OULUN AMMATTIKORKEAKOULU

Teemu Arkkukangas & Joonas Kulju

Reverse Tower Defense –peli

Reverse Tower Defense –peli

Teemu Arkkukangas & Joonas Kulju
Opinnäytetyö
Kevät 2014
Tietojenkäsittelyn koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjema, internetpalvelut ja digitaalinen media

---

Tekijä(t): Joonas Kulju & Teemu Arkkukangas
Opinnäytetyön nimi: Reverse tower defense-peli
Työn ohjaaja: Matti Viitala
Työn valmistumislukukausi- ja vuosi: Kevät 2014 Sivumäärä: 47

Tämän opinnäytetyön tarkoituksena on suunnitella ja toteuttaa peli Unityä ja Blenderiä käyttäen PC:lle. Peli on yhdistelmä tasohyppelyä ja reverse tower defeseä. Teoriassa käydään lyhyesti läpi videopelien historiaa, Unityn ja Blenderin käyttämistä sekä pelissä käytettäviä genrejä. Käytännön osiossa kerrotaan pelin ohjelmoinnista, 3D-mallien luomisesta ja kentän toteutuksesta. Opinnäytetyön tuloksena syntyy prototyyppi pelistä, jota haluttaessa voidaan jatkokehittää loppuun asti. Toinen tekijöistä keskittyy ohjelmointiin ja toinen pelin visuaaliseen ulkonäköön.

Pelissä pelaajan tehtävänä on päästä kentän loppuun elossa samalla, kun kentässä olevat tornit ampuvat pelaajaa. Pelaajaa on myös estämässä tasohyppelyelementtejä, kuten rotkoja ja liikkuvia alustoja. Päivittämistä varten pelaaja ansaitsee rahaa sitä enemmän, mitä pidemmälle hän pääsee kuolematta. Pelaaja pystyy päivittämään elämäänsä, jolloin hänen on mahdollista selviytyä pidempään. Pelaajalla on lisäksi mahdollisuus käyttää suojakilpeä, jota voi päivittää.

Opinnäytetyön tavoitteet saavutettiin ja tuloksena syntyi toimiva prototyyppi pelistä. Peli sisältää toimivat pelimekaniikat, mutta esimerkiksi visuaalinen puoli on prototyyppitasolla. Opinnäytetyön aikana opittiin lisää ohjelmoinnista, 3D-mallintamisesta ja Unityn käyttämisestä.

Asiasanat: Listaa 3-7 avainsanaa, jotka kuvaavat opinnäytetyötäsi. Blender, Unity, Peli, 3D-mallinnus, ohjelmointi

ABSTRACT

Oulu University of Applied Sciences
Bachelors degree of information technologies, internet services and digital media

---

Author(s): Joonas Kulju & Teemu Arkkukangas
Title of Bachelor´s thesis: Reverse tower defense game
Supervisor(s): Matti Viitala
Term and year of completion: Spring 2014 Number of pages: 47

The purpose of this thesis is to design and develop a videogame for PC by using Unity and Blender as tools. The videogame is a combination of platformer and reverse tower defense. In the theoretical part of the thesis we discuss the history of videogames, Unity, Blender and the genres we use in our game. In the practical part we discuss the game's programming, the creation of the 3D models and the game's level. A prototype of a videogame will be developed as the result of this thesis, which can be developed further. One of this thesis' authors focuses on programming and the other on the game's visuals.

The player's goal is to reach the end of the level while tower are shooting at him. In the game there are platform game elements to hinder the player's progress. The player earns money as he progresses in the level. The money is used to update the player's maximum health and his shield.

Goals of this thesis were achieved and as a result a prototype of the game was developed. The game includes working game mechanics, but the visuals of the game are still on prototype level. During the thesis we learned more about programming, 3d modelling and Unity.

Keywords: Blender, Unity, Game, 3D modelling, programming

SISÄLLYS

1 JOHDANTO

Nykyään pelien kehittäminen on helpottunut huomattavasti ilmaisten pelikehitysympäristöjen julkaisun myötä. Ennen pelkästään isoilla peliyrityksillä ja ammattilaisilla oli mahdollisuus tuottaa näyttäviä ja hyvälaatuisia pelejä. Nykyään on helppoa tehdä pelejä pienen opettelun jälkeen esimerkiksi Unityn avulla. Pelaaminen on yleistynyt huomattavasti esimerkiksi 90-lukuun verrattuna ja se on näkynyt peliteollisuuden nopeassa kasvussa. Varsinkin pieniä pelistudioita on perustettu paljon.

Tämän opinnäytetyön aiheena on videopelin prototyypin tekeminen pääosin Unityä ja Blenderiä käyttämällä. Jaoimme pelin tekemisen tekijöiden kesken siten, että toinen keskittyy ohjelmointiin ja toinen visuaaliseen ulkonäköön. Tekijät tekevät pelin kentän toteutuksen yhdessä. Opinnäytetyöllä ei ole toimeksiantajaa, vaan se on tekijöiden oma tuotos. Opinnäytetyöksi valittiin peli, koska tekijät ovat kiinnostuneet peliteollisuudesta. Opinnäytetyö on rajattu pelin tekemiseen siten, että raportissa ei käydä läpi esimerkiksi pelin markkinointia tai pelin jatkokehityssuunnitelmia. Raportissa ei myöskään puututa paljon äänien tekemiseen.

Peli toteutetaan tietokoneelle ja sen aiheena on Reverse Tower Defense. Pelissä yhdistelemme tower defenseä ja tasohyppelyä. Päädyimme tällaiseen ratkaisuun, koska halusimme kokeilla onnistuuko tällainen genreyhdistelmä. Peli pyritään toteuttamaan annetussa aikarajassa ja tavoitteena on luoda toimiva kokonaisuus. Toisena tavoitteena on tehdä pelistä mahdollisimman viihdyttävä.

Tässä raportissa kerrotaan pelin toteutuksesta. Toisessa luvussa kerrotaan lyhyesti pelien historiasta ja kolmannessa luvussa pelissä käytettävistä genreistä. Neljännessä luvussa kerrotaan Unityn toiminnasta yleisellä tasolla ja viidennessä luvussa Blenderistä. Kuudennessa luvussa käydään läpi pelissä käytettäviä pelimekaniikoita ja seitsemännessä luvussa pelin visuaalista ulkonäköä. Kahdeksannessa luvussa kerrotaan pelin toteutukseen liittyvistä haasteista. Yhdeksännessä luvussa käydään läpi pelin toteutus. Lopuksi pohditaan esimerkiksi opinnäytetyön aikana opittuja asioita ja viimeisessä kappaleessa on yhteenveto opinnäytetyöstä.

2 VIDEOPELIEN HISTORIA

Ihmiset ovat harrastaneet pelaamista jo vuosituhansien ajan. Vanhin löydetty lautapeli on nimeltään Senet ja se on yli 5000 vuotta vanha. Tämän lisäksi on löydetty lähes yhtä vanhoja noppia Aasiasta ja Iranista. (Radoff 2010.)

Vuonna 1958 William Higinbotham loi ensimmäisen varsinaisen videopelin nimeltä Tennis for Two ja se oli esillä yleisölle Brookhavenin valtiollisessa laboratoriossa kaksi vuotta. Tennis for Two ei kuitenkaan päässyt laajempaan levitykseen, vaan se purettiin ja sen osat laitettiin uusiokäyttöön.(Tennis For Two 2014.) 1960-luvun alkupuolella keksittiin peli nimeltä Spacewar!, joka on kahden pelaajan avaruusräiskintäpeli tietokoneelle. Spacewar! osoittautui hyvin suosituksi ja 1960-luvun loppupuolella se löytyi monesta USA:n yliopistosta. (Spacewar (videogame) 2014.)

Vasta 1970-luvulla Atarin Pong-pelin myötä videopelit alkoivat saada suurempaan suosiota kuluttajien keskuudessa. Atarin menestys jatkui, kun he toivat markkinoille Atari 2600 pelikonsolin. Tätä aikakautta tultiin myöhemmin kutsumaan videopelien kulta-ajaksi. Tänä aikana keksittiin monia peli klassikoita, kuten Space Invaders ja Pac-Man. 1980-luvun alussa Atarin suosio alkoi laskea ja sen paikalle ilmestyi uusi peliyritys nimeltä Nintendo. Nintendon pelikonsoli nousi Amerikassa suureen suosioon ja sen myötä videopelit alkoivat olla osa ihmisten elämää. (The 30 defining moments in gaming 2007.) Nintendon myötä syntyi nykypäivänäkin hyvin tunnettuja pelejä, kuten The Legend of Zelda, Mega Man, Castlevania ja kaikkein tunnetuin peli Super Mario Bros. Nintendon menestys jatkui Nintendo Game Boyn keksimisen myötä vuonna 1989. Game Boy oli ensimmäinen suuren suosion saanut käsikonsoli. Pokemon on yksi Game Boyn tunnetuimmista ja suosituimmista peleistä. Nykypäivänäkin Nintendo hallitsee käsikonsolien markkinoita. (History of videogames 2014.)

1990-luvulla PC-pelaaminen alkoi yleistyä ja peleissä alkoi olla entistä enemmän 3D-grafiikkaa. Peliyritys id Softwaren DOOM (1993) ja Wolfestein 3D (1992) ovat 1990-luvun alkupuolen tunnetuimmista peleistä. PC-pelaamisen myötä alkoi uusien peliyritysten syntyminen, joista moni on edelleenkin olemassa. 1990-luvun puolivälissä Sonyn PlayStation ja Nintendon Nintendo 64 pelikonsolit hämmästyttivät maailman suosiollaan. Ne olivat ensimmäiset suositut pelikonsolit, jotka

tarjosivat 3D-grafiikalla varustettuja pelejä. Näihin aikoihin online pelaaminenkin alkoi yleistyä, vaikka sitä oli ollutkin olemassa jo jonkin aikaa. 2000-luvun alussa Microsoft tuli pelikonsoli markkinoille omalla Xbox-konsolillaan. (History of videogames 2014.)

3 GENRET

3.1 Tower Defense

Ensimmäinen tower defensenä pidetty peli julkaistiin vuonna 1990. Tämän pelin nimi oli Rampart ja sen julkaisija oli Atari Games. (Atari Games 2014) Rampartissa pelaajan täytyi suojella linnoja hyökkääviltä laivoilta ampumalla ja korjata linnoihin kohdistuneet vauriot (Rampart (video game) 2014). Tämän pelin ilmestymisen jälkeen monet muut pelit seurasivat Rampartin kartan jalanjälkiä, joista esimerkkinä ovat 2000-luvun alussa tehdyt pelit kuten Age of Empires II, Starcraft ja Warcraft III. Myös ajatellaan, että tower defence -genre on saanut inspiraatiota vuonna 1997 valmistuneesta pelistä nimeltään Final Fantasy VII. Tämä videopeli sisälsi minipelin nimeltään The Fort Condor ja tässä minipelissä oli useita samoja elementtejä, kuin nykypäivän tower defence-peleissä. (Final Fantasy VII 2014)

Verkkoselaimissa pelattavat Adobe Flashilla toimivat tower defence -pelit tulivat vuonna 2007 Flash Element Tower Defense-pelin myötä. Heti tämän pelin jälkeen tuli peli nimeltään Desktop Tower Defense. Desktop Tower Defensestä tuli todella tunnettu ja se sai Independent Games Festival-palkinnon. Tästä pelistä tehtiin myös mobiiliversio toisen valmistajan toimesta. Muita suosion saaneita pelejä ajansaatossa olivat esimerkiksi GemCraft ja Plants vs. Zombies. (Tower defense 2014)

Vuonna 2008 tower defence -genre oli kasvanut niin tunnetuksi, että sitä voitiin alkaa toteuttamaan konsolimarkkinoilla. Esimerkkinä tästä on Defense Grid: The Awakening, jonka alustaksi tuli Xbox 360. Playstation 3 konsolille esimerkkejä tower defence -genrestä ovat PixelJunk Monster ja Savage Moon. Tower defence-genreä on tehty myös käsikonsoleille, esimerkiksi Ninjatown ja Lock's Quest Nintendo DS:lle. (Tower defense 2014)

Reverse tower defense on tower defensen alagenre. Tässä tower defensen perusidea käännetään päinvastoin. Puolustamisen sijasta hyökätään jotain ennaltamäärättyä kohdetta vastaan tornien

ampuessa pelaajaa. (The Sequel To The Best Reverse-Tower-Defense Game Is Superb, If Barely a Sequel 2014)

3.2 Tasohyppely

Tasohyppelypelit ovat todella yleisiä videopelien keskuudessa. Tasohyppelypeleissä on tarkoitus hyppiä erilaisten tasojen välillä. Klassiset tasohyppelypelit ovat jaettu klassiseen malliin 2D- ja 3D-peleihin. Tasohyppelypeleissä seuraavat elementit ovat yleisiä: yksinkertaiset kontrollit, erilaiset pelialueet, kerättävät tavarat ja pelaajan pystyminen puolustautumaan vihollisia vastaan jollain tavalla. (Tasohyppely 2014)

Tasohyppelypelit saivat alkunsa 1980-luvulla. Ensimmäisenä pidetty tasohyppelypeli oli Universalin julkaisema Space Panic, joka tehtiin vuonna 1980. Tämä peli ei kuitenkaan pitänyt sisällään kaikkia tasohyppelypelin elementtejä. Esimerkiksi pelissä ei voinut hyppiä ollenkaan, vaan pelissä kiivettiin tikkaita pitkin tasolta toiselle. Ensimmäinen peli, jossa pelaaja pystyi hyppimään esteiden ja kuilujen yli oli suosittu Donkey Kong. Tämän pelin tekijänä oli Nintendo ja julkaisuvuosi oli 1981. Pelikonsoleiden yleistyttyä tasohyppely-genrestä tuli todella suosittu. (Platform game 2014) (Super Mario 64)

3D-videopelit yleistyivät 1990-luvulla, jolloin ilmestyi paljon 3D-tasohyppelypelejäkin. Nämä pelit ovat toimintapitoisempia, kuin klassiset 2D-tasohyppelypelit. Esimerkkinä tästä on peli nimeltään Super Mario 64, jossa pelaajan täytyi kerätä 120 tähteä 15 erilaisesta kentästä. Tekemällä erilaisia tehtäviä pelaaja palkittiin tähdellä. Pelissä oli paljon erilaisia liikkeitä. Pelaaja pystyi juoksemaan, kävelemään, hiipimään, uimaan, kiipeämään ja hyppimään. (Tasohyppely 2014) Ensimmäinen ensimmäisen persoonan 3D tasohyppelypeli oli nimeltään Geograph Seal ja se julkaistiin vuonna 1994 (Platform Game 2014).

3.3 First-person shooter

First-person shooter –genressä tarkoituksena on ampua vihollisia ensimmäisen persoonan perspektiivistä 3D-ympäristössä. Yleensä tälläisissä peleissä pelaajalla on käytössään useita erilaisia aseita ja pelaajan ympärille on rakennettu tarina, joka selittää miksi pelaajan täytyy ampua vihollisia. First-person shooter – genren syntymiseen on vaikuttanut eniten id Softwaren DOOM ja Wolfenstein 3D. (First-person shooter 2014)

Nykyään tälläisissä peleissä on mahdollisuus moninpeliin. Moninpeleissä ihmiset pelaavat toisiansa vastaan erilaisissa pelimuodoissa, kuten lipun ryöstö tai kaikki vastaan kaikki. First-person shooter – peleissä on useita erilaisia kenttäsuunnittelu mahdollisuuksia.Esimerkiksi perinteisesti toteutettu kentästä kenttään –meneltemä, yksi laaja jatkuva kenttä ja laaja vapaa maailma. (First-person shooter 2014)

4 UNITY

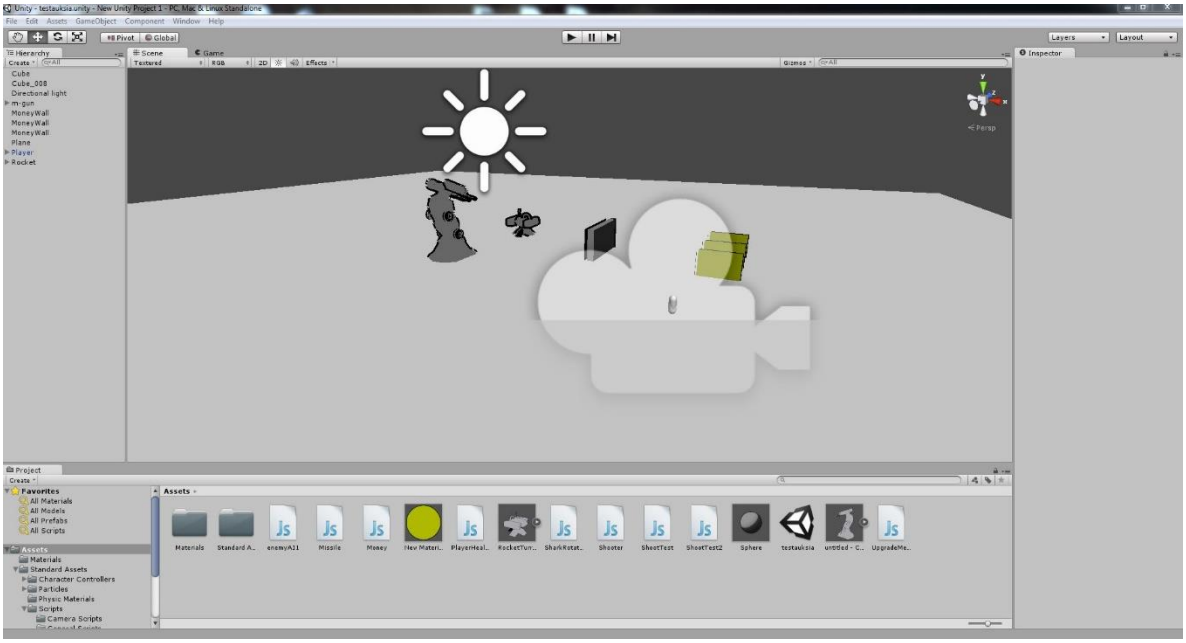
Unity on Unity Technologiesin kehittämä pelientekoympäristö, jolla voi tehdä pelejä kaikille pelialustoille. Unityn kehitys aloitetiin vuonna 2001 ja sen ensimmäinen versio julkaistiin Applen Worldwide Developers Conference-tapahtumassa. (Milestones 2014.) Unity on pääosin ilmainen ympäristö, mutta siitä on olemassa myös maksullinen versio Unity Pro. Maksullinen versio eroaa ilmaisesta siten, että Unity Pro tarjoaa ilmaisversiota enemmän ominaisuuksia ja sitä voidaan käyttää myös kaupalliseen tarkoitukseen. (Licence comparisons 2014.) Unityn tämänhetkinen versio on 4.3 (Unity 2014).

Unityä on käytetty monien pelien kehittämisessä ja hyviä esimerkkejä näistä ovat Gone Home ja Endless Space (Made with Unity 2014). Jokainen näistä kolmesta pelistä on saanut hyviä arvosteluja. Tämä osoittaa muun muassa sen, että Unityn avulla voidaan tuottaa hyvin laadukkaita pelejä (Silva 2013)(Dean 2010).

Unity sopii hyvin pienille peliyrityksille ja harrastelijoille siksi, että se on helppokäyttöinen, ja koska siitä on olemassa ilmainen versio. Helppokäyttöisyyttä tukee se, että Unityn käyttöliittymä muistuttaa esimerkiksi pelien kenttäeditoreja (Customizing Your Workspace 2014). Unity on valmis pelimoottori, joten ohjelmointia vaativat ainoastaan pelin erilaiset toiminnot kuten ampuminen ja hahmojen tekoäly (What is Unity 2014). Ohjelmoinnin voi tehdä millä tahansa koodi- tai tekstieditorilla tai Unityn mukana tulevalla MonoDevelop-ohjelmalla (Getting started with Mono Develop 2014).

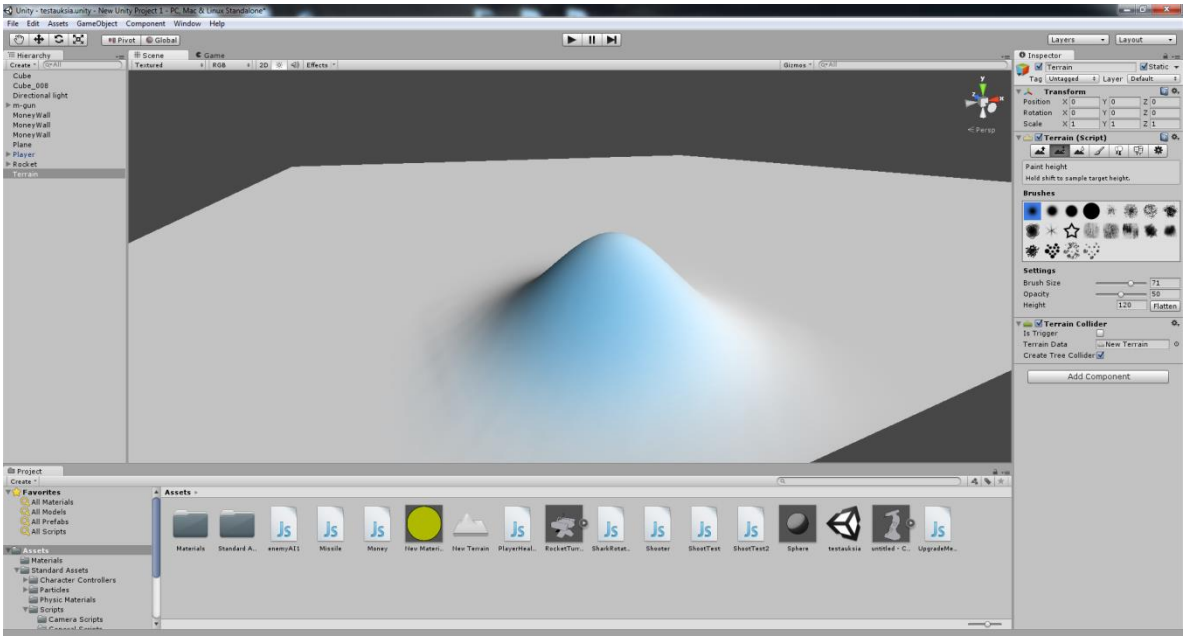
Unityn käyttöliittymä (Kuva 1.) koostuu erilaisista ikkunoista ja sitä on hyvin helppo muokata itselleen sopivaksi. Ikkunoita voi lisätä, poistaa, siirtää, suurentaa ja pienentää sekä niitä voi irrottaa kokonaan käyttöliittymästä. (Customizing Your Workspace 2014) Kolme Unityn tärkeintä ikkunaa ovat Scene-ikkuna, Game-ikkuna ja Assets-valikko. Assets-valikon rinnalla on yleensä Hierarchy-valikko. Game-ikkuna näyttää samalta, miltä varsinainen peli näyttää pelaajan näkökulmasta. Scene-ikkunassa voidaan muokata peliä esimerkiksi muokkaamalla kentän asettelua, lisäämällä ja poistamalla pelin objekteja tai lisäämällä objekteihin toiminnallisuuksia kooditiedostoilla. Assets-valikossa näkyy kaikki ne tiedostot ja objektit, joita peliprojektissa käytetään, esimerkiksi 3D-mallit ja kooditiedostot. Assets-

valikosta objektit vedetään Scene-ikkunaan, jolloin ne tulevat osaksi kenttää ja näkyvät Hierarchy-valikossa. Hierarchy-valikko sisältää kaikki tiettyyn kenttään kuuluvat objektit. (Learning the Interface 2014)(Game View 2014) Esimerkiksi kentässä 1. olevat objektit eivät näy kentän 2. Hierarchy-valikossa.



Kuva 1: Käyttöliittymä

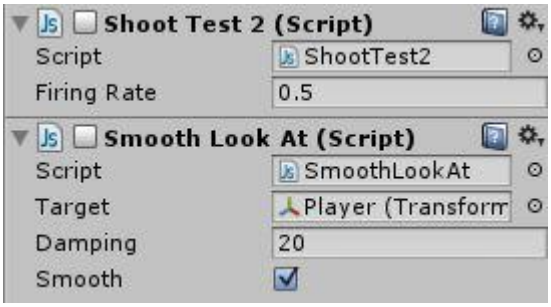
Unityssä kentän luominen aloitetaan yleensä Terrain-objektin luomisella. Tämän voi tehdä Unityn valmiilla Terrain-työkalulla tai sen voi luoda esimerkiksi 3D-mallinnus ohjelmalla. Terrainia voi muokata sopivaksi erilaisilla piirtotyökaluilla, joita käytetään myös Terrainin tekstuurien maalaamiseen (Kuva 2). Piirtotyökaluilla voidaan esimerkiksi nostaa ja laskea maanpintaa ja Terrainissa käytettävän tekstuurin voi valita itse. (Terrain Engine Overview 2014)



Kuva 2: Terrain

3D-mallit tuodaan kenttään 3D-mallinnus ohjelmista erilaisissa formaateissa. Kun 3D-malli on tuotu kenttään, sen sijaintia voi muuttaa vapaasti hiirellä (Using the Scene View 2014). Pelin 3D-malleille voi lisätä törmäyksentunnistuksen ja fysiikan. Törmäyksentunnistus huolehtii siitä, että pelaaja ei voi liikkua 3D-mallien läpi. Fysiikalla määritellään kuinka paljon esimerkiksi 3D-malli painaa ja kuinka paljon painovoima vaikuttaa siihen. Unityssä tulee mukana NVIDIA PhysX-fysiikka moottori ja Box2D-fysiikkamoottori 2D-moottori, jotka hoitavat muun muassa kappaleiden tippumisen. Peliä luodessa on kuitenkin hyvä muistaa, että fysiikan laskeminen usealle objektille on tietokoneelle raskasta ja siksi sitä kannattaa käyttää säästäen. (Unity physics 2014)

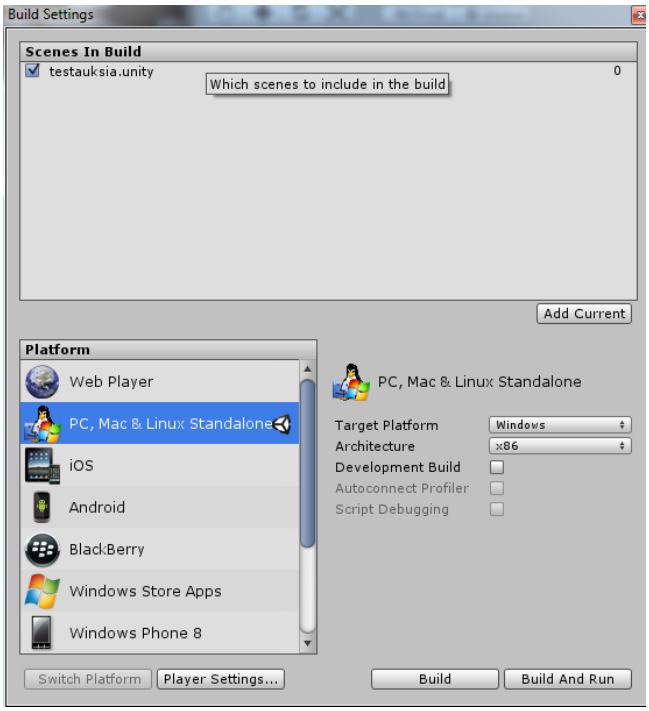
Unityssä pelilogiikka ja objektien eri toiminnot hoidetaan kirjoittamalla niille omat kooditiedostot. Eri toiminnallisuuksista voidaan kirjoittaa omat kooditiedostot, jotka liitetään siihen kuuluvaan objektiin. (Creating and Using Scripts 2014) Toiminnallisuus voi olla esimerkiksi se, että jokin objekti katsoo pelaajan suuntaan ja ampuu tätä. Kuvassa 3 3D-malliin on liitetty kaksi kooditiedostoa. Toinen niistä hoitaa pelaajaan katsomisen ja toinen ampumisen. Kooditiedostossa olevia arvoja voidaan muuttaa Unityssä ilman, että tarvitsee aukaista itse kooditiedostoa (Inspector 2014).



Kuva 3: Skripti

Peli Unityssä koostuu eri sceneistä. Jos pelissä on monta eri kenttää, voi tehdä jokaisesta kentästä oman scenen. Yleensä pelin aloitusvalikko on omana scenenä. Unityssä sceneä voi vaihtaa helposti lyhyellä koodirivillä tai editorissa kaksoisklikkaamalla sceneä. (Creating Scenes 2014)

Kun peli on valmis julkaistavaksi, se täytyy kääntää. Kääntövaiheen asetuksista (Kuva 4.) valitaan peliin tulevat scenet ja niiden järjestys. Seuraavaksi valitaan alusta, jolle peli käännetään. (Publishing Builds 2014) Unityn ilmaisversiossa on mahdollista kääntää esimerkiksi PC:lle ja nettiselaimelle. Ilmaisversio tukee myös mobiilijulkaisuja. (Download 2014)



Kuva 4: Build settings

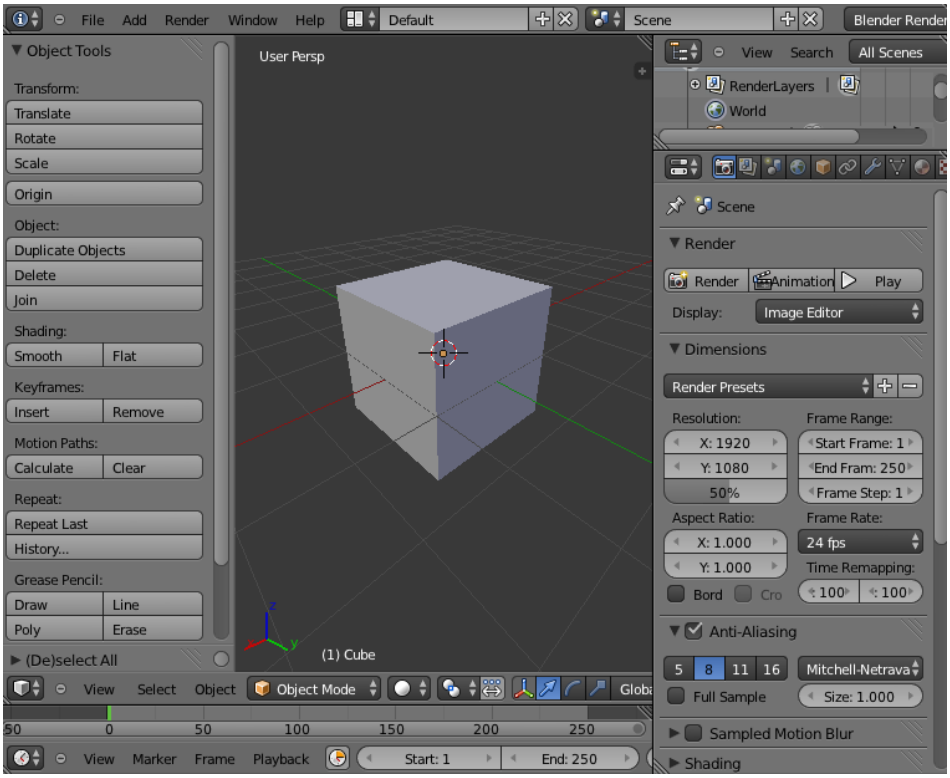
5 BLENDER

Blender on ilmainen ja avoimeen lähdekoodiin perustuva 3D-grafiikan mallinnusohjelma. Blenderillä voi tehdä 3D-malleja, tuottaa animaatiota, tehdä visuaalisia tehosteita, luoda interaktiivisia 3D-sovelluksia ja tehdä videopelejä. Hollantilainen animaatiostudio Neo Geo ja Not a Number Technologies suunnittelivat tämän sovelluksen ja se julkaistiin 2000-luvun alussa. Blender käyttää GNU (General Public License) lisenssiä, eli käyttäjillä on oikeus käyttää, kopioida, muuttaa ja jakaa edelleen tätä sovellusta ja sen lähdekoodia. (GNU General Public License 2014) Blenderiä kehitetään jatkuvasti satojen vapaaehtoisten toimesta ympäri maailman (About 2014).

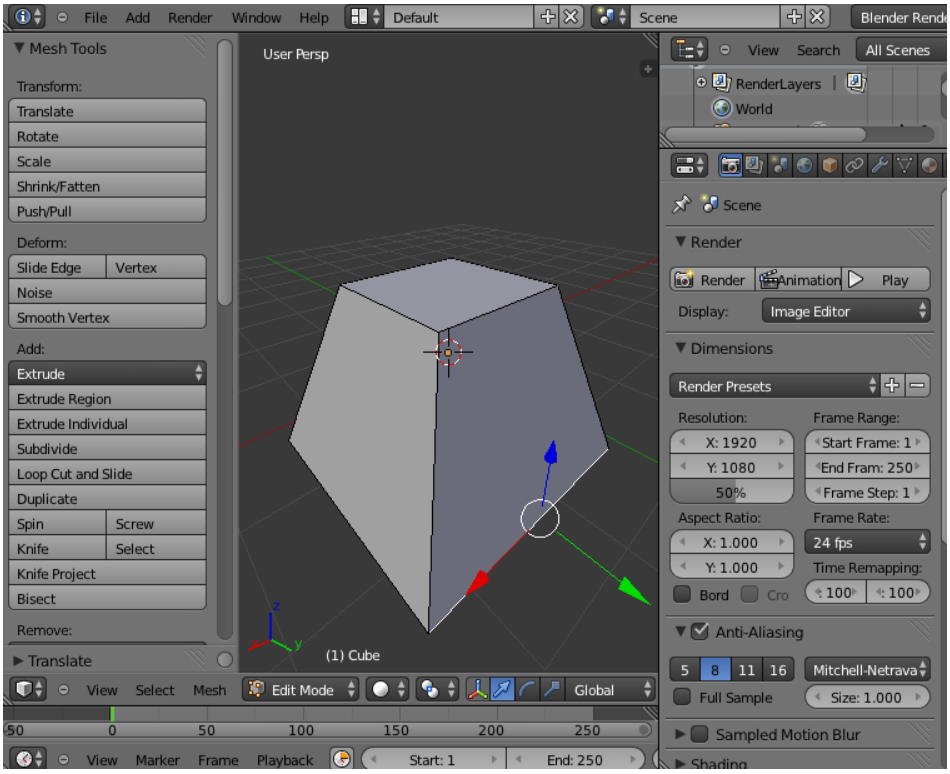
Blender on monipuolinen sovellus, joka koostuu useasta eri ominaisuudesta. Blenderin ominaisuuksia ovat 3D-mallintaminen, teksturointi, rigging-ominaisuus, nesteen ja savun simulointi, particle(hiukkas?)simulointi, pehmeän rungon simulointi, muotoilu(sculpting), animointi, match moving-ominaisuus, kameran seuraamien(camera tracking), renderöinti, videon editointi ja sommittelu. (Blender (software) 2014) (Features 2014)

Ensimmäinen ammattilaisprojekti, jossa käytettiin Blenderiä, oli elokuva Spiderman 2. Blenderiä käytettiin elokuvassa luomaan erilaisia animaatioita. Myöhemmin Blenderiä käytettiin tekemään erikoistehosteet ranskankieliseen elokuvaan Friday or Another Day. Blenderiä on käytetty myös esimerkiksi History channelilla muiden 3D-mallinnusohjelmien lisäksi. (Blender (software) 2014)

Blender on erinomainen vaihtoehto kaikille 3D-mallinnuksesta kiinnostuneille. Sen käyttöliittymä on kohtuullisen helppo oppia ja sovelluksesta löytyy todella paljon opastusvideoita. Blender koostuu useasta eri toimitilasta, joilla kaikilla on omat ominaisuutensa. Tärkeimmät näistä tiloista ovat esinetila (object mode) (Kuva 5) ja muokkaustila (edit mode) (Kuva 6). Näiden tilojen välillä voi liikkua tab-näppäintä painamalla. Esinetilaa käytetään esimerkiksi objektien koon ja sijainnin muokkaamiseen. Muokkaustilaa taas tarvitaan esimerkiksi, jos halutaan muokata 3D-mallia erinäköiseksi. Blenderin käyttöä helpottaa laaja valikoima pikanäppäimiä, joiden opeteltua mallintamisesta tulee todella nopeaa.



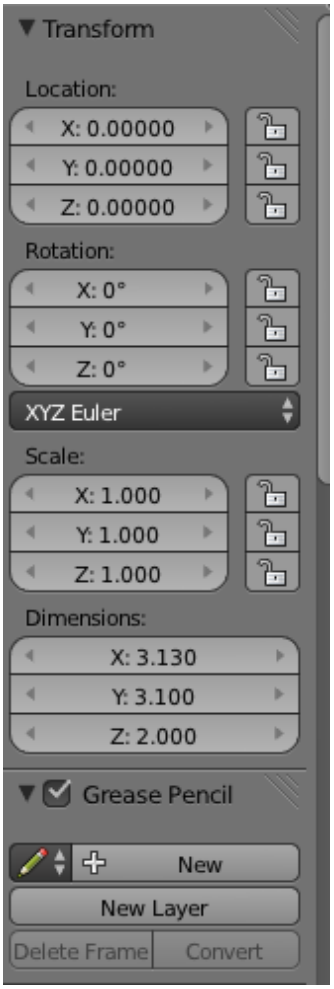
Kuva 5: Object mode



Kuva 6: Edit mode

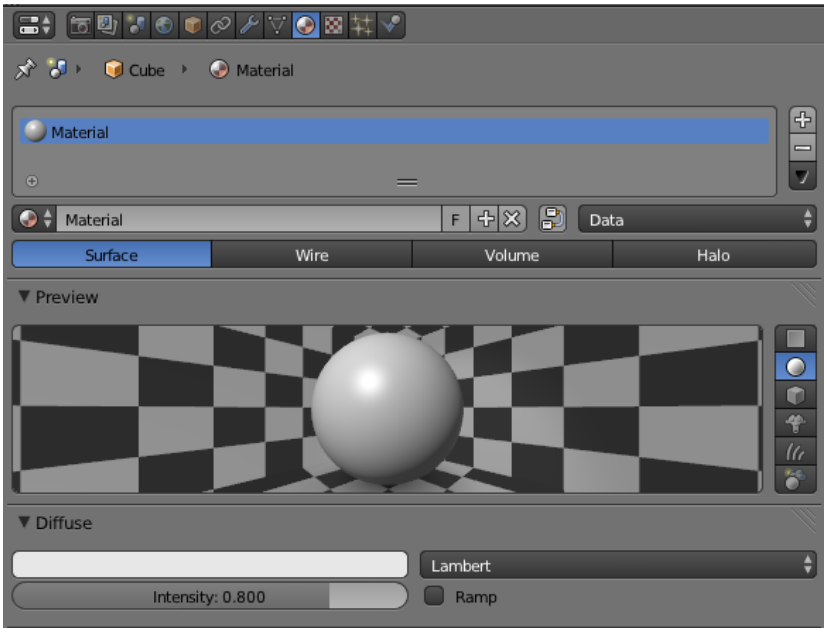
Blenderin työtila jaetaan erilaisiin osiin ja näitä osia pystyy myös hallinnoimaan helposti. Oletuksena on 3D-näkymä, jossa käyttöliittymän vasemmassa reunassa on objektityökalut (Kuva 7), joilla pystyy esimerkiksi muokkaamaan 3D-mallien sijaintia ja kokoa. Oikeassa reunassa on transform-osio (Kuva 8), josta näkee tarkempia tietoja objektista. Oikealta ylhäältä löytyy Scene-näkymä (Kuva 9) ja täältä voi nähdä mitä kaikkia objekteja on tällä hetkellä nykyisessä scenessä. Oikealta löytyy myös lista erilaisista työkaluista. Tästä listasta löytyy esimerkiksi materiaalieditori, jolla voi asentaa 3D-mallin päälle erilaisia visuaalisia materiaaleja (Kuva10).

Kuva 7: Object tools



Kuva 8: Transform

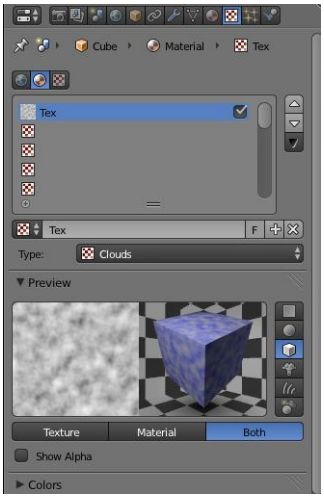
Kuva 9: Scene



Kuva 10: Material

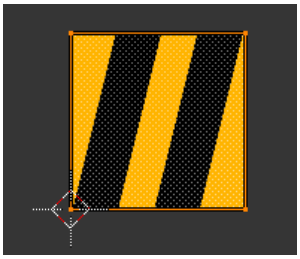
3D-mallinnuksen voi aloittaa useammalla eri tavalla. Pelissä käytetään Box modeling –tekniikkaa ja tämä aloitetaan tekemällä jokin aloitusobjekti, kuten kuutio tai pallo. Tätä objektia lähdetään muokkaamaan esine- ja muokkaustilassa. Objektia voi muokata myös muilla keinoin, esimerkiksi asettamalla sille jokin muokkaustoiminto (modifier). Kun 3D-malli on halutun näköinen, sille voidaan laittaa väriä materiaali-osiosta tai laittaa sille tekstuuri. 3D-mallin ollessa valmis sen voi renderöidä, jolloin sovellus piirtää 3D-mallista kuvan kamera-objektista katsoen. 3D-malliin voi halutessa käyttää export-toimintoa, jolloin 3D-mallia voidaan käyttää vaikka tuomalla se Unity-ohjelmaan.

Materiaali-osion lisäksi 3D-mallille voidaan myös lisätä tekstuureja eli bittikarttakuvia. Materiaalilla saa aikaan vain tasaista pintaa, mutta jos halutaan 3D-mallin päälle jotain kuviollista pintaa, täytyy käyttää teksturointia. Tekstuuri (Kuva 11) on yleensä kaksiuloitteinen bittikarttakuva, joka lisätään 3D-mallin päälle. Tämän jälkeen voidaan säätää teksturoinnin asetuksia. Esimerkiksi voidaan säätää kuinka monta kertaa bittikarttakuva toistetaan objektin pinnalle. Teksturointi on todella yleinen keino, jota käytetään esimerkiksi videopelien 3D-mallien tekemiseen. (Texture mapping 2014) (Textures common options 2014)

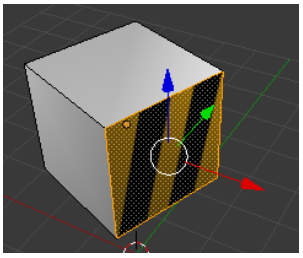


Kuva 11: Teksturointi

Normaalin teksturoinnin lisäksi Blenderissä voidaan käyttää myös UV-kartoitusta. Tällä pystytään tekemään tarkempaa teksturointia objektien pinnoille ja sitä pystyy tekemään tilassa nimeltä UV/Image Editor. Ensin täytyy valita objektin alueet, jotka halutaan UV-kartoittaa. Tämän jälkeen valitaan bittikarttakuva, joka halutaan siirtää valitulle alueelle UV/Image Editorissa (Kuva 12). Haluttua UV-kartoitusta voidaan skaalata tai kiertää tarpeen vaatiessa. Asettelun lopetettua, bittikarttakuva siirtyy 3D-objektin pinnalle (Kuva 13). (TEKSTUURIT BLENDERISSÄ 2014)



Kuva 12: UV/Image Editor



Kuva 13: UV-kartoitettu 3D-malli

6 PELIMEKANIIKKA

Pelissämme ideana on yhdistää ensimmäisestä persoonasta kuvattua peliä reverse tower defence-genren kanssa. Pelissä pelaajan tehtävänä on päästä kentän loppuun elossa samalla, kun kentässä olevat tornit ampuvat pelaajaa. Pelaajaa on myös estämässä tasohyppelyelementtejä, kuten rotkoja ja liikkuvia alustoja.

Tarkoituksenamme on luoda peli, jossa pelaaja ei pääse helposti kenttää läpi, vaan hän tulee kuolemaan pelin aikana useasti. Kuoleman jälkeen pelaajalla on mahdollisuus päivittää hahmoaan, jotta hän pääsisi kentässä pidemmälle. Päivittämistä varten pelaaja ansaitsee rahaa sitä enemmän, mitä pidemmälle hän pääsee kuolematta. Pelaaja pystyy päivittämään elämäänsä, jolloin hänen on mahdollista selviytyä pidempään. Pelaajalla on lisäksi mahdollisuus käyttää suojakilpeä, jota voi päivittää.

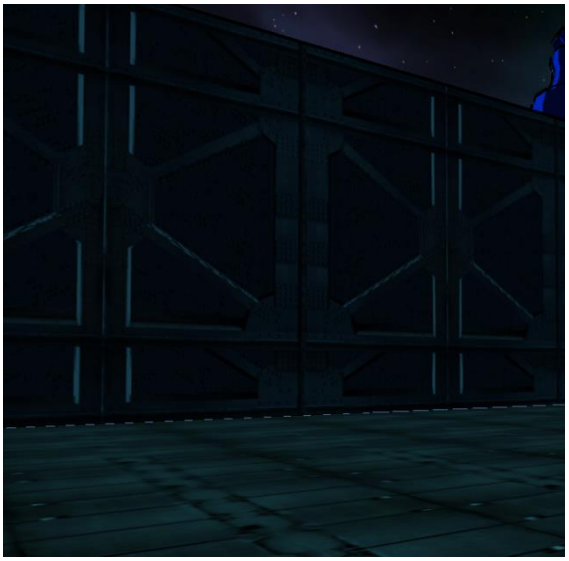
Pelissämme on kahdenlaisia torneja: ohjustorneja ja konekivääritorneja. Ohjustornit ampuvat itsestään ohjautuvia ohjuksia ja konekivääritornit luoteja. Konekivääritorni ei ammu näkyviä luoteja toisin kuin ohjustornit. Ohjustornit ja konekivääritornit eroavat siten, että ohjustornit tekevät enemmän vahinkoa, mutta ampuvat hitaammin. Ohjustornien ohjukset voivat myös osua esteisiin ja näin pelaaja voi säästyä vahingolta. Molemmilla tornityypeillä on samanlainen oma havaintoalue. Kun pelaaja astuu havaintoalueelle, torni alkaa ampua pelaajaa. Torni lopettaa ampumisen kun pelaaja poistuu havaintoalueelta.

7 PELIN VISUAALINEN ILME

Pelissä käytämme kirkkaita värejä sarjakuvatyyliin. Peli sijoittuu tulevaisuuteen ja tämän takia käytämme esimerkiksi robotteja torneina. Pelikenttä rakennetaan futuristiseen tyyliin käyttämällä sen mukaisia tekstuureja. Pyrimme luomaan selkeää ja hyvännäköistä grafiikkaa peliin. Käytämme kaikissa peliobjekteissa sarjakuvagrafiikkaa ja tämä onnistuu käyttämällä Unityn toon shader-ominaisuutta. Tämä muokkaa objekteille paksut ääriviivat ja tekee objekteista selkeämmän näköisiä. Sarjakuvatyyliä saamme myös muokkaamalla kaikki peliin tulevat tekstuurit ennen Unityyn siirtämistä.

Olemme hakeneet pelin ulkoasulle mallia pääasiassa kahdesta olemassa olevasta pelistä. Nämä pelit ovat Borderlands ja Portal. Borderlandsissa käytetään erittäin sarjakuvamaista grafiikkaa kun taas Portalissa on käytetty yksinkertaisia tekstuureja ja futuristista ilmettä. Käytimme sarjakuvagrafiikkaa pelissämme, koska mielestämme sarjakuvagrafiikka sopii todella hyvin videopeleihin ja yleensä tower defence-peleissä ei ole käytetty niin paljoa sarjakuvagrafiikkaa. Tulevaisuuteen perustuvaa teemaa käytimme, koska haluamme tuoda peliin kuvitteellista tieteisfiktiota ja tätä kautta luoda tunnelmaa peliin.

Pelikenttä sisältää paljon metallisia ja futuristisia tekstuureita. Pelialue on suljettu metallisilla seinillä (Kuva 14) ja pelialueen yläpuolella on näkyvillä musta avaruus. Pelialueen yllä loistaa myös muutamia suuria ja värikkäitä tähtiä (Kuva 15). Peliin tuodaan tunnelmaa myös äänillä. Pelin taustalla soi nopeatempoinen kappale. Peli sisältää futuristisia objekteja, kuten robotteja, leijuvia alustoja (Kuva 16) ja lasersäteitä. Tehokeinoina käytämme myös muutamaa partikkeliefektiä, mutta joudumme rajoittamaan näiden käyttöä, koska partikkelit rasittavat suhteellisen paljon pelin suorituskykyä. Päätimme, että teemme pelin, joka toimii hyvin ilman lagia. Se on parempi ratkaisu verrattuna peliin, joka on ehkä hieman paremman näköinen visuaalisesti useiden partikkelien takia, mutta tällöin pelissä ilmenee lagia. Pelissä viive vaikuttaa pelaajakokemukseen negatiivisesti. Varsinkin tasohyppely on todella vaikeaa, jos pelissä ilmenee lagia samanaikaisesti.



Kuva 14 Teksturointia

Kuva 15 Tähti

Kuva 16 Alusta

8 PELIN TEKEMISEN HAASTEET

Videopelin tekemiseen liittyy aina haasteita ja moni asia ei yleensä onnistu ensimmäisellä yrittämällä. Videopelin rakentaminen voidaan luokitella useampaan eri osa-alueeseen, kuten ohjelmointiin, 3D-mallinnukseen ja itse objektien sijoittelu pelialueelle. Videopelin tulisi olla lopulta toimiva kokonaisuus, johon vaikuttaa moni eri asia. Näitä asioita ovat esimerkiksi pelin optimointi, eli miten raskas tietokoneen on pyörittää peliä. Toinen tärkeä asia pelin onnistumisen kannalta on skriptien toimivuus. Haasteena on myös pelialueen huolellinen suunnittelu ja tekeminen. Pelin tekemiseen kuluva aika on monesti aivan liian lyhyt ja peli tulisi ehtiä testaamaan kunnolla ennen julkaisua.

Pelissä, jossa ei ole otettu huomioon pelin optimointia voi käydä huonosti. Pelissä voi olla esimerkiksi paljon yhtäaikaisia liikkuvia objekteja kerralla. Tämä voi johtaa pelissä näkyvään lagiin (lag). Lagi tarkoittaa peleissä näkyvää nykimistä eli tällöin peli ei toimi sulavasti. Lagia pelissä mitataan kuvataajuudella, joka kuvaa näyttötekniikassa sitä, kuinka monta kuvaa sekunnissa näytölle piirretään. Mitä isompi kuvataajuus on, sitä paremmalta liike näyttää pelissä. Kuvataajuuden suuruuteen vaikuttaa tietysti myös tietokoneen suorituskyky ja miten raskailla grafiikoilla peliä ajetaan. Peliin tulee paljon animoituja objekteja ja tästä syystä joudumme tutkimaan kuvataajuuden lukemia. Haasteena on se, että kuvataajuus putoaa liian alas. Tällöin peli näyttää huonommalta, mikä tekee pelin pelaamisesta hankalaa. Unityn partikkeliefektit rasittavat suorituskykyä, mikäli näitä käytetään paljon. Yksi tapa parantaa optimointia on muuttamalla skriptejä siten, että ne eivät rasita suorituskykyä niin paljon. (Kuvataajuus 2014)

Ohjelmoinnin kannalta on tärkeää, että kaikki skriptit on kirjoitettu virheettömästi ja että ne ovat sijoitettu oikeisiin objekteihin. Unity esimerkiksi ilmoittaa viallisesta skriptistä, jos skriptissä on jokin kirjoitusvirhe. Haasteellisempaa on skriptien toiminnallisuus. Skriptit eivät saa mennä sekaisin toisten skriptien kanssa. Skriptien välinen kommunikointi on yksi haaste. Skriptit lähettävät usein tietoja keskenään ja tämän toimivuus on pelin toimivuuden kannalta todella tärkeää.

Peliin tulee paljon erilaisia tilanteita, joissa pelaaja joutuu miettimään miten tästä päästään etenemään. Nämä erilaiset pelitilanteet joudutaan miettimään tarkasti. Esimerkkinä voidaan käyttää

peliin tulevia lasereita, joita pelaajan tulee pystyä väistämään. Lasereiden sijoittelu ja niiden toiminnallisuus voi olla haastavaa. Osa lasereista joudutaan myös animoimaan, joka vie paljon aikaa. Erilaisten pelitilanteiden tulisi olla myös viihdyttäviä, jotta pelaaja ei toistaisi samoja asioita. Tästä syystä joudumme käyttämään paljon mielikuvitusta, jotta peli pitäisi pelaajan mielenkiintoa yllä. Objektien sijoittelu pelialueelle on oma haasteensa.

Pelialueen toteuttaminen on myös iso haaste, koska siihen vaikuttaa useampi eri riskitekijä. Ensiksi pelialue tulee rajata hyvin ja muokata kentän pituus sopivaksi. Pelialue ei saa sisältää mitään kohtia, joista pelaaja pääsisi pelialueen ulkopuolelle. Peliä täytyy ehtiä testaamaan tarpeeksi, jotta pelistä ei tule liian hankala tai liian helppo. Suurin haaste tässä on se, että pelin tornien tekemä vahinko saadaan säädettyä sopivaksi. Pelialue täytyy myös testata hyvin, jotta kenttä ei sisällä mitään virheitä ja kaikkien objektien täytyy toimia suunnitellulla tavalla. Tämän takia testaamiselle täytyy varata hyvin aikaa.

Pelin yhtenä haasteena on saada pelistä visuaalisesti kaunis. 3D-malleista täytyy saada tehtyä robottien ulkonäköä vastaava. Vaikka 3D-mallien muodot olisivat oikeanlaiset, tarvitsevat ne silti väriä tai tekstuurit niiden pinnoille. Objekteille täytyy löytää oikeanlaiset värisävyt tai tekstuurit. Oikeat värit saadaan yleensä vain kokeilemalla, mutta tekstuurien tekeminen ja muokkaaminen vie paljon aikaa. Tekstuureja teemme itse tai käytämme valmiita tekstuureja Internetistä. Lisäksi kaikkien objektien ulkonäkö pitää saada sopimaan pelin teemaan. Peli täytyy saada näyttämään futuristiseen sävyyn hyvältä. Täytyy myös varoa, ettei pelistä tule liian realistisen näköinen, sillä tarkoituksenamme on käyttää pelissä pääosin sarjakuvagrafiikkaa. Myös peliin tulevat äänet voi olla hankala löytää, koska niiden täytyy vastata pelissä käytettävää futuristista teemaa. Yhtenä haasteena on Blenderin ja Unityn yhteensopivuus, sillä Unity ei hyväksy suoraan Blenderin blend-tiedostoja. Tiedostomuoto täytyy ensin muuttaa ensin Unityn tukemaksi fbx-tiedostomuotoon. Tämä aiheuttaa esimerkiksi sen, että Blenderissä tehty teksturointi ei toimi Unityssä ilman UV-kartoitusta. (The market-leading import pipeline 2014) (Smooth Blender -> Unity workflow 2014)

9 PELIN TOTEUTUS

9.1 Ohjelmointi

Pelissä käytimme ohjelmointikielenä Unityn JavaScriptiä. Päädyimme tähän kieleen, koska se on meille jo entuudestaan tuttu. Pelin toiminnallisuus on jaettu useaan eri JavaScript-tiedostoon. Näitä toiminnallisuuksia ovat esimerkiksi tornien suorittama ampuminen, suojakentän päälle meneminen, päivitysvalikon esille tuleminen ja pelaajan tilan tarkastaminen. Tässä kappaleessa keskitymme kertomaan pelaajalle selvimmin näkyvät pelin ohjelmoidut toiminnallisuudet.

9.1.1 Pelaajan tila

Yksi pelin kannalta tärkeimmistä toiminnallisuuksista on pelaajan tilan tarkastelu. Tämän olemme toteuttaneet JavaScript-tiedostoon nimeltä PlayerStats.js. PlayerStats-tiedosto sisältää pääasiassa funktioita, joita muut JavaScript-tiedostot voivat kutsua. Esimerkiksi kuvassa 17 päivitysvalikon JavaScript-tiedosto voi kutsua PlayerStats-tiedoston AddMaxHP-funktiota, joka kasvattaa pelaajan maksimi elämäpisteitä.

```
if(GUI.Button( Rect (38,250,180,45), "HP:ta 100gold")){
    GameObject.FindGameObjectWithTag("Player").SendMessage("AddMaxHP");
}
```

Kuva 17: Päivitysvalikko kutsuu AddMaxHP-funktiota

PlayerStats-tiedoston päätarkoitus on päivittää pelaajan elämäpisteitä ja rahatilannetta. Pelaajan ottaessa vahinkoa, pelaajalle vahinkoa tuottava elementti kutsuu PlayerStats-tiedostosta sille määriteltyä funktiota. Esimerkiksi, jos konekivääri torni ampuu pelaajaa, PlayerStats-tiedosto kutsuu TakeBulletDamage-funktiota. Tämä funktio vähentää pelaajalta ennaltamääritellyn määrän elämäpisteitä. Jokaiselle vahinkoa tuottavalle elementille on määritelty omat funktiot.

Peliin on sijoitettu näkymättömiä rahapisteitä ennaltamääriteltyihin kohtiin, joista pelaaja saa käyttöönsä rahaa hahmon päivittämistä varten. Kun pelaaja ohittaa tällaisen pisteen, piste kutsuu PlayerStats-tiedoston AddMoney-funktiota. Pelaajan saatua rahaa rahapiste deaktivoituu, eikä pelaaja voi enää saada siitä rahaa. Pelaajan herättyä henkiin rahapisteet aktivoituvat uudelleen ja niistä voi saada rahaa.

Pelin kenttään on sijoitettu tallennuspisteitä, joista pelaaja voi jatkaa kuoleman jälkeen mikäli pelaaja on päässyt tarpeeksi pitkälle. Kun pelaaja ohittaa tällaisen pisteen rahapiste kutsuu Checkpoint_X-funktiota PlayerStats-tiedostosta. X on tässä tallennuspisteen järjestysnumero, ja jos tämä on esimerkiksi 1, pelaaja on ohittanut ensimmäisen tallennuspisteen (Kuva 18).

```
function OnTriggerEnter (col : Collider){
    PlayerPrefs.SetInt("Checkpoint", 1);

}
```

Kuva 18: Tallennuspisteen järjestysnumero asetetaan

Mikäli pelaaja saa vahinkoa sen verran, että hänen elämäpisteensä vähentyvät nollaksi, pelaaja kuolee ja vahinkoa antava funktio kutsuu Dead-funktiota. Dead-funktio lähettää päivitysvalikolle viestin pelaajan kuolemasta. Päivitysvalikko käsittelee pelissämme pelaajan kuolemistilanteen. PlayerStats-tiedosto sisältää myös Alive-funktion, jota päivitysvalikko voi kutsua kun pelaaja herää eloon. Se antaa pelaajalle täydet elämäpisteet ja siirtää pelaajan tallennuspisteelle, joka on viimeksi aktivoitu.

9.1.2 Päivitysvalikko

Pelin päivitysvalikko sisältää pelaajan kuolemistapahtuman käsittelyn, päivitysvalikon esille tulemisen ja taukotilanteen käsittelemisen. Päivitysvalikko on kirjoitettu UpgradeMenu.js-tiedostoon. Kun

pelaaja kuolee, päivitysvalikko saa tästä viestin. Viestin saatua päivitysvalikko laittaa pelin taukotilaan ja ottaa pelaajalta pois käytöstä liikkumista hallitsevat komponentit. Tämän jäkeen ruudulle tulee näkyville päivitysvalikko. Valikosta löytyy painikkeet pelin jatkamiselle, pelaajan elämäpisteiden päivittämiselle ja pelaajan suojakentän elämäpisteiden päivittämiselle. Pelaajan painaessa jatka-painiketta peli siirtyy pois taukotilasta ja pelaaja herää eloon. Pelin voi laittaa taukotilaan painamalla esc-näppäintä. Tällöin näkyville tulee taukotilan valikko, jossa on painike pelin jatkamiselle.

9.1.3 Pelaajan suojakenttä

Pelaajan suojakenttää hallitseva Shield.js-tiedosto on hyvin samanlainen kuin PlayerStats-tiedosto. Shield-tiedosto tarkastelee, milloin pelaaja on laittanut suojakentän päälle, milloin suojakenttä saa vahinkoa ja milloin pelaaja päivittää suojakentän maksimi elämäpisteitä.

Pelissämme suojakentän saa päälle painamalla hiiren ykköspainiketta. Tällöin kutsutaan Shield-tiedoston ShieldUp-funktiota (Kuva 19). Suojakentälle annetaan tässä täydet elämäpisteet ja ne laitetaan näkyville. Mikäli suojakentän elämäpisteet menevät nollille, kutsutaan ShieldDown-funktiota (Kuva 19). Suojakenttä menee pois päältä ja tämän jälkeen pelaajan täytyy odottaa viisi sekuntia ennen kuin hän voi uudestaan käyttää suojakenttää.

```
function ShieldDown (){
    GameObject.FindGameObjectWithTag("Shield").GetComponent(Renderer).enabled = false;
    GameObject.FindGameObjectWithTag("Shield").GetComponent(Collider).enabled = false;
    GameObject.FindGameObjectWithTag("Player_lazor").GetComponent(Collider).enabled = true;
    Timer = startTime;
}

function ShieldUp (){
    shieldHp = maxShieldHP;
    GameObject.FindGameObjectWithTag("Shield").GetComponent(Renderer).enabled = true;
    GameObject.FindGameObjectWithTag("Shield").GetComponent(Collider).enabled = true;
    GameObject.FindGameObjectWithTag("Player_lazor").GetComponent(Collider).enabled = false;
}
```

Kuva 19: ShieldDown ja ShieldUp funktiot

9.1.4 Tornit

Pelissämme on käytössä kahdenlaisia torneja: konekivääritorneja ja ja ohjustorneja. Kummallekkin tornityypille on kirjoitettu oma JavaScript-tiedosto. Tästä huolimatta tornien ohjelmoinnit ovat hyvin samanlaisia. Molemmilla torneilla on myös käytössä Unityn valmiista asseteista käytössä SmoothLookAt.js-tiedosto. Tämä tiedosto saa tornin kääntymään pelaajaa kohti.

Konekivääritornilla on käytössä collider-komponentti, joka tarkastelee onko pelaaja tarpeeksi lähellä tornia. Kun pelaaja on tarpeeksi lähellä, tornin Shoot_MGun.js-tiedosto menee päälle ja tällöin torni alkaa ampua pelaajaa (Kuva 20). Pelaajan poistuessa tornin ampumisalueelta Shoot_MGun-tiedosto laitetaan pois päältä (Kuva 20). Tornin ampuessa pelaajaa torni lähettää PlayerStats-tiedostolle viestin siitä, että pelaajan täytyy ottaa vahinkoa. Mikäli pelaajalla on suojakenttä käytössä, torni lähettää viestin Shield-tiedostolle PlayerStats-tiedoston sijasta. Molemmille torneille on myös määritelty niiden omissa kooditiedostoissa aikaraja, jonka kulumisen jälkeen torni voi ampua uudestaan.

```
function OnTriggerEnter(col : Collider){

    if(col.gameObject.tag == "Player"){
        gameObject.GetComponent(Shoot_MGun).enabled = true;
        gameObject.GetComponent(SmoothLookAt).enabled = true;
    }
}

function OnTriggerExit (col : Collider) {
    if(col.gameObject.tag == "Player"){
        gameObject.GetComponent(Shoot_MGun).enabled = false;
        gameObject.GetComponent(SmoothLookAt).enabled = false;
    }
}
```

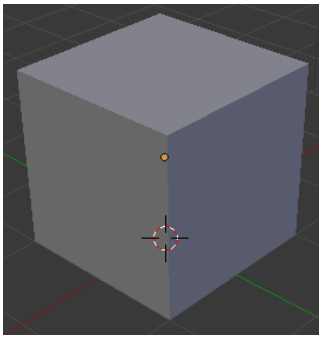
Kuva 20: Tornin toiminnallisuus laitetaan päälle tai pois

Ohjustorni eroaa konekivääritornista siten, että se ampuu fyysisen ohjuksen pelaajaa kohti. Laukaisun jälkeen ohjus hakeutuu automaattisesti pelaajaa kohti. Ohjuksen osuessa pelaajaan tai suojakenttään se lähettää vahingonotto viestin pelaajalle tai suojakentälle.

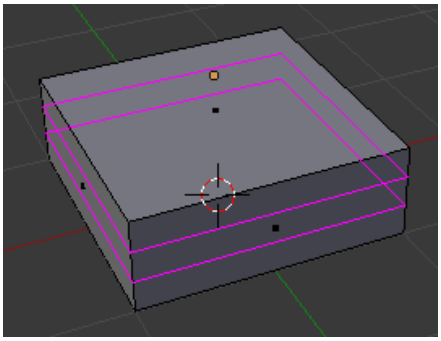
9.2 3D-mallit

3D-mallit tehtiin Blenderiä käyttäen ja peli tuli sisältämään useampia eri 3D-malleja. Näitä 3D-malleja ovat ohjustorni, konekivääritorni, ohjus, tasohyppelyalusta ja seiniin tulevien laserien alusta. Peliin tehtiin myös suojamuureja, mutta ne päätettiin jättää kokonaan pois.

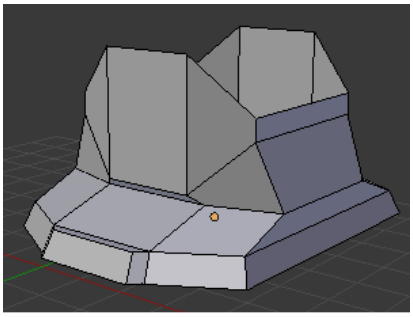
Konekivääritorni tehtiin ensimmäisenä. Blenderin avauduttua mallin tekeminen aloitettiin luomalla Cube-niminen objekti. Eli siis kuution muotoinen objekti (Kuva 21). Tämän jälkeen vaihdettiin esinetila muokkaustilaksi ja alettiin muokata kuution muotoa. Valittiin pelkästään kuution yläosa (face) ja painettiin kuutio matalaksi. Sitten käytettiin muokkaustilan loop cut and slice-ominaisuutta, jolla voitiin jakaa kuutio useampaan eri muokattavaan osaan (Kuva 22). Kun objekti oli jaettu osiin, sitä voitiin alkaa muokkaamaan venyttämällä objektin verteksejä, edgejä tai faceja. Ensin luotiin tornin pohja halutun näköiseksi (Kuva 23) ja sitten objektia rakennettiin pala palalta ylöspäin. Lisää faceja saatiin käyttämällä extrude-toimintoa.



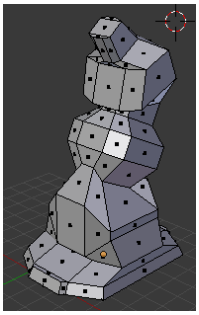
Kuva 21: Cube



Kuva 22: Loop Cut and Slice



Kuva 23: Pohja



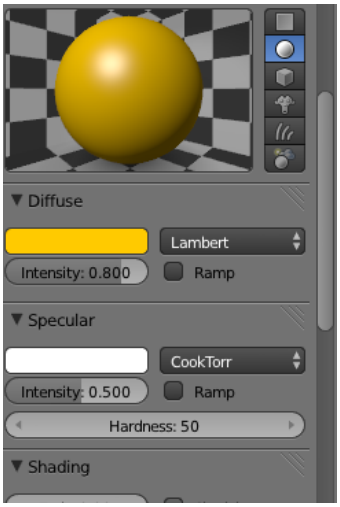
Kuva 24: Runko

Tornin rungon muodostuttua (Kuva 24) tehtiin sille hieman yksityiskohtia tekemällä uusi objekti esinetilassa. Yksityiskohdat tehtiin cylinder-objektilla. Sylinterin muotoista objektia muokattiin muokkaustilassa samalla tavalla, eli käyttäen loop cut and slice ja extrudea. Yksityiskohtien valmistuttua ne liitettiin tornin runkoon kiinni. Ensin liikutettiin yksityiskohdat oikeaan kohtaan

rungossa, sitten valittiin molemmat objektit ja painettiin lopulta join-toimintoa. Tämä liittää kaksi objektia yhdeksi kokonaisuudeksi.

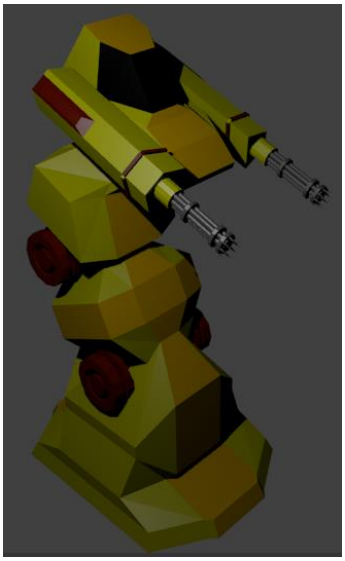
Lopuksi tehtiin vielä tornin ylin osa, eli ohjaamo. Tämä muokattiin useammasta kuutio-objektista. Itse tornin konekiväärit tehtiin useammasta eri sylinteristä. Hieman yksityiskohtia lisättiin konekiivääriin muutamalla cone-objektilla eli kartioilla. Konekiväärit liitettiin osaksi tornin ohjaamoa. 3D-malli jätettiin koostumaan kahdesta osasta, jotka ovat tornin runko ja tornin ohjaamo. Syynä tähän on se, että itse pelissä tornin rungon on pysyttävä paikallaan ja tornin ohjaamon on seurattava pelaajan liikkeitä.

3D-mallin perusvärit korvattiin lisäämällä uusia värejä material-osiosta. Uuden materiaalin lisäämisen jälkeen sille valittiin haluttu väri diffuse-valikosta ja tehtiin muita haluttuja muutoksia, kuten materiaalin heijastumiseen liittyviä muutoksia. (Kuva 25) Eri värejä saatiin tekemällä useampia eri materiaaleja. Samaan objektiin saadaan useita materiaaleja yhtä aikaa käyttämällä muokkaustilaa. Tällöin pitää ensin valita objektin haluttu face, valita haluttu materiaali ja lopulta painaa assign-painiketta. (Kuva 26) 3D-malli on nyt valmis. Lopputuloksen voi katsoa asettamalla kamera-objektin haluttuun paikkaan, säätämällä valoitusta ja lopuksi valitsemalla render-toiminto (Kuva 27).



Kuva 25: Material

Kuva 26: Assign

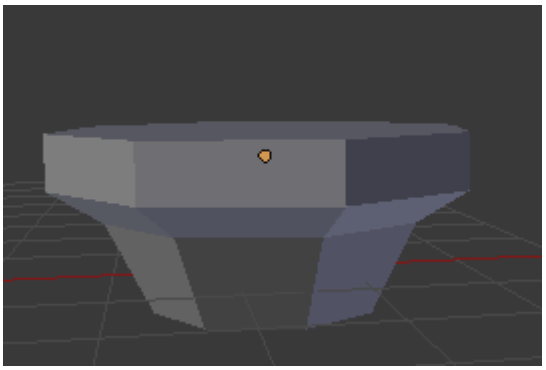


Kuva 27: Konekivääritorni

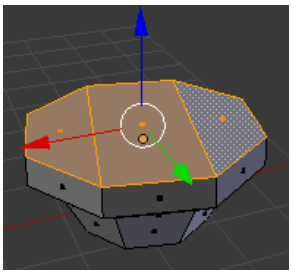
Muut 3D-mallit tehtiin samankaltaisesti, eli luomalla aloitusobjekti ja muokkaamalla sitä muokkaustilassa. Objekteja jaettiin useampiin muokattaviin palasiin ja nämä muokattiin halutun näköiseksi. Monesti useampi objekti liitettiin yhdeksi kokonaisuudeksi, mutta voi käyttää halutessa vain yhtä objektia.

Pelissä käytettävät liikkuvat tasohyppelyalustat muokattiin normaaliin tapaan muokkaustilassa (Kuva 28). Alustoja ei kuitenkaan saatu näyttämään tarpeeksi hyvältä pelkillä materiaaleilla. Tästä syystä päädyimme käyttämään UV-kartoitusta. Ensin valittiin objektista alue, joka haluttiin teksturoida UV-kartoituksella (Kuva 29). Seuraavaksi valittiin UV-kartoitusvalikko painamalla U-näppäintä (Kuva 30).

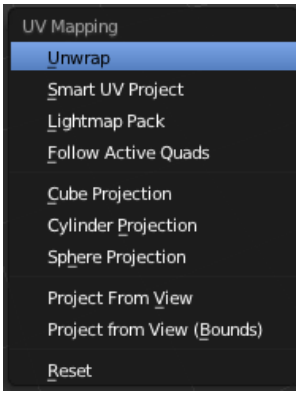
Sitten valittiin unwrap-toiminto, joka tekee valitusta alueesta kuvan, jota voitiin käyttää UV/Image Editorissa. Sitten vaihdettiin 3D-näkymä pois ja siirryttiin UV/Imgae Editoriin. Tämän jälkeen tarvittiin bittikarttakuva, jota aiotaan käyttää teksturoimiseen. Se valittiin painamalla Image-valikkoa ja sitten valitsemalla Open image. Nyt voitiin valita tallennettu bittikarttakuva ja pääsimme asettelemaan UV-kartoitettua aluetta siihen (Kuva 31). Säädettiin valittu alue haluttuun kohtaan ja käytettiin myös valitun alueen skaalausta. Skaalauksella pystyy säätämään suhdetta siitä miten suurena bittikarttakuva tulee objektin pinnalle. Objektin haluttu alue on nyt teksturoitu UV-kartoituksella. Tuloksen voi nähdä vaihtamalla näkymän varjostuksen tyyppiä. Näkymän varjostuksen tyyppi on yleensä solid, eli kiinteä. Vaihdettiin solid pois ja valitaan tilalle texture. Nyt objektin pinnalla voi huomata tehdyn UV-kartoituksen, vaikka ei ole käytetty render-toimintoa (Kuva 32).



Kuva 28: Alustan muoto

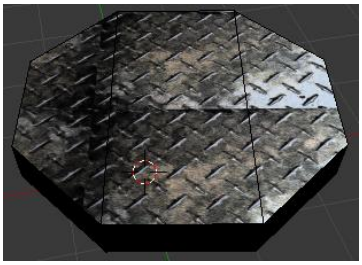


Kuva 29: Valittu alue



Kuva 30: UV Unwrap

Kuva 31: UV-kartoitus



Kuva 32: Valmis malli

9.3 Kentän toteutus

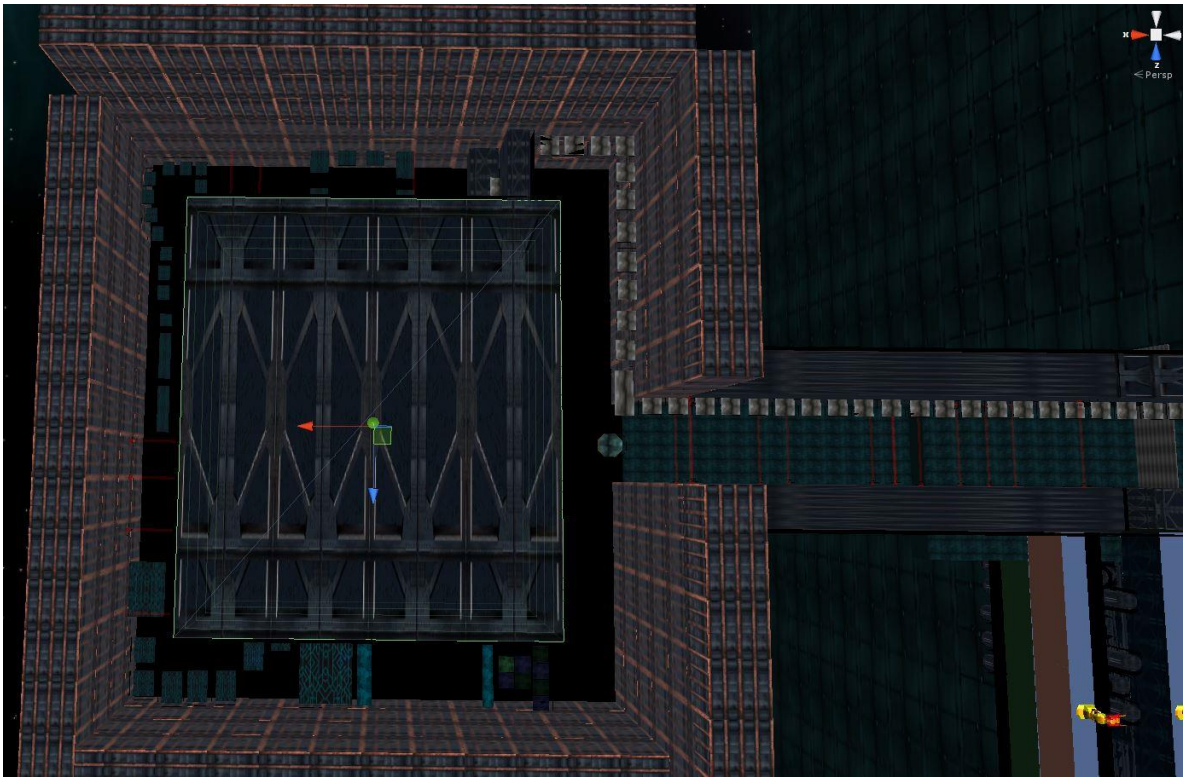
Aloitimme kentän tekemisen luomalla kentälle uuden scenen. Kentälle teimme ensiksi seinät ja lattian käyttämällä Unityn valmiita cube-objekteja. Seuraavaksi toimme kenttään ensimmäisen persoonan pelihahmon ja säädimme kentän mittasuhteet sopiviksi pelaajalle. Pelaajaan liitimme kaikki tarvittavat skriptitiedostot. Toimme tornien ja muiden käytettävien objektien 3D-mallit kenttään ja sijoittelimme

ensimmäiset tornit seinien päälle kummallekkin puolelle. Torneihin liitimme ampumis– ja pelaajanseurantaskriptit. Tässä vaiheessa toimme ensimmäiset tekstuurit kentään ja liitimme ne 3D-malleihin.

Tasohyppelyelementtejä aloimme tehdä jättämällä lattiaan rotkoja. Seuraavaksi sijoittelimme alustoja pelaajan käytettäväksi. Kentän keskiosan kaksi alustaa animoimme liikkumaan eteen ja taakse, jotta pelaaja pääsee viimeiseen osioon. Animoimme kentän ensimmäisen osion lopussa olevat alustat liikkumaan ylös ja alas eri tahdissa toisiinsa nähden. Tasohyppelyelementtien jälkeen sijoittelimme kentän ensimmäiseen osioon rahaa antavat näkymättömät seinät ja sijoittelimme loput torneista. Osion loppuun sijoitimme ensimmäisen välipisteen, josta pelaaja voi jatkaa kuoltuaan. Sijoitimme lisäksi rotkoihin partikkeliefektit. Ensimmäinen osio on nyt valmiina (Kuva 33).

Kuva 33: Kentän ensimmäinen osio

Kentän toinen osio (Kuva 34) aloitettiin samalla tavalla kuin ensimmäinenkin. Teimme siihen aluksi seinät ja lattian, minkä jälkeen toimme siihen siinä käytettävät 3D-mallit. Torneja emme käyttäneet ollenkaan, koska halusimme painottaa enemmän tasohyppelyä toisessa osiossa. Toisen osion alkuun sijoitimme lasereita, jotka animoimme liikkumaan eri tavoin. Lattiaa emme käyttäneet alun jälkeen, vaan käytimme erilaisia alustoja, joista osa animoitiin liikkumaan. Tässä vaiheessa aloimme tuoda peliin musiikkia ja toimme loput partikkeliefektit kenttään. Kolmannen osion alkuun sijoitimme toisen välipisteen.



Kuva 34: Kentän toinen osio

Kolmannessa osiossa (Kuva 35) emme käyttäneet enään seiniä, koska halusimme tehdä siitä avoimemman. Tässä osiossa emme käyttäneet lasereita, mutta otimme tornit uudestaan käyttöön. Kolmas osio sisältää paljon tasohyppelyä. Pyrimme tekemään kolmannesta osiosta haastavemman muihin verrattuna lisäämällä tornien määrää ja käyttämällä haastavia tasohyppelyelementtejä. Tasohyppelyelementit sisältävät paljon rotkoja ja liikkuvia alustoja. Esimerkiksi yksi alusta on animoitu pyörimään itsensä ympäri ja sen päälle päästäkseen pelaajan täytyy ajoittaa hyppynsä hyvin. Kolmannen osion loppuun sijoitimme savuseinämän, johon peli päättyy.

Kuva 35: Kentän kolmas osio

Lopuksi kentän valmistuttua aloimme testata peliä ja teimme pelille alkuvalikon. Aloimme säätämään tornien antamaa vahinkoa sopivaksi, jotta pelin vaikeustaso pysyisi sopivana. Säädimme lisäksi alustojen nopeutta tasohyppelyn helpottamiseksi. Pelaajalta säädimme esimerkiksi painovoimaa ja nopeutta säätteleviä arvoja. Alkuvalikkoon lisäsimme taustakuvan ja painikkeen pelin aloittamiselle.

10 POHDINTA

Tavoitteena oli toteuttaa videopeli PC:lle Unityä ja Blenderiä käyttäen. Toinen tekijöistä keskittyi ohjelmointiin ja toinen pelin visuaaliseen ulkonäköön. Opinnäytetyön tekemiseksi meillä oli aikaa noin viisi kuukautta. Jouduimme luopumaan joistain peliin tulevista ideoista ajan puutteen takia.

Peli saatiin lähes täysin valmiiksi, vaikka näytti siltä, että aika loppuu kesken. Pelissä saattaa olla vielä bugeja, koska pelin testaaminen jäi hyvin vähäiseksi. Meillä oli paljon muita koulutöitä opinnäytetyön ohella ja tästä syystä varsinkin pelin testaaminen jäi vähäiselle. Tällä hetkellä pelaaja pystyy menemään kentän läpi onnistuneesti, vaikka joutuukin päivittämään hahmoaan useampaan otteeseen. Vaikka saimme pelin valmiiksi, huomasimme pelin olevan hieman erilainen kuin alussa suunnittelimme. Esimerkiksi peli ei ole niin nopeatempoinen kuin suunnittelimme. Tämä olisi ehkä voitu välttää paremmalla kentän suunnittelulla ja käyttämällä suunnitteluun enemmän aikaa.

3D-mallien tekeminen onnistui lähes ongelmitta, koska olemme tehneet paljon 3D-mallinnusta ja peliin tulevat objektit ovat suhteellisen helppo toteuttaa. Ohjelmoinnissa oli ongelmatilanteita useasti, kun jokin skripti ei jostain syystä toiminut tai skripti suoritti vääriä asioita. Yleensä skriptien toimimattomuus johtui meidän omasta huolimattomuudesta eli skriptiä ei oltu liitetty oikeaan objektiin tai skripti oli kirjoitettu väärin. Esimerkiksi tornien ampumisen ohjelmoinnissa törmäsimme sellaiseen ongelmaan, että torni ampui omaa collideria. Sama toistui siinä, että tornit ampuivat toistensa collidereja eikä pelaaja tällöin saanut vahinkoa. Tämä ongelma korjattiin siirtämällä tornit eri layerille pelaajaan nähden.

Pelin visuaalisen ulkonäön tekeminen oli myös hidasta. Kokeilimme peliin paljon erilaisia tekstuureja. Tekstuurit oli valmistettu itse alusta lähtien tai sitten ne oli otettu Internetistä ilmaispalvelusta. Pelissä käytettiin paljon liikkuvia objekteja ja näiden objektien animointi vei todella paljon aikaa. Animaatioita ei voi kopioida objektista suoraan toiselle, vaikka animaation liike olisi samankaltaista. Tästä syystä jokainen animaatio jouduttiin tekemään erikseen. Peliin tulevat äänet löysimme onneksi nopeasti.

Peliä tehdessä opimme paljon lisää esimerkiksi ohjelmoinnista, sillä teimme paljon virheitä. Opimme myös sen, että selkeän aikataulun suunnittelu on erittäin tärkeää pelin teossa. Vielä tärkeämpää on aikataulussa pysyminen. Hyvä pelisuunnitelma ja kenttäsuunnitelma ovat myös erittäin tärkeitä. Jos suunnitelma on hyvin loppuun ajateltu, sen toteuttaminen on helppoa. Meillä suunnitteluvaihe jäi hieman vähäiseksi ja se myös näkyy pelistä. Esimerkiksi tekstuureja vaihdoimme useaan kertaan.

Raportti jäi ehkä hieman lyhyemmäksi mitä oli tarkoitus. Keskityimme enemmän pelin tekemiseen. Sen tekemiseen olimme enemmän motivoituneita kuin raportin kirjoittaminen. Pelin tekemiseen menee yleensä aina enemmän aikaa kuin siihen on varattu.

LÄHTEET

Blender Foundation 2014. About. Viitattu 6.4.2014,
http://www.blender.org/about/

Blender Foundation 2014. Features. Viitattu 6.4.2014,
http://www.blender.org/features/

Blender Foundation 2014. Textures common options. Viitattu 22.5.2014,
http://wiki.blender.org/index.php/Doc:2.6/Manual/Textures/Options

Dean, P. 6.7.2012. Endless Space Review. Viitattu 2.4.2014,
http://www.ign.com/articles/2012/07/06/endless-space-review

Future plc 14.7.2007. The 30 defining moments in gaming. Viitattu 7.3.2014,
http://www.edge-online.com/features/30-defining-moments-gaming/

Radoff, J. 24.5.2010. History of Social Games. Viitattu 6.3.2014,
http://radoff.com/blog/2010/05/24/history-social-games/

Silva, M. 15.8.2013. Grown Up Games. Viitattu 2.4.2014,
http://www.ign.com/articles/2013/08/15/gone-home-review

Toivio, T. 2012. Teksturointi. Viitattu 28.5.2014,
http://books.okf.fi/blender/teksturointi/#.U3zdK_l_t1A

Totilo, S. 2013. The Sequel To The Best Reverse-Tower-Defense Game Is Superb, If Barely a Sequel. Viitattu 30.5.2014,
http://kotaku.com/5972514/the-sequel-to-the-best-reverse-tower-defense-game-is-superb-if-barely-a-sequel

Unity Technologies. 2014. Customizing Your Workspace. Viitattu 2.4.2014,
http://docs.unity3d.com/Documentation/Manual/CustomizingYourWorkspace.html

Unity Technologies 2014. Creating Scenes. Viitattu 3.4.2014,
http://docs.unity3d.com/Documentation/Manual/CreatingScenes.html

Unity Technologies 2014. Creating and Using Scripts. Viitattu 3.4.2014,
http://docs.unity3d.com/Documentation/Manual/CreatingAndUsingScripts.html

Unity Technologies 2014. Download. Viitattu 3.4.2014,
https://unity3d.com/unity/download

Unity Technologies 2014. Game View. Viitattu 2.4.2014,
http://docs.unity3d.com/Documentation/Manual/GameView40.html

Unity Technologies 2014. Getting started with Mono Develop, Viitattu 2.4.2014,
http://docs.unity3d.com/Documentation/Manual/HOWTO-MonoDevelop.html

Unity Technologies. 2014. Inspector. Viitattu 3.4.2014,
http://docs.unity3d.com/Documentation/Manual/Inspector.html

Unity Technologies. 2014. Made with Unity. Viitattu 2.4.2014,
http://unity3d.com/showcase/gallery

Unity Technologies. 2014. License Comparisons. Viitattu 1.4.2014,
http://unity3d.com/unity/licenses

Unity Technologies 2014. Learning the Interface. Viitattu 2.4.2014,
http://docs.unity3d.com/Documentation/Manual/LearningtheInterface.html

Unity Technologies. 2014. Using the Scene View. Viitattu 3.4.2014,
http://docs.unity3d.com/Documentation/Manual/UsingTheSceneView43.html

Unity Technologies. 2014. Unity physics. Viitattu 3.4.2014,
http://unity3d.com/unity/quality/physics

Unity Technologies. 2014. Milestones. Viitattu 1.4.2014,
https://unity3d.com/company/public-relations

Unity Technologies 2014. Publishing Builds. Viitattu 3.4.2014,
http://docs.unity3d.com/Documentation/Manual/PublishingBuilds.html

Unity Technologies 2014. Smooth Blender -> Unity workflow. Viitattu 30.5.2014,
http://forum.unity3d.com/threads/109223-Smooth-Blender-gt-Unity-workflow

Unity Technologies 2014. Terrain Engine Overview. Viitattu 2.4.2014,
http://docs.unity3d.com/Documentation/Manual/Terrains.html

Unity Technologies 2014. The market-leading import pipeline. Viitattu 30.5.2014,
http://unity3d.com/unity/workflow/asset-workflow

Unity Technologies 2014. What is Unity. Viitattu 2.4.2014,
http://unity3d.com/pages/what-is-unity?gclid=CNOZ_JHLur4CFULecgodKbYAjg

Wikimedia Foundation. 2014. Atari Games. Viitattu 10.3.2014,
http://en.wikipedia.org/wiki/Atari_Games

Wikimedia Foundation. 2014. Blender (software). Viitattu 10.3.2014,
http://en.wikipedia.org/wiki/Blender_(software)

Wikimedia Foundation. 2014. Final Fantasy VII. Viitattu 10.3.2014,
http://en.wikipedia.org/wiki/Final_Fantasy_VII

Wikimedia Foundation. 2014. First-person shooter. Viitattu 30.5.2014,
http://en.wikipedia.org/wiki/First-person_shooter

Wikimedia Foundation. 2014. GNU General Public License. Viitattu 10.3.2014,
http://fi.wikipedia.org/wiki/GNU_GPL

Wikimedia Foundation. 2014. History of video games. Viitattu 6.3.2014,
http://en.wikipedia.org/wiki/History_of_video_games

Wikimedia Foundation. 2014. Kuvataajuus. Viitattu 23.5.2014,
http://fi.wikipedia.org/wiki/Kuvataajuus

Wikimedia Foundation. 2014. Platform_game. Viitattu 20.5.2014,
http://en.wikipedia.org/wiki/Platform_game

Wikimedia Foundation. 2014. Rampart (video game). Viitattu 10.3.2014,
http://en.wikipedia.org/wiki/Rampart_(arcade_game)

Wikimedia Foundation. 2014. Spacewar (video game). Viitattu 6.3.2014,
http://en.wikipedia.org/wiki/Spacewar_(video_game)

Wikimedia Foundation. 2014. Super_Mario_64. Viitattu 20.5.2014,
http://fi.wikipedia.org/wiki/Super_Mario_64#Pelaaminen

Wikimedia Foundation. 2014. Tasohyppely. Viitattu 20.5.2014,
http://fi.wikipedia.org/wiki/Tasohyppely

Wikimedia Foundation. 2014. Texture_mapping. Viitattu 21.5.2014,
http://en.wikipedia.org/wiki/Texture_mapping

Wikimedia Foundation. 2014. Tower defense. Viitattu 10.3.2014,
http://en.wikipedia.org/wiki/Tower_defense

Wikimedia Foundation. 2014. Tennis for Two. Viitattu 6.3.2014,
http://en.wikipedia.org/wiki/Tennis_for_Two