjelmiston toiminnan lisäksi siis testata sen toimintaa resurssien jaon ja tiimille jakautumisen näkökulmasta. Tiimiin osallistui kirjoittajan lisäksi toinen opiskelija. Jaoimme tehtävät siten että tuottaisin valikot, pelattavan kentän korkean tason logiikalla ja alun omalle peliluokalle ja viholliselle. Toisen opiskelijan tehtäviin kuului luokkien täydennykset, kahden pelattavan kentän luonti, taukovalikon asennus ja työn viimeistely vastaamaan suunnitelmaa. Itse peli jaettiin vastuualueisiin, jotka kukin toteuttaisi itselleen sopivana ajankohtana. Lopulliseksi työksi suunniteltiin Dungeon Crawlerin hengessä tehty peli. Kirjoittajan osuus onnistui odotetusti ja tuotti pelattavan alun projektille.

ABSTRACT

KYMENLAAKSON AMMATTIKORKEAKOULU
University of Applied Sciences
Information Technology

KUUSISTO, KIM	Creating a Game using Unreal Development Kit
Bachelor’s Thesis	52 pages
Supervisors	Marko Oras, laboratory engineer
May 2014	
Keywords	UDK, UE, Game Engine

The game industry has grown over the past few years as a significant part of the entertainment industry. The growth shows in Kymenlaakso as an increase in the number of firms. Many companies are using the Unreal Development Kit game development system In their projects. UDK will increase its importance as the new console generation and the new Unreal Engine 4 by Epic Games are getting older. At the time of Bachelor’s Thesis, the development environment is free Unreal Engine 3.

The purpose of this study was to explore the UDK's supply of development opportunities and to learn how to use it to the degree that a development team would be able to create a small example of a game. The purpose of the thesis was also to test the software's resource allocation and functionality in general from the team's perspective. The team took part in the author's in addition to one other student. The tasks were divided in such a way that the writer would produce menus, a high-level logic for the course and the beginning of the game and the enemy class. Coworkers tasks included the categories of supplements, two playable courses, installation of pause menu, and finishing work to meet the plan. The game itself was divided into areas of responsibility, each to undertake its own time. The final job was planned dungeon crawler like game. Writers part of the game succeeded as planned and produced playable start for the project.

SISÄLLYS

LYHENTEET JA TERMIT

2D	*"Two Dimensional".* Kuvaus objektille, jolla on vain leveys ja korkeus.
3D	*"Three Dimensional".* Kuvaus objektille, jolla on leveys, korkeus ja syvyys.
Dungeon Crawler	*"Vankityrmä Konttaus".* Pelimuoto, jossa pelaaja asetetaan tilanteeseen, jossa hänen on matkattava sokkeloisen maaston läpi päämääräänsä vihollisten läpi. Peli sisältää kehittyviä elementtejä ja vihollisia sekä kerättäviä objekteja.
Emulaattori	*Emulator.* Ohjelmiston laajennus, joka simuloi haluttua kohdealustaa.
Isometrinen	*Isometric.* Pelille asetettu näkökulma, jossa pelin kuvakulma on x, y ja z -akselilla sama.
Level Editing	*Level Editing.* Level Editing on kentän muokkaamista haluttuun tarkoitukseen. Muokkaus kohdistuu pääasiassa valmiiden resurssien asetteluun ja pelin asetusten muokkaamiseen.
Merkki	*Tag.* Merkki on käyttäjien itse määrittelemä tai valmiista listasta valittu määritelmä objektille.
Pelialusta	*Platform.* Pelialusta on laitteisto, jolle peli suunnataan.
Solmu	*Node.* Solmu on Unreal Kismetin yksittäinen toiminto-objekti, johon voidaan liittää muita toimintoja.
UDK	*Unreal Development Kit.*Pelinkehitys-ympäristö-kokonaisuus, joka pohjautuu Unreal Engineen (UE).
UE	*Unreal Engine* .Epic Games -yrityksen luoma pelimoottori, joka on todella suosittu korkeamman kehitystason peleissä..

UI	*User Interface.* Käyttäjän näkemät graafiset objektit, jotka sijaitsevat näytöllä 2D-tasossa. Useimmiten valintaruudut ja indikaattorit ovat tässä tasossa.

1 JOHDANTO

Kymenlaakson ammattikorkeakoulu (Kyamk) on korkeakoulu, joka toimii yhteistyössä alan osaajien kanssa ja tarjoaa opiskelijoille mahdollisuuden oppia oman alansa uusimpia tekniikoita.

Kyamklla on tietotekniikan alan suuntautumisvaihtoehtona ohjelmistotekniikka. Tällä suuntautumisvaihtoehdolla on mahdollista erikoistua haluamalleen alalle aina web-ohjelmoijasta mobiilipeli-ohjelmoijaksi. Ongelmia kohdatessaan voi opiskelija kääntyä monitaitoisten laboratorioinsinöörien tai lehtoreiden puoleen, jotka mieluusti auttavat opiskelijaa.

Kyamkin ıiloissa toimii GameLab-opiskeluympäristö, jossa on alan uusinta tekniikkaa ja työkaluja sekä virikkeitä. Näiden toimitilojen kautta opiskelija saa nopeasti kontakteja haluamaansa suuntaan ja useasti verkottautuukin nopeasti työelämää varten.

Opiskelujeni aikana olen törmännyt työharjoitteluiden ja työnhaun ohessa toistuvasti kysymykseen: "Osaatko käyttää UDK:ta?". Liian monesti olen joutunut vastaamaan kieltävästi. Siksi päätin opiskella UDK:n käyttöä. Koska pelin tuottaminen on työelämässä projektipohjaista ryhmätyöskentelyä, päätin ottaa opiskelijatoverini mukaan opinnäytetyöhön. Yhteistyöskentely jaettiin siten, että molemmat opiskelivat käyttöä itsenäisesti ja ympäristön toimivuutta testattiin ryhmässä luomalla peli. Peli jaettiin vastuualueisiin, jotka jaoimme keskenämme.

Itse toimintaa kuvaava peli on "Dungeon Crawler" -henkinen peli, jossa pelaaja asettuu robotin saappaisiin ja pyrkii navigoimaan radasta toiseen. Pelissä ihmissotilaat pyrkivät hävittämään lähestyvän uhan. Pelaajalla on käytettävissään valikoima aseita, joita hän käyttää tarvittaessa edetäkseen. Pelaaja tulee kohtaamaan tilanteita, jolloin hänen on suoritettava kevyttä pulmanratkontaa.

2 TYÖKALUJEN KUVAUKSET

Projektissa käytettiin pääasiassa UDK:n sisältämiä työkaluja sekä muutamia avustavia kolmannen osapuolen ohjelmistoja. Näitä tarvittiin UI:n ja unreal

skrptien tuottamiseen. UDK sisältää erilaisia editoreita, joista käytetyimmät ja siten tärkeimmät vaativat esittelyn ennen niiden käyttöä.

2.1 Unreal Development Kit

Unreal-kehitysalusta muodostuu ammattimaisista työkaluista, joilla on luotu monia menestyksekkäitä tuotoksia. Työkalut ja pelimoottori ovat ilmaisia ja tämänhetkinen versio on 3.

UDK mahdollistaa pelien teon PC:lle, Macintoshille ja iPhonelle. Harvinaisemmin käytetty mutta mahdollinen tapa on myös kehitys 3D-renderöintikohtauksille ja animaatioille.

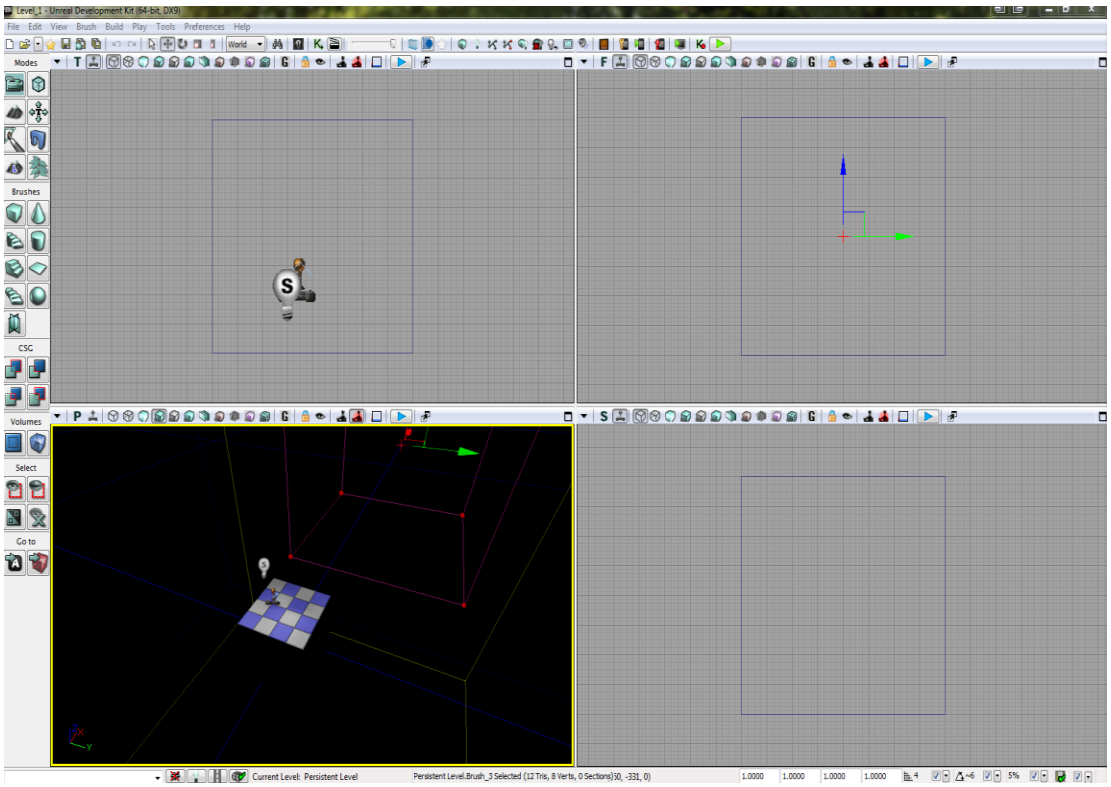
Kehitystyökalut sisältävät tarpeellisia työkaluja yksittäisten resurssien tekemiseen ja muokkaamiseen, mutta kirjoituksen pääpaino on tärkeimmillä työkaluilla, jotka mahdollistavat suoran vaikutuksen pelattavuuteen (3D Buzz. 30.11.2009).

Useimmat kehitystyökalut ovat suoraan editorissa, mutta tarvittavia muita ohjelmia on etsittävä asennuskansion "Binaries"-alikansiosta.

2.1.1 Unreal ED

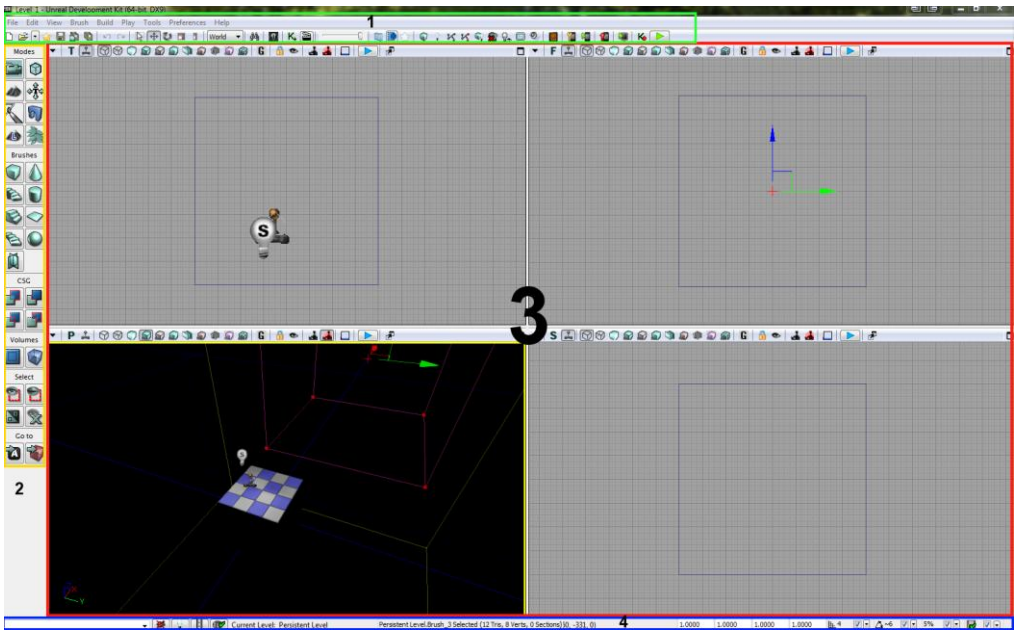
Unreal ED on kehitysalustan päämuokkain, josta käyttäjä pääsee helposti muihin tarvittaviin muokkaimiin, kuten Unreal Kismetiin. Päämuokkaimen tärkein käyttötarkoitus on pelin asetusten ja kentän ulkoasun muokkaus (Level Editing).

Päänäkymä on ensikertalaiselle todella hämmentävä, mutta pienellä harjoittelulla työkalut sisäistää nopeasti (Learning Unreal Engine 3). Pienen hämmennyksen ja opettelun jälkeen käyttäjä huomaa, että suurin osa työkaluista on löydettävissä moneen kertaan ja että nelikenttänäkymä (kuva 1) on asettelun kannalta tärkeä ja käytännöllinen.



Kuva 1. Unreal ED -perusnelikenttänäkymä.

Editori on jaettu eri osa-alueisiin, joiden tärkeys ja toimintakeskeisyys on käyttäjän hallittava, jotta muokkaaminen onnistuisi sujuvasti. Nämä pääalueet on jaettu ja numeroitu selkeyttämään editorin alueita (kuva 2).



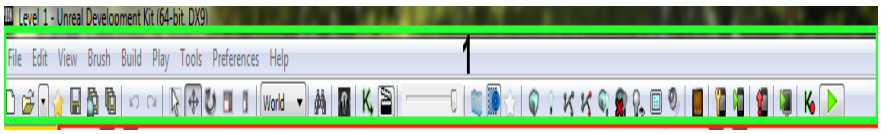
Kuva 2. Unreal ED -nelikenttänäkymä on jaoteltu osa-alueisiin.

Ylimpänä muokkaimessa (kuva 3) sijaitsee hallintatyökalut, jotka keskittyvät muiden editorien avaamiseen, pelikentän asetusten muokkaamiseen ja tallentamiseen sekä tarvittavien kentän resurssien uudelleen rakentamiseen. Uudelleen rakentaminen on vaadittavaa, sillä pelimoottorin on esimerkiksi geometriaa siirrettäessä laskettava valaistus uudelleen, jotta varjot piirtyisivät oikein.

Vasemmassa laidassa editorista (kuva 4) löytyy niin sanottu työkalulaatikko, joka keskittyy omien kategorioidensa mukaisiin tehtäviin. Näitä työkaluja käytetään yleensä 3D-työikkunan kanssa (kuva 5). Samat työkalut ovat löydettävissä hallintatyökalujen alta tai hiiren oikealla klikkauksella valitusta objektista. Käyttäjän toimintatavoista riippuen työkalupalkki on hyödyllinen työkalu, joka mahdollistaa tärkeisiin elementteihin pääsyn.

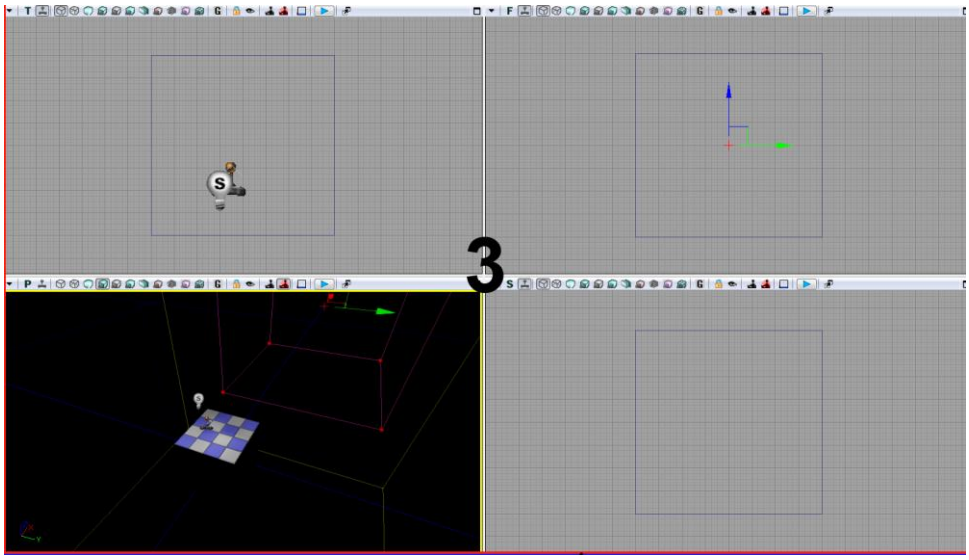
Suurimman ja pelottavimman osa-alueen editorista muodostaa työikkunoista muodostuva editorinäkymä (kuva 5). Lähtökohtaisesti työikkunat kuvaavat samaa maailmaa eri näkökulmista (ylhäältä, edestä, 3D-maailmassa, sivusta) ja kolme neljästä näkymästä on luotu vain ääriviivoista. Aktiivinen työskentelyikkuna korostuu keltaisella reunuksella. Halutessaan työskentelijä voi suurentaa haluamansa työikkunan koko nelikentän kokoon painamalla ikkunansa oikeasta laidasta löytyvää nappia. Kustakin työikkunasta löytyy niiden ylälaidasta samat työkalut, jotka muokkaavat näkymää. Tärkein ikkuna muokkaamisessa on 3D-näkymä, jossa tekijä voi muokata maailmaa samalla tarkastellen, miltä maailma pelaajan silmin näyttäisi.

Alimpana editorista löytyy informaatiopalkki (kuva 6). Tänne keskittyvät pääasiassa tiedot valitun objektin sijainnista ja skaalasta. Samaisesta palkista on myös mahdollista pikaisesti säätää objektin liikuttamiseen liittyviä nopeuksia, jotka perustuvat vetoristikkoon. Vetoristikko on suhdeverkosto joka luodaan maailmaan ja joka vaikuttaa objektin muutosten kokoon. Alapalkista voidaan säätää yhden solun kokoa. Esimerkiksi objektia liikutettaessa se liikkuu yksikön verran, joka on yksi solu verkossa. Täten saadaan todella tarkkoja siirtoja ja esimerkiksi seiniä luotaessa voidaan olla varmoja, etteivät ne mene päällekkäin. Informaatiopalkista voi myös vaihtaa nopeasti automaattisen tallennuksen aikavälin.

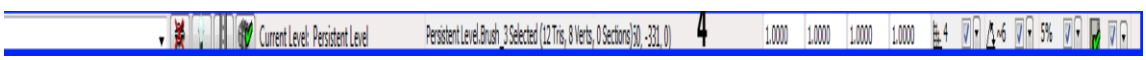


Kuva 3. Unreal ED -muokkaimen hallintatyökalut ovat helposti löydettävissä.

Kuva 4. Unreal ED -työkalulaatikko on nopeakäyttöinen muokkaajan ystävä.



Kuva 5. Unreal ED -nelikenttä on kartanmuokkaajan työmaa.



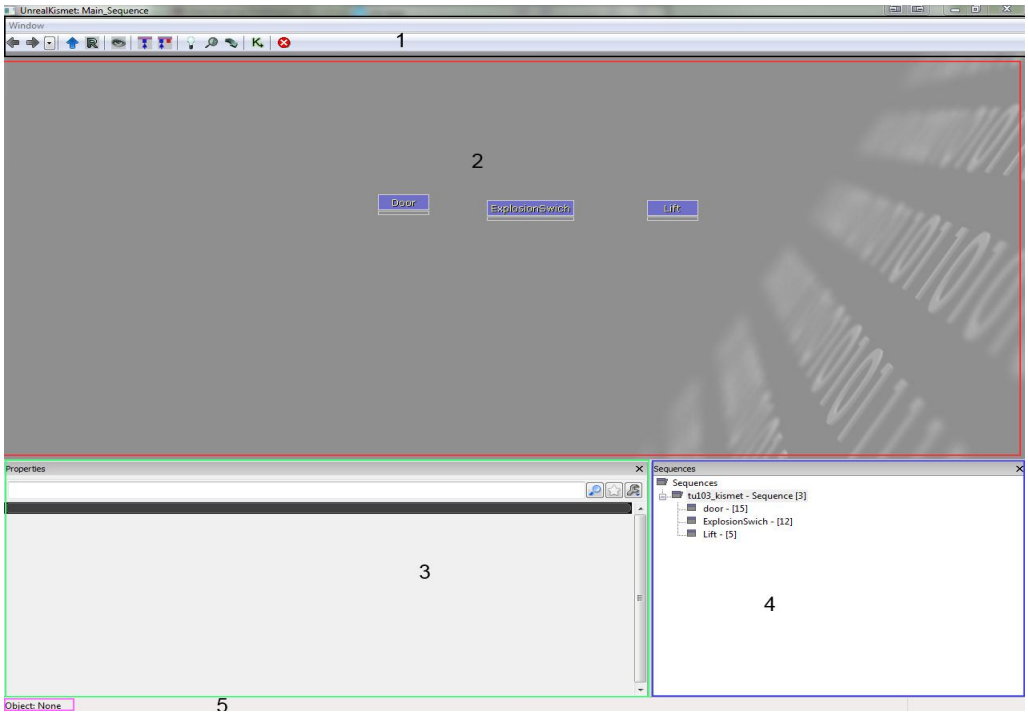
Kuva 6. Unreal ED -informaatiopalkki pitää käyttäjän kartalla.

2.1.2 Unreal Kismet

Unreal Kismet on työkalu, joka mahdollistaa pelielementtien teon koskematta yhteenkään skriptiin. Järjestelmää käytetään pääasiallisesti kentän suoran toiminnan luomiseen, ja se onkin pääasiallisesti tarkoitettu artisteille ja suunnittelijoille. Koodin luojat joko keskittyvät luomaan skriptejä, jotka lisäävät ominaisuuksia Kismetiin (Unreal Script - ".uc") tai luokkia, jotka integroituvat pelin asetuksiin tai omiin käytettyihin luokkiin.

Peli on täysin mahdollista luoda kirjoittamatta riviäkään koodia. Järjestelmä on täydellinen kuvallisesti hahmottaville ihmisille ja henkilöille, joilla ei ole koodin kirjoittamisesta kokemusta (UDK: Introduction to Kismet). Ehdottomia positiivisia puolia puhtaaseen skriptaamiseen on, ettei Kismetin tuotosta tarvitse rakentaa binääriin, kuten skriptit. Tekijän on mahdollista vaihtaa arvoja ja käynnistää peli suoraan editorissa (kuva 14).

Kismet on oma editorinsa (kuva 7) Unreal ED -muokkaimessa ja sen saa avattua joko "View/UnrealKismet" tai painamalla pääeditorin hallintapalkin "Kismet"-ikonia. Avautuva ikkuna sisältää pääeditorissa auki olevan tason korkean tason logiikan.



Kuva 7. Unreal Kismet -editorin näkymä.

Ikkunan ylimmässä laidassa (kuva 8) on löydettävissä päätyökalupalkki, joka keskittyy Kismetin sisäiseen navigointiin ja näkyvyyteen.

Suurimman osan tilasta vie editori-ikkuna (kuva 9), johon käyttäjä voi luoda tapahtumia käyttämällä hiiren oikean näppäimen alta paljastuvia toimintoja. Liitos pelimaailmaan voidaan luoda valitsemalla haluttu objekti päämuokkaimessa ja luomalla siitä muuttuja Kismetissä.

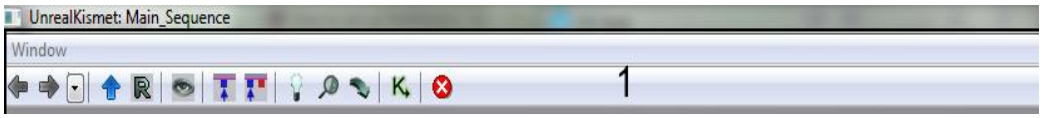
Monimutkaisten logiikoiden syntyessä voi kokonaisuuden hahmottuminen hämärtyä tässä visuaalisessa koodaamisessa, ja siksi on mahdollista luoda alafunktioita. Tästä esimerkkinä on luotu "Door"- alifunktio (kuva 9), joka sisältää oven avautumistoimintoja. Tarkemmin toimintoja (kuva 13) voi tarkastella tuplaklikkaamalla alifunktion nimeä tai valitsemalla sen alifunktioiden listauksesta (kuva 11).

Kismetin työkalut ovat solmutyökaluja, joiden toiminnot liitätetään toisiin vetämällä nuolia. Luotujen solmujen ominaisuuksia voidaan muokata valitsemalla haluttu solmu ja tarkastelemalla sen ominaisuuksia vastaavan nimisessä ikkunassa (kuva 10). Tästä ikkunasta käyttäjä voi vaihtaa arvoja.

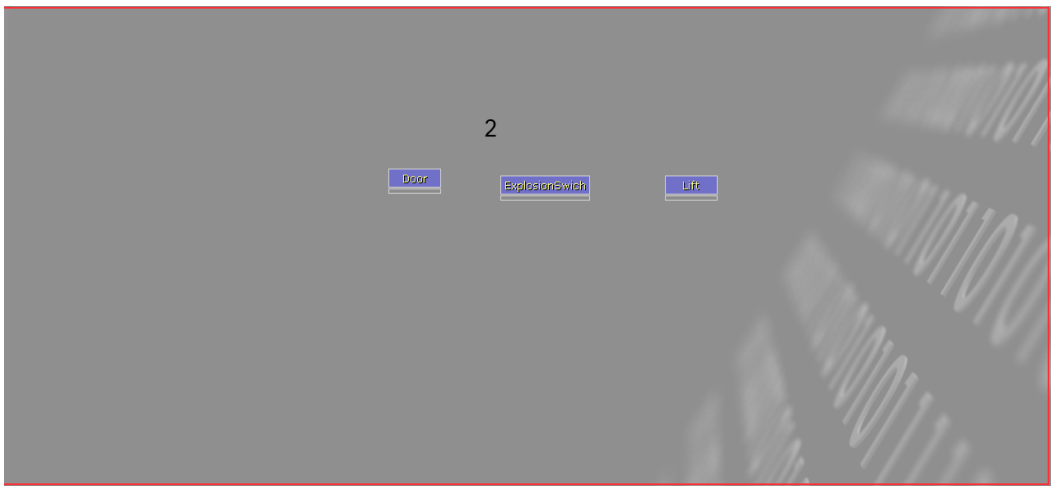
Oikeassa alareunassa esiintyvä "Sequences" -ikkuna sisältää luodun tason kaikki alifunktiot ja mahdollistaa nopean siirtymisen niiden välillä (kuva 11).

Alimpana (kuva 12) löytyy nimialue, joka nimeää valitun solmun.

Sekä asetukset että alifunktiolistausikkuna voidaan sulkea tai irrottaa omaksi ikkunakseen tai poistaa. Liitos takaisin tapahtuu ikkunan raahaamisella ja ikkunan palautus navigointivalikon "Window" -kohdasta.

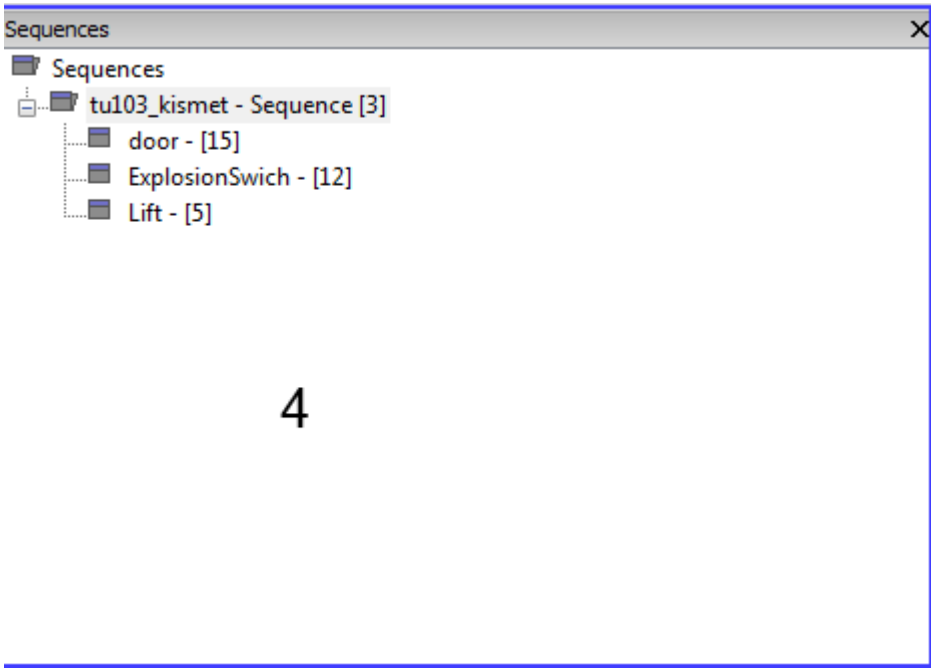


Kuva 8. Unreal Kismet -päätyökalupalkki mahdollistaa nopean liikkumisen editorissa.

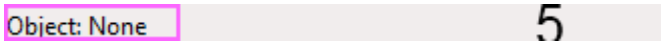


Kuva 9. Unreal Kismet -editori-ikkuna on käyttäjän pääikkuna.

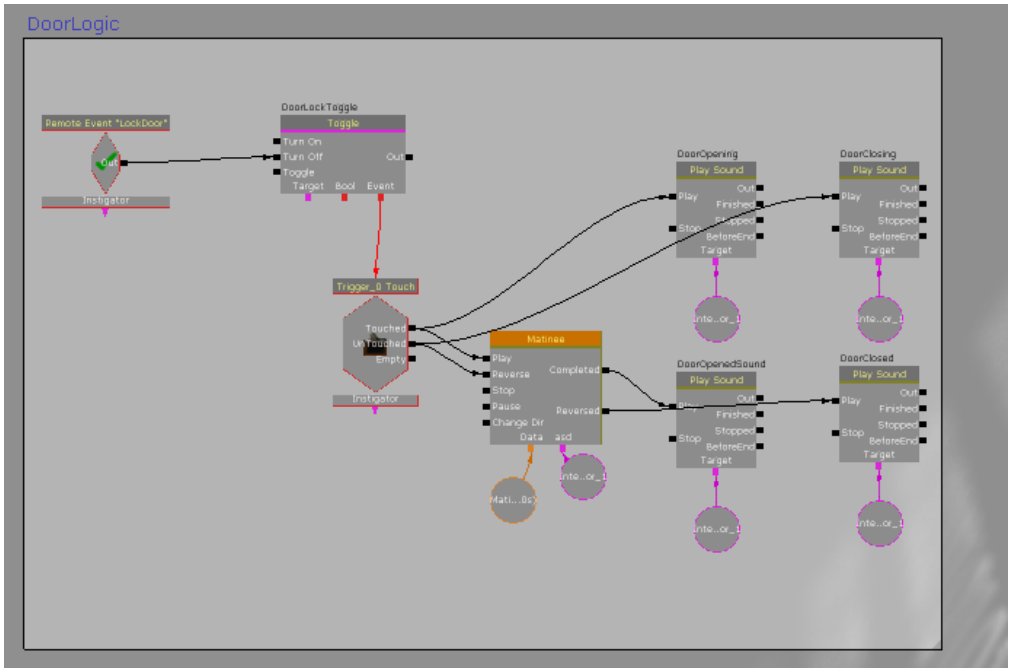
Kuva 10. Unreal Kismet -ominaisuusikkuna näyttää valitun solmun ominaisuudet.



Kuva 11. Unreal Kismet "Sequences" -ikkuna mahdollistaa alifunktioiden välillä nopean siirtymisen.



Kuva 12. Unreal Kismet -nimialue kertoo käyttäjälle valitun solmun nimen.



Kuva 13. Unreal Kismetillä on luotu pelin logiikkaa.

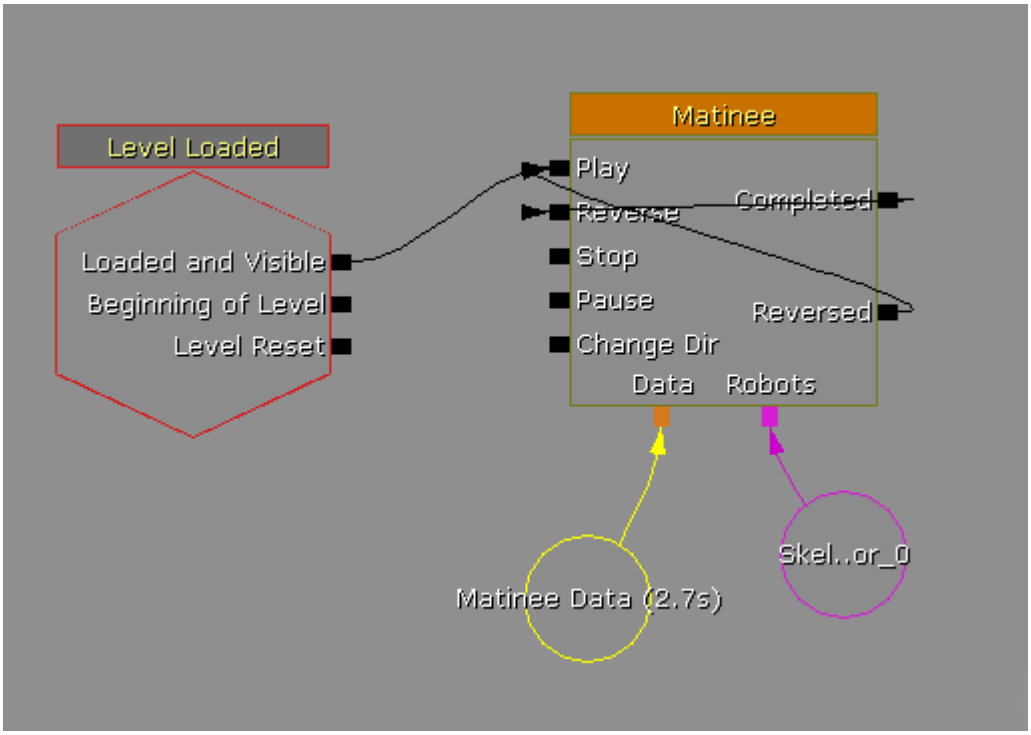
Kuva 14. Unreal Kismetillä on luotu toimiva ovi.

2.1.3 Unreal Matinee

Unreal Matinee (kuva 16) on animointityökalu, jolla voidaan luoda pelimaailman objekteille tarvittavia tapahtumia ajan kuluessa. Työkalulla voidaan luoda erilaisia animaatioita, jotka voidaan Unreal Kismetillä yhdistää pelimaailman tapahtumiin (Tutorials - UE3 Cutscenes).

Yksittäinen Matinee-animaatio voidaan liittää yhteen tai useampaan objektiin, mikäli objektit ovat identtisiä tarvittavilta osin ja ovat Matinee-yhteensopivia. Esimerkiksi: mikäli objekti on staattinen, ei sitä voida animoida.

Matinee, kuten muutkin erilliset editorit, ovat käynnistettävissä päämuokkaimen hallintatyökalupalkista tai "View"-valikon alta. Koska animaatio vaatii objektin ja Kismet-hallinnan, on parasta aloittaa valitsemalla objekti, jota halutaan animoida, ja liittämällä se Kismetissä matinee-animaatioon. Tällöin helpoin tapa avata matinee on tuplaklikata Kismetissä matineeta vastaavaa solmua.



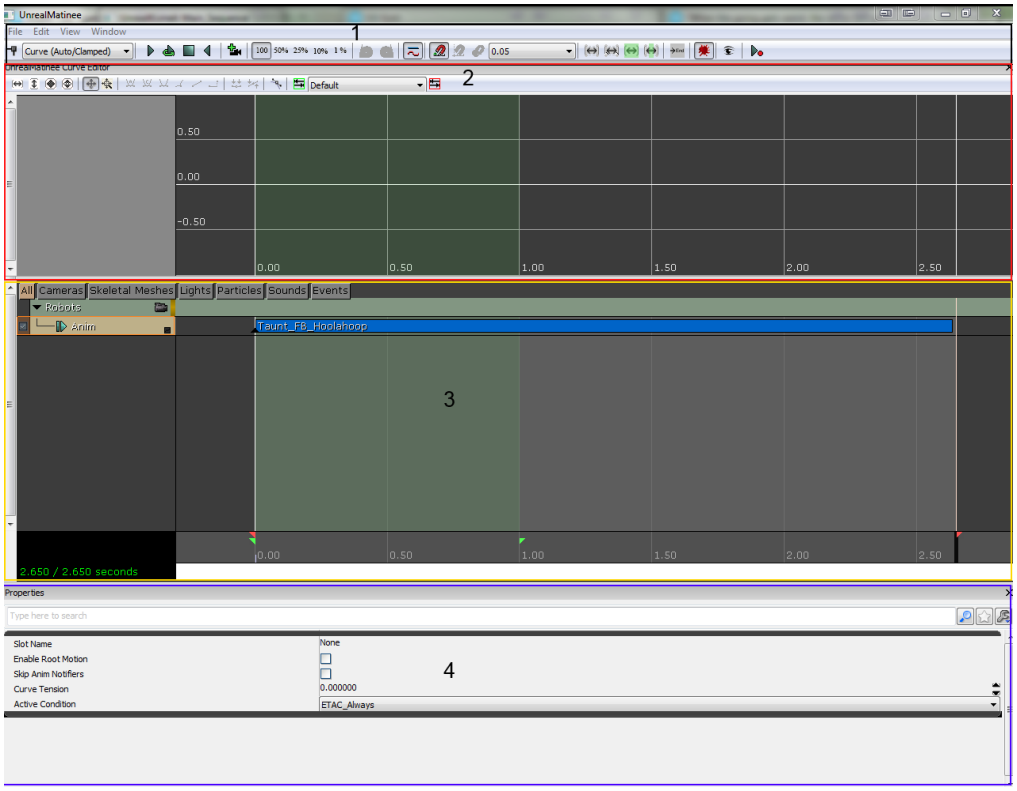
Kuva 15. Unreal Kismetillä on luotu logiikka Matinee-animaation toimimista varten.

Ylimpänä ohjelmassa (kuva 17) on jälleen hallintapaneeli, jolla hallitaan Matineen kurvimuokkainta (kuva 18) ja animaation aikalinjaa (kuva 19). Alimpana (kuva 20) on löydettävissä ominaisuudet-ikkuna, joka nimensä mukaisesti kertoo aikajanalta valitun kohteen ominaisuudet. Samalta linjalta voidaan myös käynnistää animaatio ilman pelin käynnistystä, jolloin Unreal ED -pääohjelmassa voidaan 3D-työikkunasta seurata animaation toimintaa.

Hallintapalkin alta on löydettävissä kurvimuokkain, joka mahdollistaa animaation eri kohtien tarkemman muokkaamisen. Tätä voi käyttää esimerkiksi liikeradan pehmentämiseen. Työkalu ei ole välttämätön tavalliselle käyttäjälle. Animaattori hallitsee Matineen paljon paremmin.

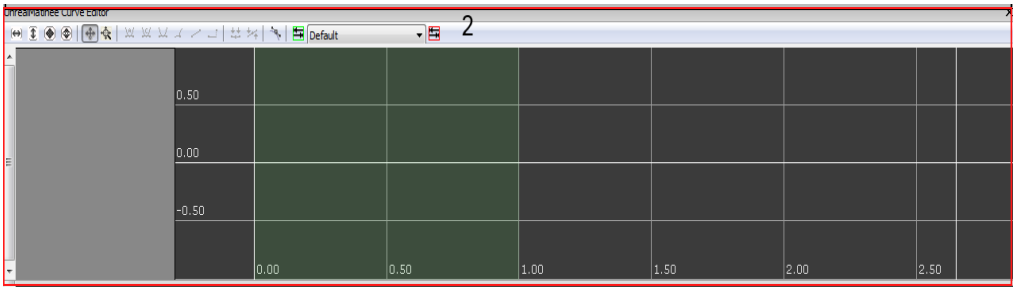
Animaatiolla löytyy aikalinja ja luokkaryhmittely keskeltä Matinee-ikkunaa. Luokkaryhmittely mahdollistaa animaation samanaikaisten toimintojen luokittelun tarpeen mukaan. Useimmiten käyttäjät tekevät yhden Matinee-animaatiota per käyttötarkoitus, joten useammat ryhmät jäävät vähäiselle yleensä. Vaadittavaa on kuitenkin, että yksi ryhmä on luotu ja sen alle on tehty hiiren oikealla klikkauksella halutunlainen rata. Mikäli halutaan tehdä animaatio hahmolle,

valitaan animaation hallintarata ja asetetaan sille haluttu animaation kohta annetusta animaatiokokoelmasta. Vastaavasti liikuteltavaa objektia varten valitaan liikkumiseen tarkoitettu rata ja annetaan aikajanalle halutut avainkehykset. Jotta liikkuminen saataisiin aikaiseksi, valitaan haluttu avainkehys ja asetetaan se Unreal ED-muokkaimessa haluttuun kohtaan. Unreal-moottori laskee avainkehysten välisen liikkeen ja kurvimuokkaimella voidaan säätää liikkeen sujuvuus. Onnistuneella animaatiolla peliä elävöitetään huomattavasti (kuva 21).

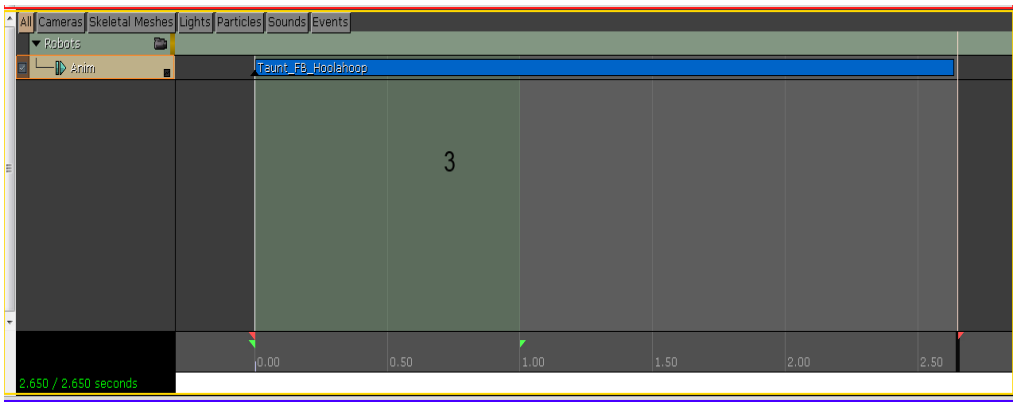


Kuva 16. Unreal Matinee -perusnäkymä.

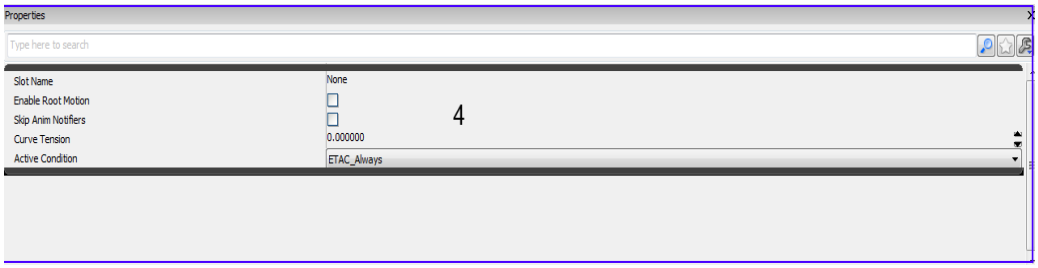
Kuva 17. Unreal Matinee -hallintapaneeli mahdollistaa nopean testaamisen editorin sisällä.



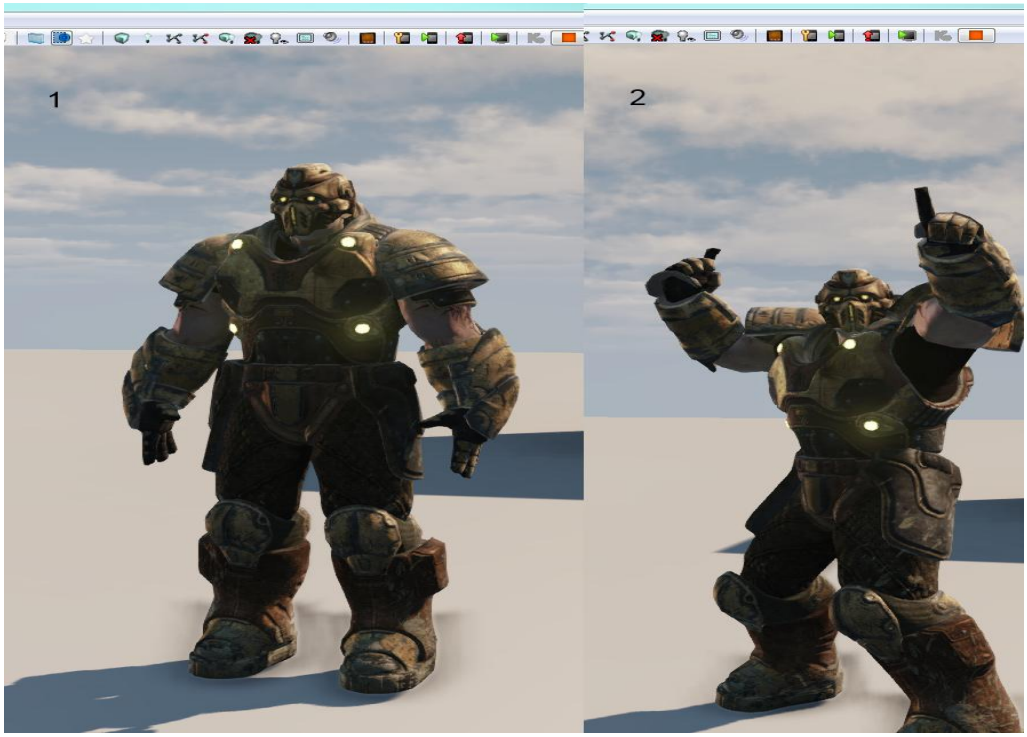
Kuva 18. Unreal Matinee -kurvimuokkain mahdollistaa animaation vaiheiden muokkauksen.



Kuva 19. Unreal Matinee -aikalinja näyttää animaation kulun ja keston.



Kuva 20. Unreal Matinee -ominaisuusmuokkain mahdollistaa valitun kohteen ominaisuuksien muokkaamisen.



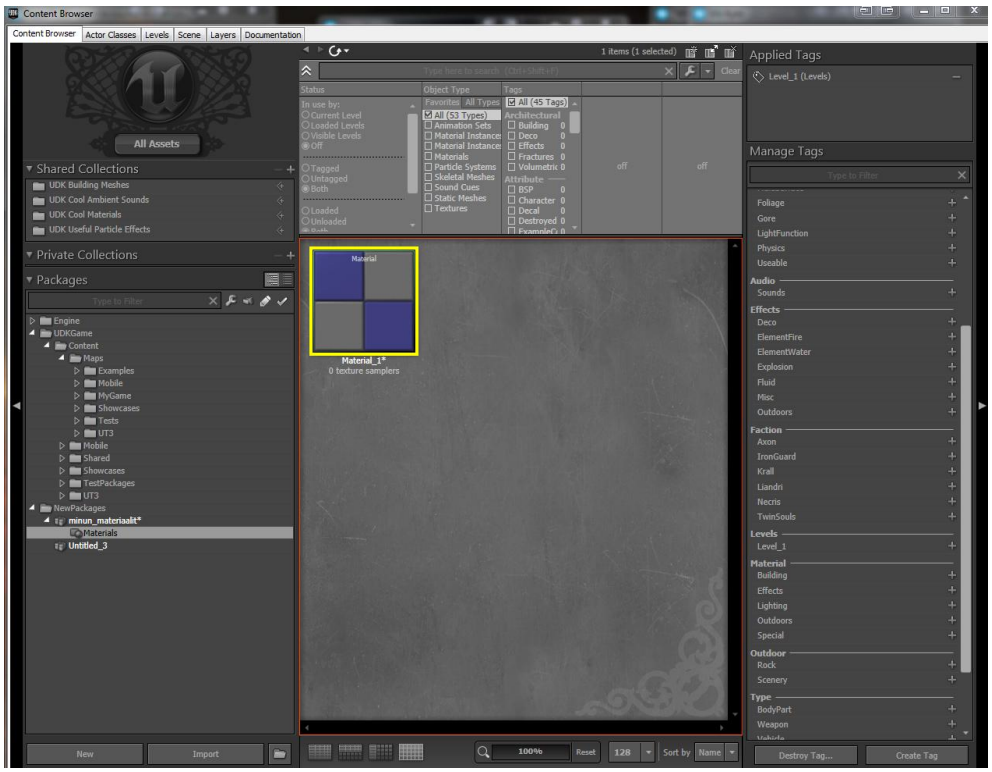
Kuva 21. Animoimaton ja animoitu hahmo Unreal -pelimoottorissa.

2.1.4 Unreal Content Browser

Unreal Content Browser on nimensä mukaisesti sisältöselain. Kehitysympäristön sisältöselain mahdollistaa resurssien, kuten 3D mallien, tuomisen suoraan järjestelmään. Ohjelma hallinnoi kaikkia tuotuja resursseja paketteina (.upk) ja muokkaakin resurssin omaksi tiedostokseen tuodessa ohjelmaan (Roberts).

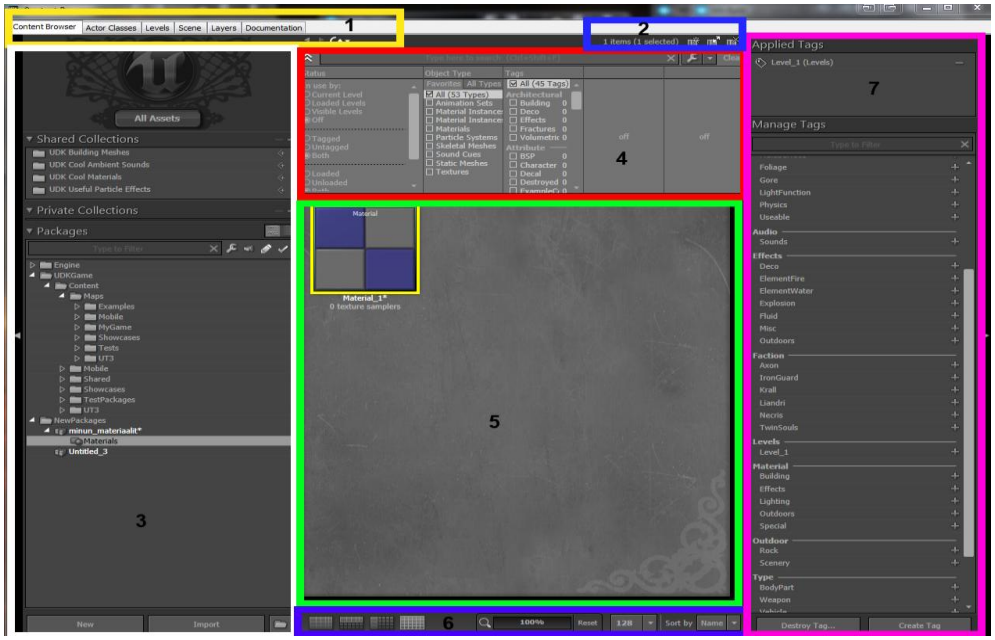
Sisältöselain mahdollistaa eri tiedostojen muokkaamisen omilla muokkaimillaan. Lisäksi kukin resurssi voidaan määritellä kuulumaan käyttötarkoituksensa mukaan omaan luokkaansa. Sen lisäksi objektille voidaan antaa valmis tai itse luotu merkki, joka mahdollistaa erilaisen erottumisen, mikäli etsitään tiettyjä resursseja.

Esimerkiksi: Esineen käyttämä tekstuurimateriaali voidaan määritellä kuulumaan kategoriaan "materiaalit" ja on lisätty projektin pakettiin nimeltä "minun_materiaalit" sekä kyseinen materiaali liitetty merkkiin "Level_1" (kuva 22).



Kuva 22. Kuvassa on Unreal Content Browser ja esitetty esimerkkitilanne.

Unreal Content Browser jakaantuu seitsemään tärkeään osa-alueeseen (kuva 23), joiden olemassaolon tiedostaminen on tärkeää. Ohjelma on monipuolinen ja monella tapaa muokattavissa, mutta useasti käyttäjä tulee pysymään muutamassa tärkeimmässä kohdassa.



Kuva 23. Unreal Content Browserin näkymä on jaoteltu.

Aivan ylimpänä ikkunassa (kuva 24) sijaitsevat kaikki mahdolliset toiminnot. Samaiseen ohjelmaan on mahdutettu myös monia peliin vaikuttavia asetuksia, ja tätä kautta niihin pääsee nopeasti, ellei halua etsiä niitä päämuokkaimen ikkunavalikosta. Tärkeimpänä on ensimmäinen kohta, johon nyt perehdytään.

Ehkei tärkeimpänä, mutta käyttäjälle mieleisimpänä on ikkunan ylälaidassa (kuva 25) löydettävissä informaatiopalkki. Palkista voidaan valmiilla asetuksilla nopeasti muokata ikkunan näkymiä valmiisiin pohjiin ja katsoa, kuinka monta objektia on ikkunassa valittu.

Vasemmasta laidasta (kuva 26) on löydettävissä resurssiselain, jolla voidaan tarkastella Unreal Development Kitin resursseja aivan tiedostotasolla. Kansiorakenne siis vastaa asennuskansiosta löydettävää kansiorakennetta. Käyttäjän on kuitenkin huomioitava, että resurssit eivät näy selaimessa, ellei niitä ole tuotu (Import) sinne. Tärkeää on myös huomata, että resurssit ovat käytössä vain, jos ne on ladattu ohjelman muistiin. Ladatut resurssit, ja siten käyttökelpoiset, näkyvät vaaleampina. Ikkuna voidaan piilottaa käyttämällä vasemmasta laidasta löydettävää nuolta.

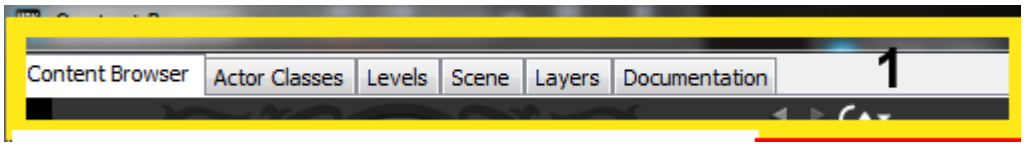
Ikkunan keskeltä ja ylälaidasta (kuva 27) on löydettävissä suodatin. Suodattimen avulla voidaan valitusta paketista etsiä haluttua resurssia eri kriteerien mukaan, kuten resurssin tyyppi, merkki, nimi tai sen osa. Merkittävää on huomata, että tulokset vastaavat valitun paketin sisältöä. Useimmat ensikertalaiset haluavat etsiä yleisesti resurssipaketeista, mutta päätyvät etsimään yksittäisestä paketista. Tästä ongelmasta pääsee eroon painamalla resurssiselaimen ylälaidassa löytyvää "All Assets" -nappia.

Suurin ja tärkein alue on aivan keskellä ikkunaa (kuva 28). Tästä alueesta nähdään kaikki resurssit, jotka on löydettävissä valitusta paketista ja, jotka vastaavat suodattimen arvoja. Ikkunasta on nähtävissä resurssin tyyppi, nimi ja tarkempia tietoja, kuten tekstuurin resoluutio. Käyttäjä valitsee täältä haluamansa resurssin ja muokkaa sitä halutessaan hiiren oikealla näppäimellä tai luo uuden haluamansa resurssin toimien samoin, mutta valitsematta objektia. Resurssin siirtäminen pääeditoriin toimii seuraavasti: käyttäjä voi joko raahata resurssin ikkunasta

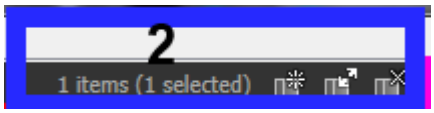
päämuokkaimen työikkunaan tai valita haluamansa ja käyttämällä jälleen hiiren oikeata näppäintä työikkunassa.

Resurssien esikatselualuetta voi muokata haluamansa näköiseksi käyttämällä sen alta löytyvää muokkausaluetta (kuva 29). Tämä alue mahdollistaa erilaisen näkymän, esikatselukuvakkeiden koon määrittämisen ja lajitteluperusteen.

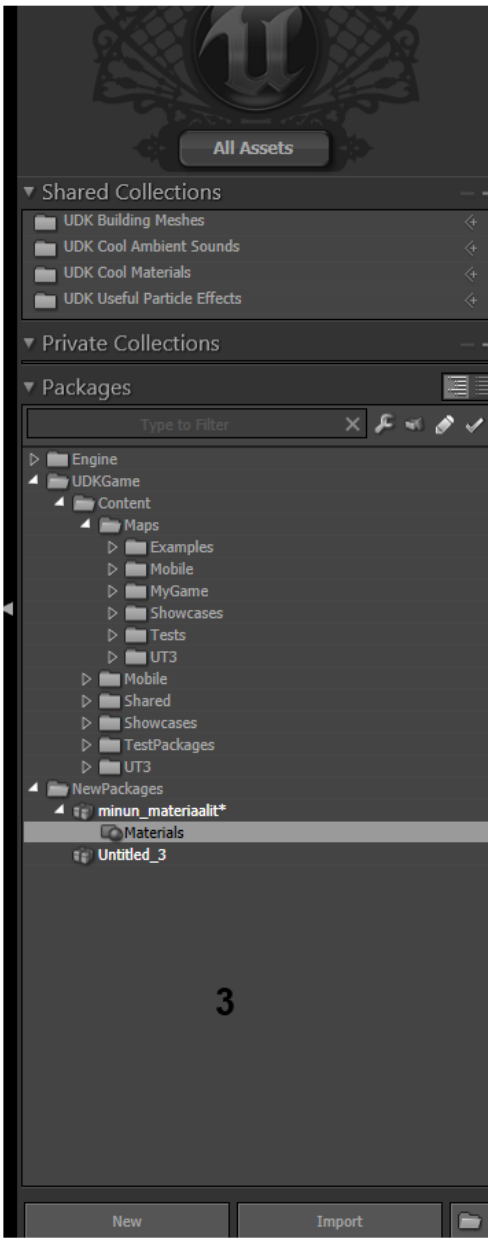
Suurimmille resurssimäärille on elintärkeää, että niille on annettu tietty määrä merkkejä. Niitä voidaan yksittäisestä valitusta resurssista muokata ikkunan oikeasta laidasta löytyvästä merkkimuokkaimesta (kuva 30). Täältä käyttäjä voi valita valmiita merkkejä ja antaa niitä resursseille. Halutessaan käyttäjä voi luoda, poistaa tai muokata olemassa olevia merkkejä tästä ikkunan osasta. Kuten pakettiselaimessa, niin myös merkkimuokkaimessa on löydettävissä nuoli sen reunassa. Tämä vastaavasti piilottaa kyseisen ikkunan osan.



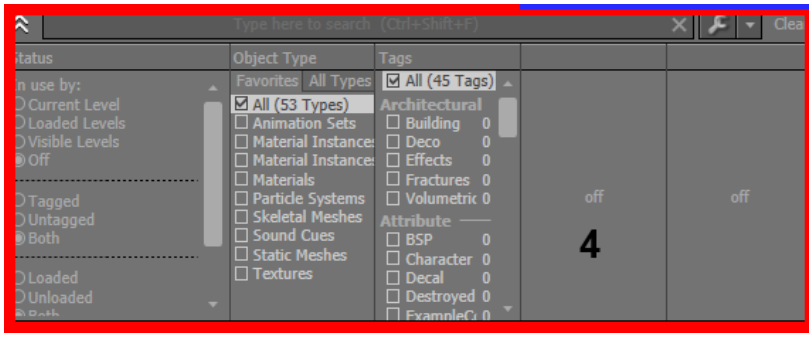
Kuva 24. Unreal Content Browserin kautta pääsee vaikuttamaan moneen asetukseen.



Kuva 25. Unreal Content Browserin informaatiopalkki tarjoaa käyttäjälle tietoja.

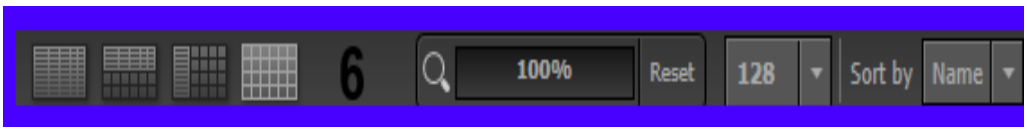


Kuva 26. Unreal Content Browserin resurssiselain mahdollistaa pääsyn asennuskansion rakenteeseen.

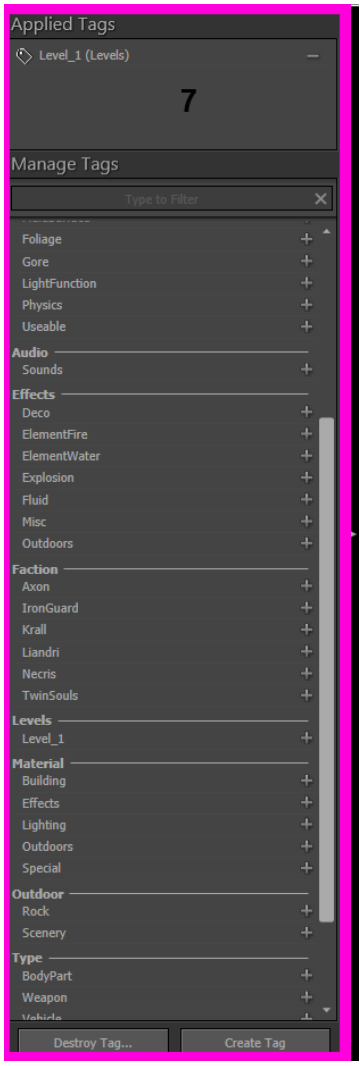


Kuva 27. Unreal Content Browserin suodatin on käyttäjän elintärkeä työkalu.

Kuva 28. Unreal Content Browserin esikatselualue näyttää suodattimen läpi tulleet kohteet.



Kuva 29. Unreal Content Browserin esikatselualueen muokkain mahdollistaa omanlaisen näkymän muodostamisen.



Kuva 30. Unreal Content Browserin merkkimuokkain mahdollistaa nimensä mukaisesti merkkien muokkaamisen.

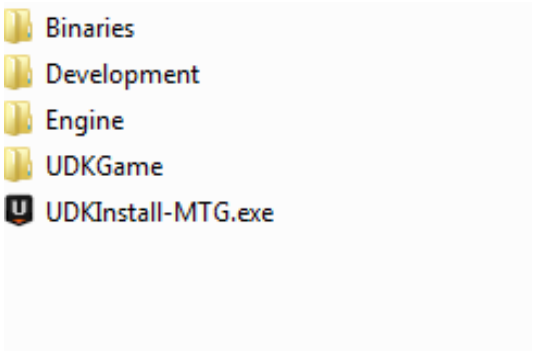
2.1.5 Unreal Frontend

Unreal Frontend on ohjelma, joka on etsittävä asennuspolun "Binaries"-kansion alta. Frontend on pääasiallisesti pelin lopputuotoksen asennuspaketin luontiohjelma, mutta mahdollistaa erilliset toimenpiteet. Ohjelma mahdollistaa luotujen skriptitiedostojen lisäämisen binääritiedostoihin, tarvittavien pakettien lisäämisen asennuspakettiin ja kokonaisuuden lisäämisen yhteen asennuspakettiin halutulle alustalle.

Paketoitavaan tuotteeseen valitaan Frontendistä halutut karttatiedostot (How to Cook a Level in UDK). Ongelmatilanteiden tapahtuessa järjestelmä ilmoittaa ilmoitusikkunassaan sattuneista virheistä. Esimerkiksi: Mikäli skriptilähde on

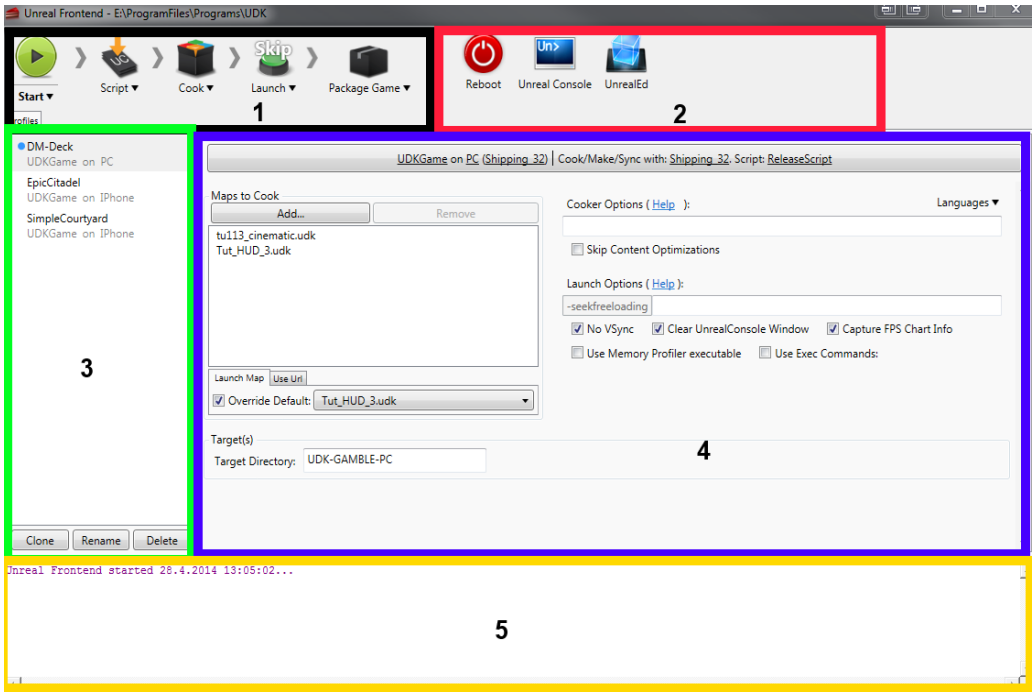
lisätty, mutta lähdekansiossa skriptit ovat kansion juuressa eivätkä "Classes"-alikansiossa, ilmoittaa järjestelmä, ettei löydä tiedostoja. Varoitusilmoitus on seuraavanlainen "SkriptikansionNimi/*.uc".

Valmis asennuspaketti (esim. ".exe") luodaan UDK:n asennuskansion juureen valitulla sovellusnimellä (kuva 31).



Kuva 31. Unreal Frontendin tuottama valmis pelin asennuspaketti.

Itse ohjelma jakautuu viiteen perusosaan, joita kaikkia käyttäjä ei tule kokonaisuudessaan käyttämään.



Kuva 32. Unreal Frontendin näkymä on jaoteltu.

Ylimpänä ikkunasta on löydettävissä suoritusjärjestysrivi ja lisätyökalupalkki.

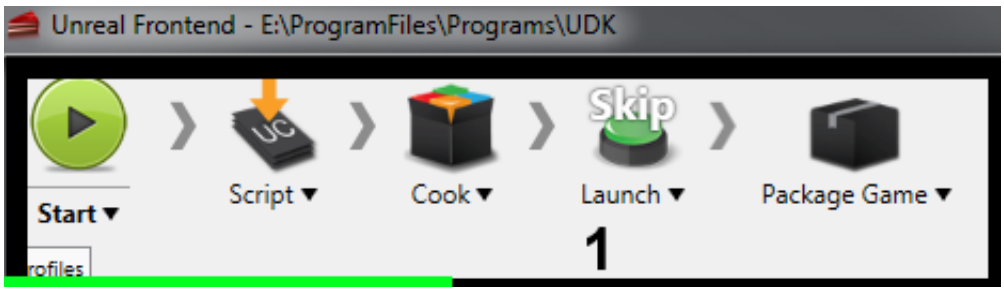
Suoritusjärjestysrivi (kuva 33), joka myös sisältää aloitusnapin, on kokoelma työkaluja, jotka Unreal Frontend käy läpi järjestyksessä luodessaan asennuspakettia. Käyttäjä voi halutessaan valita yksittäisen osa-alueen halutessaan esimerkiksi vain rakentaa skriptit binääriin. Yksittäisiä alueita voidaan myös säätää ohitettavaksi (skip), jolloin pakettia tehdessä esimerkiksi ohitetaan pelin testikäynnistys.

Suoritusjärjestysrivin vieressä (kuva 34) on löydettävissä lisätyökalupalkki, josta käyttäjä voi käynnistää päämuokkaimen ja Unreal-pelimoottorin sisäisen komentotulkin tai uudelleenkäynnistää haluamiaan komponentteja.

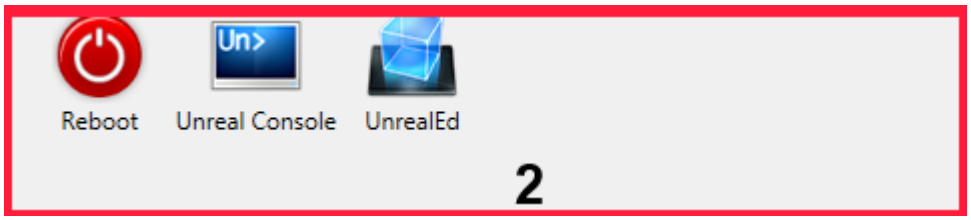
Vasemmasta laidasta ikkunaa (kuva 35) on löydettävissä rakennusprofiilit, jotka tehdään kohdelaitteen mukaan. Profiilit sisältävät valmiiksi aseteltuja tietoja kohdealustalle rakennettaessa peliä. Profiilit ovat tärkeässä asemassa, kun peliprojekti julkaistaan useammalle pelialustalle.

Suurin ja tärkein alue on pelin rakentamisen muokkausalue (kuva 36). Täältä käyttäjä voi valita, mitä karttoja halutaan pistää peliprojektiin mukaan. Lisäksi voidaan määrittää kohdealusta ja tarvittavia muita tietoja, kuten käytetäänkö muistin profilointityökalua. Tärkeintä on kuitenkin valita oikeat kartat ja vaihtaa käynnistettävän kartan kohdalle "Override default" (ylikirjoita oletus) haluttu käynnistyskartta, joka useimmiten sisältää käynnistysvalikon.

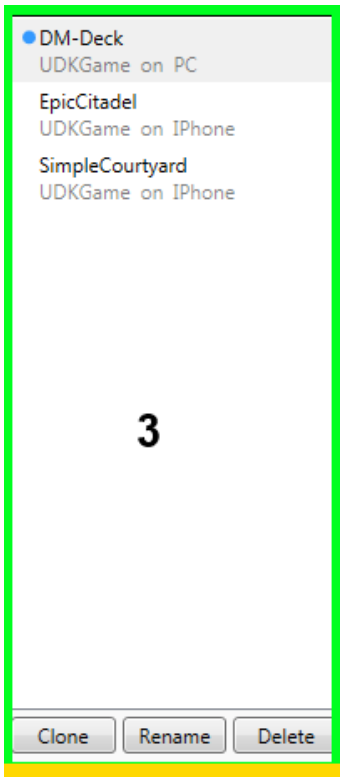
Alimpana muttei vähäisimpänä (kuva 37) on löydettävissä ikkunasta sen konsoli, johon se tulostaa tietoja suorittaessaan tehtäviä. Täältä on löydettävissä syntyneet ongelmat käytettäessä ohjelmaa. Skriptien rakentamisessa tämä on kätevää, sillä käyttäjä näkee, mistä ongelma tarkalleen tulee.



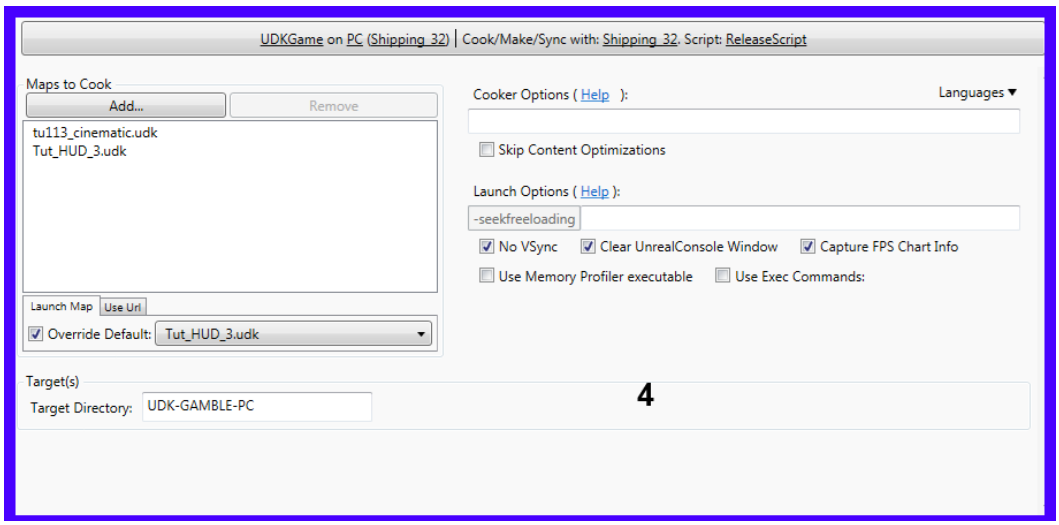
Kuva 33. Unreal Frontendin suoritusjärjestysrivi on käyttäjälle yksi tärkeimmistä alueista.



Kuva 34. Unreal Frontendin lisätyökalupalkki mahdollistaa editorin hallinnan.



Kuva 35. Unreal Frontendin rakennusprofiilimanageri on kohdelaitteiden välillä liikuttaessa kätevä työkalu.



Kuva 36. Unreal Frontendin rakentamisen muokkausalue on laitteistolle rakentamisen tärkein säätöalue.

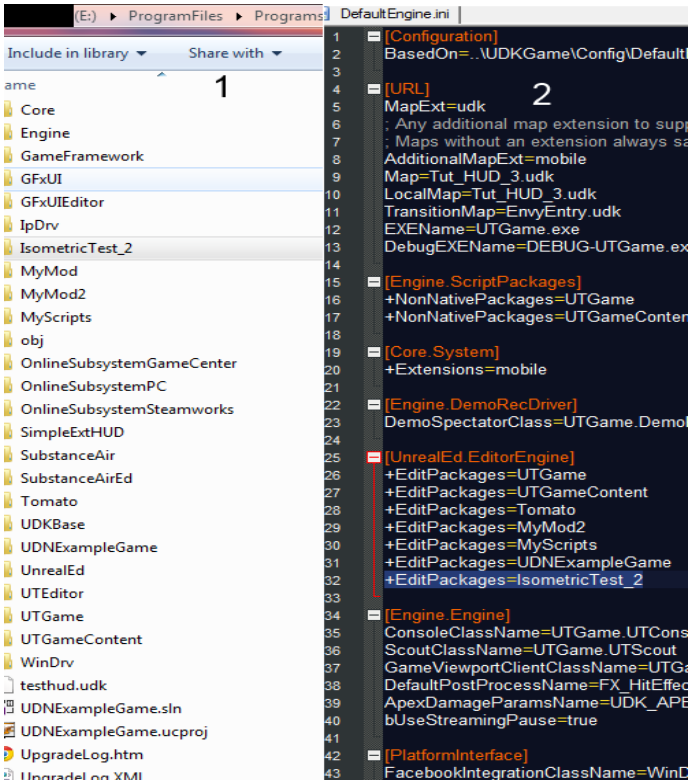
Kuva 37. Unreal Frontendin tietokonsoli näyttää toimintojen edistymisen, virheet ja muut tiedot.

2.1.6 Unreal Script (.uc) ja luokkarakenne

Unreal Script on Unreal-moottorin käyttämä objektipohjainen skriptauskieli, joka muistuttaa huomattavasti Javaa (Adamson). Unreal Skriptiä voidaan kirjoittaa useilla tekstieditoreilla ja näihin voidaan kytkeä päälle automaattitäydennys sekä syntaksitunnistus pienellä vaivalla. Kaikkein suosituin vaihtoehto käyttäjien keskuudessa on Visual Studio ja sen liitännäinen, joka mahdollistaa Unreal Script tuen.

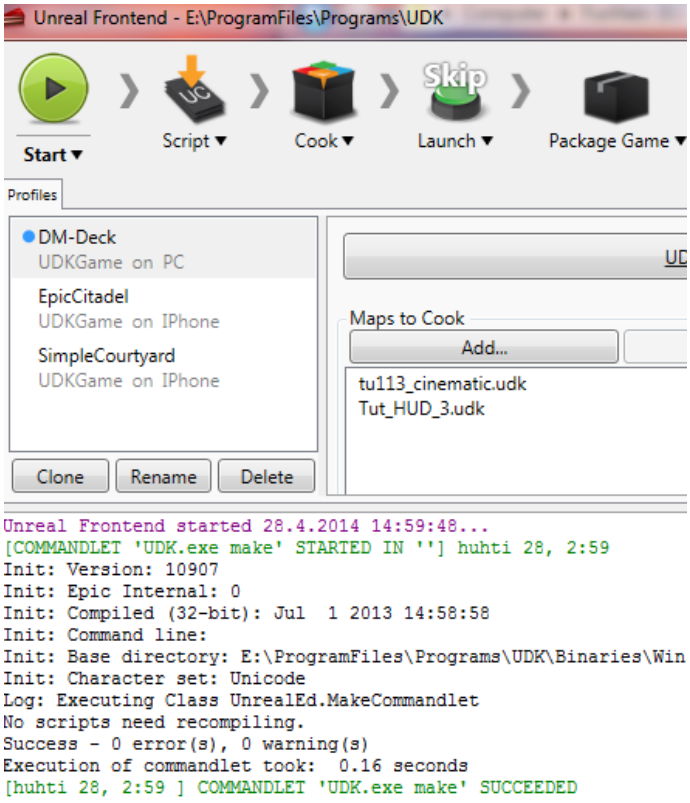
Skriptien on sijaittava asennuskansion alaisessa polussa "UDK/Development/Src" ja halutussa kansiossa (kuva 38, kohta 1). Halutun kansion on sisällettävä "Classes"-kansio, jossa itse skriptit sijaitsevat. Kansion nimi, jossa tarvittavat skriptit sijaitsevat, on lisättävä editorin tietoihin, jolloin järjestelmä tietää, mistä hakea tarvittavia tiedostoja. Tiedot siis lisätään "DefaultEngine.ini"-tiedoston

"[UnrealEd.EditorEngine]"-kategorian alle muodossa "+EditPackages = IsometricTest_2" (kuva 38, kohta 2).



Kuva 38. Skriptejä varten on asetettava koodikansio otettavaksi huomioon.

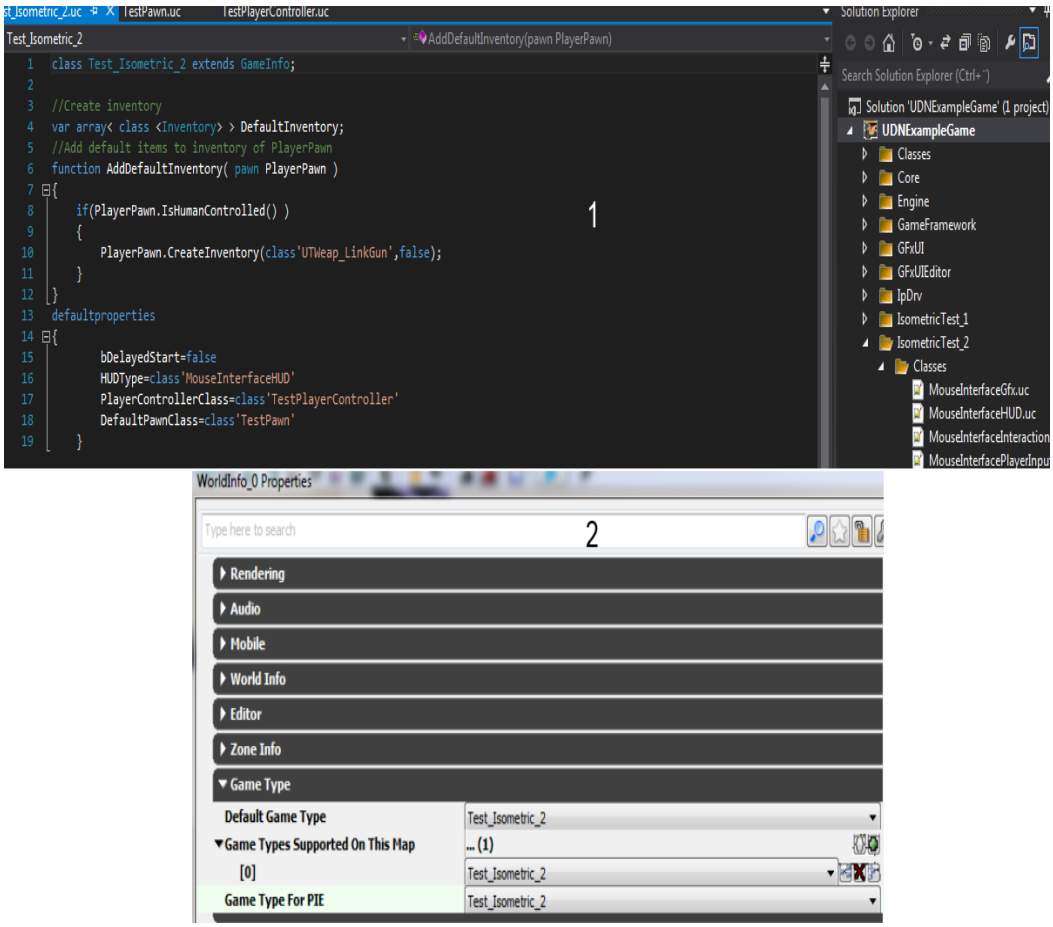
Lisäyksen jälkeen skriptit voidaan ajaa binääreihin joko Unreal Frontend-ohjelman scriptien rakennuksella tai käynnistämällä editori. Editorin käynnistyessä tämä tunnistaa, että tiedostoja on rakennettava, ja kysyy lupaa niiden rakentamiseen. Kolmas vaihtoehto on Visual Studion liitännäisen asetuksista säätää tarvittavat asetukset, jotta se osaa rakentaa tarvittavat lähteet oikeaan binääriin.



Kuva 39. Unreal Frontendillä koodien uudelleenrakentaminen onnistui.

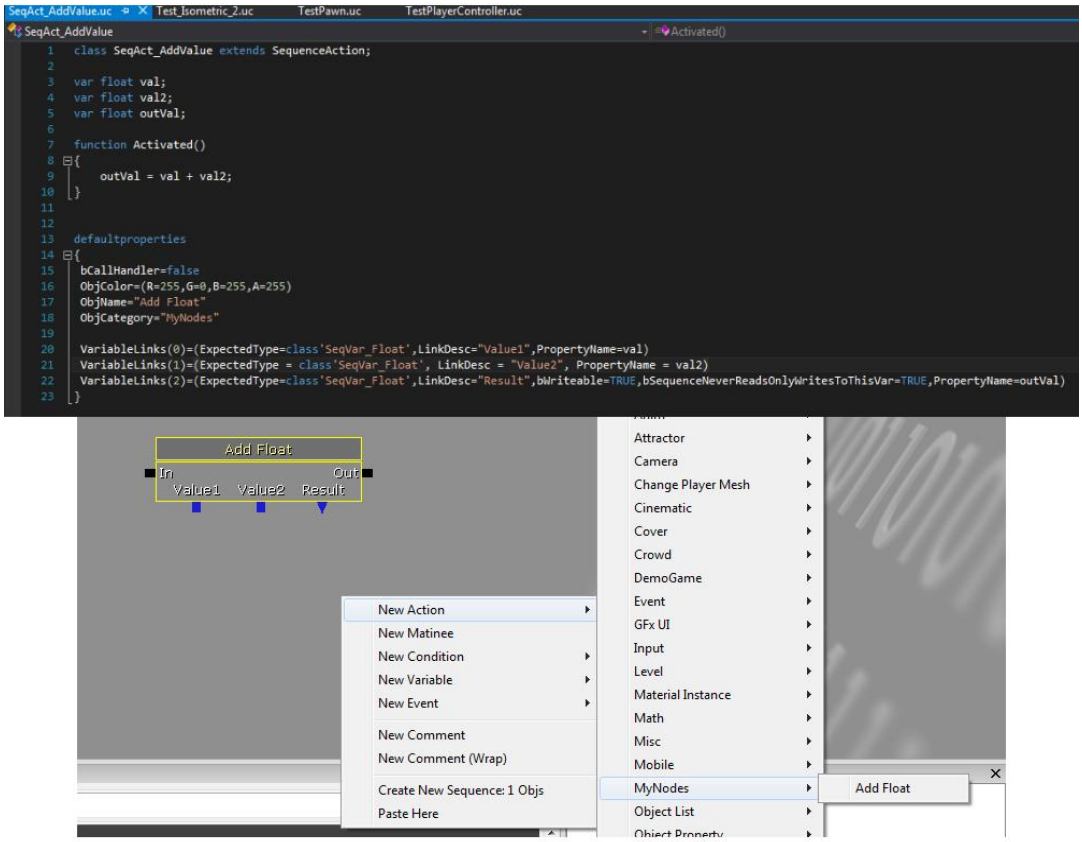
Skripiten luokkarakenne on sekava ja usein dokumentaatio on ajasta niin paljon jäljessä, että ainut tapa tuottaa järkevää tuotetta on tutkia olemassa olevia luokkia. Hyvää ohjeistusta voi saada tarkastelemalla UDK:n perehdytysmateriaalia.

Usein omat skriptit peritään jostain olemassa olevasta luokasta ja kuormitetaan halutut arvot (kuva 40, kohta 1). Uusien luokkien itsenäinen teko onnistuu, mutta käytännössä pelille on annettava ainakin pelityyppi, joka asetetaan kuhunkin rakennettuun karttaan erikseen. Pelityypin omalle kartalle tai tasolle voi määrittää päämuokkaimen "View/WorldProperties"-valikon alta (kuva 40, kohta 2).



Kuva 40. Koodillisesti luotu oma peliluokka, joka on otettu karttaan käyttöön.

Unreal Kismetiin työkalujen skriptaaminen on huomattavasti yksiselitteisempää ja selkeämpää (kuva 41). Työkaluja voidaan luoda helpoiten käyttämällä nFringen valmiita skriptipohjia, jolloin haluttu luokka laajennetaan koskemaan esimerkiksi tapahtumajaksoa. Tällöin luotu logiikka löytyy Unreal Kismetistä tapahtumakategorian alta.



Kuva 41. Unreal Script, joka liittyy saumattomasti Unreal Kismet -muokkaimeen.

Uusi versio (4) UDK:sta tuo mukanaan C++ -ohjelmointikielen tuen ja siten mahdollistaa vapaampaa toimintaa. Vapaampi toimiminen mahdollistuu, sillä C++ tukee useita avoimen lähdekoodin kirjastoja.

2.1.7 UI

Käyttäjän näkemät valikot voidaan luoda UDK:ssa joko Canvasta tai Scaleformia käyttäen. Vaikka molemmat ovat teknisesti vielä käytettäviä, on Canvas jäämässä pois ja sitä ei suositella kuin nopeita esimerkkejä varten.

2.1.7.1 Canvas

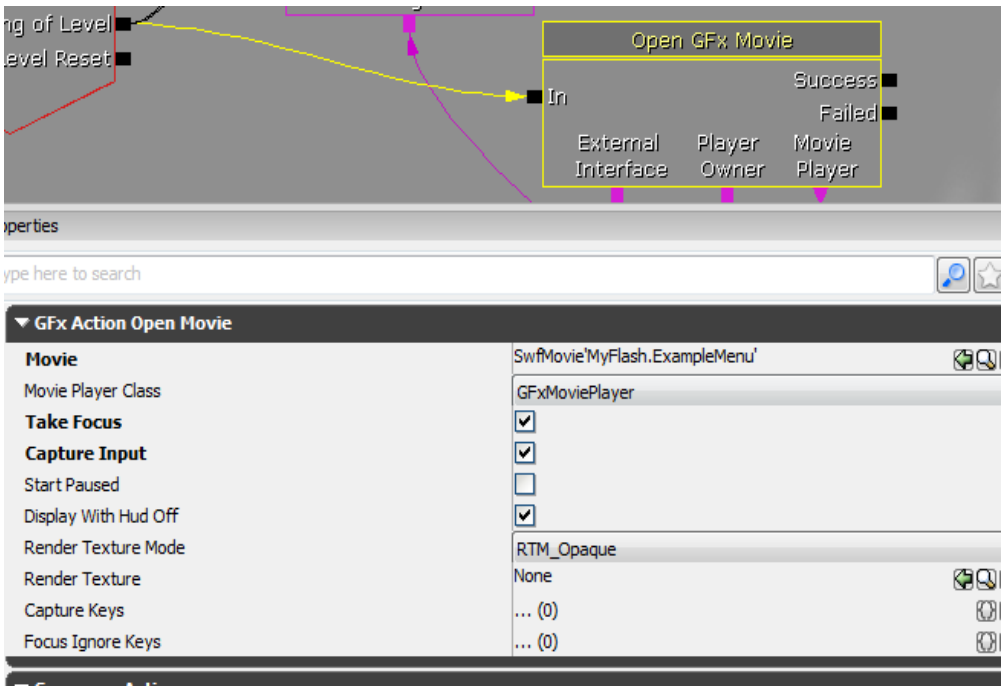
Canvas on perinteinen koodillinen piirtämistapa, jolla voidaan ruudulle luoda objekteja (Canvas Technical Guide). Valitettavasti esimerkiksi nappuloita tehtäessä täytyy käyttäjäsyötteelle tehdä alusta alkaen omat käsittelijät.

Tämä ei sovellu isompiin projekteihin, sillä tiimissä on aina suunnittelijoita ja artisteja, joiden on helposti ja nopeasti päästävä muokkaamaan graafisia ominaisuuksia, kuten objektien sijainteja.

2.1.7.2 Scaleform GFx

Scaleform GFx (tai lyhyesti Scaleform) on ryhmä työkaluja (skriptejä ja Flash-liitännäinen), joilla suunnittelijat voivat helposti muuttaa tai luoda käyttäjävalikoita peliin. Graafisen ulkoasun muuttuessa ei artistin tarvitse vaivata koodin kirjoittajia vaan hän pystyy tekemään haluamansa muutokset suoraan flash-projektiin.

Käytännössä artisti lisää Adobe Flash -ohjelmistoonsa kehitysympäristön mukana tulleen liitännäisen ja voi luoda flah tuotoksia, jotka voidaan lisätä Unreal-projektiin (Doyle). Linkki flashin ja peliprojektin välillä luodaan Action Scriptillä, jota GFx-skriptit tai Unreal Kismet -työkalut pystyvät tunnistamaan (kuva 42).



Kuva 42. Unreal Kismetissä on käytetty scaleform-elokuvan avaussolmua.

2.1.8 Kansiorakenne

Kun Unreal Development Kitin on asennettu, löytyy asennuspolun alta kuukauden mukainen versio kehitysympäristöstä. Version alla löytyy seuraavat alikansiot: "Binaries", "Development", "Engine" ja "UDKGame". Nämä kansiot ovat asennuksen juuressa, jonne lähtökohtaisesti luodun pelin asennuspaketti julkaistaan.

"Binaries" eli binäärikansio sisältää nimensä mukaisesti binääritiedostoja. Kansiosta on löydettävissä muun muassa editorin, esimerkkipelin ja FrontEndin ajettavat tiedostot.

"Development"-kansio sisältää käytännössä koodit, joita käytetään peliä luotaessa.

"Engine" sisältää tarvittavat komponentit, jotta kehitysalusta toimisi.

"UDKGame" sisältää peliprojektille olennaiset sisällöt, skriptejä lukuun ottamatta. Saman kansion alta löytyvässä "Config"-kansiossa voidaan myös yksittäisen pelin asetuksia säätää.

Työtehtävästä riippuen käyttäjä tulee viettämään aikaansa eri kansioissa: koodin kirjoittaja "Development" -kansion alaisissa koodikansioissa, muut toimenkuvat löytyvät "UDKGame/Content"-kansion alta.

2.2 Kolmannen osapuolen ohjelmistot

Unreal Engine 3 käyttää käytännössä omaa skriptikieltään (Unreal Script). Kieli on hyvin Javan omainen objektipohjainen kieli. Yleisimmille tekstieditoreille löytyy käyttäjien tekemiä lisäosia, jotta kielen syntaksi ja automaattitäydennys toimisi. UDK ei sisällä koodieditoria.

2.2.1 Visual Studio 2012 Ultimate ja nFringe

Projektissa käytettiin Visual Studio 2012 Ultimatea, joka itsessään ei tunnista Unreal Script -syntaksia, mutta useiden käyttäjien mainostaman nFringe-lisäosan avulla se onnistuu. Pixel Minenen kehittämä nFringe on Visual Studiolle kehitetty

lisäosa juuri Unreal Scriptiä varten (About nFringe). Uusimmassa versiossa (Unreal Engine 4) ei tällaisia tarvita, sillä se tukee alusta alkaen c++ ohjelmointikieltä. Syntaksi ei ole ainut positiivinen asia, sillä nFringe tuo Visual Studioon itsessään mahdollisuuden luoda "Unreal Projectin", jolloin kehittäjä voi pienellä vaivalla päästä käsiksi kaikkiin koodeihin, joita kehitysalusta pitää sisällään. Usein yhteisö ja dokumentaatio eivät ole ajan tasalla. Tällöin kehittäjän paras työkalu on Visual Studion etsi-komento. Tätä ei voida muilla ilmaisilla työkaluilla saavuttaa samantasoisesti. Lisäosa mahdollistaa haluttaessa myös koodien suoran rakentamisen binääriin sekä pelin käynnistyksen ilman editoria.

Lisäosa on ilmainen ei kaupallisiin tuotteisiin. Se haluaa silti varmistaa projektin nimen ja sähköpostin, kun sitä lisätään Visual Studioon. Tätä "rekisteröintiä" ei tarkisteta mitenkään, joten sattumanvaraiset arvotkin käyvät.

2.2.2 Adobe Flash

Kätevin tapa luoda käyttäjävalikoita on luoda Flash-tiedostoja, joissa Action Script on yhteydessä Unreal Scriptiin tai Unreal Kismet -työkaluihin. Työkaluna ei ole pakollista käyttää Adoben Flash -työkaluja, mutta vaatimus on, että tuotos on flash-tiedosto ja että siinä on vähintään Action Script 2 -tuki.

3 TYÖN TOTEUTUS

Oman pelin työstäminen käytännössä on haaste, sillä on hankalaa päättää aloituskohdasta. Työskentelyä ei paranna itsenäisen työskentelyn aikatauluttamattomuus. Kokemuksen mukaisesti oli järkevintä aloittaa suunnittelulla.

3.1 Suunnitelma ja projektin tavoite

Pelin teon tärkein vaihe on suunnittelu, sillä hyvin suunniteltu on puoliksi tehty. Kun ajatukset on kirjattu muistiin, voidaan niitä helposti karsia rajoitteiden mukaan pois ja säilyttää koherentti rakenne. Tilanteen mukaan keksityt ratkaisut eivät pitkässä teoksessa vaikuta järkevältä.

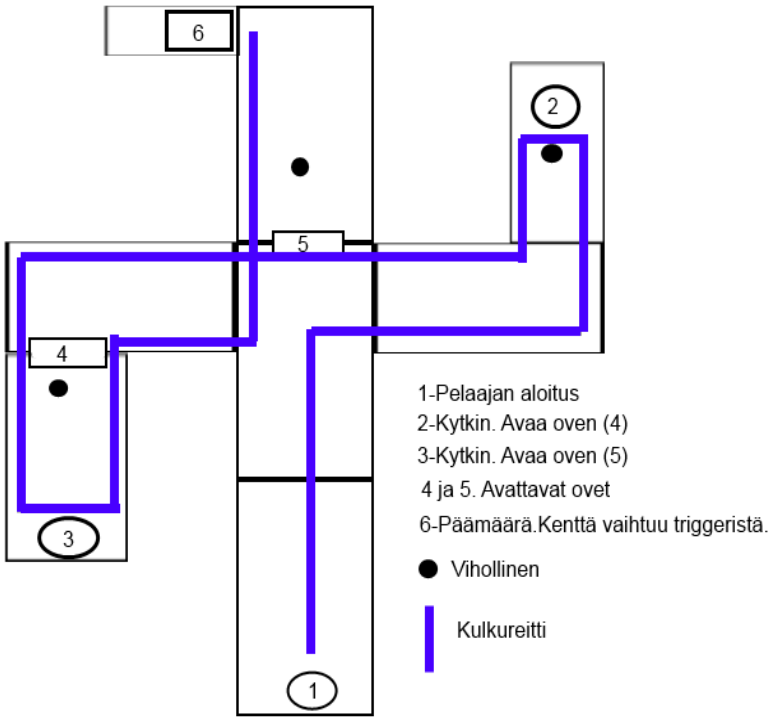
Peli suunniteltiin useamman illan aikana yhdessä toisen tiimin jäsenen kanssa (Winai Prathumwong). Tällöin kirjasimme syntyneet ajatukset muistiin. Seuraavana suunnittelukertana aloitimme tarkastamalla edelliskerran ajatukset ja karsimme pois huonoilta tuntuvat tai teknisesti toimimattomat ajatukset. Suunnitteluun vaikuttivat suosiota nauttineet "Diablo"- ja "Torchlight"-pelisarjat sekä isometristen pelien suosion kasvu.

Päädyimme suunnittelemaan isometrisen pelin, jossa robottiyhdyskunta tuhoaa turmeltuneita biologisia olentoja. Suunnitelman mukaan pelaaja tuhoaisi olentoja ja keräisi niiden hallussa olevaa biomassaa ja teknologiaa kehittääkseen itseään. Alustavat kaaviot luotiin erilaisista esinevalikoista ja kykypuista, mutta ne karsittiin käytetyn ajan painottuessa UDK:n opetteluun. Lopulta päädyimme karsimaan suunniteltuja ominaisuuksia ja päädyimme muodostamaan kevyemmän tavoitteen.

Peliprojektin tavoitteena oli luoda Unreal-pelimoottorin ominaisuuksia esittelevä peli, jossa pelaaja etenee kentästä toiseen ja tuhoaa ihmisvihollisia. Lopullisessa tuotteessa pelaajan näkökulma olisi isometrinen ja toiminnot olisivat hiirestä riippuvaisia. Pelin aikana pelaaja tulee ratkaisemaan kevyitä pulmia.

Opinnäytetyön aikana tuotin valikot, vihollis- ja pelipääluokan pohjan ja yhden toimivan pelikartan Unreal ED -muokkaimella. Tarvittavan kentän luonti aloitettiin luomalla toimintakaavio, joka määrittää pelaajan kulkemisen ja toimintamahdollisuudet (kuva 43).

Ryhmätoverini sai työtehtäväkseen oman taukovalikon käyttöönoton, kaksi pelattavaa kenttää, peliluokan ja vihollisen toiminnan täydennyksen sekä syntyvien ongelmien korjaamisen. Hänen vastuullaan on projektin viimeistely.



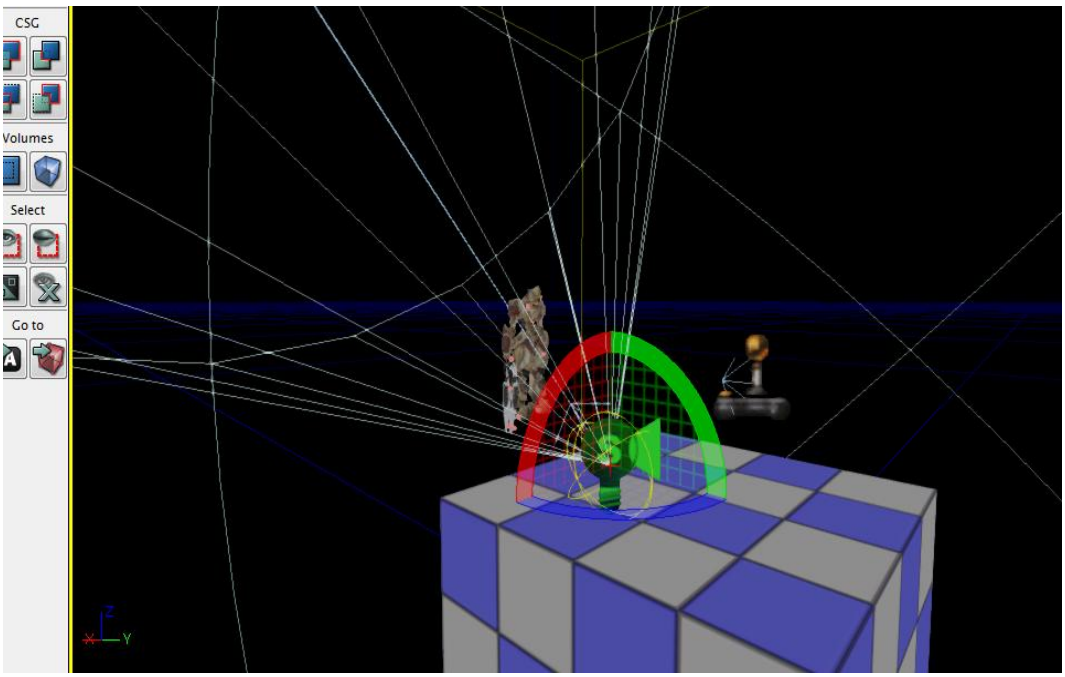
Kuva 43. Microsoft Paint -ohjelmalla on tuotettu karttasuunnitelma.

3.2 Käytännön toteutusta

Käytännön toteutuksissa työstettiin omien osa-alueiden työtehtäviä siten, että lopputulos on pelattava.

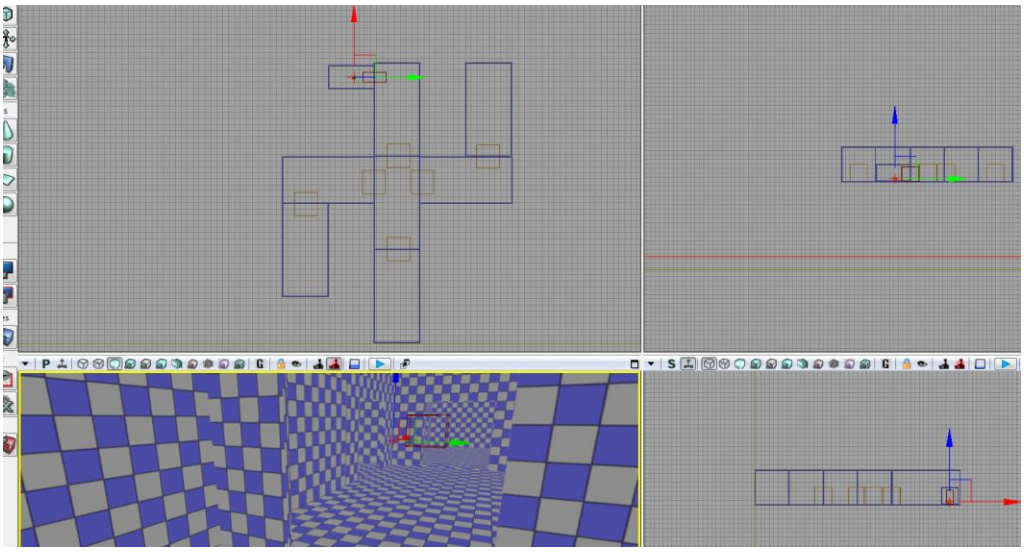
3.2.1 Tarvittavien tasojen luonti

Karttoja ja niihin liittyvien resurssipakettien säilytystä varten luotiin kansiorakenne. Tämän jälkeen loin neljä tarvittavaa karttaa. Nämä sisälsivät kolme erilaista valikkoa ja yhden pelattavan kartan. Valikoita varten luotiin hyvin yksinkertainen kartta, jossa oli valaistus, kamera ja animoitu hahmo (kuva 44).



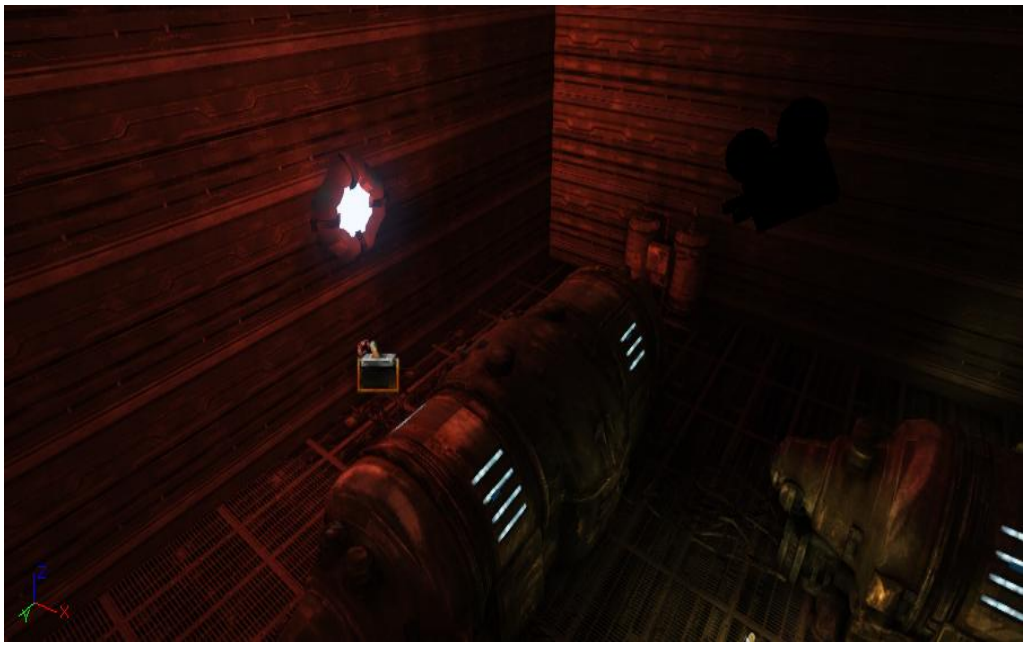
Kuva 44. Karttaan luotu asetelma on valikkoa varten.

Pelattavan kartan tekeminen aloitettiin luomalla laatikkotyökalulla huoneita, jotka yhdistettiin toisiinsa samaisen työkalun poistavalla ominaisuudella. Poistotyökalulla huoneiden välille luotiin oviaukkoja. Lisäksi tasosta poistettiin huoneiden katot, jotta myöhemmin voitaisiin lisätä haluttu kuvakulma. Tähän asetelmaan ensimmäiseksi sijoitettiin pelaajan syntymispiste ja aktivointipiste, joka laukaisee tason voittamisen. Nämä esineet sijoitettiin pelikenttään, jotta sitä voitaisiin testata (kuva 45).



Kuva 45. Huoneita on luotu lisäämällä ja poistamalla geometriaa.

Yksinkertaisten huoneiden ollessa käyttökelpoisia siirryttiin lisäämään huoneisiin yksityiskohtia teksturoimalla, lisäämällä niihin esineitä ja asettamalla valaistuksia (kuvat 46 ja 47). Yksityiskohdat ja valaistus luovat tärkeän osan peliä, sillä ne ovat täydellisiä tunnelman luojia ja voivat helpottaa pelaamista (kuva 48).



Kuva 46. Huoneisiin luodaan tunnelmaa yksityiskohdilla.

Kuva 47. Erilaisilla valaistuksilla on tärkeä osa tunnelman luonnissa.

Yksityiskohtien luonnin jälkeen keskityttiin peliominaisuuksien luontiin Unreal Kismetillä ja siihen luoduilla Unreal-skripteillä. Nämä ominaisuudet mahdollistivat muun muassa elokuvatyyppisen itsestään liikkuvan kameran (kuva 49), vihollisten luonnin, liikkuvan hissin ja ovien avaamisen sekä tiettyjen ehtojen täyttyessä valikkoihin siirtymisen.

Kuva 48. Yksityiskohdilla selkeytetään pelaajan kulkua.

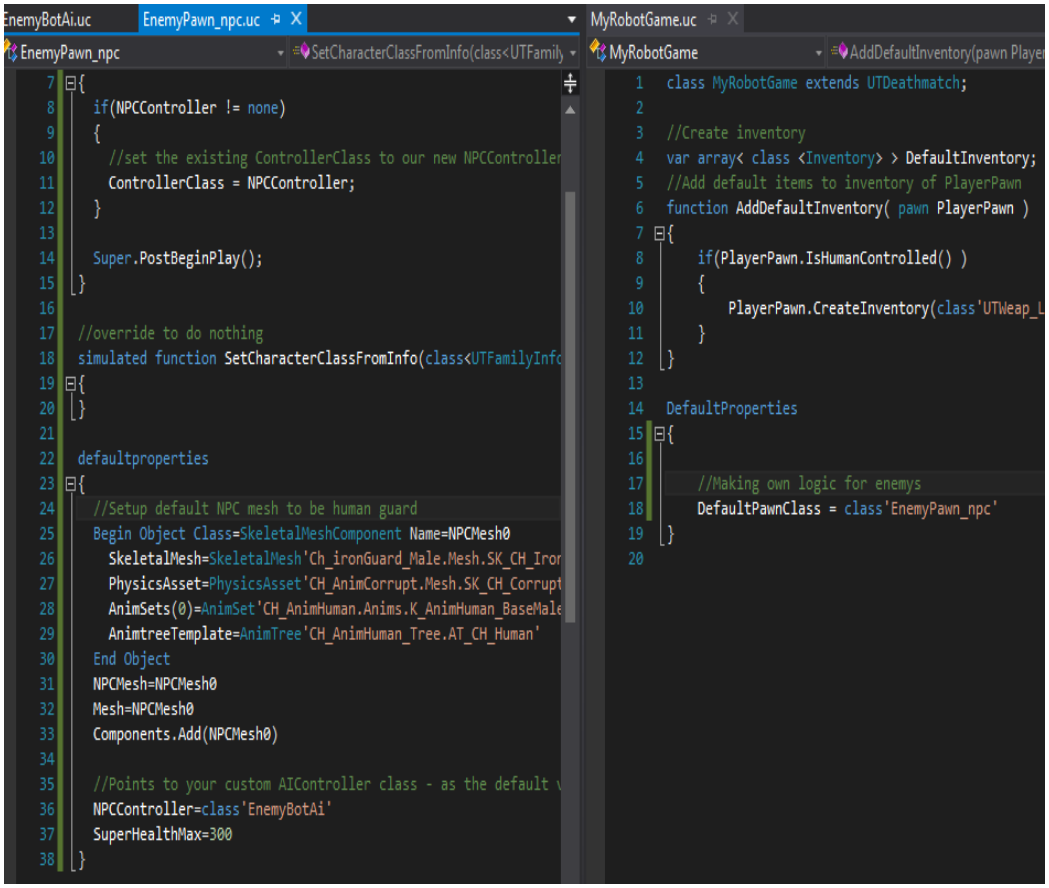
Kuva 49. Elokuvakamera mahdollistaa dramaattiset oven avaukset.

3.2.2 Koodit

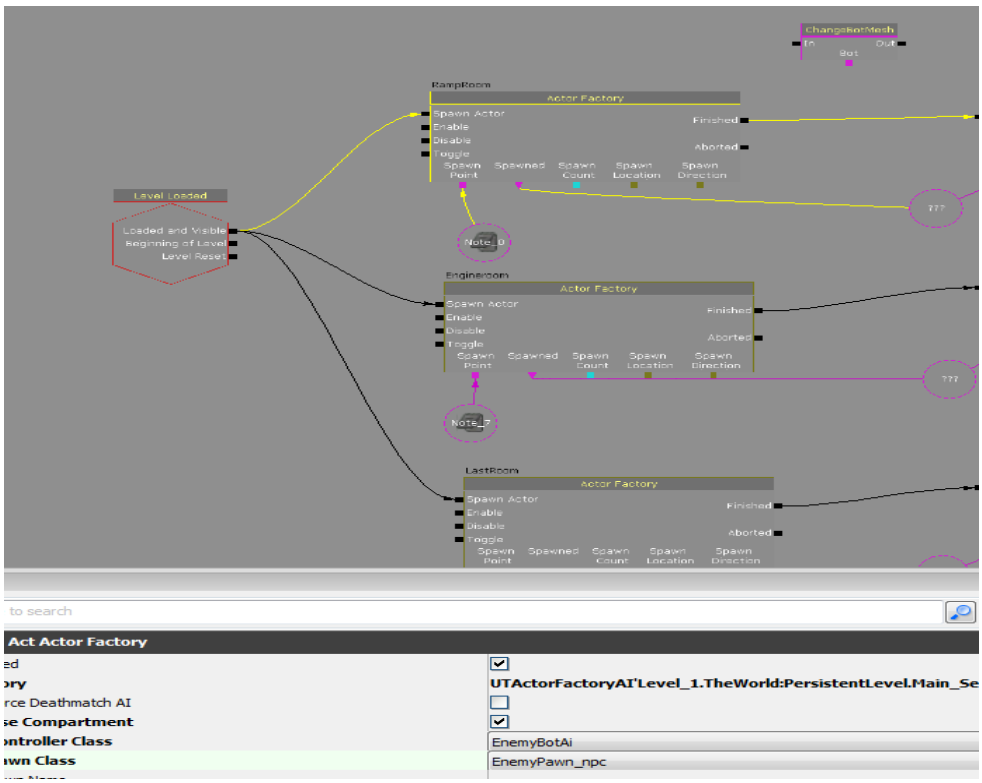
Työn aikana oli tarkoitus luoda vihollisen ja pelin pääluokan skriptit sekä tarvittavia työkaluja Unreal Kismet -ympäristöön.

Pääluokka määritetään kartan maailman asetuksista ja se kertoo pelimoottorille, millaisia sääntöjä tulee noudattaa (kuva 50). Peliasetuksessa koodi laajennetaan, joko koskemaan olemassa olevaa pelityyppiä tai peliinformaatioluokkaa.

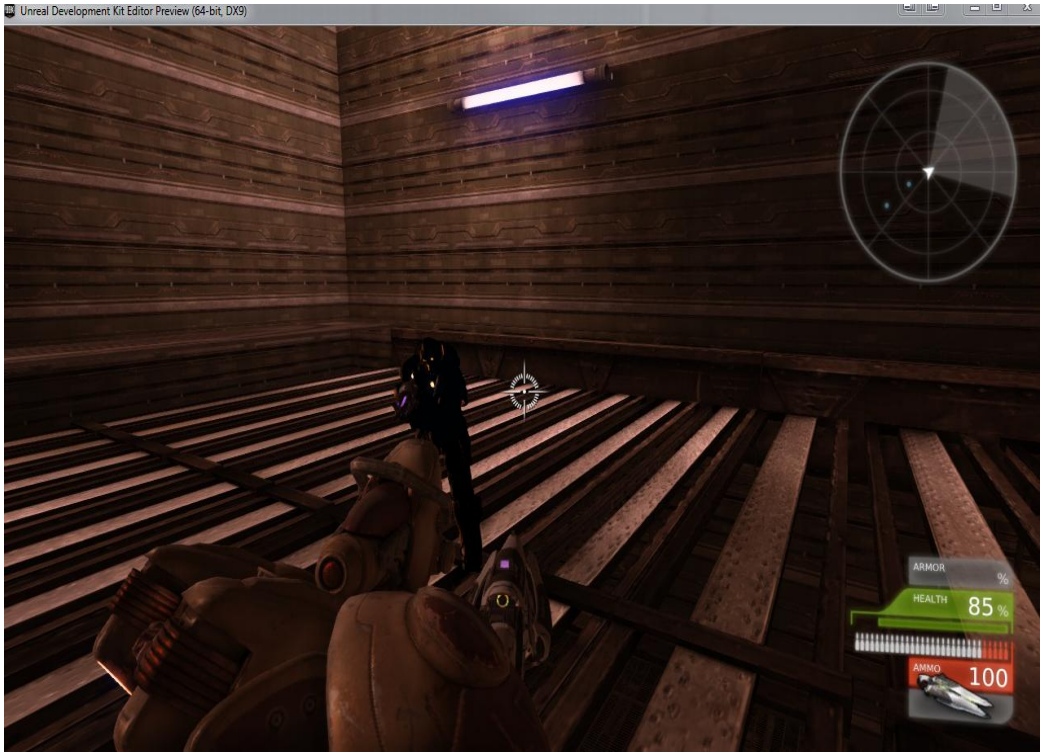
Pelaajan tärkein vastustaja kartassa on vihollinen, ja se luodaankin helposti Unreal Kismetin solmulla "Actor Factory" (kuva 51). Luodulle viholliselle tulee luoda logiikkaa hallitsemaan sen toimintaa ja ominaisuuksia. Onnistuneesti luotu vihollinen voi olla omanlaisensa näköinen (kuva 52) ja toimia, miten tekijä on sen määritellyt (kuva 53). Lisäksi omaan peliin tulee luoda oma hallintakoodi pelaajan toimintaan tai käyttää olemassa olevaa luokkaa.



Kuva 50. Omalle peliluokalle on luotu vihollinen ja sille ominaisuuksia.



Kuva 51. Vihollisten luonti Unreal Kismetillä ja omalla vihollisluokalla.



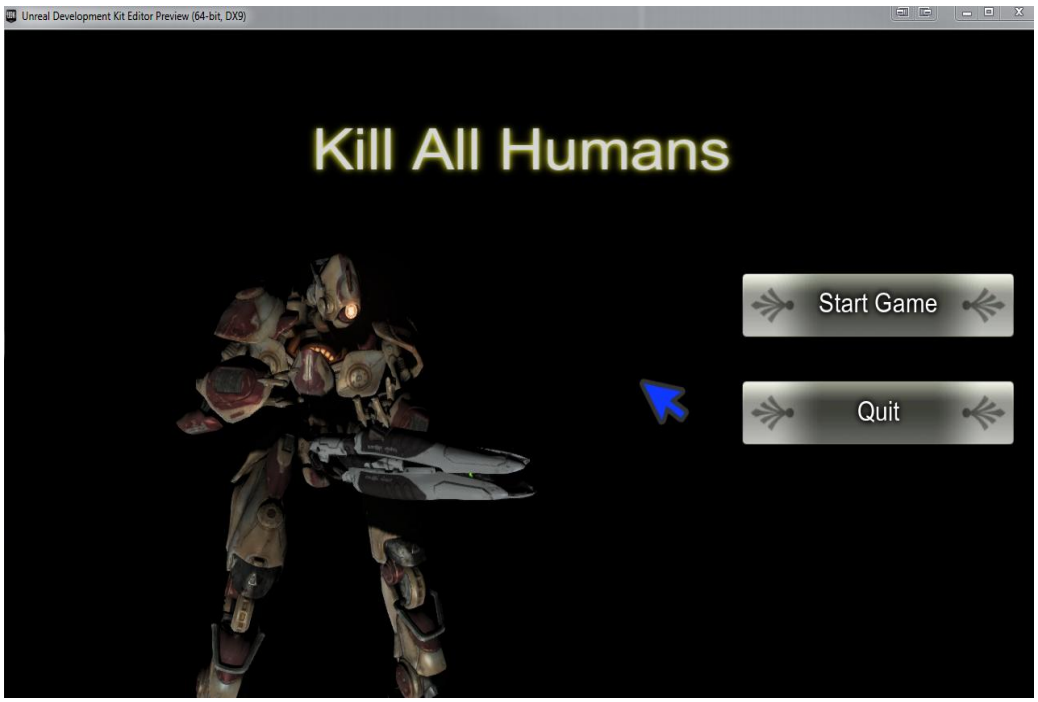
Kuva 52. Luodulla vihollisella on oma ulkoasunsa ja se tunnistaa pelaajan viholliseksi.

Kuva 53. Pelaajan saapuessa vihollisen näkökenttään, se hyökkää.

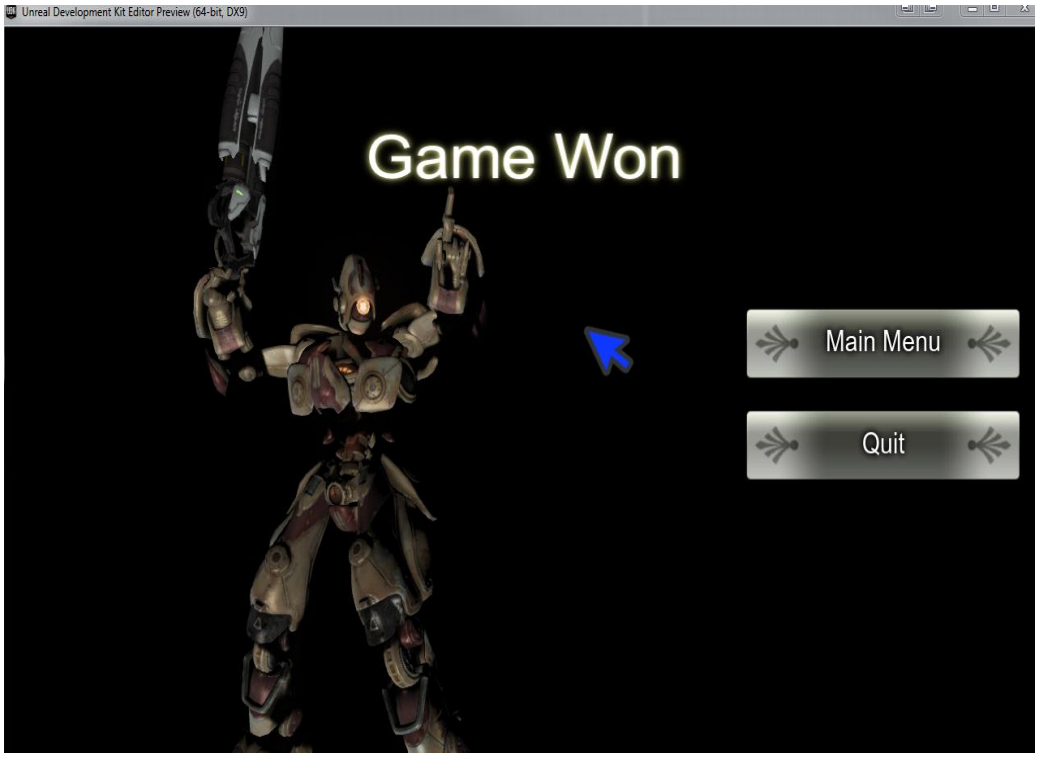
3.2.3 Valikot

Työtä tehtäessä kaikki materiaalit olivat vapaan lisenssin alaisia tai pelimoottorin valmiiksi sisällytettyjä resursseja. Suunnitelmana oli luoda 2D-tasossa olevat flash-valikot, joiden taustat olisivat läpinäkyviä, jotta voitaisiin luoda pelikenttiä, jotka keskittyvät ainoastaan valikoihin. Näissä kentissä yhdistetään animoidut 3D-esineet 2D-tasossa oleviin valikoihin. Tuloksena on elävöitynyt valikko (kuvat 54 ja 55).

Erillisissä pelikentissä sijaitsevien valikoiden lisäksi luotiin taukovalikko, joka on aktivoitavissa kesken pelaamisen. Taukovalikko on luotu ja logiikka asennettu. Pelin pääluokan puuttuessa sitä ei ole kiinnitetty käyttöön. Tilannetta korjaa oletusarvoinen taukovalikko, joka on muokattu toimimaan luotujen valikoiden rinnalla.



Kuva 54. Yhdistämällä 3D-esine 2D-valikkoon saadaan elävyyttä.



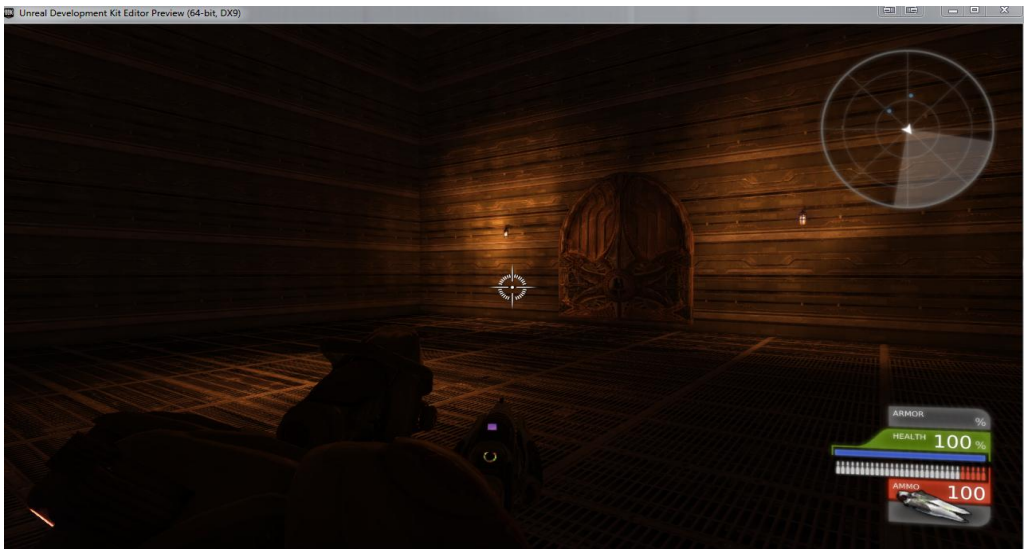
Kuva 55. Animoimalla 3D-esine saadaan valikkoon lisää elävyyttä.

3.3 Pelitestaus

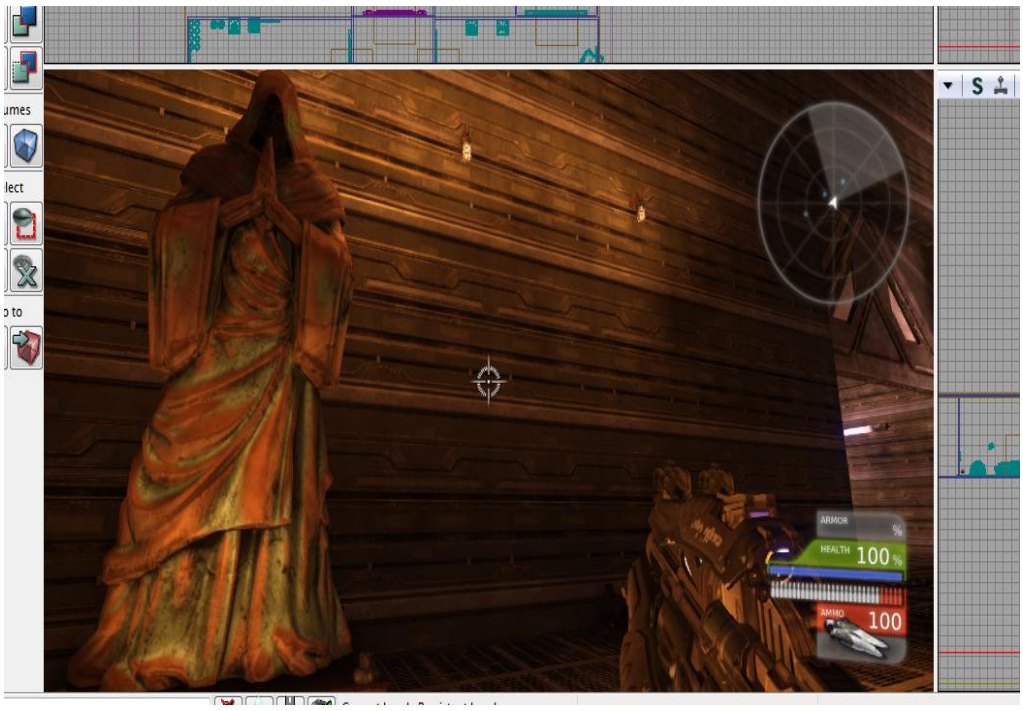
Työn aikana harjoitettiin pelin teon vähiten arvostettua osuutta - pelitestausta. Testaus on yksi tärkeimmistä työkaluista virheiden ja pelin toimivuuden tarkastelussa. Pelaaminen myös kertoo nopeasti, onko toimintalogiikka mielekästä ja hauskaa sekä onko tunnelma haluttu (kuva 56).

Testaus useasti ylenkatsotaan perusteilla: "Se ei ole tuottavaa" ja "testausta voidaan harjoittaa tekijöiden keskuudessa". Useasti kehittäjät testaavat itse tuotostaan ja sokeutuvat teoksensa epäloogisuudelle, sillä he ovat kerta toisensa jälkeen testanneet samaa kohtaa. Opinnäytetyön aikana harjoitettiin kolmea erilaista itsenäistä testausta. Lisäksi pyydettiin ulkopuolisilta tahoilta vapaata mielipidettä heidän katsomanaan aikana.

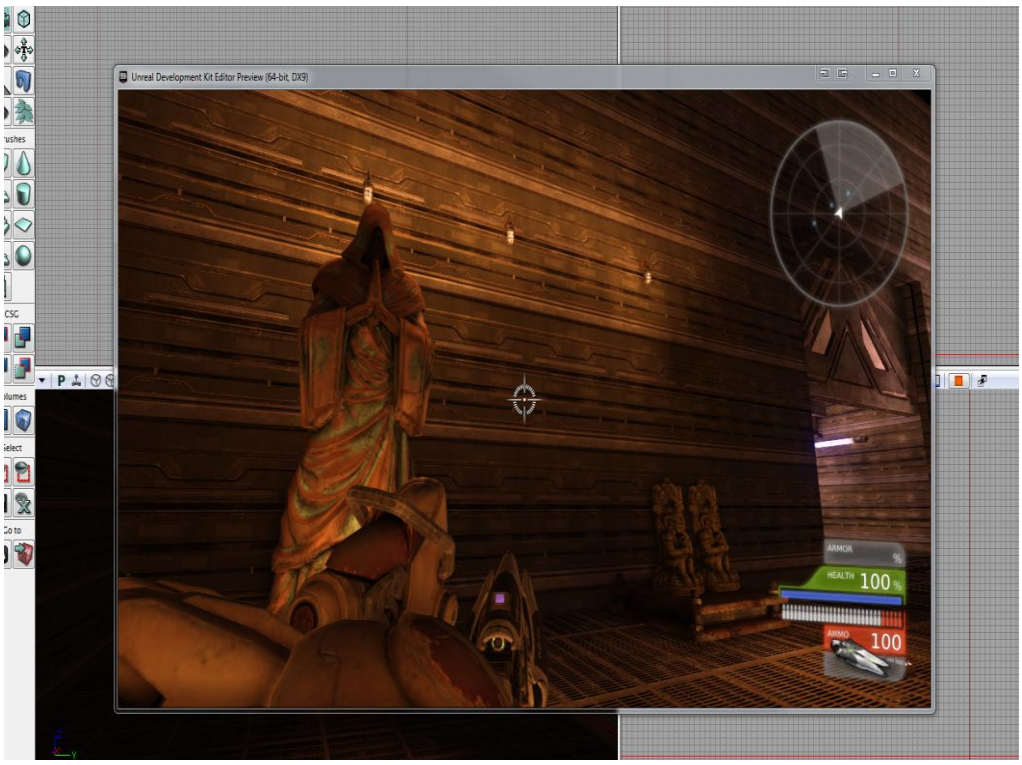
Unreal Development Kit mahdollistaa kolmenlaista testausta, jotka kaikki ovat tärkeitä omissa kohdissaan tuotantoa. Nopein ja tärkein tapa testata on suoraan Unreal ED -muokkaimen 3D-työikkunassa (kuva 57). Tämä testaustapa on kätevä yksittäisten ominaisuuksien testauksessa, sillä peli käynnistyy ikkunassa kohdasta, jossa ikkuna sillä hetkellä on. Suurempaa testausta varten voi valita Unreal ED -muokkaimesta pelin käynnistämisen kohdealustan emulaattorissa (kuva 58). Emuloitu peli alkaa alusta, ihan kuin pelaisi peliä kohdealustalla. Tämä testaus on tärkeä kokonaisuuden hahmottamisessa ja sitä tehdään harvemmin. Viimeinen ja yksi tärkeimmistä testauksista on lopputestaus (kuva 59). Tässä testauksessa varmistetaan, että lopputuote asentuu oikein eikä kohtaa tavallisimmissa tilanteissa ongelmia.



Kuva 56. Pelin testaaminen antaa selvän käsityksen tunnelman toimivuudesta.



Kuva 57. Peliä voi nopeasti testata pääeditorin 3D-työnäkymässä.



Kuva 58. Pelin varsinaista lopputuotosta voidaan simuloida kohdealustalle.

Kuva 59. Lopullinen tuote kannattaa aina testata.

4 YHTEENVETO

Opinnäytetyön tekemiseen aikaan julkaistiin kehittyneempi versio Unreal-moottorista (versio 4). Projektissa otettiin käyttöön vanhemman moottorin kehitysalusta, sillä useat yritykset voivat käyttää vanhempaa vielä jonkin aikaa. Lisäksi vanhan päälle on parempi lähteä opettelemaan uusia konsepteja. Nykyinen versio on ilmainen.

Kehitysalustalla saadaan äärettömän hienoja tuotoksia aikaiseksi, mutta se ei silti tee siitä kovinkaan helppokäyttöistä. Ohjelmisto päivittyy melkein kuukauden välein, ja useasti voidaan skriptien luokkia muuttaa hyvinkin radikaalisti. Ongelmaa ei syntyisi, jos materiaalia päivitettäisiin. Korvaavista muutoksista ilmoitetaan ainoastaan muutoslokissa. Muutosten löytäminen ei ole läheskään käyttäjäystävällistä. Käyttäjäkunnalta kysyminen keskustelupalstoilta ei aina takaa tuloksia, sillä useimmiten vastaukset viittaavat tutustumaan dokumentaatioon. Vaihtoehtoisesti vastausta joutuu odottamaan vuosia.

On hyvin todennäköistä, että uuden version myötä käyttäjäkunta aktivoituu enemmän ja uusi C++ -ohjelmointikielen tuki tulee lisäämään kiinnostuneiden määrää. Valitettavasti kehitysalustan saa toistaiseksi vain maksullista lisenssiä vastaan. Kiinnostus on ollut kehitysympäristöä kohtaan kova jo vuosia, ja siksi odotan julkaisua innolla. Mielestäni monet yritykset kiinnostuvat virtaviivaisemmasta kehitysympäristöstä.

Itse opinnäytetyön työstämisen olisin aloittanut paljon aikaisemmin, mikäli olisin ollut tietoinen dokumentaation virheellisyydestä jo aikaisemmin. Vaikka ajatus ohjelmiston testikäytöstä myös niin sanotusti tiimikäytössä olikin todella hyvä, oli se myös rajoittava tekijä. Suunnitteluvaiheessa tuli aikatauluongelmia ja kommunikaatiokatkoksia, joiden jättämien turhien "tyhjäntoimittajahetkien" tilalla koetin opiskella dokumentaatiota ja toteuttaa esimerkkipelejä, joilla pystyin esittelemään yksittäisiä ominaisuuksia.

Itse opinnäytetyön tekemiseen ei löytynyt yhtenäistä kunnollista ohjetta, joka vastaisi koulutusalaani. Kieliopillisia ohjeistuksia ja turhankin tarkkaa pilkunviilausta löytyi enemmän kuin tarpeeksi, mutta yhtenäistä ja selkeää ohjeistusta sisällön rakenteeseen ei löytynyt tarpeen mukaan. Lisäksi

toiminnalliset ohjeet ja aikataulu opinnäytetyön vaiheista oli todella hankala löytää. Tarvittaessa osaavat laboratorioinsinöörit osasivat onneksi ohjeistaa oikeaan suuntaan.

LÄHTEET

3D Buzz. 30.11.2009. Using UDK. Saatavissa: https://forums.epicgames.com/threads/710137-3D-Buzz-Video-Tutorials-Using-UDK [viitattu 12.4.2014].

About nFringe. Saatavissa: http://pixelminegames.com/nfringe/ [viitattu 15.4.2014].

Adamson T. Introduction to UnrealScript. Saatavissa: http://www.udk3developer.com/PDF/USTutorial-1.pdf [viitattu 15.4.2014].

Canvas Technical Guide. Saatavissa: http://udn.epicgames.com/Three/CanvasTechnicalGuide.html [viitattu 17.4.2014].

Doyle, Matt. 3.9.2010. Getting Started with Scaleform & UDK. Saatavissa: https://forums.epicgames.com/threads/743567-Getting-Started-with-Scaleform-amp-UDK [viitattu 17.4.2014].

How to Cook a Level in UDK. Saatavissa: http://www.ehow.com/how_6029103_cook-level-udk.html [viitattu 12.4.2014].

Learning Unreal Engine 3 - UE3 Editor. Saatavissa: http://waylon-art.com/LearningUnreal/ [viitattu 12.3.2014].

Roberts, Bucky. Lecture 16: Unreal Development Kit UDK Tutorial - 16 - Content Browser. Saatavissa: http://freevideolectures.com/Course/3214/Unreal-Development-Kit-UDK/16 [viitattu 14.3.2014].

Tutorials - UE3 Cutscenes. Saatavissa: http://www.hourences.com/tutorials-ue3-cutscenes/ [viitattu 18.3.2014].

UDK: Introduction to Kismet. Saatavissa: http://www.worldofleveldesign.com/categories/wold-members-tutorials/petebottomley/udk-kismet-introduction.php [viitattu 17.3.2014].