HAAGA-HELIA
ammattikorkeakoulu

Kentän luominen peliprojektiin

Vesa Mankinen

Opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

2012

HAAGA-HELIA
ammattikorkeakoulu

Tietojenkäsittelyn koulutusohjelma

Tiivistelmä

19.11.2012

Tekijä tai tekijät Vesa Mankinen	**Ryhmä** TM07
Opinnäytetyön nimi Kentän luominen peliprojektiin	**Sivu- ja liitesivumäärä** 38
Ohjaaja Heikki Hietala	

Viime vuosina visuaaliset efektit tietokonepeleissä ja elokuvissa ovat kehittyneet melkein fotorealistisiksi. Teknologia on antanut artisteille mahdollisuuden luoda melkein realistisia ympäristöjä peleihin tai elokuviin.

Opinnäytetyön tarkoituksena on oppia 3D mallinnuksen perusteet, grafiikan luominen Blender ja Photoshop ohjelmia käyttäen ja luoda tarvittava grafiikka peliprojektin testi kenttään.

Case osa opinnäytetyöstä sisältää grafiikan suunnittelun ja luomisen peliprojektiin. Luominen sisältää objektien mallintamisen ja teksturoinnin. Mallin typologin suunnittelun ja viimeiseksi kaiken yhdistämisen valmiiksi objekteiksi ja peliprojektin testikentäksi.

Työn lopputuloksena syntyi graafinen osuus peliprojektin testikentästä, jota käytetään myöhemmin testiympäristönä pelin muiden osien luomisessa ja testauksessa.

Asiasanat
Blender, mallintaminen, teksturointi, grafiikka

HAAGA-HELIA
University of Applied Sciences

Abstract

November 19, 2012

Degree Programme in Information Technology

Author or authors Vesa Mankinen	**Group** TM07
The title of thesis Creating a level to game project	**Number of pages and appendices** 38
Supervisor Heikki Hietala	

In recent years the visual effects in computer games and movies have advanced close to photorealism. Technology gave artists the means to create almost realistic environments in games or in movies.

The purpose of this Thesis is to learn the means to create 3D graphics with software's like Blender and Photoshop and create needed graphics to build a test level in a game project.

The case part of the thesis consists of designing and creating graphics for the game project. Creation consists of modeling and texturing of objects and planning the typology of the object and finally putting them all together to create finished objects and test level.

Result of this thesis was to build a test lever for the game project, which will later function as a testing environment for other aspects of the game.

Key words
Blender, modeling, texturing, graphics

Sisällys

Keskeiset käsitteet 1
1 Johdanto 2
1.1 Tavoitteet 3
1.2 Rajaus 3
2 3D suunnittelun historia 4
3 Pelien historia 4
3.1 Arkadipelit 5
3.2 3D pelit ja niiden tekniikka 5
4 Pelimoottoreiden tehtävät 6
5 3D suunnittelun perusteet 7
5.1 Mallin osat 7
5.2 Mallinnus 8
5.3 Modulaarisuus 9
5.4 Colliderit 10
5.5 Tekstuurit 10
5.6 Mallin UV ja sen purkaminen 11
5.7 Shaders 11
5.8 Mapit 12
6 Kentän luomisessa huomioitavat osiot 14
6.1 Kentän visuaalisuus 14
6.2 Kenttä tyypit 14
7 Case: Testi kentän luominen 16
7.1 Suunnittelu 17
7.2 Objektien luominen 21
7.3 Objektien viimeistely ja kentän luonti 31
8 Pohdinta 35
Lähteet 37

Keskeiset käsitteet

Vertices on piste 3D mallissa, jossa yhdistyy ainakin kaksi reunaa**(Edge).**

Edges ovat reunoja, jotka luovat pintoja**(Faces).**

Polygonal Faces ovat tasoja ja pintoja mallissa.

Low Polygon on 3D mallinnus tapa, jossa yritetään tehdä hyvää jälkeä pienellä määrällä pisteitä.

Framerate eli kuvataajuus tarkoittaa kuinka monta kuvaa piirtyy näytölle sekunnissa.

Collider on 3D malliin lisättävä ylimääräinen objekti, jonka avulla peleissä lasketaan ja hallitaan törmäyksiä.

Tekstuurilla tarkoitetaan 2D pintaa mikä annetaan mallille.

Shaderit ovat pieni ohjelmia, jotka tuovat ylimääräisiä efektejä malleihin.

Diffuse Map on 2D tekstuuri, jossa tuodaan esille mallin värit.

Specular Map on 2D tekstuuri, jolla vaikutetaan, miten valo vaikuttaa malliin.

Normal Map on 2D tekstuuri, jolla luodaan ylimääräisiä yksityiskohtia malleihin. Yksityiskohdat toimivat vain valaistuksen kanssa.

Opacity Map on 2D tekstuuri, jolla vaikutetaan mallin läpinäkyvyyteen.

Mesh on kokonaisuus vertexeja, edgeja ja faceseja, jotka määrittelevät yhdessä objektin muodon.

1 Johdanto

Nykyään peliteollisuuden liikevaihto kilpailee elokuvateollisuuden liikevaihdon kanssa. Englannissa vuonna 2009 peliteollisuus ylitti jo elokuvanteollisuuden liikevaihdon. 80 ja 90- luvun aikoina pelien luontiin riitti parikin tekijää. Pienet tiimit loivat sen ajan pelejä. Tiimeissä yksi henkilö pystyi toimimaan monissa rooleissa. Tämän mahdollisti tekniikan yksinkertaisuus ja rajoitukset. Pelien graafinen ulkonäkö ja niissä käytettävien kenttien tai ympäristöjen luominen oli erittäin yksinkertaista.

Nykyajan PC- , konsoli- ja älypuhelimien peleissä on alettu yhä enemmän käyttämään 3D-tekniikkaa ympäristön luomiseen. Tämä on luonut peliteollisuudelle ja peliprojekteille tarpeen erillisille artisteille ja kentänluojille. Koko peliala on jakaantunut erilaisiin erikoisosaajiin, jotka tekevät yhtä osaa tai aluetta projekteissa. Enää ei oleteta, että yksi henkilö pystyisin täysin hallitsemaan tai ehtisi tehdä kaikki pelinteossa tarvittavat osa-alueet.

Vaikka tekniikka on tehnyt suuria harppauksia eteenpäin. Se ei silti vieläkään kykene luomaan reaaliaikaista fotorealistista kuvaa ja videota. Tämän takia pelinkehittäjät joutuvat käyttämään erilaisia vaihtoehtoisia menetelmiä, luodakseen vaikutelma aidosta ympäristöstä saadakseen melkein fotorealistisia ja reaaliaikaisia grafiikoita peleihinsä. Näiden menetelmien opetteleminen ja hallitseminen ovat yksi tärkeimpiä keinoja luoda uskottavia ja fotorealistisia ympäristöjä nykyajan peleihin.

1.1 Tavoitteet

Opinnäytetyö tehdään suomalaiseen peliprojektiin nimeltä Terra Rapio. Projekti on ollut suunnittelussa vuoden verran. Peli sijoittuu osittain nykyiseen maailman dystopiaan. Opinnäytteen tavoitteena on luoda testipelikenttäprojektiin ja samalla selvittää mitä kaikkea uskottavan pelikentän pitäisi pitää sisällään. Tavoitteisiin kuuluu myös tekniikan oppiminen, jotta sitä voidaan käyttää myöhemmin peliprojektissa hyväksi. Tästä opinnäytetyöstä saatu osaaminen tullaan hyväksikäyttämään ja osittain koko opinnäytetyö tulee olemaan myös apuna muille projektin jäsenille, jos he tulevat tekemään tulevaisuudessa pelikenttiä tai siinä tarvittavia osioita.

1.2 Rajaus

Opinnäytetyössä keskitytään pelimaailman visuaalisten elementtien tuottamiseen eri työvaiheiden kautta: 3D- mallintaminen, mallien teksturoiminen, sekä mahdollisimman uskottavan pelimaailman luomien. Nämä on tarkoitus luoda itse tuotettuja elementtejä ja alan käytäntöjä noudattaen. Tekniikoilla joita opinnäytteessä käytetään yritetään luoda realistista grafiikkaa. Opinnäytetyössä otetaan myös huomioon tekniikan rajoitukset, mitkä esimerkiksi tulevat vastaan, jos pelikenttä luodaan mobiiliympäristöön. Eli opinnäytetyössä ei tulla huomioimaan parin vuoden sisällä markkinoille tulleita uusia tekniikoita, kuten DirectX 11:n tuoma Tesselaatio, koska mobiiliympäristössä prosessointiteho ei ole vielä tarpeeksi suuri.

Kentässä tarvittavat 3D-mallit tehdään Blender- sovelluksella. Blender on 3D mallinnus sovellus, joka on maksuton käyttää. Tekstuurit luodaan joko itse ottamista valokuvista tai Internetistä löydetyistä vapaasti käytettävistä kuvista ja ne muokataan Photoshop sovelluksessa. Pelikenttä muutetaan semmoiseen muotoon, että sen voi viedä Unity3D sovellukseen, jolla itse peli luodaan.

2 3D suunnittelun historia

3D suunnittelu aloitettiin 1960- luvulla. Suunnitteluja William Fetter yritti maksimoida Boeing lentokoneen ohjaamoa, jolloin hän loi ensimmäisen tietokoneella tuotetun ortografisen projektion ihmisestä. Fetter kehitti termin tietokonegrafiikka, jolla hän kuvaili tuotoksensa. Tämä aloitti suuren ketjureaktion, joka on mullistanut koko maailman viihde-, mainos- ja media-alan. (Raj 2007.)

Toinen 3D suunnittelun pioneereista oli Ivan Sutherland, joka vuonna 1963, suunnitteli Sketchpad:in, jossa henkilö kykeni piirtämään valokynällä kuvia tietokoneen ruudulle. Tämä oli ensimmäinen prototyyppi graafisille käyttäjärajapinnoille. Tämä ominaisuus on nykyään välttämätön nykyajan tietokoneavusteiselle suunnittelulle**(CAD)**.(Shklyar)

Ensimmäiset kaupalliset CAD:it olivat isoilla auto ja lentokone teollisuudella. Vain suurilla yrityksillä oli varaa tietokoneisiin, joissa oli tarpeeksi prosessointikykyä. Tietokoneiden hinnan laskiessa, myös kehitys CAD:n PC sovelluksille alkoi. Tämän jälkeen CAD:ista tuli universaali sovellus kaikilla rakennusaloilla. (Raj 2007.)

1970-luvulla CAD:eilla tuotettiin 2D piirustuksia, jotka olivat samalla tasolla käsin piirrettyjen kanssa. 1980-luvulla tekniikan kehittyessä kyettiin luomaan 3D malleja. Nykypäivänä CAD on yksi päätyökaluista, jota käytetään tuotteiden suunnittelussa. (Raj 2007.)

3 Pelien historia

Videopelit keksittiin 1950 -luvulla. Niitä pelasivat vain muutamat, sen ajan isoilla tietokoneilla. Ensimmäiset peliohjelmoijat olivat Yhdysvaltalaisia opiskelijoita, jotka toimivat tietokone laboratorioissa isoissa yliopistoissa ja työntekijöitä Brookhavenin kansallisessa laboratorioissa. Sen ajan peleissä oli joko erittäin yksinkertaiset grafiikat tai ei grafiikkaa ollenkaan ja ne näytettiin pienistä mustavalkoisista oskilloskooppi ruuduista. (Rogers 2010, 4-6.)

3.1 Arkadipelit

Arkadipelejä alkoi ilmestymään baareihin 1970 -luvulla. Ensimmäisissä arkadi peleissä grafiikka renderöintiin vektori grafiikalla, eli kuvat luotiin viivoista. Myöhemmissä arkadipeleissä, kun väri rasterigrafiikka keksittiin, kuvat luotiin piste jonoista, joita kutsutaan pikseleiksi, alkoivat sarjakuvien inspiroimat pelit ilmestymään kuluttajille. Näistä esimerkiksi Pac-Man vuonna 1980 tai Donkey Kong vuonna 1982. Näistä peleistä tuli pop ikoneita melkein yhdessä yössä. 1980-luvun puolivälissä arkadi pelit levisivät joka paikkaan. Peligenret ja teemat muuttuivat laajemmiksi ja peliohjaimet ja kabinetit tulivat realistisemmiksi ja ulkonäöllisesti hienoiksi. Tekniikan kehittyessä 1980-luvun lopussa, isojen arkadi pelikoneiden ylläpito kustannukset alkoivat olla suuria ja kotijärjestelmät alkoivat kilpailla arkadipelien kanssa. Nopeasti kotijärjestelmien grafiikka ja teho menivät ohi arkadipelikoneista. (Rogers 2010, 4-6.)

3.2 3D pelit ja niiden tekniikka

Ensimmäinen reaaliajassa renderöityvä 3D peli oli Zig-Zag, joka vuonna 1984 ilmesty 8 bittiselle ZX Spectrum tietokoneelle. Myöhemmin pelistä ilmestyi myös muille tehokkaammille koneille. 3D pelien käännekohta tapahtui vuonna 1992, kun Wolfenstein 3D julkaistiin 16 bittiselle PClle. Se oli ensimmäinen peli missä käytettiin tekstuureita. Vuonna 1996 julkaistu Quake oli seuraava läpimurto 3D tietokone peleissä. Siinä jokainen peli malli oli 3Dta. (Valient 2001.)

Vuosi 1997 oli vallankumouksellinen vuosi 3D peleille. Ensimmäinen 3D kortti(**accelerator**) luotiin PClle. Se oli Voodoo 3Dfx:ltä. Kortti toi monia uusia ominaisuuksia, kuten bilineaarisen suodatuksen tekstuureille ja sumu. Tämä antoi kehittäjille mahdollisuuden jättää vaikeimmat kohdat pelimoottorin ohjelmoinnissa väliin. Tästä syystä pelit olivat paremman näköisiä, nopeampia ja sisälsivät enemmän yksityiskohtia. Samoihin aikoihin pelien kehittäjät alkoivat käyttää OpenGL:ää. Se on ei objekti orientoitunut kirjasto, joka mahdollistaa laadukkaan ja nopean renderöinnin 2D ja 3D grafiikoille. OpenGL:ää kehitetään ja päivitetään ARBn**(Architecture review board)** toimesta, johon kuuluu yrityksiä kuten SGI, NVIDIA ja 3D Labs. OpenGL kehittyi hitaasti, koska järjestelmien laajennuksia eli niitä toimintoja mitä ei löydy OpenGL:ssä, ei valvota millään tavalla. Jokainen laitevalmistaja määrittelee omat laajennuksensa, jotka

toimivat vain tietyillä näytönohjaimilla. Tämän takia kehittäjien täytyy kirjoittaa erinäköisiä koodi yhdistelmiä saataville laajennuksille, että näytönohjaimille. Tämä taas nykypäivänä sotii OpenGL:n tarkoitusta vastaan, joka on olla laitteistoriippumaton. (Valient 2001.)

Microsoft on kehittänyt oman kilpailijan OpenGL:lle. Se on nimeltään DirectX, olio pohjainen kirjasto, joka mahdollistaa erittäin matalan pääsyyn grafiikka, ääni ja syöte laitteistoon Windowsista. Sitä on kehitetty yhdessä suurien laitevalmistajien kanssa ja tämä mahdollistaa nopeamman reaktion uusien tekniikoiden tukemiseen, kuin OpenGL:ssä. (Valient 2001.)

Vuonna 1999 NVIDIA julkaisi uuden grafiikkasirun GeForce256. Sirua kutsuttiin GPU:ksi(**geometry processor unit**). Ero vanhoihin siruihin oli tapa, miten siru prosessoi renderöintia. Vanhoissa siruissa annettiin laskennan tapahtua CPU:ssa(**computer processor unit**) ja siru itsessään vaan muunsi, valaisi ja teksturoi kolmion. GPU itse laski nämä tapahtumat omassa prosessorissaan. Vuonna 2001 syntyi toinen vallankumouksellinen muutos arkkitehtuuriin. NVIDIA julkaisi GeForce3:sen. Tämä siru tarjosi ohjelmoitavia vertexin ja pikselin prosessointi kaistan. Kehittävät pystyivät tämän avulla luomaan omia muutoksia ja vaikutuksia grafiikkaan, mikä ei ennen ollut mahdollista GPU siruilla. Näitä muutoksia kutsutaan sharedeiksi. (Valient 2001.)

4 Pelimoottoreiden tehtävät

Pelimoottorilla tarkoitetaan monimutkaista ohjelmistoalustaa, jonka vastuulla on pelin visualisointi ja äänimaailma. Se käsittelee käyttäjän antamia syötteitä ja tarjoaa resurssin hallintaa, animaatioita ja fysiikkaa.(Valient 2001.)

Pelimoottorin konsepti on hyvin yksinkertainen. Se hoitaa normaaleja peliin liittyviä töitä, kuten renderöinnin ja fysiikan, jotta pelinkehittäjät voivat keskittyä yksityiskohtiin, jotka tekevät pelistä uniikin. Moottoreissa on usein uudestaan käytettäviä komponentteja, joita manipuloimalla kyetään tuomaan peliin elämää. Latauskohdat, mallien animaatiot, törmäyksen havaitseminen objektien välillä, fysiikka, graafiset käyttöliittymät ja jopa osa pelin keinotekoisesta älystä voivat olla kaikki komponentteja, jotka luo-

vat yhdessä pelimoottorin. Toisaalta pelin sisältö, mallit ja tekstuurit, ajatukset objektien törmäysten taustalla ja miten objektit toimivat pelimaailmassa, ovat komponentteja, jotka tekevät pelistä pelin ja ne ovat sisältöä mitä pelinkehittäjät luovat.(Ward, 2008.)

Pelimoottorit ovat tietyllä tavalla uniikkeja peliteollisuudessa, niissä ei ole yhteistä peliteollisuuden standardia. Pelimoottoreita on monia erilaisia, joista pelinkehittäjät joutuvat valitsemaan omansa omien tarpeidensa ja osaamistasonsa mukaan. Isoimpia nimiä pelimoottorimaailmassa ovat Unreal Engine, CryENGINE ja Source Engine. Toisaalta jotkut pelistudiot eivät valitse mitään valmiita pelimoottoreita vaan luovat täysin omansa. (Totten 2012, 20.)

Pelimoottorien erot ovat suuria. Jotkut ovat enemmän keskittyneet skriptaukseen ja toiset ovat enemmän grafiikka -pohjaisia käyttöliittymiä(**art-base interfaces**), jotka näyttävät, miltä peli näyttää samalla, kun peliä ja sen maailmaa kehitetään. Yksi grafiikka -pohjaisista käyttöliittymän omaavista pelimoottoreista on Unity. Se tarjoaa parhaimman kokonaisuuden grafiikkaan keskittyneille (**art-focused**) pelinkehittäjälle. Se antaa käyttäjän viedä suoraan objekti listalta omia tekeleitään kenttään.(Totten 2012, 20.)

5 3D suunnittelun perusteet

5.1 Mallin osat

3D malli voidaan jakaa eri osiin**.** Näitä osia ovat Polygonal Face, Edges ja Vertices. Niitä manipuloimalla, muokkaamalla ja muovaamalla saadaan luotua mallien muodot. Osat ovat yleismaallisia, jolloin suurin osa mallinnusohjelmista tukee niitä. (Castillo, Novak 2008, 170.)

Vertices ovat pisteitä, jossa kohtaavat ainakin kaksi reunaa. Nämä pisteet ovat mallissa helposti muokattavissa ja niillä voi saada suuri muutoksia malliin, muuttamatta itse mallin geometriaa. (Castillo, Novak 2008, 172.)

Edges eli reunat ovat jokaisen polygonin sivuissa. Ne ovat viivoja, jotka ympäröivät jokaista polygonal facea. Jokainen reuna jakaa polygonin eri segmenttiin. Reunoja voi lisätä malliin, jolloin on helpompi muokata sen muotoja. (Castillo, Novak 2008, 171.)

Polygonal Face on geometrian osa mistä malli rakentuu ja pinta mille tekstuuri asetetaan. Tätä osaa kutsutaan joko face tai poly nimellä. Riippuen käytössä olevasta mallinnus ohjelmasta. (Castillo, Novak 2008, 171.)

5.2 Mallinnus

Mallinnuksella tarkoitetaan geometrian luomista, jolla täytetään pelikenttä. Kaikki objektit mitä halutaan kenttään on mallinnettava. Jokainen malli luodaan polygoneista. Pelimoottori joutuu laskemaan jokaisen polygonin, joten on tärkeää, että mallit ovat yksinkertaistettuja. Tätä kutsutaan Low Polygon mallinnukseksi. Polygonit ovat mallissa olevia neliöitä tai kolmioita. Pelimoottori päättää lasketaan mallin tiheys neliöinä vai kolmioina. Mallintamisessa helpoin tapa mallintaa on luoda mallit neliöistä. Jotkin pelimoottorit vaativat mallin osat kolmioina, jolloin malli voidaan myöhemmin muuttaa kolmioiksi tai antaa pelimoottorin muokata malli haluttavaksi. (Castillo, Novak 2008, 168.)

Pelinkehittäjillä on haasteena luodessaan objekteja, että heidän työnsä pitää pystyä renderöimään reaaliajassa. Jokaisen kuvan kohdalla, peliohjelman pitää kyetä laskemaan 3D mallit, valot, tekstuurit ja vuorovaikutukset peliobjektien välillä, jotta ne pystytään näyttämään pelaajan näytöllä. Pelimoottoreissa on niin sanottu polygon budjetti eli suurin mahdollinen määrä polygoneja, mitä voin näyttää kuvaruudulla samaan aikaan. Tämä antaa mallintajille ylärajan mallien kokoihin ja yksityiskohtiin. Hyvä mallintaja osaa luoda tehokkaita malleja, joissa ei ole turhia polygoneja. (Totten 2012, 13-16.)

Low Polygon -mallinnuksessa yritetään saada malli näyttämään hyvältä mahdollisemman pienellä määrällä polygoneja. Low Polygon -mallinnuksella ei tarkoiteta, että mallin ulkonäössä pitäisi tehdä suuria uhrauksia, vaan ideana on, että malli suunnitellaan älykkäästi käyttämään juuri tarvittavat polygonit, eikä yhtään enempää. Tämä mahdollistaa vielä tarkempien mallien luomisen, kun polygoneja vapautuu tarpeettomista

kohdista mallia ja niitä voidaan lisätä kohtiin, joissa tarvitaan enemmän yksityiskohtia. Kentän luomisessa on hyväksi myös ottaa huomioon, mihin polygoneja käyttää. Suuri määrä polygoneja pienissä kivissä tai muissa vähemmän näkyvissä malleissa, vie polygoneja suurimmilta malleilta. Kentissä olisi parempi käyttää enemmän polygoneja isommissa malleissa ja jättää pienemmille malleille yksityiskohtien luonnit tekstuurien ja Normal mapien varaan. Tällöin illuusio kentän yksityiskohdista tulee paremmin esille. (Castillo, Novak 2008, 169-170.)

5.3 Modulaarisuus

Pelikenttien objektit viedään ensin muistiin, jonka jälkeen niitä voidaan vasta käyttää kentässä. Kun objekti on muistissa sitä voidaan käyttää niin monta kertaa, kun halutaan, niin ettei sitä tarvitse ladata uudestaan muistiin. Tämän takia on tärkeää, että objektien mallintamisessa otetaan huomioon modulaarisuus. Modulaarinen kentän rakentaminen on yksi optimaalisimmista tavoista luoda kenttä. Se ei kuormita laitteen prosessoreja tai näytönohjaimia ja niiden muisteja niin paljon kuin, jos luotaisiin kokonaisia objekteja suuri määrä. Jos uniikkeja objekteja luotaisiin hirveästi ja käytettäisiin samassa pelikentässä, se vaatisi paljon enemmän muistia, johon nämä objektit tallennettaisiin. Modulaarisuus tulee tässä vaiheessa tarpeelliseksi, jotta laitteen suorituskyky ei kärsisi. (Castillo, Novak 2008, 123.)

Modulaarinen objekti on objekti, joka voi toimia yhdessä muiden objektien kanssa luoden täysin uuden objektin. Esimerkiksi voidaan luoda joko kymmenen erilaista puu mallia, jolloin pelimoottori joutuu viemään muistiin jokaisen niistä. Jos sama idea tehtäisiin modulaarisina objekteina, luotaisiin puusta erilliset mallit rungosta, lehdistä, oksista ja köynnöksistä. Näin niistä voidaan luoda loputon määrä uniikkeja objekteja ja niistä pitää viedä muistiin vain tarvittavat modulaariset mallit. Tällä tavalla saadaan luotua peleihin realistisempia ympäristöjä, jossa ei kaikki kokonaiset objektit ole toisiensa näköisiä. (Castillo, Novak 2008, 123.)

Modulaaristen kokonaisuuksien rakentamista voi käyttää missä vain pelikenttää. Kunhan objektit ja geometriat ovat luotu modulaarisesti. Silloin pelikentän luojat voivat rakentaa uniikkeja maailmoja valmiina olevista paloista. Modulaarinen rakentaminen

hyödyntää pelimoottorien instanssia eli sitä, että objekti pitää sijaita muistissa, jolloin sitä voi käyttää niin monta kertaa kun haluaa. (Castillo, Novak 2008, 126.)

5.4 Colliderit

Törmäysten havaitseminen pelimoottoreissa on Collider komponentin työ. Luomalla objektille ympärille näkymättömän verkon, annamme objektille Collider komponentin. Tämä komponentti usein jäljittelee objektin muotoa ja sen tehtävä on raportoida kaikista törmäyksistä muiden Colliderien kanssa. On olemassa kahdenlaisia Collider komponentteja, Primitiivisia ja Meshejä. Primitiivinen tarkoittaa 3D terminä, yksinkertaista objektia kuten laatikkoa, palloa tai kapselia. Primitiivistä collideria käytetään usein sen takia, että se on laskennallisesti yksinkertaisempi. Vaikka objekti itse olisi monimutkainen, niin primitiivinen collider silti pysyy esimerkiksi pallona. (Goldstone 2011, 16.)

Mesh collider taas on raskaampi laskennallisesti. Se voi olla aivan objektin muotoinen. Mitä yksityiskohtaisempi ja tarkempi collider on, sitä enemmän se vaatii tietokoneelta tehoja. (Goldstone 2011, 16.)

5.5 Tekstuurit

Tekstuurit ovat kaksiulotteisia kuvia, jotka kääritään 3D objektin ympärille. Nämä tekstuurit voidaan ottaa valokuvista, tehdä itse tai yhdistää kumpaakin. Tekstuurin tehtävä on määritellä objektin pinta, väri tai tekstuuri ja visualisoida kaikki ylimääräiset yksityiskohdat, joita ei olla mallinnettu polygoneilla. (Gahan 2011, 10.)

Pelintekijöille ei ole niin paljon ylellisyyksiä, kun mennään pelin sisällön yksityiskohtiin. He joutuvat työskentelemään paljon pienemmillä polygoni määrillä. Tämän takia tekstuurit toimivat suuressa osassa pelinkehittäjien luodessa objekteja peleihin. (Totten 2012, 88.)

5.6 Mallin UV ja sen purkaminen

Tekstuurin lisäys malliin vaatii mallin UVW:n purkamisen, jotta tekstuuri voidaan asettaa malliin oikein**.** Hyvä tapa tähän on ottaa malli ja purkaa se eri osiin 2D pinnalle. (Castillo, Novak 2008, 181.)

UV purkaminen on prosessi, jossa mallin geometria puretaan osiin ja muunnetaan 2D kuvaksi, johon tekstuuri kartta voidaan lisätä. Prosessissa leikataan malli osiin tietyistä saumoista. UV:n purkaminen on yksi vaikeimmista ja monimutkaisimmista osioista 3D objektin luonnissa. UV:n purkaminen aloitetaan luomalla objektiin saumat. Saumojen luonnissa on hyvä pitää mielessä, että leikatut UV kappaleet ovat järkeviä, jolloin tekstuurien lisääminen niihin helpottuu. Esimerkiksi jos on objektina talo. Hyvä tapa on yrittää jakaa katon mesh talon muista osista, jolloin asioiden kokonaisuus pysyy selkeänä. (Totten 2012, 88-89.)

5.7 Shaders

Shaderit ovat miniohjelmia, jotka prosessoivat graafisia efektejä reaaliajassa. Esimerkki shaderistä on reaaliaikainen heijastus objektin pinnalla tai reaaliaikainen varjostus. Shaderit ovat erittäin voimakkaita visuaalisia efektejä. Usein niiden näkijät eivät tiedä minkä takia peli, jota he pelaavat näyttää niin hyvältä. Korkea yksityiskohtainen taso on usein luotu reaaliaikaisilla heijastuksilla(**reflektion**), Normal mapeilla tai Specular mapin prosessoinnista reaaliajassa. (Ahearn 2012, 81-85.)

Nykyaikaisilla grafiikkaprosessoreilla on kaksi päätyypin shaderiä. Ensimmäinen on Vertex shaders, joka manipuloi verticeja ja niiden attribuutteja reaaliajassa. Toinen on Pixel shader, joka manipuloi renderöityjä pikseleitä reaaliajassa. (Ahearn 2012, 81.)

Reaaliaikainen vertexien tai pikseleiden manipulointi tekee shadereista todella vaikuttavia, mutta samalla ne vievät todella paljon prosessointitehoja. Itse yhden vertexin tai pikselin manipuloimiseen ei paljon tehoa mene, mutta kun peleissä ja kentissä voi olla yhteensä miljoonia pikseleita ja vertexeja. (Ahearn 2012, 82.)

Shaderit tarvitsevat usein 2D tekstuureita tai komponentteja inputiksi. Peleissä on nykyään jo käytössä efektejä, joiden luomisessa ei enää käytetä yhtä tekstuuria vaan efektit ja yksityiskohdat ovat monen shaderin ja tekstuurin summia. Niiden luominen vaati yhä enemmän suunnittelua ja normaalista poikkeavaa ajattelua tekstuureiden luojalta. Nykyajan normaalit shaderit vaativat kuvan, joka on helposti luotavissa kuvan käsittely ohjelmilla. Näitä ovat esimerkiksi, Color map joka myös tunnetaan nimellä Diffuse color. Bump ja Normal maps, Specular maps, Illumination maps ja Opacity maps. (Ahearn 2012, 82-83.)

5.8 Mapit

Diffuse map on kuva, joka sisältää väri tiedon pinnasta. Tekstuurin pitäisi määrittää pinnan päävärit ja yksityiskohdat. Hyvässä Diffuse mapissa ei pitäisi olla ollenkaan kohdistettua valaistusta. Ainoa valaistus mitä tekstuurissa pitäisi olla on ympäröivä valaistus. Tämä tarkoittaa, että tekstuurin syvennykset olisivat tummempia ja kohoamat taas vaaleampia. Tekstuurissa ei saa olla valon aiheuttamia varjoja tai korostuksia.(Gahan 2011, 10.)

Normal map ja Bump map luovat 3D syvyyden litteälle pinnalle**.** Bump map on mustavalkoinen ja luo rajallisen 3D syvyyden. Normal map on värillinen map, johon on valaistuksen informaatio sisälletty. Normal mapissa voidaan laskea valo jokaisella pikselillä, jolloin ei tarvita lisätä ylimääräistä geometriaa, saadakseen luotua varjoja ja korostuksia. Normal mapeilla voidaan pyytää 3D ohjelmaa kohtelemaan jokaista pikseliä, kuten se olisi reagoimassa valoon, jolloin efekti on täysin sama, kun korkeasti mallinnetussa 3D mallissa. (Ahearn 2012, 97-98.)

Tämän päivän Normal mapit antavat meille mahdollisuuden keskittää enemmän polygoneja objektin siluettiin. Negatiivisena puolena Normal mapit eivät pysty itse muuttamaan objektin siluettia. Normal mapin efekti käyttäytyy parhaiten tietyistä kuvakulmista, mutta kun tietyn kulman yli siirrytään, tullaan huomaaman, että kyseessä on vain 2D tehoste, joka päivittyy reaaliajassa. (Ahearn 2012, 99.)

Specular mapissa vaikutetaan mallin tapaan reagoida valoon. Huippuvalo(**specular highlight**) on se kirkas spotti, joka ilmestyy useimmille pinnoille kun valo osuu siihen. Spotin koko riippuu siitä minkälainen materiaali on kyseessä. Specularity map kontrolloi tätä efektiä. Samalla mapilla voi myös määritellä miten eri osat siinä toimivat, joissakin kohdissa mappia voidaan määritellä, että kyseessä on kirkas materiaali, joka toimii kuten peili ja toisessa kohtaa mappia valon vaikutus on taas todella pieni. (Ahearn 2012, 89.)

Specularity mapissa käytetään harmaansävyjä, jotka vaikuttavat materiaaliin tietyin tavoin. Specularity kartta luodaan usein suoraan Diffuse kartasta, muuttamalla se mustavalkoiseksi kuvankäsittelyohjelmilla. (Ahearn 2012, 90.)

Opacity map päättelee onko kuva kiinteä, läpinäkyvä tai jotain niiden väliltä. Opacity mapia käytetään, kun on tarvetta läpinäkyvyydelle, kuten ikkunoille. Myös efekteissä, kuten savussa tai räjähdyksissä käytetään Opacity mappia.(Ahearn 2012, 92.)

Masking eli Alpha mapissa käytetään yhtä määriteltyä väriä, joka tekee siitä väristä läpinäkyvää. Sen käyttö on kevyempää kuin Opacity mapin, mutta se ei näytä niin hyvältä. Alpha mappia on hyvä käyttää skenaarioissa, missä on paljon päällekkäin elementtejä, joissa on läpinäkyvyyttä, esimerkiksi metsissä ja viidakoissa. (Ahearn 2012, 92.)

Illumination mapissa väri paletti on valkoisen ja läpinäkyvän välillä. Valkoinen osa kertoo mitkä alueet mallista pitäisi loistaa itse. Yksi tunnetuimmista tavoista käyttää illumination tekstuuria on valaistu ikkuna pimeässä kaupunkimaisemassa (Totten 2012, 20.)

6 Kentän luomisessa huomioitavat osiot

Jokaisella kentällä on oltava syy olemassa oloon. Syy miksi se on olemassa ja mikä historia sillä on taustalla. Jokaisen ympäristön pitäisi aina kertoa tarina, ilman että pelinkehittäjän pitäisi selittää sitä. Tämä luodaan kentän teemalla ja ympäristön elävöittämisellä. Kentän ympäristön elävöittämisellä, luodaan kentälle persoona. Laittamalla ympäristöön vihjeitä, jotka pelaaja näkee katsoessa ympäriinsä. Ne voivat luoda yhdessä tarinan siitä mitä kentässä on tapahtunut ja mitä ympäristö ei kerro, sen pelaaja itse täyttää mielikuvituksellaan. (Galuzin 2011, 14-16.)

6.1 Kentän visuaalisuus

Pelikentässä on parempi saada muutama asia ja elementti toimiviksi ja näyttämään hyvältä, kun yrittää saada se täyteen kaikkea. Tärkeää on suunnitella kenttä hyvin ja valita teema kentälle, jonka jälkeen keskittyä muutamaan yksinkertaiseen ominaisuuteen, malliin ja tekstuuriin(Galuzin 2011, 27.)

Teema on yhdistävä idea, tiivistys kentän ympäristöstä. Se kertoo tarinaa läpi pelin, kentästä kenttään. Teemassa voi olla tarina elementtejä, visuaalisia tai pelattavuus elementtejä. (Galuzin 2011, 29.)

6.2 Kenttä tyypit

Saari kentän tyylisissä pelikentissä pelaajalle annetaan iso ympäristö tutkittavaksi. Antaen heille täydellisen vapauden liikkua ja tehdä mitä haluavat. Tämän tyylinen kenttä vaatii paljon suunnittelua, mutta usein myös on sen arvoinen. (Galuzin 2011, 92.)

Saaren tapaiset kentät toimivat hyvin moninpeleissä, koska se tukee moni erilaisia pelitapoja. Pelikentän suunnittelussa kannattaa jakaa saaren tapaiset kentät eri alueisiin, joissa on jokin tunnistettava piirre. Tämä helpottaa pelaajien navigointia pelissä.(Rogers 2010, 220.)

Kujan tyyliset pelikentät luovat suunnatun peli kokemuksen. Pelaajalla on määränpää, joka pitää saavuttaa ja kenttä on luotu auttamaan pelaajia saavuttamaan sen. Kujat voi-

vat olla joko kapeita tai sitten laajempia, joissa pelaajille annetaan illuusio vapaudesta ja tilasta. Kuja antaa pelinkehittäjälle monia etuja. Kehittäjän on helpompi luoda alueille kohtia missä jokin asia tapahtuu, kun kehittäjä tietää mistä suunnasta pelaaja saapuu ja mihin suuntaan pelaajan on määrä edetä kentässä. Kentässä on myös mahdollista luoda skripteja, jotka aiheuttavat pelissä tapahtumia, kun tietää mihin suuntaan pelaaja katsoo.(Rogers 2010, 219.)

7 Case: Testi kentän luominen

Case:na tein testi kentän peliprojektiin Terra Rapio. Projektissa on mukana noin 16 ihmistä, joista jokaisella on oma erikoisalansa. Peliprojektissa tärkein tavoite on saada tekninen versio valmiiksi ja saada siitä osaamista ja kokemusta yleisesti peliprojektista. Melkein jokainen peliprojektissa oleva henkilö tekee myös päivätöitä, jolloin projektissa voi olla viivästymisiä tai tekijät voivat joutua pitämään taukoa produktista. Oma roolini projektissa on kentän luoja ja mallintaja eli luon muutaman muun henkilön kanssa peliprojektin graafista ulkoasua.

Testi kenttä on sisäkenttä, jolloin siinä ei tarvitse ottaa huomioon niin paljon tekniikan rajoituksia ja muita laajan ja ison ympäristön vaatimuksia. Case keskittyi pelikentän suunnittelemiseen, luomiseen ja viimeistelyyn graafisella tasolla. Pelikenttä rakennettiin Blenderillä tehdyistä mallinnuksista ja Photoshopilla muokatuista kuvista.

Ensimmäinen vaihe kentän luomisessa on sen suunnittelu. Sain projektilta aiheen, joka on pieni slummi kylä, ison maanalaisen siilon sisällä. Siilon korkeus on 36 metriä ja leveys 18 metriä. Samaan aikaan kun toteutin omaa osaani projektista, tekivät muut projektin jäsenet omia osioitaan projektiin. Heiltä sain valmiiksi tehdyn aloitus tilanteen, joka oli osa isompaa kokonaisuutta. Aloitustilanteessa oli kaksi sylinteriä(Kuvio 1.), toinen niistä oli kenttänä toimivan siilon ulkoseinän Collider ja toinen oli graafinen osa. Lopuksi niihin luodut objektit ja Colliderit yhdistettiin.

(Kuvio 1. Kaksi sylinteriä, joissa normaalit käännetty toisinpäin. Aloitus tilanne.)

7.1 Suunnittelu

Siilo on isompi osa laajempaa kenttää, siinä on yksi sisäänkäynti sen pohjalla ja kaksi sen yläpäässä. Ideana suuremmassa mittakaavassa oli se, että pelaaja ensin saapuu toiselle siilon yläpäässä olevalle sisäänkäynnille, josta näkee koko siilon sisällön. Ikäväkseen pelaaja huomaa, ettei pääse siiloon yläpäässä olevasta sisäänkäynnistä ja lähtee etsimään siilon pohjalla olevaa sisäänkäyntiä. Tämä kohta ei ole sisälletty opinnäytteeseen.

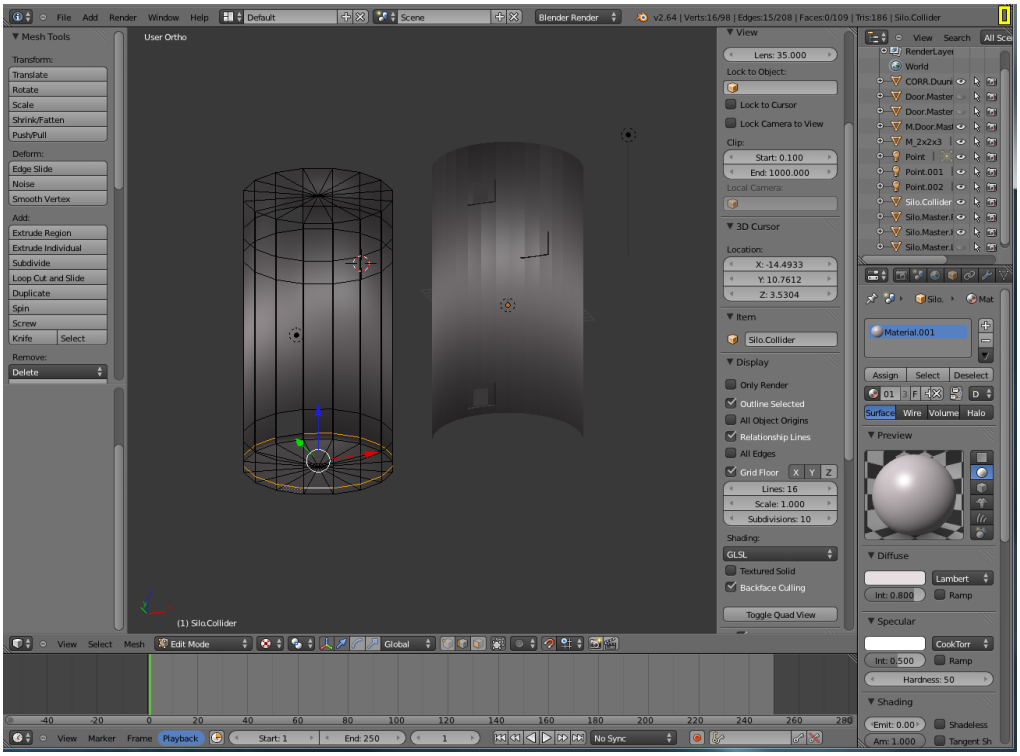
Siilon pohjalta pelaajan on päästävä siilon yläpäässä olevalle toiselle sisäänkäynnille. Siilon sisällä oleva pieni kylä, on pahasti ränsistynyt ja osa siilon seinästä on homeen peitossa. Pelaajalla on oltava myös suora näköyhteys yläpäässä olevalle sisäänkäynnille. Tällöin pelaaja tietää minne on edettävä.

Visualisointi on yksi tärkeä osa kenttä suunnittelua. Ennen kuin aloin tekemään mallinnuksia ja teksturointeja kenttään. Aloitin tekemällä visualisointia eri objekteilla. Visu-

aloin kentän suurimmat objektit ja kohteet, jolloin oli helpompi pohtia miten pelaaja siinä liikkuisi ja miten kenttä toimisi.

Visualisoinnissa usein kannattaisi piirtää alustavia piirroksia kentästä ja kentän objektien siluetteja. Itse jätin nämä tekemättä ajan puutteen takia. Visualisoinnin tein Blenderillä. Loin siinä nelikulmioista ja sylintereistä osia, joista rakensin ensimmäisen visualisoinnin kentästä. Visualisoinnin toteuttamisessa Blenderissä en tajunnut ottaa huomioon Collidereiden luomista samalla. Vaan loin erikokoisia, isompia mallinnuksia seiniksi ja lattioiksi. Colliderit olisi hyvä olla samankokoisia kuin siihen myöhemmin liitettävä objekti. Tietenkin tähän vaikuttaa myös Collidereitten tarkoitus objektissa, joissakin objekteissa voidaan käyttää myös isompia Collidereita, jos sille on pelattavuuden kannalta hyvä tarkoitus.

Visualisoinnissa toin Blenderiin valmiin tiedoston(Kuvio 2.), missä sylinterit sijaitsivat. Siihen aloitin mallien luomisen. Käytin sylintereitä ja kuutioita, joita muokkasin tarpeen mukaan seiniksi, lattioiksi ja portaiksi.

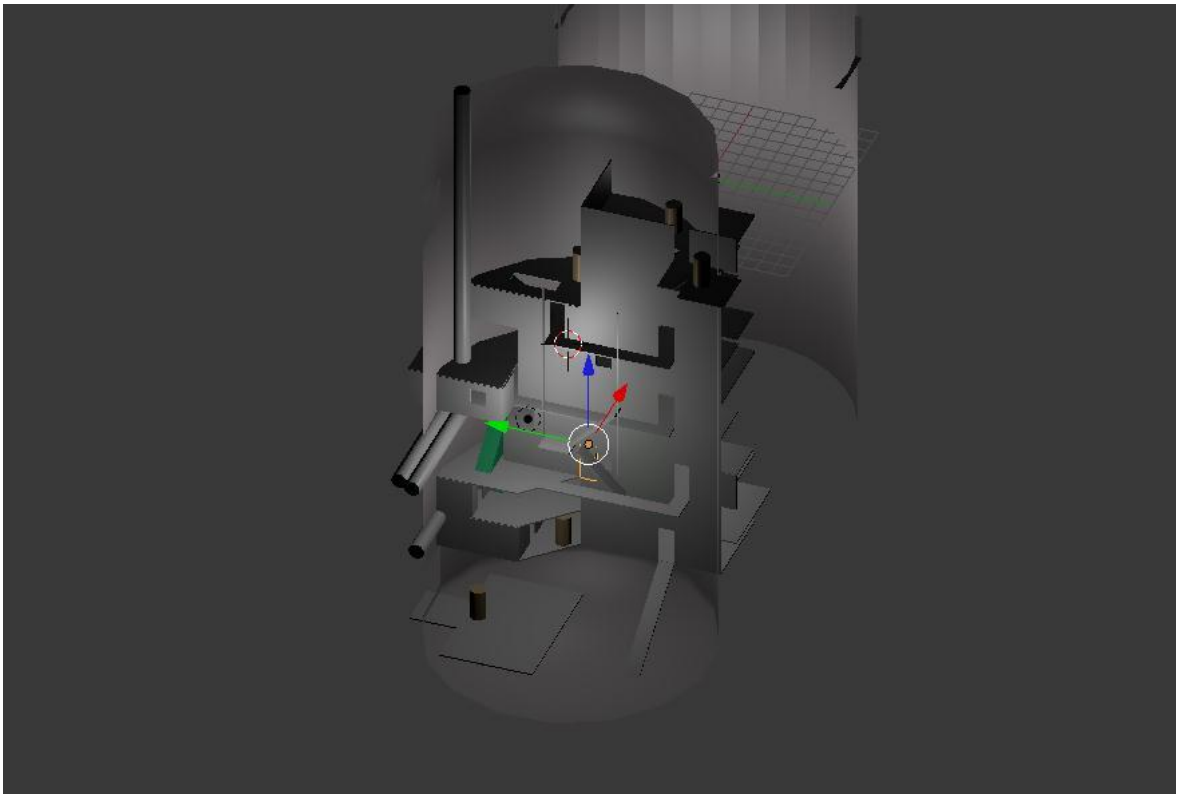


(Kuvio 2. Valmiit sylinterit.)

Kun perusobjekti on luotu, sitä voidaan muokata liikuttelemalla sen osia. Blenderissä voi valita mitä osaa objektista haluaa muokata. Näistä vaihtoehtoina ovat vertex eli pisteet, edges eli reunat ja faces eli tasot. Suurimman osan muokkaamisesta tein tasoja muokkaamalla. Pisteiden tai reunojen muokkaus olisi ollut liian yksityiskohtaista ja aikaa vievää. Uusien pisteiden luominen ja niiden muokkaus on pääosassa, jos luodaan yksityiskohtaisia malleja, mutta loin vain visualisointiin tarkoitettuja objekteja, jolloin yksityiskohdilla ei ollut merkitystä. Vaan tärkeämpää oli visualisoinnissa saada kokonaiskuva pelikentästä.

Blenderissä on oma yksikkö, jota käytetään mallien kokojen hallinnoimisessa. Kun työssäni on kyse pelikentästä ja siihen eri objektien ja mallien luomisesta. Tärkeää on varmistaa, että oma Blender yksikkö on sama kuin muilla projektissa olevilla. Tämä sen takia, jos tulevaisuudessa projektiin liittyy muita mallintajia, niin objektit ovat oikeassa koossa ja niiden siirtely toiseen Blender -sovellukseen tapahtuu ongelmitta.

Lopputuloksena visualisoinnissa(Kuvio 3.) sain suuripiirteisen hahmotelman kentän rakenteesta ja pientä ajatusta minkälaisia objekteja sinne voisin luoda. Ruskeat sylinteri kuvastivat pelaajaa, syynä tähän oli se, että en voinut vielä testata pelikenttää tai sen toimivuutta. Joten jouduin käyttämään sylintereitä apuna, jotta pelikentän osien koot pysyivät oikean kokoisina ja pelaaja pystyi liikkumaan kentässä oikein.



(Kuvio 3. Suunnittelun lopputulos.)

Tarvittavien objektien määrää jouduin pohtimaan ennen kuin aloin luomaan objekteja. Liian monet ja yksityiskohtaiset mallit voivat vaatia liian paljon laskentatehoja tietokoneelta tai laitteelta, jolla peliä pelataan. Tämä voi aiheuttaa pelin hidastumisen. Pieni määrä yksinkertaisia malleja voi luoda kentästä liian epärealistisen ja tuhota pelin tunnelman. Objektien määräksi suunnittelin 5-7 eri seinä mallia, 2-3 eri lattia mallia ja portaat. Materiaaleja eli tekstuureita seinille 3-4 ja lattialle 2-3. Kentän teemana oli slummimainen kylä, joten pidin tärkeänä, että sen ulkonäkö kuvastaa myös slummia, jossa seinät ja lattiat on luotu halvoista saatavista materiaaleista, kuten pahvista, pellistä ja puusta. Objekteissa koin tärkeämmäksi hyvälaatuiset tekstuurit, kuin itse mallin yksityiskohdat. Myöhemmin jos on tarvetta pystyn luomaan ja korjaamaan jo tehtyjä malleja yksityiskohtaisimmiksi. Oppimisen takia tärkeämpi oli saada luoda mallit nopeasti ja opetella eri tekstuureiden ja mappien käyttö malleissa.

Suunnittelun lopuksi tutkin mahdollisuuksia käyttää valmiita tai melkein valmiita tekstuureita työssäni, jolloin työn luominen nopeutuisi. Internetissä olisi löytynyt paljon käyttökelpoisia tai melkein käyttökelpoisia tekstuureita, joita olisin voinut Photoshopilla muokata halutuiksi. Tekstuureiden etsimisessä ja muokkaamisessa menisi oma aikansa,

mutta ei niin paljon kuin kokonaan omien tekstuurien luomisessa alusta asti. Valmiiden tekstuureiden käyttämisessä pitäisi ottaa huomioon, että saako tekstuuria käyttää vai tulisiko siinä oikeuksien kanssa ongelmia. Omien tekstuurien luominen alusta asti, olisi antanut mahdollisuuden luoda juuri haluttu tekstuuri sen teeman mukaisesti, mikä kenttään halutaan. Tekstuurien luominen veisi paljon enemmän aikaa kuin valmiiden tekstuurien käyttö tai valmiiden muokkaaminen halutuiksi kokonaisuuksiksi. Myös omat taitoni eivät ehkä ole niin hyvä, että pystyisin luomaan aivan niin tarkkoja tekstuureita, joita haluaisin käyttää malleissa. Tämän ja aika rajoitusten takia tämä vaihtoehto olisi haastavampi kuin muut vaihtoehdot, joten päädyin käyttämään melkein valmiita tekstuureita, joita myöhemmin muokkaisin halutuksi.

Kun olin saanut suunnittelun tehtyä, aloin luomaan objekteja, joista rakensin pelikentän. Objektien luomista varten olin hakenut Internetistä ilmaisen Blender -nimisen mallinnus sovelluksen.

7.2 Objektien luominen

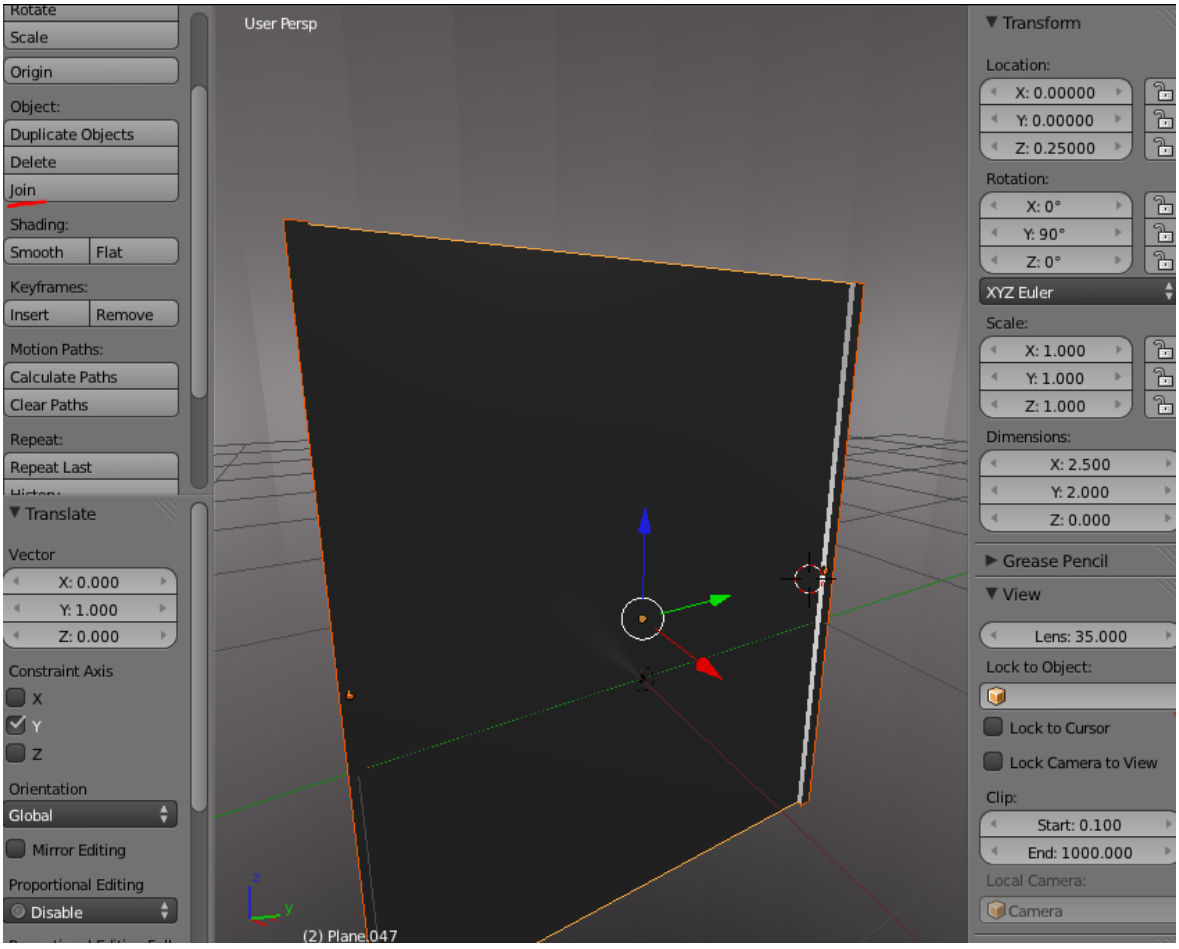
Objektien luomisen aloitin seinien malleista. Seinien mallien koissa piti ottaa huomioon pelaajan koko ja kerroksen koko. Pelikenttä koostui monesta kerroksesta, jolloin myös pelaajan oli mahduttava liikkumaan niissä ja myös kokea ettei pelaajan hahmon pään kohdalla oleva kamera kokoajan liiku katossa kiinni. Objektien koissa käytin Blenderin omaa koko yksikköä. Pelaaja on kaksi metriä korkea ja kerrokset ovat kaksi ja puoli metriä korkeita.

Modulaarisuus on tärkeä osa objektien luomista. Kenttää luodessa on tärkeää, että pystyn käyttämään samoja objekteja ja luomaan niistä kokonaisuuksia, jotka näyttävät hyvältä ja toimivilta. Modulaarisuuden otin objekteissa huomioon. Luomalla metrin ja kahden metrin versiot seinistä. Pystyin luomaan tarvittavat palikat, joilla kykenen luomaan minkä tahansa kokoisia rakennuksia.

Seinissä olevia ikkunoita varten luodaan joko oma isompi objekti, jota käytetään yksin muiden modulaaristen osien kanssa tai luodaan pienempiä osia, joista myöhemmin

luodaan seiniä, joissa on ikkunalle paikka. Päätin tehdä pienempiä osia, joita pystyn käyttämään myöhemmin uudestaan rakentaessa isompia kokonaisuuksia.

Seinäpaloja päätin luoda kolmesta eri materiaalista. Aaltopellistä, pahvista ja puusta. Niistä jokaisesta loin pari eri versiota, joita kykenin käyttämään eri kohdissa kenttää. Peltiseinän luomisen aloitin luomalla yhden 2.5x0.1x0.1 kokoisen palkin. Tämän jälkeen loin plane objektin. Luomani planen koko oli 2.5x2. Tästä palasta tuli seinä ja palkista seinän reuna. Koska plane on vain yksipuolinen, eli siitä näkee läpi toiselta puolelta, niin jouduin luomaan siitä duplikaatin ja kääntää sen näkymään toiseen suuntaan. Tällöin minulle jäi kaksipuolinen plane. Tein palkista duplikaatin. Asetin nämä palkit seinän kummallekin reunalle ja yhdistin valitut mallit yhdeksi objektiksi(Kuvio 4.).

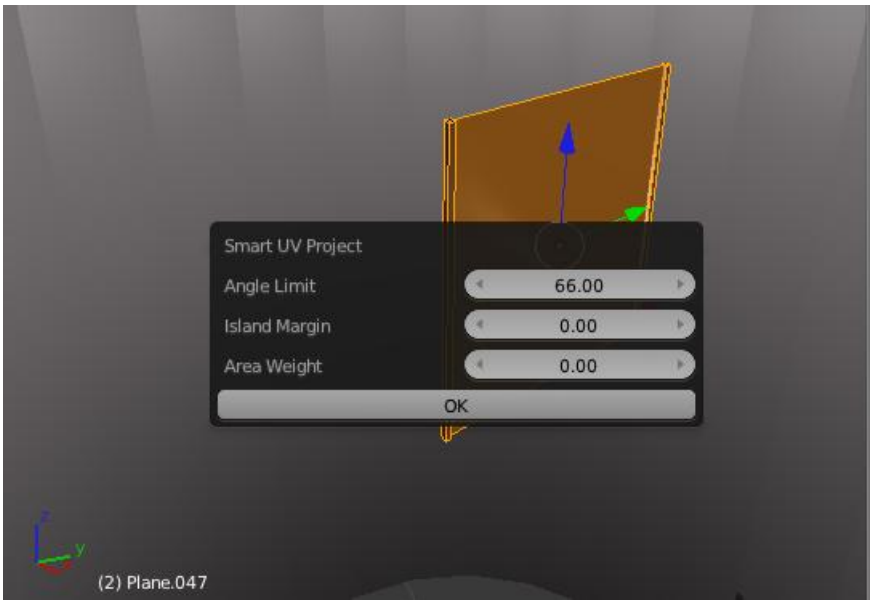


(Kuvio 4. Objektien muutto yhdeksi objektiksi.)

UV map generoidaan objekteista, jotta niihin on helpompi luoda räätälöityjä tekstuureita. Malleihin voidaan myös lisätä tekstuureita suoraan, ilman UV mapin generoimis-

ta, mutta silloin tulos ei usein vastaa sitä mitä halutaan. Varsinkin jos on kyseessä yksityiskohtainen ja tarkka tekstuuri monimutkaiselle mallille. UV mapien käyttö on tarpeellista myös pelien näkökulmasta, jossa mallit ja tekstuurit yritetään luoda pieni kokoisiksi, jotta pelit eivät veisi niin paljon tilaan kovalevyiltä tai muistista.

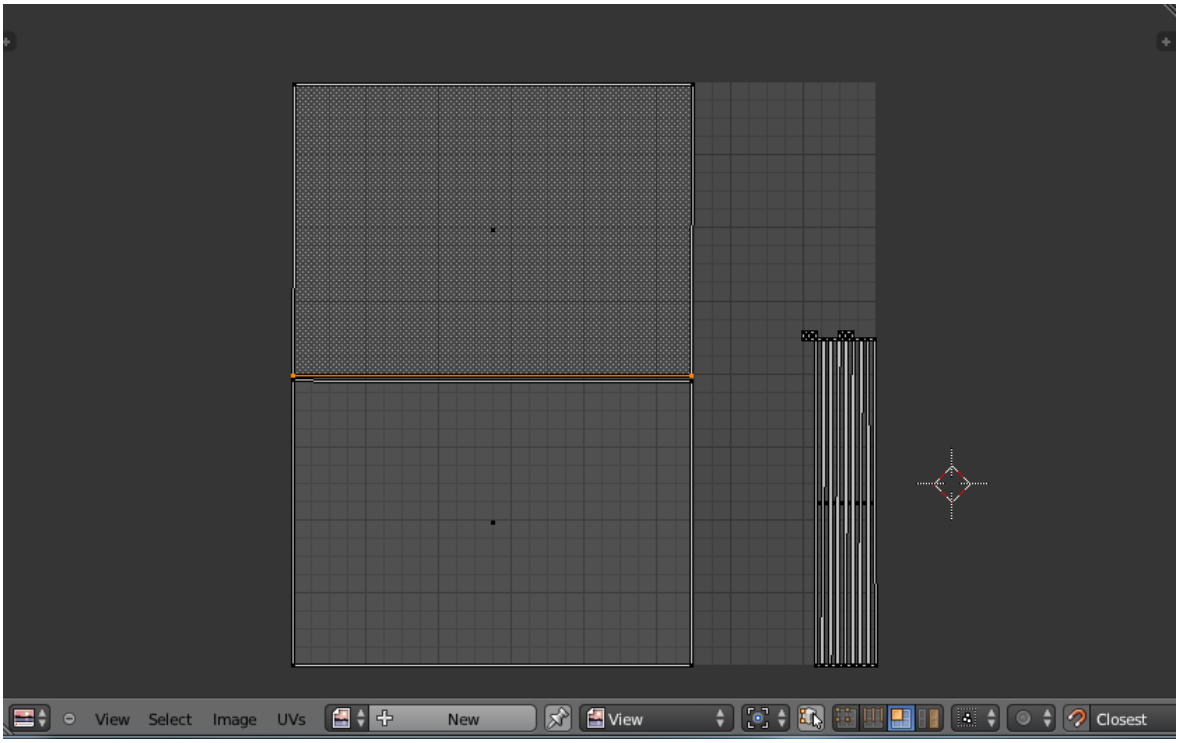
Blenderissa UV mapin generoiminen onnistui sovelluksen sisään rakennetulla Smart UV Project ominaisuudella. Haluttu malli piti valita ja käyttää työkalua, jolloin Blender antoi valikon missä pystyi muokkaamaan automaattisesti generoituvan mallin arvoja.(Kuvio 5.)



(Kuvio 5. Smart UV Project.)

Kun olin valinnut halutut säädöt. Blender generoi haluttujen arvojen mukaan UV mapin valitusta mallista. Editoriin syntyneet objektin sivujen osat, ovat muokattavissa. Editorissa kannattaa jo suunnitella tekstuurin käyttö ja mappaus. Smart UV Projectilla luodut UV mapit voivat olla välillä outoja tai ne jättävät paljon käyttämätöntä pinta-alaa tekstuuriin. Tämän takia niitä voidaan myös muokata käsin.

Muokkasin planen vertexeja haluamaksi kokonaisuudeksi (Kuvio 6.). Laitoin planen sivujen pinnat vierekkäin vaakasuoraan, jolloin planen pinnat voidaan mapata suuremmiksi. Tämä mahdollistaa tarkemman tekstuurin mikä tulee näkyviin objektiin. Yritin myös pitää planessa oikean suhteen, jottei tekstuuri näytä hassulta siinä.



(Kuvio 6. Muokattu UV map.)

UV mapien luomisen jälkeen, generoin niistä Blenderillä kuvatiedoston, jonka siirsin Photoshoppiin. Kuva tiedoston päälle loin tarkat tekstuurit. Tekstuurit piti muokata halutuiksi ennen tekstuurien lisäämistä UV map kuvatiedostoon.

Tekstuurien muokkauksen tein Photoshopilla. Kuvat jotka muokkasin halutuiksi tekstuureiksi, otin nettisivulta, www.cqtextures.com. Pelikentän graafinen ulkonäön halusin olla slummimainen. Tämän takia valitsin aaltoilevan peltiseinän, pahvin ja lauto- ja tekstuureiksi.

Kuvan muuttaminen tekstuuriksi, jota voidaan käyttää saumattomasti vierekkäin, vaati hyvin vähän työtä. Alkuperäinen kuva(Kuvio 7.) pitää jakaa 4 osaan ja tämän jälkeen siirtää vasemmalla alareunassa oleva osa oikeaan yläreunaan ja oikeassa yläreunassa oleva osa edellisen osan paikalle ja samanlailla tehdä myös oikean alareunan ja vasemman yläreunan osille. Tällä tavoin jo yhteen sopivat osat jäävät uuden tekstuurin ulkoreunoille ja minulle jäi vain keskikohdan muokkaaminen yhteen sopivaksi.

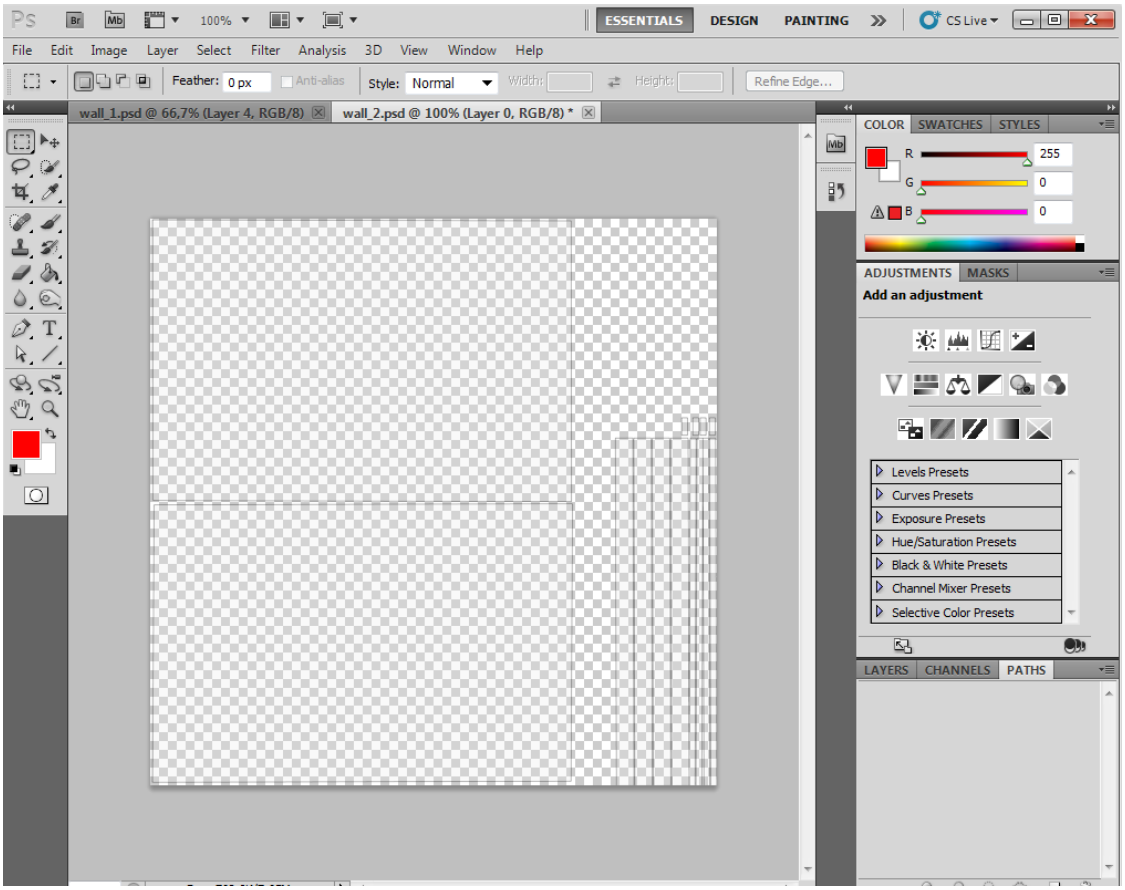
(Kuvio 7. Alkuperäinen kuva.)

Muokkauksen tein Photoshopin healing tool työkalulla. Lopputulokseksi (Kuvio 8.) syntyi tekstuuri, jota voidaan käyttää saumattomasti. Saumaton tekstuuri objekteissa helpottaa objektien käyttöä pelikentässä. Samanlaisia objekteja voidaan asettaa vierekkäin ja silti ylläpitää illuusiota, että kyseessä on vain yksi suuri objekti.

(Kuvio 8. Esimerkki tuloksesta.)

Työnlaatu riippui täysin siihen pistetystä ajasta. Itse tein tämän asian todella nopeasti ja tekstuuria joudun varmasti korjailemaan myöhemmin. Onneksi kyseessä oli rähjäinen, ruostunut ja puolimätä peltiseinä, jolloin tekstuurin ei tarvinnut olla kovin hääppöinen. Kun olin muokannut tarpeelliset tekstuurit sopiviksi. Pystyin aloittamaan itse Diffuse mapin luomisen eli tekstuurin, jossa näkyisi vain väri arvot.

Diffuse mapia varten tarvitsin Blenderissä aikaisemmin luodun UV mapin(Kuvio 9.), johon lisäsin halutut tekstuurit. Aikaisemmin luodun UV mapin ansiosta, näin suoraan mitä tekstuureita minun piti laittaa Diffuse mapin eri osiin.



(Kuvio 9. UV map Photoshopissa.)

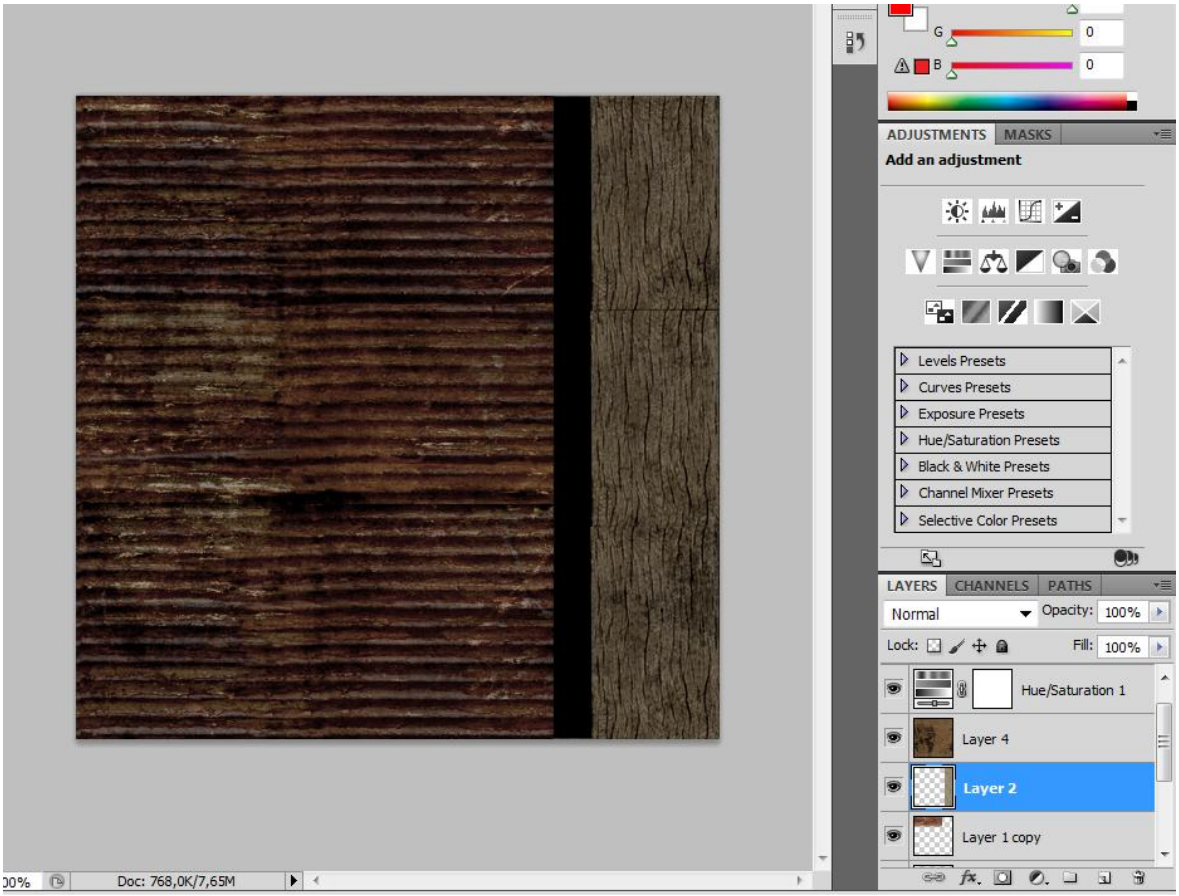
Diffuse mapia luodessa yritin käyttää Photoshopin layereita. Layerit mahdollistavat helpon muokkaamisen jälkikäteen, jos tulee tarvetta muuttaa jotakin osaa Diffuse mapista. Perusideana mapien taustaksi loin mustan layerin, joka kuvastaa käyttämätöntä osaa mapista. Sen päälle loin layerit eri tekstuureille. Kaikki tekstuurit voitaisiin myös pistää samalle layerille, mutta silloin mapin muokkaaminen olisi hankalampaa. Monia layereita käyttämällä sain mahdollisuuden muokata ja lisätä ominaisuuksia, jotka vaikuttavat vain yhteen layeriin eli yhteen tekstuuriin.

Photoshop tarjoaa melkein rajattomasti muokkaus mahdollisuuksia tekstuureille. Esimerkiksi halusin saada tekstuureistani likaisempi ja kuluneempia. Tätä varten hain toisen tekstuurin Internetistä. Tekstuurin piti sisältää likaa tai tahroja (Kuvio 10.), jotka näyttäisivät hyvältä omissa tekstuureissani.

(Kuvio 10. Likainen tekstuuri.)

Lisäsin tekstuurin(Kuvio 10.) Diffuse mappiin tekemällä uuden layerin, johon lisäsin likaisen tekstuurin ja muutin sen normal arvon Linear Burn arvoksi, jolloin Likaisen tekstuurin upotin Diffuse mappiin, muuttamalla layerin sekoitus asetusta. Eli sitä miten layerissa oleva kuva sekottuu sen alla olevan layerin kuvaan. Muuttamalla sen Linear Burn asetukseksi. Tällöin kuvassa tummat kohdat tulevat paremmin esille ja heijastaa sekoitettua väriä. Viimeiseksi laskin layerin täytön 40%, jolloin sekoituksen aiheuttama lopputulos ei olisi niin vahva. Tuloksena sain hiukan haluttua vahvemman efektin Diffuse mapiin(Kuvio 11.). Saatu tulos oli hiukan liian vahva omaan tarpeeseeni, joten muutin Diffuse mapista vielä Hue/Saturation arvoja. Saturation arvon muutin -50:ksi, jolloin sain kuvan hiukan neutraalimmaksi. Tämän jälkeen tyydyin lopputulokseen(Kuvio 12.).

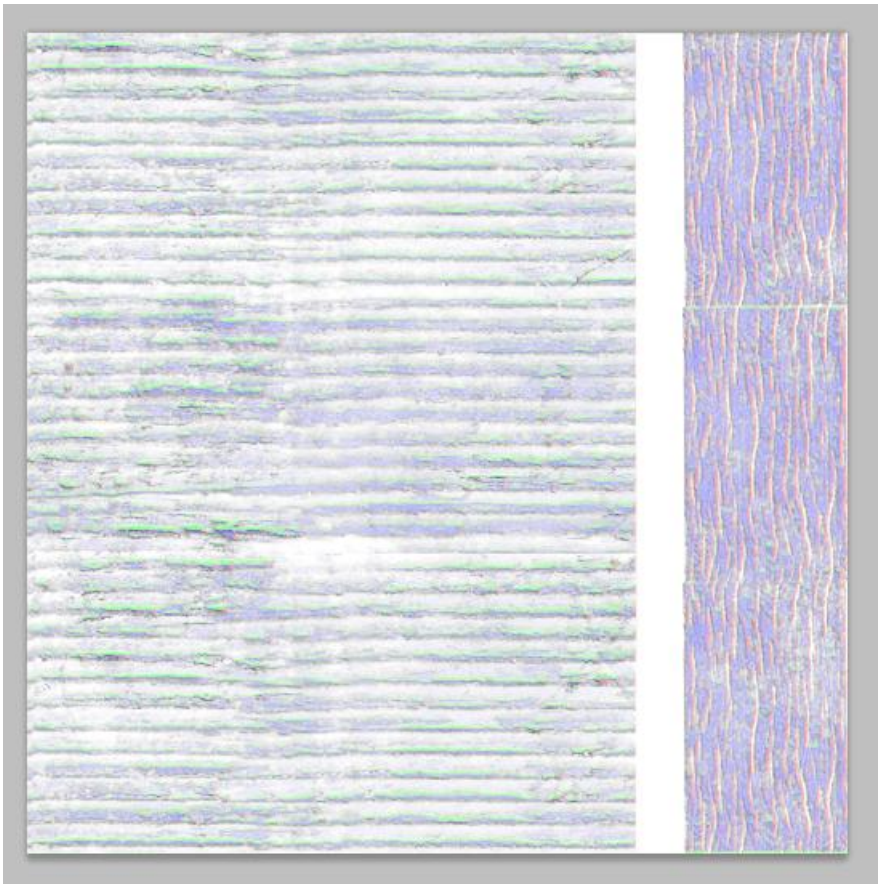
(Kuvio 11. Likainen Diffuse Map.)



(Kuvio 12. Valmis Diffuse map.)

Diffuse mapin valmistuttua, tein Normal mapin. Normal mapin luomiseen on tehty Photoshopiin lisäosa. Hain sen Nvidian nettisivuilta.
https://developer.nvidia.com/nvidia-texture-tools-adobe-photoshop

Normal mapin luominen onnistui helposti, kun työkalu oli asennettu. Kykenin sillä luomaan normaalista Diffuse mapista Normal map(Kuvio 13.). Minulla oli Diffuse mapin osat eri layereilla, joten jouduin ne yhdistämään. Paras tapa tähän oli kun otin kopiot alkuperäisistä layereista ja tämän jälkeen yhdistin tekeleen yhdeksi kokonaiseksi Diffuse mapiksi. Viimeiseksi tarvitsin vielä Specular mapin, jottei ruostunut pelti loista valaistuksessa oudosti.



(Kuvio 13. Normal Map.)

Specular map on mustavalkoinen kuva(Kuvio 14.). Sen saa helpoiten luotua, kun ottaa Diffuse mapin ja tekee siitä mustavalkoisen. Yksi keino tähän on käyttää Photosho-

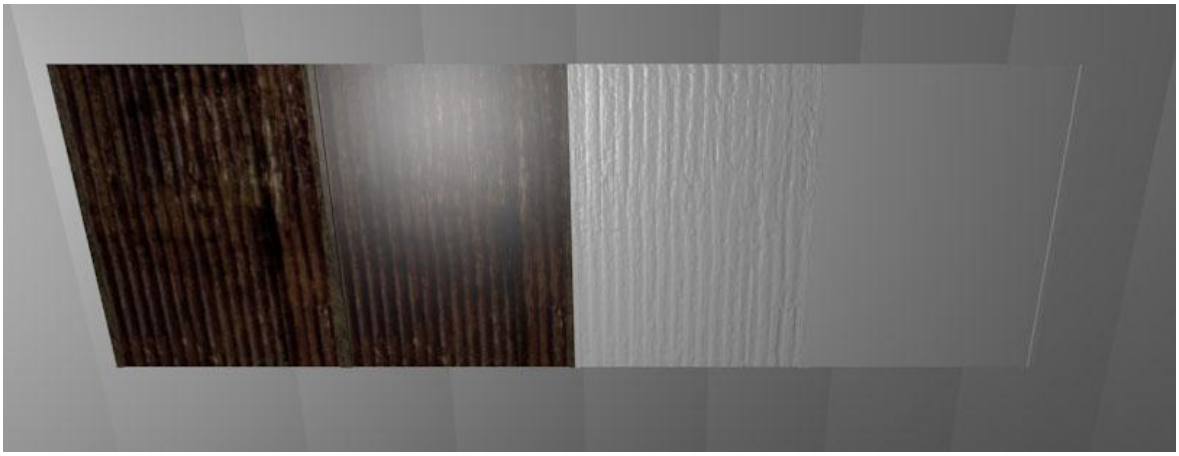
pin Black and White työkalua. Kun Specular map oli valmis, oli aika pistää tekstuurit aikaisemmin luotuun seinämalliin.

(Kuvio 14. Specular Map.)

7.3 Objektien viimeistely ja kentän luonti

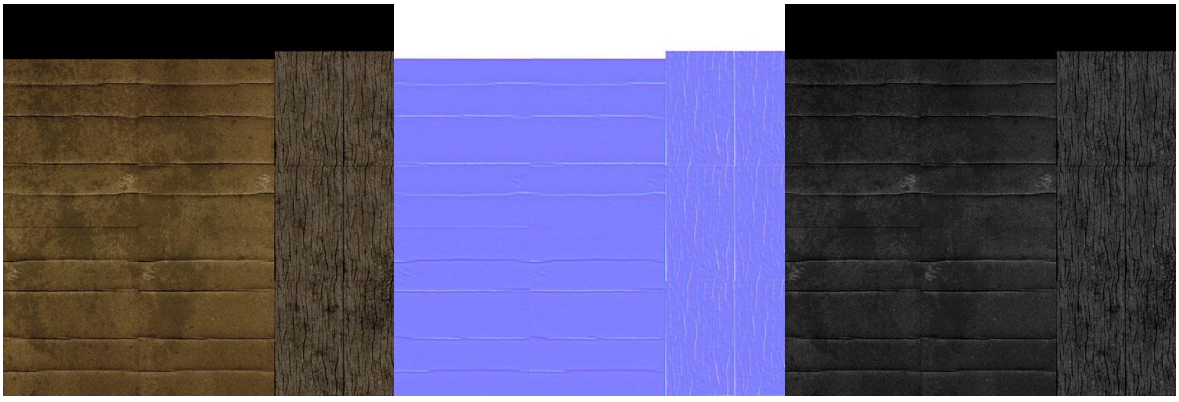
Aikaisemmin olin luonut mallit ja generoinut niihin UV mapit Blenderissä ja tehnyt niihin eri mapit Photoshopissa. Tämän jälkeen oli aika yhdistää kaikki tekeleet valmiiksi teksturoiduiksi objekteiksi. Ensimmäinen askel oli luoda Blenderissä materiaali, valitsin halutun mallin ja loin sille oman materiaalin. Materiaaliin lisäsin sen jälkeen malliin tarkoitetut mapit. Materiaali oli tämän jälkeen valmis. Blenderissä materiaaleja voidaan käyttää useissa eri malleissa.

Ennen kuin Blender kykeni käyttämään tehtyjä materiaaleja, siinä piti muuttaa Mapping asetuksen Coordinate kohta UVksi, jolloin malli osasi käyttää siihen generoituja UV mapeja tekstuurien käytössä. Tämän jälkeen malli tekstuureineen(Kuvio 15.) oli valmis.

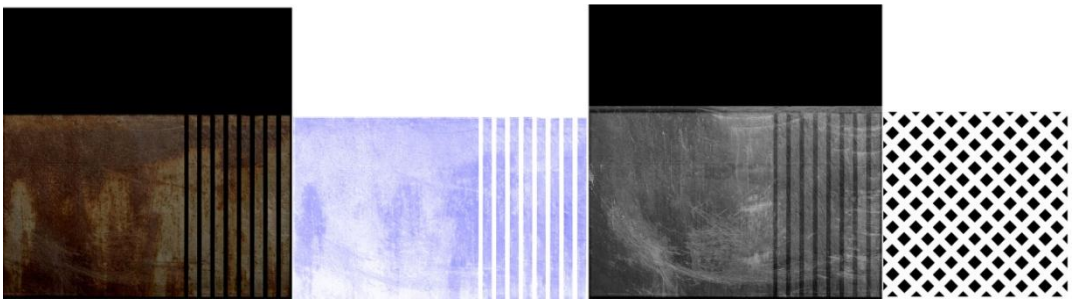


(Kuvio 15. 1.malli: Diffuse, Normal ja Specular map, 2. malli: Diffuse map, 3. malli: Normal map, 4. malli: ei textuureita.)

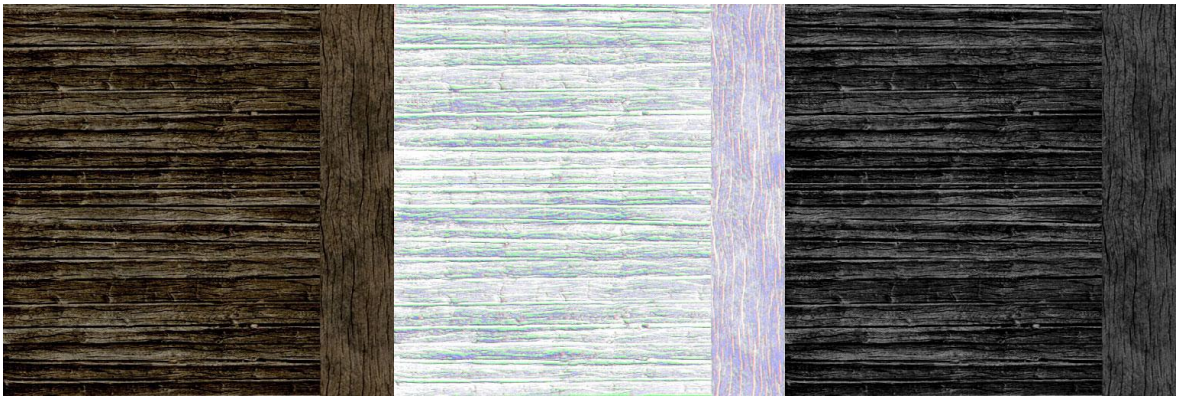
Kenttää varten halusin luoda vielä erilaisia seinä malleja. Tein lopulta yhdet tekstuurit pahviseinää(Kuvio 16.) varten, yhdet tekstuurit lautaseinää(Kuvio 18.) varten ja lopulta metalliset tekstuurit metallisia ritilöitä(Kuvio 17.) varten. Lattiaa varten loin vielä kaksi erilaista tekstuuria. Käytin aikaisemmin mainittuja tapoja luodessani pelikentän muut osat.



(Kuvio 16. Pahviseinän tekstuurit, 1. Diffuse map, 2. Normal map. 3. Specular map.)

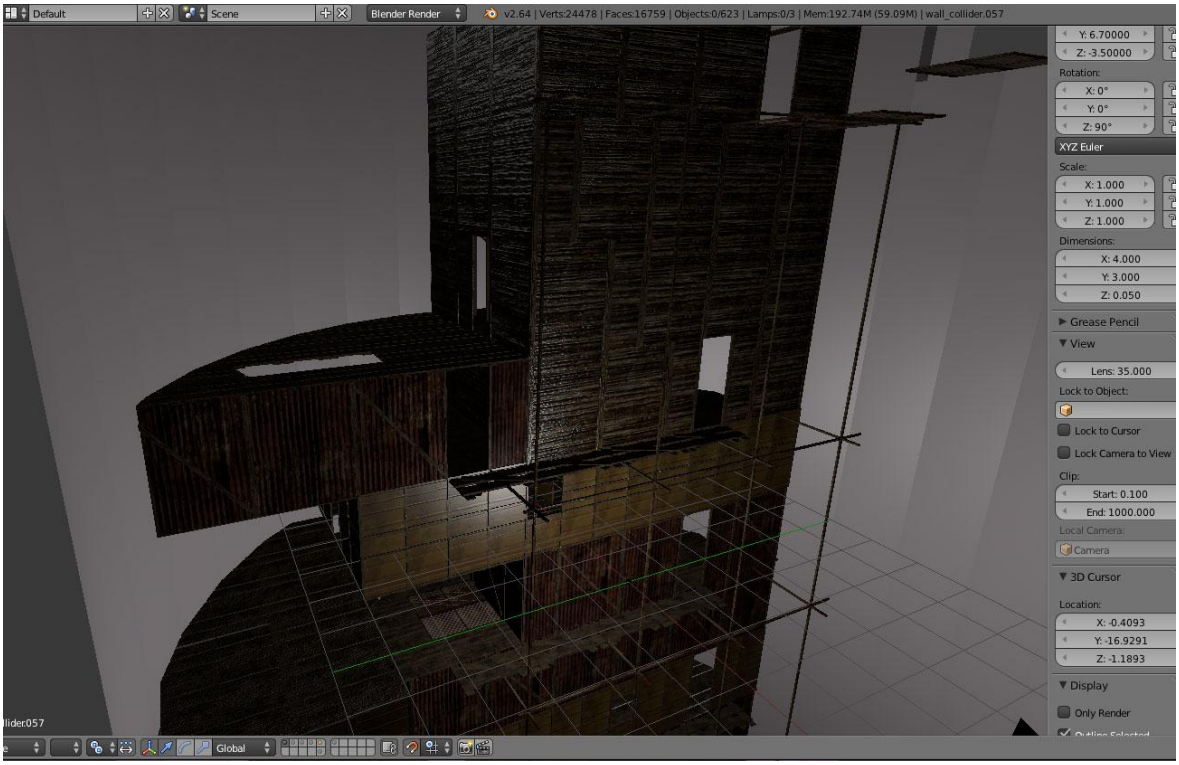


(Kuvio 17. Ritilän tekstuurit, 1. Diffuse map, 2. Normal map, 3. Specular map, 4. Alpha map.)

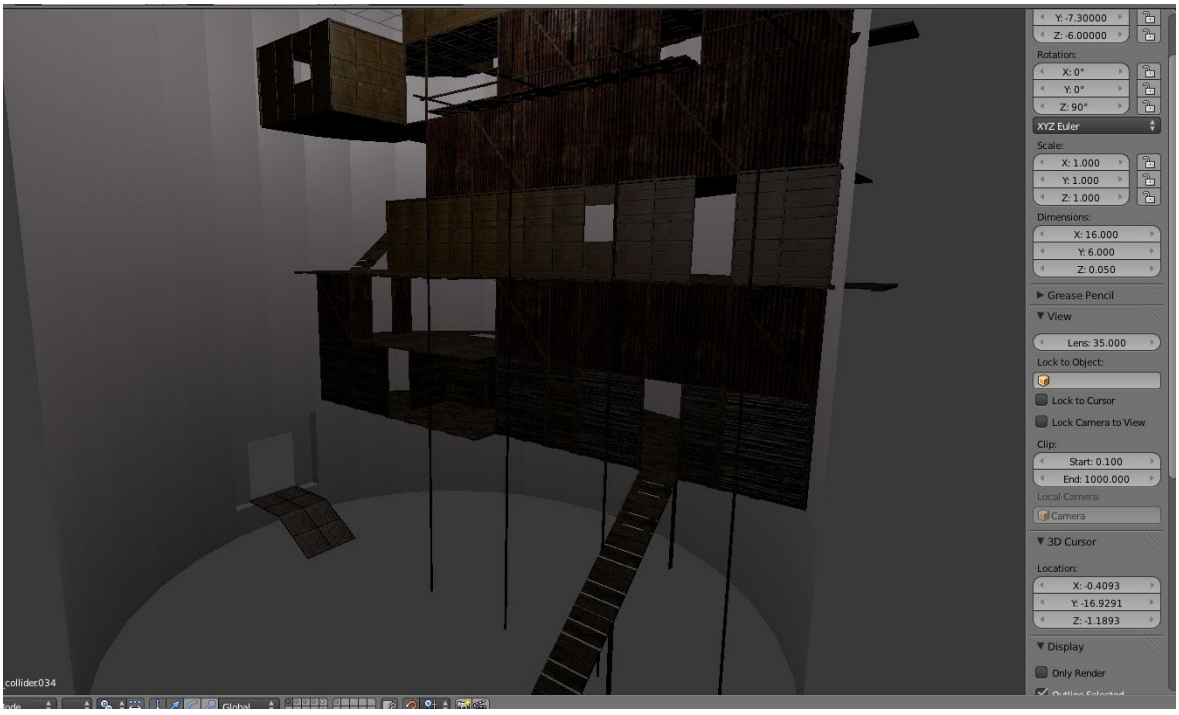


(Kuvio 18. Puuseinän tekstuurit, 1. Diffuse map, 2. Normal map, 3. Specular map.)

Kenttää luodessani, jouduin muuttamaan suunnittelussa tekemiäni päätöksiä. Jouduin jättämään paljon yksityiskohtia pois. Pelikentän raamina toimineen sylinterin seinämille suunnitellut putket jäivät tekemättä. Pelikentän loin yhden ja kahden metrin kokoisista seinä paloista ja parista eri lattiapalasta. Kentän kokoaminen käyttäen osittain alkuperäistä suunnitelmaa kävi nopeasti, jonka jälkeen graafinen osuus pelikentästä yläosiosta(Kuvio 19.) ja alaosiosta(Kuvio 20.) olivat valmiit. Lopuksi vielä loin objekteille collideriit, jotka olivat mallien kokoisia ja yhdistin ne yhdeksi kokonaiseksi collideriksi, jotta sen siirto Unity3D sovellukseen olisi helppoa. Tämän jälkeen kenttä oli valmis siirrettäväksi Unity3D sovellukseen.



(Kuvio 19. Lopputulos yläkerta.)



(Kuvio 20. Lopputulos alakerta.)

8 Pohdinta

Tämän opinnäytetyön tekoon meni noin puoli vuotta, siihen kuului myös tekniikan opetteleminen, sovellusten oppiminen ja peligrafiikka teorian oppiminen. Opinnäytetyö antoi minulle hyvän osaamisen Blender sovelluksesta, jolla loin 3D mallit ja Photoshop sovelluksesta, jolla muokkasin ja loin tekstuurit. Tutustuin myös Unity3D sovellukseen, vaikka jouduin myöhemmin rajaamaan tämän osion pois opinnäytetyöstäni.

Alkuperäisessä suunnitelmassa olisin halunnut myös viedä kentän grafiikan Unity3D sovellukseen ja luoda siihen valaistukset ja partikkeli efektit. Opinnäytteen edetessä, jouduin rajaamaan suunnitelmaa, oman osaamiseni ja aikani puutteen takia. Jouduin tämän takia myös viitekehystä muokkaamaan ja poistamaan sieltä muutamia kohtia jotka olisivat vasta olleet mukana Unityssä tehtävässä työssä. Olisi ollut mielenkiintoista vielä jatkaa opinnäytetyötä. Peliprojektista mihin objektit tein tullaan tekemään ainakin tekninen demo. Mallinnukset ja niiden teksturoinnit, mitkä loin opinnäytetyön aikana. Tulevat olemaan osana isompaa kenttä kokonaisuutta ja niitä käytetään peliprojektissa ensimmäisen testi version teossa.

Opinnäytetyötä kirjoittaessa huomasin, että pelinteossa käytetyt termit olivat usein englanninkielisiä, eikä niihin ole kunnollisia suomennoksia. Yritin viitekehystä kirjoittaessa ottaa huomioon asian ja aukaista termistöä.

Toteutusta tehdessäni huomasin, ettei aika oikein riittänyt ja työmäärää oli enemmän kun olin odottanut. Suunnittelun jälkeen tajusin, että olisin voinut myös luoda suunnittelussa objektin Colliderit, jotta niitä ei olisi tarvinnut myöhemmin luoda uusiksi. Tämä lisäsi työ määrää jonkin verran. Objekteja luodessani jouduin karsimaan ylimääräisiä yksityiskohtia, mitä olisin halunnut luoda. Esimerkiksi seiniä tehdessäni olisin voinut luoda niille paksummat Colliderit ja käyttää teksturoituja planeja ja muita pieniä objekteja, täydentämään ja luomaan illuusiota syvällisemmästä ja yksityiskohtaisemmasta seinästä. Colliderin paksuus taas olisi auttanut siihen, ettei pelaaja olisi päässyt liian lähelle seinää, jolloin illuusio olisi pysynyt.

Pelikenttää ei ehditty opinnäytetyöprosessin aikana testaamaan. Kenttä vietiin onnistuneesti Unity3D sovellukseen, mutta itse pelin ohjelmointia ja testi hahmon luomista ei tehty. Graafinen puoli opinnäytetyöstä onnistui projektin jäsenten mukaan hyvin. Tulevaisuudessa sitä voidaan vielä parantaa, kun projektissa saamme muut osuudet tarpeeksi pitkälle, jolloin voimme testata myös pelikentän kuormitusta testikoneella ja kuinka paljon enemmän yksityiskohtia voimme luoda pelikenttään ennen kuin testikoneella tulee ongelmia laskentatehojen kanssa.

Opinnäytetyön aihe oli laaja ja sen takia myös siitä jätettiin paljon syvällisempää tietoa pois. Opinnäytetyötä voisi jatkaa myöhemmin vaikka tutkimalla Tesselaatiota tai muita uudempia Sharedeita. Myös itse valmiin pelikentän toimivuutta voisi tutkia. Miten pelaaja liikkuu kentässä ja miten kentän rytmitys toimii eli kentän hiljaiset osuudet ja niiden välissä olevat nopeat toimintaa vaativat osuudet, joita nykyajan peleissä käytetään rytmittämään pelikenttiä. Vaihtoehtoisesti voisi myös tutkia ohjelmointia ja toiminnallisuuden luomista valmiiseen pelikenttään.

Lähteet

Alex Galuzin 2011, Ultimate Level Design Guide. Luettavissa: http://www.worldofleveldesign.com/categories/books_dvds/ultimate-level-design-guide-11day-level-design-guide-ebooks.php Luettu: 20.9.2012

Andrew Gahan 2011, 3ds Max Modeling for Games, Second Edition, Elsevier Inc., Oxford, UK.

Chris Totten 2012, Game Character Creation with Blender and Unity. John Wiley & Sons, Inc., Indianapolis, Indiana, USA.

Deep Raj 2007, History of Cad. Luettavissa: http://www.articlesbase.com/software-articles/history-of-cad-112609.html Luettu 14.11.2012

Dmitry Shklyar, 3D Rendering History. Luettavissa: http://www.cgsociety.org/index.php/CGSFeatures/CGSFeatureSpecial/3d_rendering_history_part_1._humble_beginnings Luettu 14.11.2012

Jeff Ward 2008, What is Game Engine? Luettavissa: http://www.gamecareerguide.com/features/529/what_is_a_game_.php?page=2 Luettu: 16.9.2012

Mary Bellis, Computer and Video game history. Luettavissa: http://inventors.about.com/library/inventors/blcomputer_videogames.htm Luettu: 17.9.2012

Michal Valient 2001, 3D Engines in games. Luettavissa: http://www.dimension3.sk/downloads/3d_engines_in_games.pdf Luettu 14.11.2012

Luke Ahearn 2012, Third Edition 3D Game Textures, Elsevier Inc., Oxford, UK.

Paul Steed 2005, Modeling a Character in 3DS Max, 2nd Edition, Wordware Publishing, Inc., Texas, USA.

Scott Rogers 2010, Level up the Guide to Great Video Game Design, John Wiley & Sons, Ltd., West Sussex, UK.

Travis Castillo, Jeannie Novak 2008, Game Development Essentials: Game Level Design, Delmar Cengage Learning, New York, USA.

Unity3d. Luettavissa: http://docs.unity3d.com/Documentation/Components/class-MeshCollider.html Luettu: 27.7.2012

Will Goldstone 2011, Unity 3.x Game Development Essentials, Packt Publishing Ltd., Birmingham, UK.