Saimaan ammattikorkeakoulu
Tekniikka Lappeenranta
Tietotekniikan koulutusohjelma
Tietojärjestelmien kehitys

Julian Parman ja Jussi Juuti

Mechanical Warfare

Opinnäytetyö 2015

Tiivistelmä

Julian Parman ja Jussi Juuti
Mechanical Warfare, 46 sivua
Saimaan ammattikorkeakoulu
Tekniikka Lappeenranta
Tietotekniikka
Tietojärjestelmien kehitys
Opinnäytetyö 2015
Ohjaajat: lehtori Mikko Huhtanen, Saimaan ammattikorkeakoulu

Tässä opinnäytetyössä luotiin tornipuolustuspeli Unityllä käyttäen 3D-malleja. Opinnäytetyön peli on demoversio. Peli on suunniteltu käyttäen suppeaa pelisuunnitelmaa.

Pelin 3D-mallit on luotu käyttäen Blender 3D-mallinnusohjelmaa. Pelin grafiikat on tuotettu GIMP2-kuvankäsittelyohjelmalla. Pelin viholliset ja tornit on toteutettu 3D-malleina. Pelissä on tarkoituksena puolustaa omaa tukikohtaa vihollisaalloilta, jotka tulevat tietä pitkin.

Lopputuloksena saatiin aikaan toimiva demoversio Mechanical Warfare tornipuolustuspelistä.

Asiasanat: 3D-malli, Blender, Gimp 2, grafiikka, Unity

Abstract

Julian Parman, Jussi Juuti
Mechanical Warfare, 46 pages
Saimaa University of Applied Sciences
Technology Lappeenranta
Degree Programme in Information Technology
Information System Development
Bachelor´s Thesis 2015
Instructor(s): Mikko Huhtanen, Saimaa University of Applied Sciences

The purpose of this thesis is to create a tower defense game using Unity and 3D models. The game of this thesis is a demo version. Game has been designed by using compact game design.

The game's 3D models are created using Blender 3D modelling software. The graphics are created using GIMP2- picture manipulation software. The game's enemies and towers are created as 3D-models. The purpose of the game is to defend your base from waves of enemies that approaches along the road.

The final result was functional demo version of the Mechanical Warfare tower defence game.

Keywords: 3D-model, Blender, Gimp 2, graphics, Unity

Sisältö

Termit ja käsitteet

3D-malli	Kolmiulotteinen graafinen malli.
3D-mallinnus	3D-mallien teko 3D-mallinnusohjelmalla.
3D-osoitin	Blenderissä siirrettävä piste x,y,z-koordinaatistossa.
Algoritmi	Tarkasti kuvattu toimintaohje, ongelman ratkaisuun tai tehtävän suoritukseen.
Animaatio	Erilaisten kuvien sarja, joilla luodaan illuusio liikkeestä.
A* Pahtfinding Project	Unityn lisäosa vihollisten reitin hakua varten.
Arraymodifier	Arraymodifier on Blender-työkalu, joka luo taulukon kopioidusta objektista ja sille poikkeaman haluttuun kohtaan.
Audacity	Ilmainen äänenmuokkausohjelma.
Boo	Oliopohjainen ohjelmointikieli.
Blender	Blender-säätiön tukema ilmainen 3D-mallinnusohjelma.
C#	Microsoftin luoma ohjelmointikieli.
Console-ikkuna	Unityn ikkuna, jossa näkyvät viestit, varoitukset ja virheet.
Demo	Esitysversio, johdatetaan sanasta demonstration.
DPS	Damage per second eli vahinko sekunnissa.
fbx-tiedosto	Autodeskin omistama tiedostomuoto.
FPS-peli	First Person Shooter, ensimmäisen persoonan räiskintäpeli.
Fysiikkamoottori	Pelimoottorin osa, joka mm. sisältää törmäyksen tunnistuksen ja putoamisen simuloinnin.
GIMP2	Ilmainen piirto-ohjelma.
Grafiikka	Pelin graafiset ominaisuudet.

Grafiikkamoottori	Pelimoottorin grafiikasta huolehtiva komponentti.
JavaScript	Netscape Communications Corporationin kehittämä oliopohjainen komentosarjakieli.
Koordinaatisto	Geometrinen järjestelmä alueen kuvaamiseen ja mittasuhteisiin.
Käyttöliittymä	Pelaaja pelaa peliä käyttöliittymän kautta.
Mobiilipeli	Puhelimen tai tablettietokoneeseen tarkoitettu peli.
Ohjelma	Tietokoneen ohjattavaksi tarkoitettu käskykokonaisuus.
Osumapiste	Mitataan vihollisen tai tornin kestävyyttä.
Parametri	Funktiolle välitettyä tietoa.
Pelikenttä	Pelin toiminnallinen alue, jossa voi liikkua tai pelata peliä.
Projektiili	Nopeasti lentävä kappale.
Pelimoottori	Pelin kehityksen runko, joka sisältää valmiita komponentteja.
Peliobjekti	Unityn objekti, muokattava objekti, johon voi laittaa skriptejä ja toimintoja.
Päivitysikkuna	Ikkuna, jossa voidaan päivittää torni, jos resursseja on tarpeeksi.
Realiaikainen strategiapeli	Reaaliajassa pyörivä strategiapeli.
Resurssi	Mechanical Warfaren rahayksikkö.
Satunnaislukugeneraattori	Algoritmi, joka arpoo satunnaisia lukuja annetuista maksimi- ja minimiarvoista.
Skripti	Ohjelmistokooditiedosto.
SphereCollider	Unityssä oleva pallon muotoinen törmäystunnistin.
Strategia	Tapa tai menetelmä saavuttaa haluttu tavoite.

Tasohyppely	Peli, jonka pelimekaniikka on tasolta tasolle hyppimistä
Tekstuuri	Kuva, jota käytetään peliobjekteissa ja 3D-malleissa.
Terrain assets	Unityn kehittämä ilmainen 3D-malli ja tekstuuripaketti.
Tukikohta	Pelaajan tärkeä rakennus, jota täytyy suojata vihollisilta.
Tornipuolustuspeli	Strategiapeligenre, jossa puolustetaan tietä ja estetään vihollisia pääsemästä kentän laidasta laitaa.
Unity	Unity Technologiesin kehittämä pelimoottori ja pelin kehitysympäristö.
Vuoropohjainen strategiapeli	Strategiapelimuoto, joka toimii vuoropohjaisesti, esimerkiksi shakki.

1 Johdanto

Peliala on kasvava ala, johon tarvitaan uusia innovatiivisia osaajia luomaan uusia ideoita ja tapoja. Suurimmat pelialan yritykset keskittyvät pääosin tietokone- ja konsolipelien tuotantoon, jotka ovat suosittuja. Kuitenkin nousevana alana ovat mobiilipelit, jotka ovat nykyisin todella suosittuja ja joita pelaavat kaikenikäiset. Mobiilipelejä on helpompi luoda, ja niitä tulee markkinoille jatkuvasti.

Tämän opinnäytetyön tavoitteena on suunnitella ja toteuttaa demoversio Mechanical Warfare -tornipuolustuspelistä. Pelistä tehdään myöhemmin kokoversio mobiilialustalle. Peli toteutetaan Unity-ohjelmalla ja 3D-mallit luodaan Blenderillä. Tässä opinnäytetyössä käsitellään Mechanical Warfare-pelin kehitystä, peliohjelmointia, käytettyjä 3D-malleja ja niiden mallintamisesta, käytettyjä ohjelmia ja Unityn lisäosia. Pelin äänet on muokattu Audacity-äänenmuokkausohjelmalla.

Koska Mechanical Warfare on demo, siihen ei saada aikaiseksi tiettyjä asioita, joita esimerkiksi kokoversiossa olisi hyvä olla. Demoversioon ei tule kunnollisia tuhoutumisanimaatioita, jossa vihollisyksiköiden tuhoutuminen on animoitu, tai visuaalisesti havaittavaa vaurioitumista. Uusia kenttiä ei ole tarjolla, vaan kaikki tapahtuu yhdellä kentällä. Vihollis- ja tornimalleja tulee olemaan vain neljä erilaista. Torneilla ja vihollisilla on vain yksi oma ääni käytössään.

2 Pelisuunnittelu

Tässä luvussa kerrotaan pelisuunnittelusta yleisesti ja Mechanical Warfare-demon pelisuunnittelusta. Aiheina on tasapaino, toteutus, pelimekaaniikka ja käytetty pelisuunnittelumalli.

2.1 Yleisesti

Suunnittelu on tärkeä osa projektia, koska siinä tuodaan esiin projektin tavoite, idea, toteutustapa, arkkitehtuuri, asiakas, määritelmät ja muita vastaavia tärkeitä käsitteitä. Kaikki projektit alkavat ideasta, joka muutetaan suunnitelmaksi. Suunnitelmassa tuodaan tärkeimmät asiat esiin: tavoite ja idea.

Suunnitelmassa tuodaan esille menetelmät ja keinot päästä tavoitteeseen ja kerrotaan, miten idea sai alkunsa. Suunnitelmat pitävät sisällänsä tarkat tiedot ja keinot, jotka välitetään eteenpäin kehittäjille, jotka alkavat toteuttaa projektia suunnitelman mukaan. Hyvin suunniteltu on puoliksi tehty.

Pelisuunnittelussa täytyy tuoda esille vähintään pelin ideat, pelimekaniikka, tasapainotus ja juoni. Peli-ideoita on helppo toteuttaa, mutta vaikea saada toimivaksi kokonaisuudeksi itse pelissä. Esimerkkinä tästä on miina, jonka voi laittaa Mechanical Warfaressa pelin kentälle tuomaan yksinkertainen hidaste vihollisille. Toteutukseen ei ollut aikaa ja miina jouduttiin karsimaan pois pelintoteutuksesta, mutta se jäi suunnitelmaan.

Pelimekaniikka on tärkeä osa peliä, koska siinä päätetään, millainen peli on pelattavuudeltansa ja toimivuudeltansa. Pelimekaniikka on suurelta osin pelin ydin ja siihen pohjautuvat ideat ja toiminnot. Pelimekaniikka on täten pelin luuranko, johon lisätään tarvittavat elementit: hahmot, grafiikka, tekstuurit, äänet ja kaikki muut vastaavat ulkoiset ja näkyvät seikat. Pelimekaniikan keskeisin pohja tulee aina riippumaan pelin pelilajityypistä.

Pelisuunnittelukokonaisuuden pitää olla hyvin koottu ja sitä jatketaan pelinkehityksen myötä ja joskus jopa pelin valmistumisen jälkeen. Tällöin peliä tasapainotetaan, jotta peli olisi tasapainoinen. Tasapainotus onkin pelikehityksen vaikeimpia käsitteitä. Esimerkiksi shakki on epätasapainoinen peli, vaikka kummallakin puolella on samat nappulat ja liikkeet. Shakin epätasapaino alkaa vuoroista. Ensimmäisenä siirtävällä on suurempi todennäköisyys voittaa tämän edun takia. Peli on vaikea saada tasapainoiseksi, koska siinä pitää ottaa huomioon pelaajan taidot ja pelin hahmot. Jos peliä tasapainotetaan pelaajien taitojen mukaan, huonommilla pelaajilla tulee olemaan vaikeampaa kuin kokeneilla.

Pelin juonen tulee olla myös tärkeässä asemassa, jos peli on esimerkiksi seikkailupeli. Tällöin taustatarina ja pelin juoni tulee olemaan keskeisemmässä asemassa toteutuksessa. Mechanical Warfare -pelin juoni ei ole keskeinen asia. Juonen tarkoitus on vain tuoda esiin, mitä pelaaja tekee ja miksi pelaaja tekee,

mitä hän tekee. Pelin taustatarinan ja juonen olemassaolo ei kuitenkaan muuta pelaamista tai pelattavuutta mitenkään.

2.2 Videopelilajityypit

Videopelejä on useita erilaisia, ja ne jaetaan pelin pelimekaniikan mukaan erilaisiin lajityyppeihin. Lajityyppejä ovat esimerkiksi strategiapelit, tasohyppelypelit ja ensimmäisen persoonan räiskintäpelit. Kaikille peleille löytyy vielä omat alalajit, jotka muuttavat pelimekaniikkaa hiukan mutta pysyvät silti samassa lajityypissä. Esimerkiksi reaaliaikaiset strategiapelit toimivat reaaliajassa, kun taas vuoropohjainen strategiapeli toimii vuoropohjaisesti. Molemmat kuuluvat strategiapelilajiin, mutta ovat mekaniikaltaan erilaiset.

Mechanical Warfare -pelilajityyppi on strategiapeli, ja se on alalajiltaan tornipuolustuspeli. Tornipuolustuspeli on pelimalli, jossa pelaajan tehtävänä on tuhota tiellä kulkevia vihollisia ja estää niitä pääsemästä kentän päästä päähän. Tuhotakseen viholliset pelaajan täytyy pystyttää torneja, jotka ampuvat vihollisia. Tuhouduttuaan vihollinen tuottaa resursseja, joilla parannetaan jo olemassa olevia torneja tai rakennetaan uusia. Torneja on yleensä useita erilaisia malleja, joista jokainen torni erikoistuu johonkin, esimerkiksi tietyt tornit hidastavat ja toiset ampuvat nopeasti.

2.3 Mechanical Warfare -pelin suunnittelu

Mechanical Warfare -peli on suunniteltu käyttäen suppeaa pelisuunnitelmaa (LudoCraft) laajan pelisuunnitelman sijaan, koska suunnittelijat toimivat myös kehittäjinä. Tästä syystä Mechanical Warfare ei tarvitse laajaa pelisuunnitelmaa. Suppeassa pelisuunnitelmassa käydään läpi kaikki olennaiset ja tärkeät asiat, joita tullaan tarvitsemaan pelin kehityksessä, kun taas pelisuunnitelmassa käydään läpi yksityiskohtaisesti kaikki aiheet ja asiat, koska toteuttajana toimii joku muu henkilö tai yritys.

Aluksi haettiin vastauksia seuraaviin kysymyksiin: mikä peli, pelin idea, hahmot, kohderyhmä, alusta ja pelin nimi. Kun nämä oli saatu sovittua, alettiin täyttää suppean pelisuunnitelman pohjaa.

3 Mechanical Warfare

Tässä luvussa käydään läpi pelin tavoite, joka pitää sisällänsä pelin sisällön, pelin juonen, joka pitää sisällänsä tarinan, viholliset ja tornit. Pelaaja taistelee tekoälyä vastaan, joka luo vihollisaaltoja.

3.1 Pelin tavoite

Pelin tavoitteena on antaa pelaajalle haastava pelikokemus, jossa on otettu huomioon onnekkuus ja taito. Vihollisaallot on luotu satunnaislukugeneraattorilla, joten vihollisaallot ovat kokonaisuudeltansa tuntemattomia. Mikään aalto ei ole samanlainen ja näin ollen uudelleen pelattavuus on mielenkiintoisempi. Pelin alkuvalikko on tehty siten, ettei se liity itse peliin mitenkään. Ajatuksena on katsoa, kun kaksi tornia ampuu palloa. Tornit eivät aina osu palloon, mutta sitä jääkin katsomaan. Pallo pomppii pitkin kenttää ja kanuunat ampuvat sen perään. Kun tykin projektiili osuu palloon, pallo lentää muualle saaden uutta nopeutta iskusta. Tämä ratkaisu on toimiva ja innovatiivinen.

3.2 Pelin juoni

Mechanical Warfaressa juoni ei ole tärkeässä asemassa, vaan täytteenä luomassa jonkinlaista taustaa sille, miksi pelissä tapahtuu, mitä siinä tapahtuu. Pelin juonessa vihollinen on P-valtio, joka hyökkää pelaajan yhdistynyttä valtiota vastaan. Vihollinen on laskenut joukot rantaan ja pyrkii tuhoamaan pelaajan tukikohdan ja joukot. Taustatarinassa EU on kaatunut ja kaikki maat ovat hajonneet useiksi pieniksi valtioiksi. Pelaaja toimii useiden valtioiden liitossa ja vastuksena on P-valtio, joka yrittää valloittaa maat ja liittää pelaajan valtiot P-valtioon. Sodan julistuksen syy oli valtioiden liittäminen P-valtioon.

3.3 Pelin loppuratkaisu

Pelaaja palkitaan pelisuorituksesta tähdillä, jos hän läpäisee kaikki kahdeksan vihollisaaltoa. Jos tukikohta ei ota yhtään vauriota, pelaaja palkitaan viidellä tähdellä. Jos tukikohta ottaa vauriota, pelaaja palkitaan sen mukaan, kuinka paljon osumapisteitä on jäljellä. Tähdillä ei voi tehdä mitään, ne vain kuvaavat

pelaajan onnistumista. Kun peli on loppu ja tähdet on saatu, pelaaja voi palata takaisin päävalikkoon tai pelata peli uudestaan läpi.

3.4 Viholliset

Viholliset on luotu Blender 3D-mallinnusohjelmalla. Jokainen malli on luotu fiktiivisesti käyttäen omaa mielikuvitusta, katsomalla mallia oikeista ajoneuvoista ja aseista. Koska peli sijoittuu tulevaisuuteen, ajoneuvojen rakenne poikkeaa muista olemassa olevista ajoneuvoista. Vihollisia on neljä kappaletta.

3.4.1 Taistelurobotti T-12

Robotti on kävelevä tietokoneohjattu vihollinen, joka on aseistettu kahdella konekiväärillä, kuten kuvassa 1 näkyy. Robotti kävelee hitaasti ja on kestävyydeltään heikoin vihollinen.

Kuva 1. Robotti

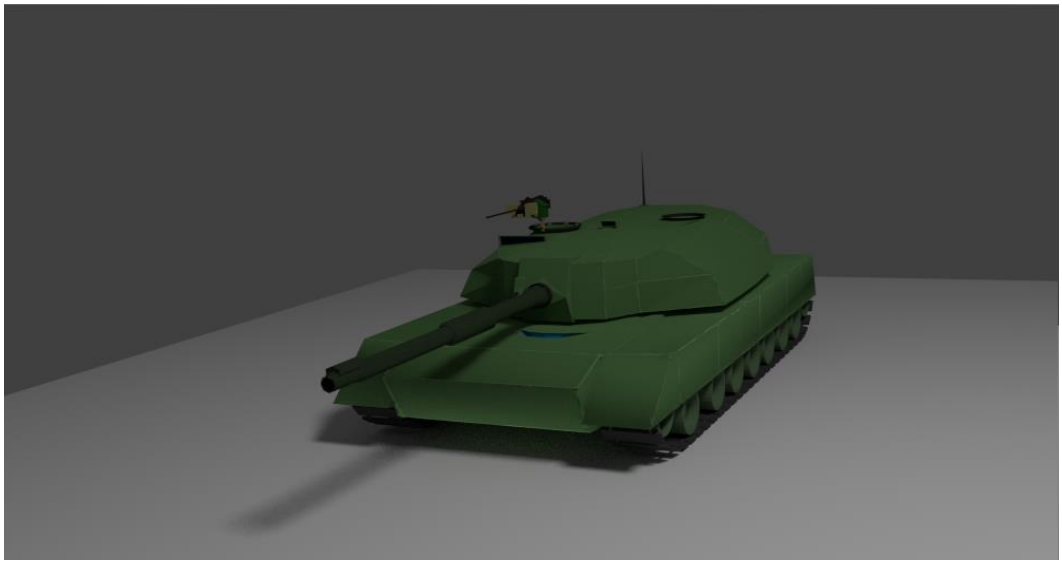
3.4.2 Maastoauto

Maastoauto on nopea auto, joka on aseistettu yhdellä konekiväärillä. Auto on panssaroitu kevyesti, joten se kestää jonkin verran luoteja, mutta ei räjähdyksiä tai raskaimpia tykkejä. Nopeutensa ansiosta maastoauto pääsee nopeasti pois tornien kantamalta ja kevyt panssari takaa kohtalaisen kestävyyden konekivääreitä vastaan. Kuvassa 2 näytetään, minkä näköinen maastoauto on.

Kuva 2. Maastoauto

3.4.3 Taistelupanssarivaunu TPV-A1

Taistelupanssarivaunu on raskaasti panssaroitu ja aseistettu ajoneuvo. A1 on aseistettu raskaalla tykillä, joka tekee siitä vaarallisen vihollisen kentällä. Sen paksu panssari kestää useita osumia ja tarvitsee raskasta kalustoa tuhoamaan sen, kuten kuvassa 3 näkyy.



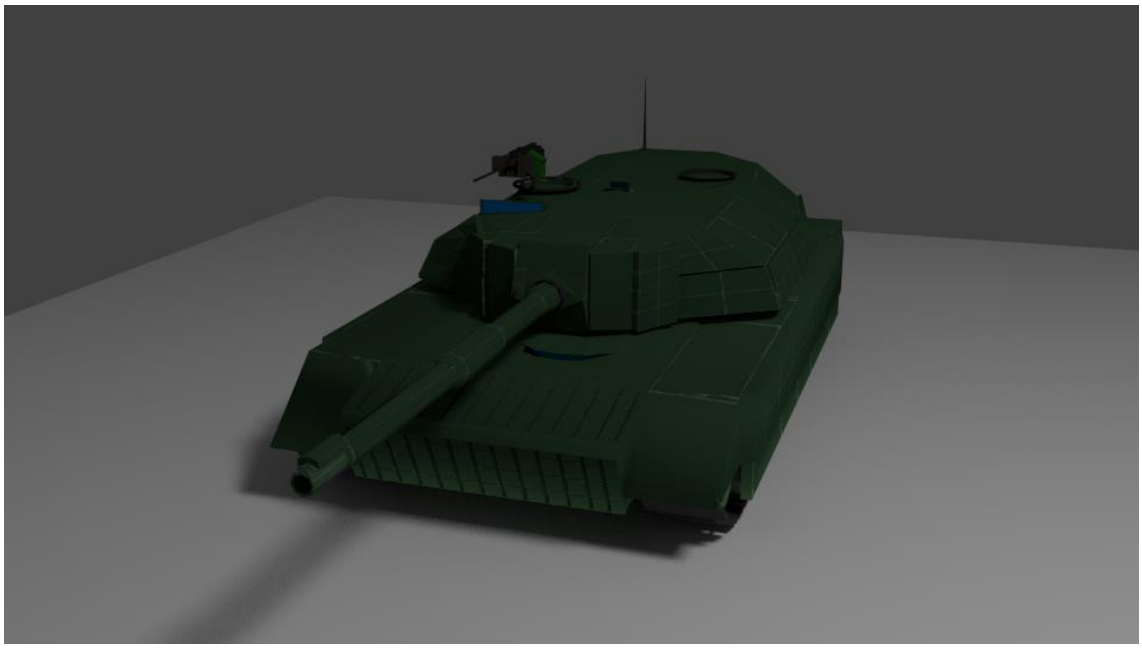
Kuva 3. TPV-A1

Panssarivaunujen tykit on animoitu Blenderillä. Animointi toteutettiin aikajanalla, eli aikajanan kohdassa 1 tykinmallia siirrettiin tankkiin sisäänpäin eri kohtaan. Kohtaan merkittiin kohta 2. Sitten tykkiä siirrettiin taas eteenpäin

aloitusasemaansa, eli kohta 3. Tämä loi yksinkertaisen animaation, jossa tykin putki pumppaa taaksepäin ja siirtyy sitten taas eteen. Samaa menetelmää käytettiin TPV-A2:ssa.

3.4.4 Raskastaistelupanssarivaunu TPV-A2

TPV-A2 on vielä raskaammin panssaroitu versio TPV-sarjasta, kuten kuvassa 4 näkyy. Siihen on lisätty reaktiivipanssaripaloja, jotka lisäävät sen kestävyyttä ja nostavat sen painoa. A2:nen on hitaampi versio TPV-sarjasta. TPV-A2:seen on päivitetty tähtäysjärjestelmä ja päivitetty tykki mahdollistaa uudempien ammusten käytön, tehden siitä tulivoimaisemman kuin A1:sestä. Raskas panssari ja päivitetty tykki tekevät A2:sesta hyvin kestävän ja vahinkoa tuottavan vihollisen.



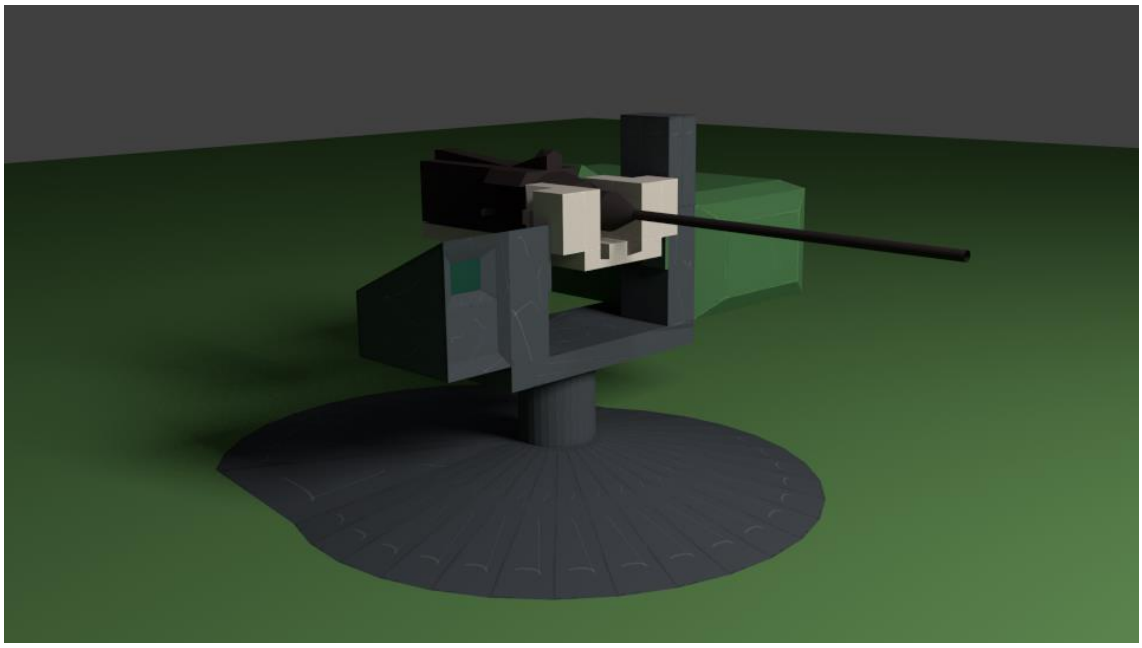
Kuva 4. TPV-A2

3.5 Tornit

Pelaajan tehtävänä on rakentaa puolustustorneja kentälle puolustamaan etumaastoa vihollisilta.

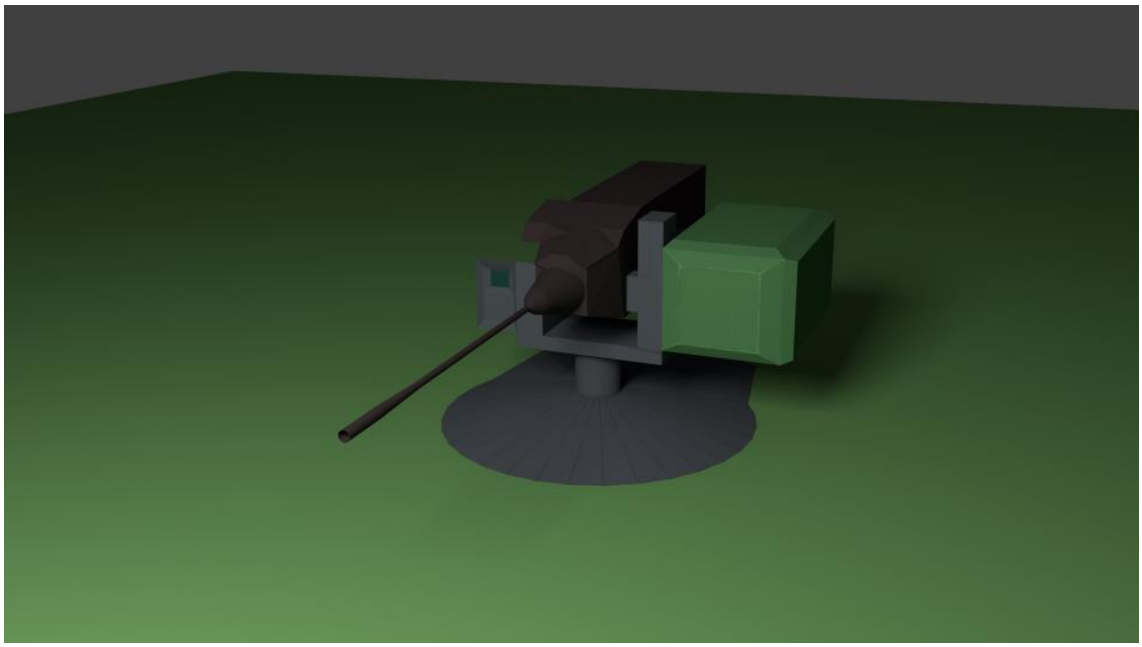
3.5.1 Konekivääritorni

Konekivääritorni on automatisoidulla alustalla toimiva sarjaa ampuva ase. Sen raskas kaliiperi tekee siitä vaarallisen kevyesti panssaroituja ja nopeita kohteita kohtaan. Sen etuna on jatkuva tulitus, joka hitaasti mutta varmasti tuhoaa sen kantamalla olevat kohteet. Päivitykset nostavat konekiväärin kaliiperia kone-tykiksi, joka taas ampuu raskaampia luoteja ja nostaa konekivääritornin tuhovoimaa. Kuitenkin raskaasti panssaroituja vastaan konekivääri ei tee mitään vahinkoa. Kuvassa 5 on Mk I:sen mallin konekivääri, joka on ensimmäinen ostettava konekivääri.



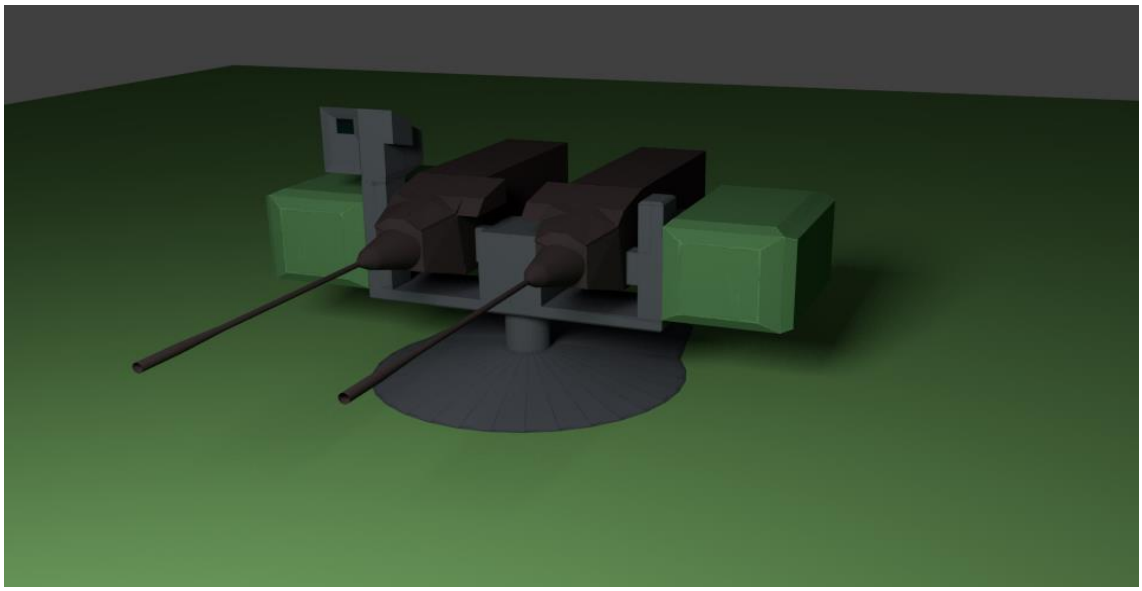
Kuva 5. Mk I konekivääri

Kuvassa 6 on Mk I ensimmäinen päivitys Mk II, joka on raskaampi ja ampuu suurempaa kaliiperia, kuin Mk I.



Kuva 6. Mk II konetykki

Mk II on toinen päivitys, kuvassa 7 on Mk III, joka on viimeisin päivitys konekiväärisarjassa.

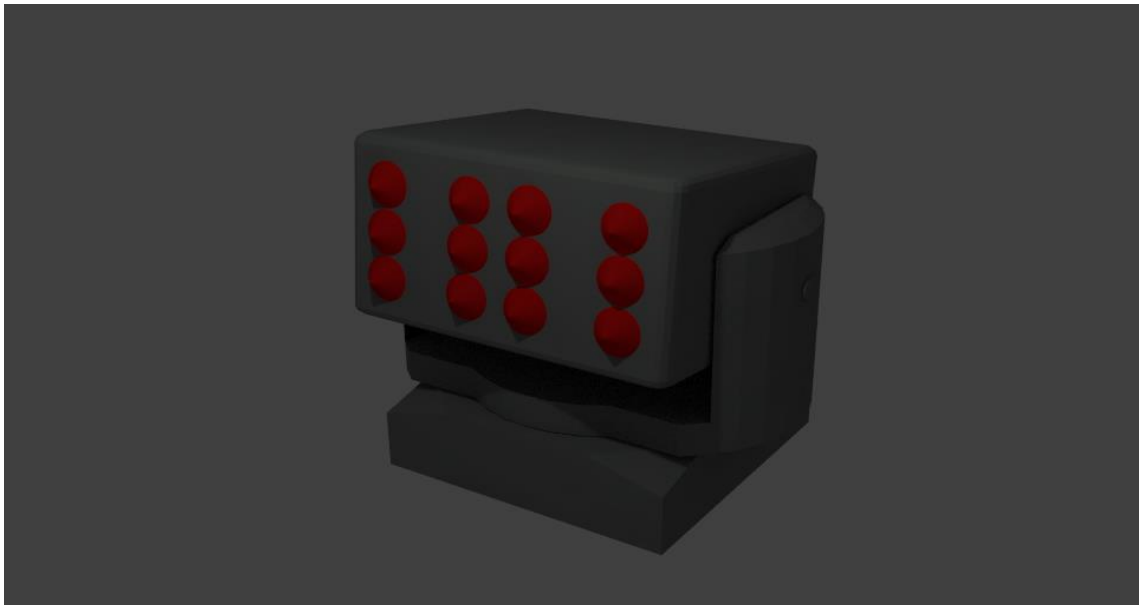


Kuva 7. Mk III konetykki

Mk III:sessa on kaksi Mk II konetykkiä yhdellä alustalla.

3.5.2 Ohjustorni

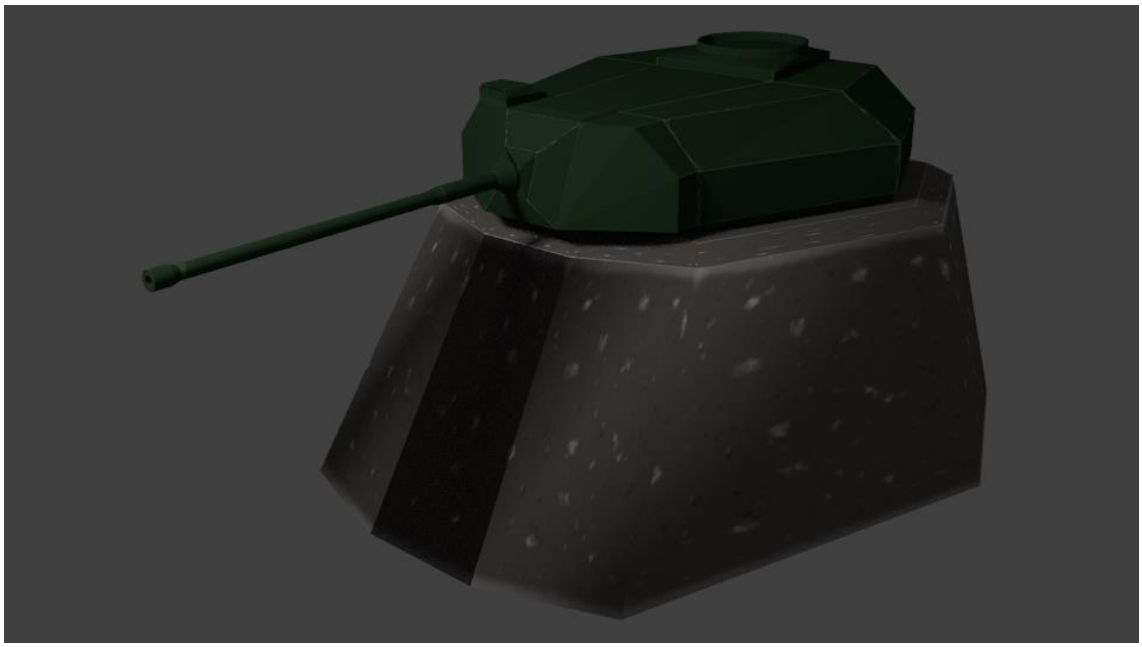
Ohjustorni laukaisee kaksi ohjusta vihollista kohti, kun ohjus osuu kohteeseen se räjähtää ja tekee tuhoa kohteeseen. Ohjuksessa kohtalainen vahinko, joten se toimii hyvin kaikkia vastaan, mutta on rakenteellisesti heikko. Päivitetyt versiot ampuvat enemmän ohjuksia ja kantama kasvaa. Kuvassa 8 Mk III ohjustorni.



Kuva 8. Ohjustorni

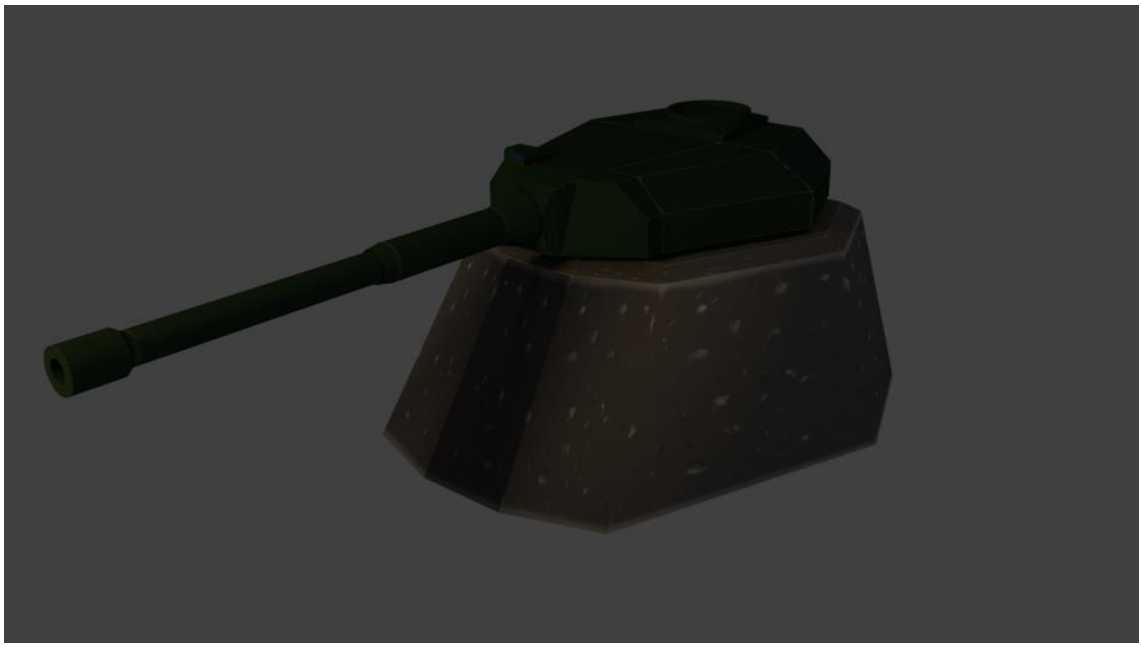
3.5.3 Tykkitorni

Tykkitornit ovat teräsbetonialustalla olevia taistelupanssarivaunun torneja. Tykkitornit ovat aseistettu erilaisilla tykeillä, jotka ampuvat panssarinläpäiseviä kranaatteja. Torni lukittuu kohteeseensa ja avaa tulen. Eri versiot parantavat tornin tehokkuutta ja kasvattavat tykin läpimittaa. Samalla tykkien vahinko ja kantama kasvavat. Parannukset kehittävät myös tornin kestävyyttä. Kuvassa 9 näkyy ensimmäinen tykki Mk I.



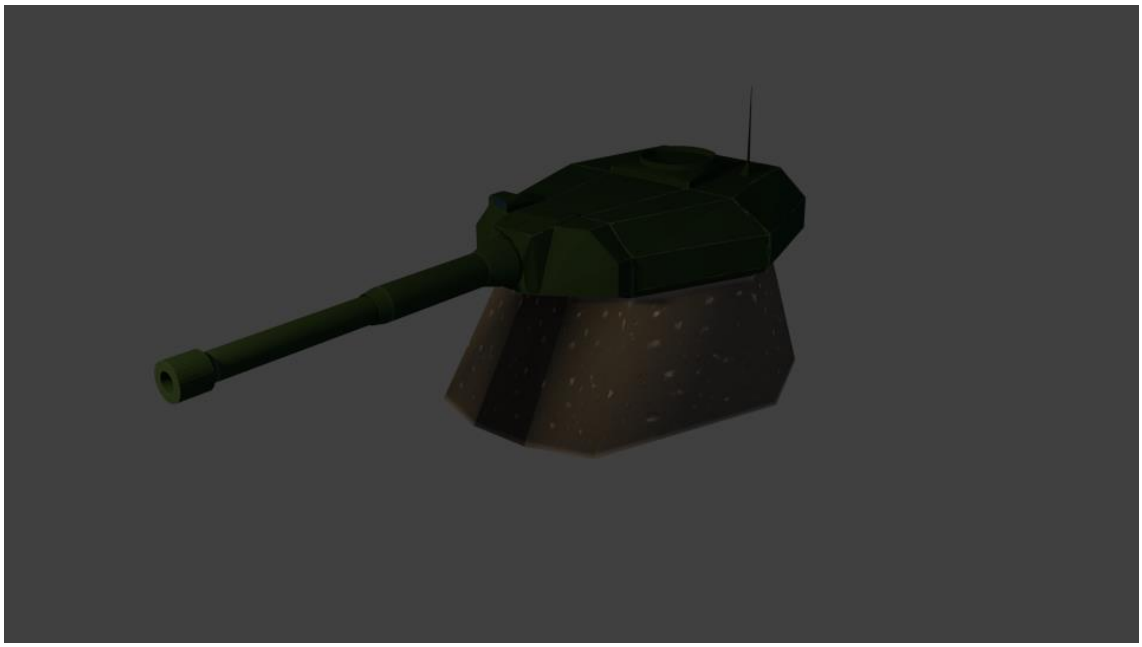
Kuva 9. Tykki Mk I

Mk I on parhaiten panssaroitu aloitustykki, jossa on nopea tulinopeus ja kohtalainen vahinko. Kuvassa 10 näkyy Mk II, Mk I:sen seuraava päivitys.



Kuva 10. Tykki Mk II

Tykki Mk II:sessa on suurempi kantama ja vahinko kuin Mk I:sessä mutta se ampuu hitaammin. Kuvassa 11 on viimeisin päivitys tykkisarjassa, eli Mk III.



Kuva 11. Tykki Mk III

Tykki Mk III:n kantama on suurimpia, mitä pelissä on, ja siinä on tehokas tykki. Se kuitenkin ampuu hitaasti verrattuna muihin tykkeihin suurten ammusten takia.

3.5.4 Kranaatinheitintorni

Kranaatinheitintorni ampuu kaaressa lentävän kranaatin vihollista kohti ja kranaatti räjähtää osuessaan maahan tai viholliseen. Kranaatti tekee tuhoa ympärilleen. Kranaatinheitin toimii parhaiten ryhmässä oleviin vihollisiin, jolloin mahdollisimman moni vihollinen vahingoittuu. Päivitetyt versiot ampuvat nopeammin ja kauemmas. Kuvassa 12 on Mk I kranaatinheitin.

Kuva 12. Kranaatinheitin

Torneilla on omat heikkoutensa ja vahvuutensa, jotka tasapainottavat peliä. Konekiväärit ovat halpoja ja heikkoja eivätkä ne vaurioita panssarivaunuja. Ohjustornit taas ampuvat ohjukset kerrallaan, mutta niissä on hidas lataus. Viholliset voivat tuhota tornit, mikä on pelaajalle riski. Jos pelaaja rakentaa useita heikompia torneja, esimerkiksi ohjustorneja, niin tornit voivat tuhoutua nopeasti vihollisaallon aikana.

4 Pelin kehitys

Tässä luvussa kerrotaan pelin kehityksestä yleisesti ja Mechanical Warfare pelin kehityksestä. Luvussa kerrotaan peli-ideasta, minimaalisista vaatimuksista, kehitysprosessista, tasapainoituksesta, ekonomiasta ja C#-skripteistä.

4.1 Yleisesti

Pelin kehitys aloitetaan ideasta, josta kirjoitetaan suunnitelma. Suunnitelmaan lisätään kaikki tarvittavat tiedot ja ideat, jotka kehitetään ja laitetaan peliin. Pelin idean täytyy olla selvillä ja samoin pelimekaniikka. Mechanical Warfare -peli on tornipuolustuspeli, joka on reaaliaikainen strategiapeli. Pelin mekaniikka siis tulee olemaan ylhäältä päin kuvattu strategiapeli.

4.2 Pelin idea

Pelin ideana oli tuottaa tornipuolustuspeli, jossa on ampuvia vihollisia. Vihollisyksiköt ampuvat pelaajan puolustustorneja ja tuhoavat tornit. Tämä tarkoittaa sitä, että pelaajan pitää suunnitella tornien laittaminen siten, että heikommat tornit eivät ole etulinjassa, ja vahvemmat tornit ottavat osumat vastaan. Toinen kriittinen idea oli se, että viholliset ilmestyvät sattumanvaraisesti algoritmin avulla. Algoritmi valitsee sattumanvaraisen vihollisen ja paikan. Tämän jälkeen vihollinen syntyy kentälle ja lähtee kävelemään tukikohtaa päin. Kolmas idea oli tukikohta, joka näkyy kuvassa 13 vasemmassa alanurkassa, johon tiet päättyvät. Jos tukikohta tuhoutuu, peli päättyy. Normaalisti tornipuolustuspeleissä viholliset marssivat kentän päästä päähän ja päästyään toiseen päähän ne katoavat ja vievät pelaajalta osumapisteitä. Mechanical Warfaressa viholliset jäävät kentälle ja ampuvat tukikohtaa ja lopulta tuhoavat sen. Pelaajan tehtävänä on estää vihollisia tuhoamasta tukikohtaa.

4.3 Pelin minimaaliset vaatimukset

Aivan aluksi piti suunnitella, mikä on tornipuolustuspelin minimaalisin vaatimustaso, ja siitä herää kysymys: Mikä tekee tornipuolustuspelistä tornipuolustuspelin? Kysymys kuulostaa yksinkertaiselta, mutta sitä se ei välttämättä ole. Tornipuolustuspeli pitää yksinkertaisimmillaan sisällänsä tien, jota pitkin viholliset kulkevat; tornin ostovalikon, mistä ostetaan tornit; viholliset, jotka liikkuvat tietä pitkin; tornit, jotka ampuvat vihollisia ja ekonomia, jossa pelaaja saa resursseja tuhotuista vihollisista, jotta pelaaja voi ostaa lisää torneja. Tässä on oletettu yksinkertaisuus koko tornipuolustuspelissä. Kun yksinkertaisuus oli selvänä, alettiin lisäämään täytettä pelimekaniikkaan. Pelin täytteenä ovat 3D-mallit, grafiikka, tekstuurit, äänet ja animaatiot.

4.4 Pelin kehitysprosessi

Peli toteutettiin käyttämällä ketterää ohjelmistokehitysmenetelmää. Kummallakin tekijällä oli oma peliobjekti, joka piti saada toimimaan. Esimerkiksi kumpikin tekijä toteutti samaan aikaan eri tornia. Testausta suoritettiin koko

ajan, jotta virheet löydettäisiin mahdollisimman aikaisin ja ne olisi helpompi korjata.

Mechanical Warfare -pelin kehitys aloitettiin tornien ja vihollisten 3D-mallien tekemisellä. Kehittäjillä ei ollut aikaisempaa 3D-mallintamiskokemusta, joten ensimmäisen mallin tekemiseen kului eniten aikaa, jotta oppi käyttämään Blenderin työkaluja. Loput 3D-mallit valmistuivat huomattavasti nopeammin. Kun mallit oli saatu valmiiksi, ne tallennettiin fbx-tiedostoiksi ja siirrettiin Unityn. Tässä vaiheessa oli aluksi ongelmia, koska mallit eivät menneet aivan oikein päin Unityn. Tämä johtui siitä, että Unityssä ja Blenderissä on erilaiset koordinaatistot. Ongelma ratkaistiin kääntämällä mallit Blenderissä siten, että z-akseli osoittaa eteenpäin, y-akseli ylöspäin ja x-akseli vasemmalle. Näin mallit saatiin oikein päin Unityssä.

Seuraavaksi alettiin tekemään pelikenttää, joka näkyy kuvassa 13. Kumpikin kehittäjä teki oman version, josta parempi valittiin. Kentällä kulkevaa tietä muutettiin useampaan kertaan, että tornien paikat saataisiin sijoitettua hyvin tien sivuille. Lopulta päädyttiin kahteen tiehen, jotka yhdistyvät hieman ennen tukikohtaa. Vihollisia tulee satunnaisesti kumpaakin tietä pitkin, mikä lisää vaikeustasoa ja pelin kiinnostavuutta. Viholliset piti saada liikkumaan tietä pitkin haluttuun kohteeseen eli tukikohdan luokse. Vihollisten liikuttaminen toteutettiin A* Pathfinding Project -lisäosalla, joka hakee nopeimman reitin pisteestä -A pisteeseen -B.

Kuva 13. Pelikenttä

Tornien on tarkoitus aloittaa ampumaan ensimmäistä vihollista, joka astuu tornin kantaman sisäpuolelle. Jos vihollinen menee kantaman ulkopuolelle, torni vaihtaa kohdetta. Torneja on neljä erilaista, joten jokaiselle piti ohjelmoida oma skripti. Tornien täytyi osata ottaa oikean verran ennakkoa, jotta ammukset eivät menisi joka kerta ohi. Ennakon laskukaava on (1) (Blasi 1997, 21). Konekivääritorni ja tykkitorni eivät eroa paljoa toisistaan, mutta konekivääritorni ampuu sarjan ammuksia ja tykkitorni ampuu vain yhden laukauksen. Molemmat käyttävät samaa ennakon laskua. Ohjustorni ei tarvitse ennakon laskua, koska ohjukset lukittuvat kohteeseen. Kranaatinheitin oli vaikein toteuttaa, koska siinä täytyi käyttää fysiikan heittoliikkeen kaavaa (2).

$$\frac{vihollisen\ nopeus}{ammuksen\ nopeus} * matka \quad (1)$$

$$\sqrt{\left(\frac{matka * vihollisen\ nopeus}{Sin(2 * ampumiskulma)}\right)} \quad (2)$$

Pelikentälle on sijoitettu ruutuja, joille pelaaja voi sijoitella torneja. Tornien hinnat näkyvät peliruudun alalaidassa sijaitsevien tornipainikkeen yläpuolella, kuten kuvassa 14 näkyy. Jos pelaajalla on tarpeeksi resursseja, hinta on vihreä, ja jos resursseja on liian vähän hinta näkyy punaisena. Kun pelaaja valitsee haluamansa tornin ja ruudun painamalla hiiren vasenta painiketta ruudun yläpuolella, testataan aluksi, onko ruutu vapaa. Jos ruutu on vapaa, torni voidaan sijoittaa ruutuun ja resursseista vähennetään tornin hinta, mutta jos ruudussa on jo toinen torni, niin valittua tornia ei voida sijoittaa samaan ruutuun. Ruuduissa olevia torneja pystyy myös päivittämään parempiin versioihin. Kun pelaaja painaa ruudussa olevaa tornia, niin ruudulle ilmestyy päivitysikkuna, jossa kysytään, haluaako pelaaja päivittää tornin parempaan versioon. Päivitysikkunassa on myös upgrade- ja cancel-painike, jos pelaajalla on tarvittava määrä resursseja upgrade-painike on aktiivinen ja sitä voi painaa, jolloin torni päivitetään. Pelaajalle ilmoitetaan päivitettävissä olevat tornit sinisellä nuolella, joka tulee tornin päälle, jos pelaajalla on tarpeeksi resursseja päivitykseen. Ruudussa olevan tornin voi myös myydä, painamalla sell-painiketta ja tämän jälkee valitsemalla ruudussa olevan tornin, tämän jälkeen pelaaja saa puolet tornin hinnasta resursseihin ja torni häviää ruudusta.

Kuva 14. Tornien sijoitus ja ostosysteemi

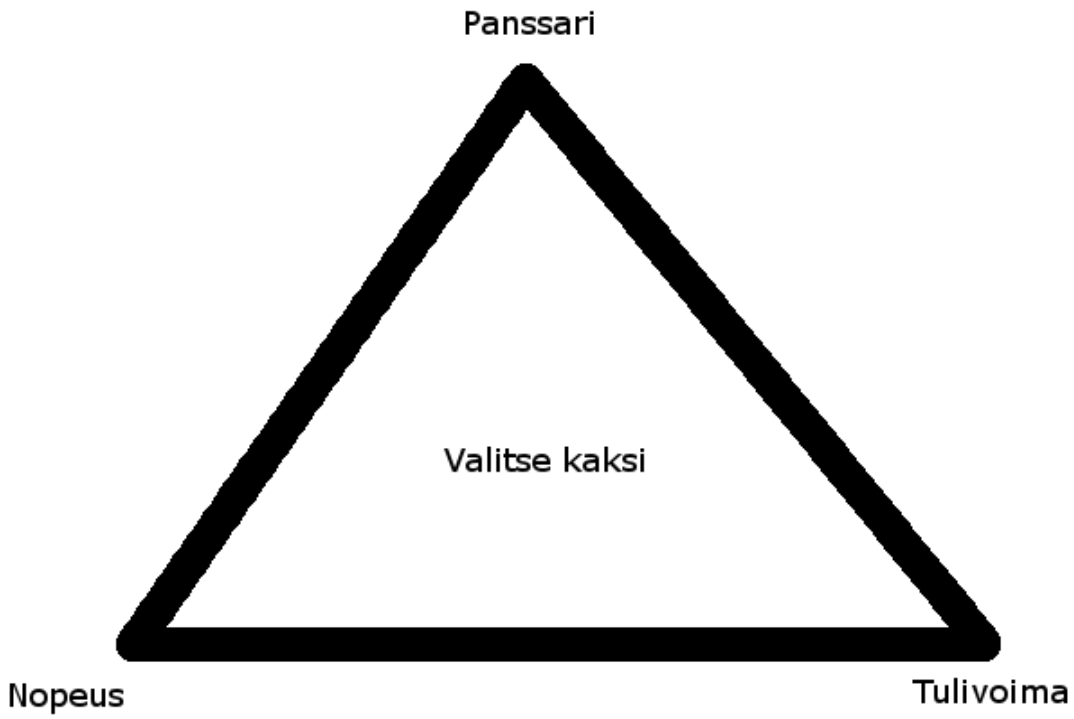
Vihollisten ja tornien kantamat tehtiin Collider:illa eli törmäyksen tunnistuksella. Jokaisella tornilla on oma sphereCollider eli ympyrän muotoinen törmäyksen tunnistus, jos jokin tulee törmäystunnistuksen sisälle, niin tornit ja viholliset lukkiutuvat tähän kohteeseen ja alkavat tulittamaan.

4.5 Pelin tasapaino

Tasapaino pelissä on vaikea toteuttaa. Pelissä vihollisten ja tornien tasapainon löytäminen vaatii suunnittelua ja toteutusta. Yleisesti tasapainon korjaamiseen ja tasapainoittamiseen kuluu useita vuosia, jopa pelin julkaisun jälkeenkin. Mechanical Warfaren pitää ottaa huomioon tornien ja vihollisten kestävyys, vahinko ja kantama. Torneilla on vielä lisäksi hinta, eli paljon tornin sijoittaminen kentälle maksaa. Tornit tekevät vahinkoa vihollisiin ja viholliset tekevät vahinkoa torneihin ja tuhotuista vihollisista saa resursseja, joita käytetään tornien rakentamiseen ja päivittämiseen. Tällä menetelmällä saadaan jatkuva resursseja tarve, jolloin pelaajalle harvoin jää ylimääräistä resursseja käytettäväksi. Tämä ratkaisu nostaa myös pelin vaikeustasoa, jolloin suuremmissa aalloissa vihollisten lukumäärä voi tuhota pelaajan puolustuksen nopeasti. Torneille laskettiin Excel-taulukolla hinta, kestävyys, panssari, vahinko ja vahinko per sekunti. Vahinko per sekunti tunnetaan paremmin DPS eli

Damage per second, joka johdetaan kaavasta (3) (Damage per second). Tornien säännöt ovat selvät. Mikään puolustustorni ei tee kaikkia tehtäviä hyvin, vaan niissä on oltava tasapaino. Tämä tarkoittaa sitä, että jos torni on hyvä pysäyttämään vihollisia, sen heikkous on joko hinta tai kestävyys, joskus molemmat. Vihollisilla taas on kolme ominaisuutta: panssari, nopeus ja tulivoima. Näistä muodostuu kolmio, josta valitaan kaksi vaihtoehtoa, kuten kuvassa 15 näkyy.

$$DPS = Vahinko \,/\, (Latausaika + Hidastukset) \quad (3)$$



Kuva 15. Tasapainokolmio

Tornien ominaisuudet ovat samanlaiset, mutta nopeudella kuvataan latausnopeutta. Tasapainon löytäminen on haastavaa, koska jotkut tornit toimivat paremmin kuin toiset. Testaamisella ja suunnittelulla löytyy kuitenkin sopiva tasapainoratkaisu. Kolmio ei kuitenkaan anna täydellistä kuvaa tasapainosta, vaan idean, miten ideat tulisivat toimimaan. Esimerkiksi robotti ei ole nopea tai hyvin panssaroitu, mutta robotilla on kohtalainen tulivoima. Kuvassa 16 näytetään tornien statistiikat, joissa näkyvät kaikki tarvittavat tiedot

torneista. Kuvassa 17 taas näytetään vihollisten statistiikat, joista saa selvän kuvan vihollisten kestävyydestä, tulivoimasta ja nopeudesta.

Robot	Humvee	TPV-A1	TPV-A2	Range	HP	Armor	Cooldown	Reload	Per shot	Burst	Dmg	DPS	Malli	Hinta
6/ 18	2/ 6	0	0	35	60	0	0,2(2)	1	7	3	21	15	KKmk1	30
8/ 24	4/ 12	0	0	35	70	1	0,2(2)	1	9	3	27	19.28571429	KKmk2	60
8/ 48	4/ 24	0	0	40	100	2	0,2(4)	1	9	6	54	38.57142857	KKmk3	110
19	15	5	0	40	60	0	0	2	20	1	20	10	TykkiMk1	65
29	25	15	10	50	80	1	0	4	30	1	30	7.5	TykkiMk2	120
49	45	35	30	60	100	2	0	6	50	1	50	8.333333333	TykkiMk3	200
24/ 48	20/ 40	10/ 20	5/ 10	35	60	0	0	5	25	2	50	10	OhjusMk1	80
24/ 72	20/ 60	10/ 30	5/ 15	40	80	1	0	5	25	3	75	15	OhjusMk2	160
24/ 96	20/ 80	10/ 40	5/ 20	45	100	2	0	6	25	4	100	16.66666667	OhjusMk3	240
34	30	20	15	40	60	0	0	6	35	1	35	5.833333333	Krhmk1	100
34	30	20	15	50	80	1	0	5	35	1	35	7	Krhmk2	200
34/ 68	30/ 60	20/ 40	15/ 30	55	100	2	0	6	35	2	70	11.66666667	Krhmk3	300
					100	0					0		Tukikohta	

Kuva 16. Tornien balanssitaulukko

	Robot	Humvee	TPV-A1	TPV-A2
DPS	1.25	1.764705	2	2.5
DMG	2*6	4*3	8	10
Armor	1	5	15	20
HP	60	100	200	300
Speed	15	25	20	16
Reload	4	3	4	4
Cooldown	0.2 (4)	0.2(2)	0	0
Palkkio	500	600	800	1000
Range	36	38	40	45

Kuva 17. Vihollisten balanssitaulukko

Kuva 16 taulukossa näytetään tornin panssarit, latausajat, projektiilin vahinko, sarjan vahinko ja tornin vahinko per sekunti. Lisäksi taulukossa näkyy tornin malli ja hinta. Erikseen taulukkoon on myös laskettu mahdollinen vahinko viholliseen per laukaus tai sarja. Taulukosta saa kuvan, miten tornit pärjäävät vihollisia vastaan. Taulukkoa käytetään apuna tasapainotuksessa ja se on tehty siten, että kun muutoksia tekee numeroihin, niin taulukon tiedot päivittyvät. Kuvan 17 taulukko toimii samalla tavalla, mutta siinä on lisäksi palkkio.

4.6 Ekonomia

Kun tornit tuhoavat vihollisia, vihollisista saa palkkion. Palkkio on summa resursseja, joita käytetään uusien tornien rakentamiseen tai tornien

päivittämiseen. Eri vihollisesta saa eri määrän resursseja. Koska viholliset tuhoavat torneja, pelaajan täytyy rakentaa uudet tilalle. Tämä tarkoittaa sitä, että resurssien kerääminen on vaikeata, koska sitä täytyy kuluttaa koko ajan rakentamiseen ja päivitykseen. Pelin alussa pelaajalla on vähän resursseja.

4.7 Ohjelmointi

Peli on ohjelmoitu käyttäen C#-ohjelmointikieltä. Pelissä on kaiken kaikkiaan 32608-riviä koodia. Tästä määrästä 2584 -riviä on itse tehtyä koodia. A* Pathfinding Project -lisäosa sisältää eniten koodirivejä.

4.7.1 Tornit

Kun vihollinen tulee tornin kantaman sisäpuolelle, vihollinen lukitaan kohteeksi ja torni alkaa ampumaan. Jos vihollinen tuhoutuu tai kohde liikkuu tornin kantaman ulkopuolelle, kohde vapautuu ja lukitaan uusi kohde, jos vihollisia on kantamalla. Torneissa on latausnopeus, joka määrittää, kuinka usein torni pystyy ampumaan. Torneissa on myös ennakon otto, joka on mainittu aikaisemmin. Torni ennakoi vihollisen liikkeen ja ampuu vihollisen eteen varmistaen, että projektiili osuu. Kuvissa 18 ja 19 näkyy tykkitornin ohjelmointikoodi.

```
using UnityEngine;
using System.Collections;

public class Tykki : TurretBase {

    public float latausAika = 3f;  //Tykin latausaika
    public float torKaanto = 5f;   //Tykin kääntönopeus
    public float tuliTauko = 0.25f;//Tykin kääntöaikaan liittyvä
    public GameObject krKranaatti; //Tykin ammus
    public GameObject tuliEffekti; //Tykin ampumisliekki
    public Transform kohLukitus;   //Vihollisen paikka
    public Transform tykinSuunta;  //Kohta mistä ammus lähtee.
    public Transform tykTorni;     //Kääntää tornin tähtäämään
    private float nextFireTime;    //Seuraava ampumis aika
    private float nextMoveTime;    //Seuraava kääntymis aika
    private Quaternion tykKaanto;  //Tykin kääntö
    public Animator anim;          //Animaattori
    //ennakko
    Rigidbody rb;                  //Vihollisen rigidbody

    void Start () {
        anim = GetComponent<Animator>();
    }

    void Update () {
        if(kohLukitus) //Jos on vihollinen
        {
            CalculateAimPosition(kohLukitus.position); //Laske tähtäys
            if(Time.time >= nextMoveTime) //Jos voi kääntää tornia
            {   //Tykkitornin kääntö
                tykTorni.rotation = Quaternion.Lerp(tykTorni.rotation,
                                                    tykKaanto, Time.deltaTime*torKaanto);
            }
            if(Time.time >= nextFireTime) //Jos voi ampua
            {
                tykTulta();   //Ammu
                audio.Play ();//Toista ääni
            }
        }
    }
```

Kuva 18. Tykkitornin koodi osa 1

```
    void OnTriggerStay(Collider other) //"Törmäystunnistus"
    {
        if (!kohLukitus){ //Jos kohdetta ei vielä ole
            if (other.gameObject.tag == "Enemy") //Jos peliobjektin tag on "Enemy"
            {
                nextFireTime = Time.time+(latausAika * 0.5f); //Lataus
                kohLukitus = other.gameObject.transform; //Vihollisen paikka
                //Ennakko
                rb = other.gameObject.rigidbody; //Vihollisen rigidbody
            }
        }
        else if(kohLukitus.tag == "Tuhottu") //Jos peliobjektin tag on "Tuhottu"
        {
            kohLukitus = null; //Vihollista ei ole
            return;

        }
    }

    void OnTriggerExit(Collider other) //Kohde poistuu
    {
        if(other.gameObject.transform == kohLukitus)
        {
            kohLukitus = null; //Kohdetta ei ole
            anim.Play ("Idle");//Toista idle animaatio
        }
    }

    void CalculateAimPosition(Vector3 kohSij)
    {   //Ennakon laskukaava: target speed/pattern speed x matka
        //Ennakon laskemista
        float matka = Vector3.Distance(kohLukitus.position,tykTorni.position);
        float ennakko = (rb.velocity.magnitude / 150f * matka);
        //Lisää ennakko täytäyspisteeseen
        Vector3 tahtPiste = new Vector3(kohSij.x + ennakko, kohSij.y -1,kohSij.z + ennakko);
        tykKaanto = Quaternion.LookRotation(tahtPiste - transform.position);//Käännä tornia
    }

    void tykTulta() //Ammu
    {
        nextFireTime = Time.time+latausAika; //Seuraava ampumis aika
        nextMoveTime = Time.time + tuliTauko;//Seuraava kääntymis aika
        //Alusta ammus ja tuliefekti
        Instantiate(krKranaatti, tykinSuunta.position, tykinSuunta.rotation);
        GameObject newexplosion = (GameObject)Instantiate(tuliEffekti,tykinSuunta.position,
                                                          tykinSuunta.rotation);
        Destroy (newexplosion,1); //Tuhoa peliobjekti
        anim.Play ("Tykki");      //Toista ääni
    }
}
```

Kuva 19. Tykkitornin koodi osa 2

4.7.2 Viholliset

Kun vihollisen kantaman sisäpuolelle tulee torni, torni lukitaan kohteeksi ja vihollinen alkaa ampua tornia. Jos torni tuhoutuu, kohde vapautuu ja lukitaan uusi torni kohteeksi, jos sellainen on kantaman sisäpuolella. Vihollisilla on latausnopeus, joka määrittää, kuinka usein vihollinen pystyy ampumaan. Jos vihollinen pääsee tukikohdan luo, se alkaa ampua tukikohtaa ja pelaaja menettää osumapisteitä. Peli päättyy, kun osumapisteet menevät nollaan ja tukikohta tuhoutuu. Tuhouduttuaan viholliset antavat resursseja pelaajalle, joita pelaaja käyttää ostaakseen uusia torneja tai päivittääkseen vanhoja.

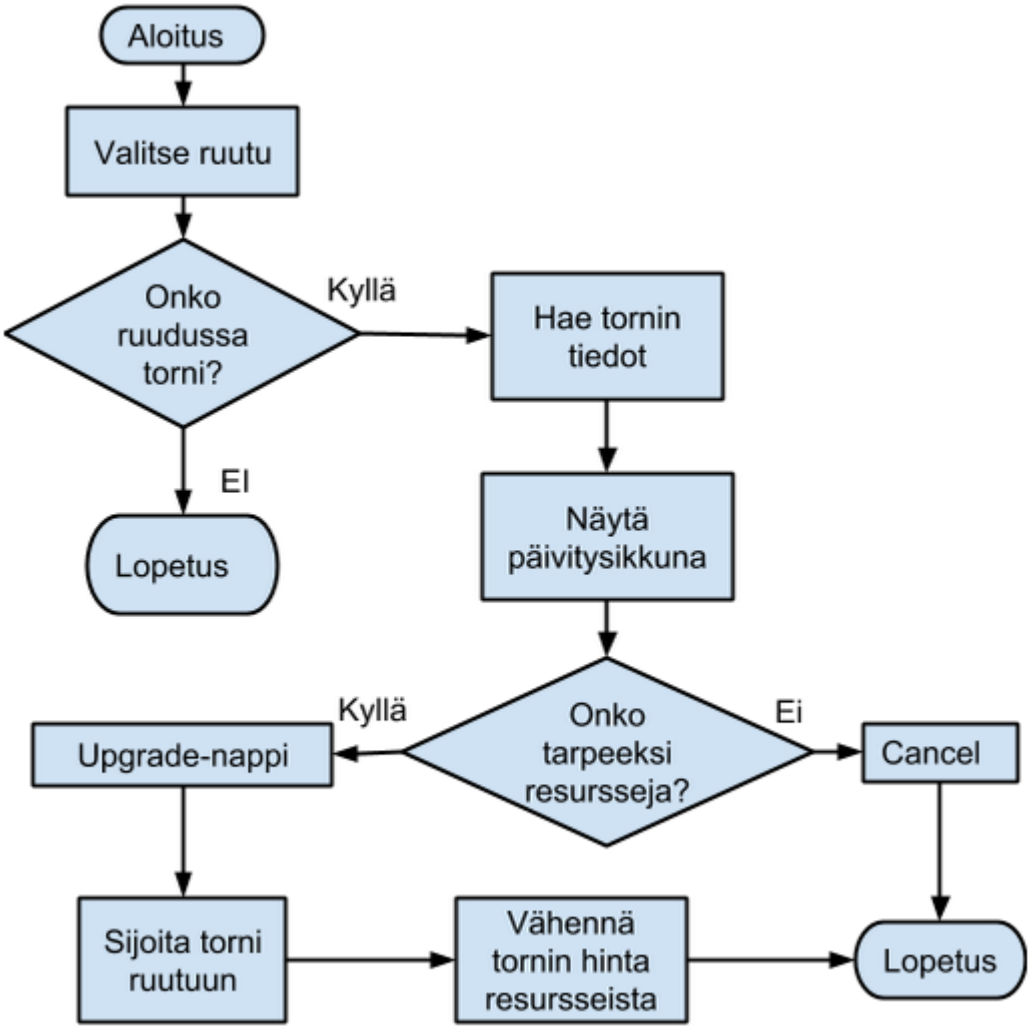
Vihollisilla on pathfinding scripti, jonka avulla ne hakeutuvat pisteestä A pisteeseen B.

4.7.3 Tornien sijoitus

Tornien sijoitusskriptiä käytetään tornien ostamisessa, sijoittamisessa, myymisessä ja päivityksessä. Pelikentällä on ruutuja, joille pelaaja voi sijoittaa torneja rakentaakseen puolustusta vihollisia vastaan. Tornien osto toimii peliruudun alalaidassa olevilla tornipainikkeilla, kun joku näistä valitaan, niin haetaan tornin hinta ja tornin numero. Tämän jälkeen painetaan pelikentällä ruutua, ensimmäiseksi tarkistetaan, onko ruutu vapaa ja riittävätkö resurssit tornin ostoon. Jos resurssit riittävät ja ruutu on vapaa, niin tornin sijoitus onnistuu. Ruutu muutetaan varatuksi ja resursseista vähennetään tornin hinta.

Myynti tapahtuu painamalla peliruudun alalaidassa olevaa “Sell”-painiketta ja tämän jälkeen painetaan ruudussa olevaa tornia, joka halutaan myydä. Ensimmäiseksi skriptissä haetaan ruudussa olevan tornin tiedot, joista tarvitaan ainoastaan myyntihinta. Myyntihinta lisätään resursseihin ja peliobjekti tuhotaan, tämän jälkeen vaihdetaan tornin numero sellaiseksi, että kentälle ei voi vahingossa sijoittaa torneja ilmaiseksi.

Torneja voidaan päivittää, kun ruudussa olevan tornin päälle ilmestyy sininen nuoli. Ruudussa olevaa tornia painetaan ja avautuu päivitysikkuna. Skriptissä haetaan aluksi ruudussa olevan tornin tiedot, jotka ovat päivitettävä tornipeliobjekti, päivityshinta, päivitysnimi ja myyntihinta. Sitten tarkistetaan, riittävätkö resurssit päivitykseen, jos ne riittävät, päivitysikkunassa oleva “Upgrade”-painike muuttuu aktiiviseksi, ja päivityksen voi vahvistaa. Ikkunassa on myös “Cancel”-painike, jolla ikkunasta pääsee pois. Päivityksen vahvistuksen jälkeen tuhotaan vanha tornipeliobjekti ja sijoitetaan uusi tornipeliobjekti. Päivitysikkuna sulkeutuu ja resursseista vähennetään päivityshinta. Kuvassa 20 näkyy vuokaavio tornien päivityksestä.



Kuva 20. Kaavio tornien päivityksestä

4.7.4 Vihollisaallon luoja -skripti

Skripti tarkistaa aaltonumeron ja sitten kertoo aaltonumeron kertoimella 5. Näin saadaan vihollisten lukumäärä, jota käytetään vihollisten synnyttämiseen. Viholliset ohjelma arpoo taulukosta. Skriptillä on ehto: jos aalto on pienempi kuin neljä, niin silloin ohjelma ei pistä kentälle panssarivaunuja, ja jos aalto on suurempi kuin 6, kaikki taulukon arvot on vapaita käyttää. Kun viholliset on tuhottu, levelmaster saa tiedon ja lopettaa aallon. Heti kun aalto on lopetettu, skripti tarkistaa, onko viimeinen aalto, jos aalto ei ole viimeinen, niin ohjelma aloittaa seuraavan aallon. Skripti synnyttää vihollisia valiten ajan kahden sattumanvaraisen luvun väliltä ja valitsee kahden pisteen väliltä, mihin se sijoittaa uuden vihollisen. Levelmaster synnyttää vihollisia, kunnes vihollisten

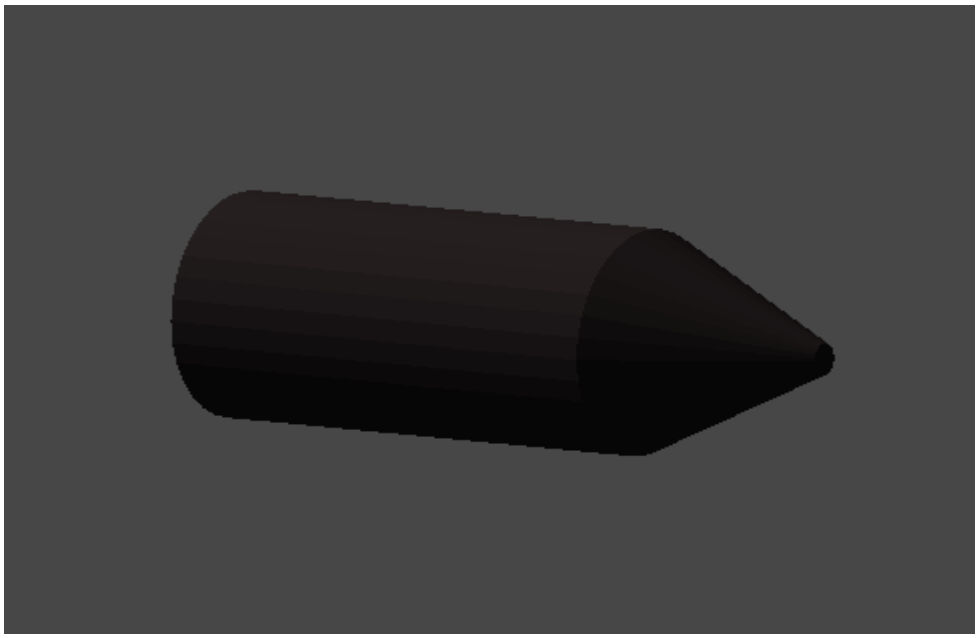
lukumäärä on täynnä, tämän jälkeen se jää odottamaan, onko kaikki viholliset tuhottu.

4.7.5 Osumapiste-skripti

HP eli osumapisteet on yksi pelin ytimistä. Vihollisilla, torneilla ja tukikohdalla on osumapisteet, jotka kuluvat, kun kohde ottaa osumaa. Jos HP on 0, niin kohde tuhoutuu. Osumapisteet skripti toimii yksinkertaisesti. Aina kun projektiili osuu kohteeseen, osumapiste-skripti saa viestin osumasta. Tämän jälkeen skripti vähentää vastaanotetun vahingon panssarin arvolla. Jos osuma on suurempi kuin panssari, niin se, mitä panssari ei estänyt, vähenee kohteen osumapisteistä. Kun vihollisen osumapisteet pienenevät nollaan, vihollinen tuhoutuu ja antaa resursseja. Kun vihollinen tai torni ottaa tarpeeksi osumaa, niin se alkaa savuttamaan. Savuttava kohde ilmoittaa pelaajalle, että se on menettänyt paljon osumapisteitä ja on tuhoutumassa. Kun vihollinen tuhoutuu, se syttyy palamaan ja sen nopeus hidastuu rajusti. Hetken kuluttua vihollinen räjähtää ja katoaa kentältä. Torneissa ei ole tätä toimintoa, vaan tornit räjähtävät ja katoavat.

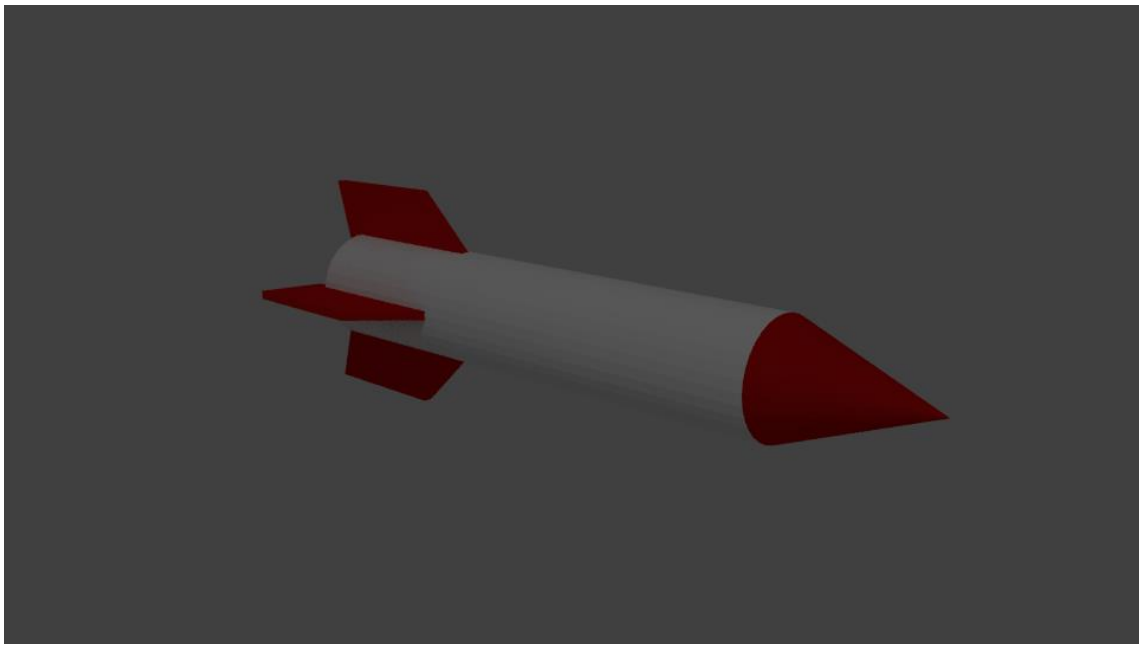
4.7.6 Projektiilit

Kranaatti toimii skriptillä, joka saa kranaatin lentämään eteenpäin. Siitä kohdasta, mihin se syntyy, kranaatti lentää eteenpäin. Kranaatilla on kantama, jonka tehtävänä on estää sitä lentämästä loputtomiin. Jos kranaatti ylittää kantaman, se tuhoutuu. Jos kranaatti osuu viholliseen, niin kranaatti lähettää viestin osumapisteskriptille. Viesti pitää sisällänsä kranaatin vahingon. Osuessaan kohteeseen kranaatti tuhoutuu. Kranaattien ohella samaa ohjelmaa käyttävät konekiväärin patruunat, vihollisten konekivääriaseet ja tykit. Kuvassa 21 näkyy käytössä oleva kranaattimalli. Malli on kaikissa aseissa sama, mutta koko vaihtelee.



Kuva 21. Kranaatti

Ohjukset ovat erilaisia, ohjukset ovat hakeutuvia. Kuvassa 22 näkyy ohjuksen malli. Ohjuksissa on punainen pää ja siivekkeet. Runko taas on valkoinen. Ohjukset hakeutuvat ohjustornin lukitsemaan kohteeseen. Ohjus liikkuu eteenpäin ja kääntyy kohdetta päin. Osuessaan viholliseen ohjus räjähtää ja tekee vahinkoa viholliselle. Räjähtäessään ohjuksella on vaikuttava alue, jossa kaikki sen kantamalla olevat viholliset ottavat vahinkoa.



Kuva 22. Ohjus

Kranaatinheittimen kranaatti on erilainen verrattuna ohjukseen ja tykkikranaattiin. Kranaatinheittimen kranaatti räjähtää osuessaan maahan, toisin kuin ohjukset ja kranaatit, jotka räjähtävät osuessaan viholliseen. Kranaatinheittimen kranaatin ei tarvitse osua viholliseen aiheuttaakseen vahinkoa. Kranaatinheittimen kranaatti toimii samalla periaatteella kuin ohjuksen räjähdys, mutta sen vaikutusalue on suurempi ja vahinko on paljon merkittävämpi.

4.8 Ongelmat

Blenderin ja Unityn välillä on tunnettu ongelma. Ongelman syy on koordinaatisto. Blender käyttää oikeakätistä koordinaatistoa, kun taas Unity käyttää vasemman kätistä koordinaatistoa. Suurimpana erona on se, että Blenderissä Z-akseli on ylöspäin, kun taas Unityssä Y-akseli on ylöspäin. Ongelmana Blenderistä tullut mallin ‘eteenpäin’ kulkeva akseli Z-osoitti väärään suuntaan, mikä tarkoittaa sitä, että mallit osoittivat väärään suuntaan. Ongelman ratkaisu oli kuitenkin yksinkertainen. Mallit piti kääntää osoittamaan oikeaan suuntaan Blenderissä siten, että mallit olivat Z-akselin suuntaisesti Blenderissä.

Mallien keskipiste piti myös määrittää Blenderissä, koska ilman sitä tornit kääntyivät siten, että ne irtosivat rungosta. Tämän korjaus oli kuitenkin nopea ja yksinkertainen. Blenderissä valittiin kohdennettava malli, eli tässä tapauksessa torni ja painetaan näppäinyhdistelmää Shift+vasen Ctrl+alt+c ja valitaan keskipisteeksi mallin keskusta, tai 3D-osoittimen sijainti.

Suurin ongelma Blenderissä oli ketjujen tekeminen Unityyn. Panssarivanujen ketjujen oli alunperin tarkoitus liikkua pelikentällä. Kuitenkin Blenderissä animoidut ketjut eivät liikkuneet Unityssä. Tämä johtui siitä, että Blenderissä tehdyt ketjut olivat vain yksi pieni palanen, johon oli liitetty useita kopioita samasta palasesta. Tämä loi illuusion kokonaisesta ketjusta. Tähän käytettiin työkalua nimeltä ArrayModifier. Kun mallista tehtiin .fbx-tiedosto, ketjuista tuli kiinteä. Tästä syystä ketjut eivät pyöri. Lopulta päädyimme ratkaisuun, missä ketjuista tehtiin kiinteämmät ja ne eivät pyöri mihinkään suuntaan. Myöskään ketjujen renkaat eivät pyöri.

Törmäystunnistusta käytetään myös osumatunnistukseen. Tämä johti ongelmaan, jossa kantama ja osumatunnistus joutuivat ristiriitaan keskenänsä. Ongelma ratkaistiin käyttämällä tasoja eli layer. Jokaisen tornin ja tykin lapsiobjektiin laitettiin uusi sphereCollider. Tälle laitettiin nimi ‘kantamataso’. Näin tornit ja viholliset tunnistivat ja ottivat osumaa oikealla tavalla ilman ristiriitoja törmäystunnistuksen kanssa.

5 Käytetyt ohjelmat

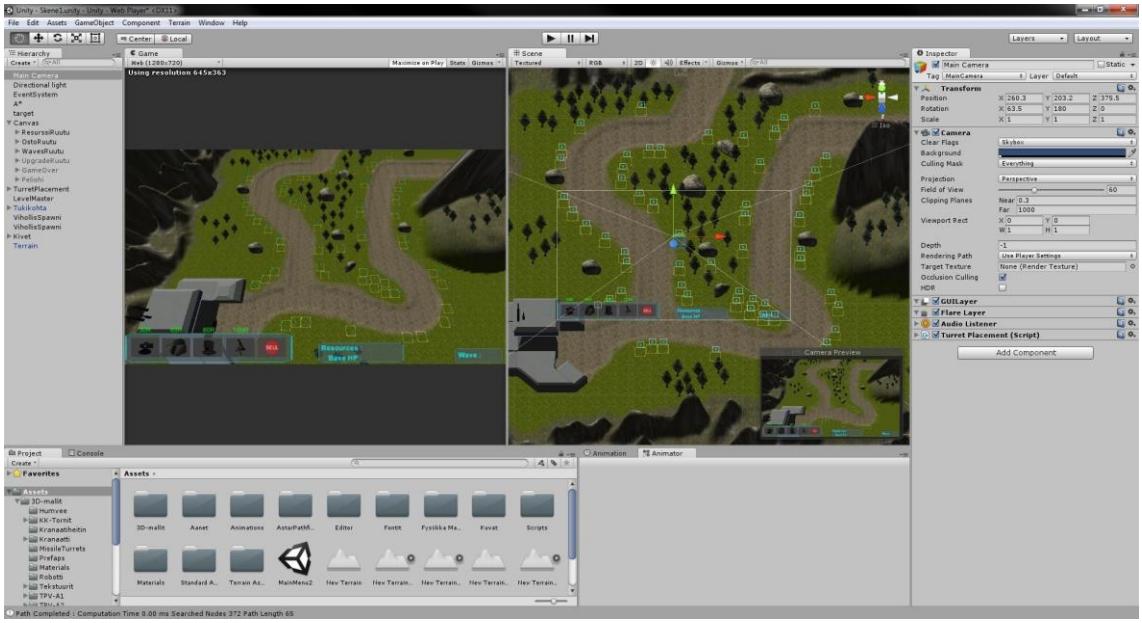
Tässä luvussa käsitellään opinnäytetyössä käytettyjä ohjelmia. Ohjelmat valittiin, koska ne ovat ilmaisia ja kaikista paitsi Blenderistä oli jo aikaisempaa kokemusta. 3D-mallit luotiin Blender 3D-mallinnusohjelmalla. Mallit siirettiin fbx.tiedostoina Unity-pelimoottoriin, jolla peliä kehitettiin. 3D-mallien tekstuurit toteutettiin GIMP 2:lla. Äänet muokattiin Audacityllä.

5.1 Unity

Unity on monialustainen pelimoottori, joka sisältää pelimoottorin ja integroidun kehitysympäristön (Unity (game engine)). Monialustainen tarkoittaa tukea monelle eri käyttöjärjestelmälle ja laitteelle. Unityn on kehittänyt Unity Technologies. Unity Technologies on perustettu vuonna 2004 Tanskassa (Unity Technologies). Unitystä on saatavilla ilmais- ja Pro-versiot. Pro-version lisenssi on maksettava, jos pelin tuotot vuodessa ovat yli 100 000 $. Unity Pro maksaa 1500 $ tai 75 $ kuukaudessa (Unity 3D store). Pro-versio sisältää enemmän toimintoja kuin ilmaisversio, mutta ilmaisversiolla pärjää hyvin ja saa toimivia pelejä aikaiseksi. Tässä projektissa käytettiin ilmaista versiota.

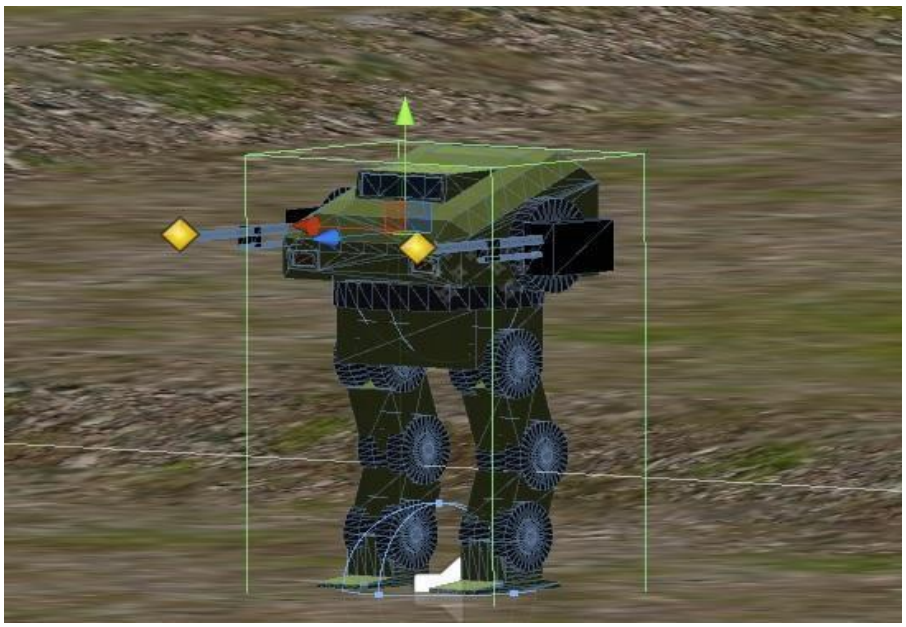
Pelimoottori sisältää mm. fysiikkamoottorin, törmäysten tunnistuksen, grafiikkamottorin. Unityn käyttöliittymässä on peli- ja scene-näkymä, inspector-, hierarchy-, project- ja console-ikkunat. Scene-näkymässä muokataan pelin kenttää. Pelinäkymässä nähdään pelikameran kuva. Inspectorissa nähdään kaikki peliobjektin sisältämät komponentit, komponentin asetukset ja parametrit. Hierarchy näyttää scenen sisältämät peliobjektit. Project-ikkunassa nähdään projektin sisältämät kansiot ja tiedostot. Console-ikkunassa nähdään log-

merkinnät, varoitukset ja virheilmoitukset. Kuvassa 23 näkyy Unityn käyttöliittymä.



Kuva 23. Unityn käyttöliittymä

Peliobjekti on pelissä näkyvä tai näkymätön objekti, jolle voidaan lisätä komponentteja tai omia skriptejä, joilla kontrolloidaan objektin toimintaa pelissä. Sille voidaan myös laittaa grafiikkaa, kuten tekstuureita, muotoja ja 3D-malleja. Kuvassa 24 on robotti-peliobjekti.



Kuva 24. Robotti-peliobjekti

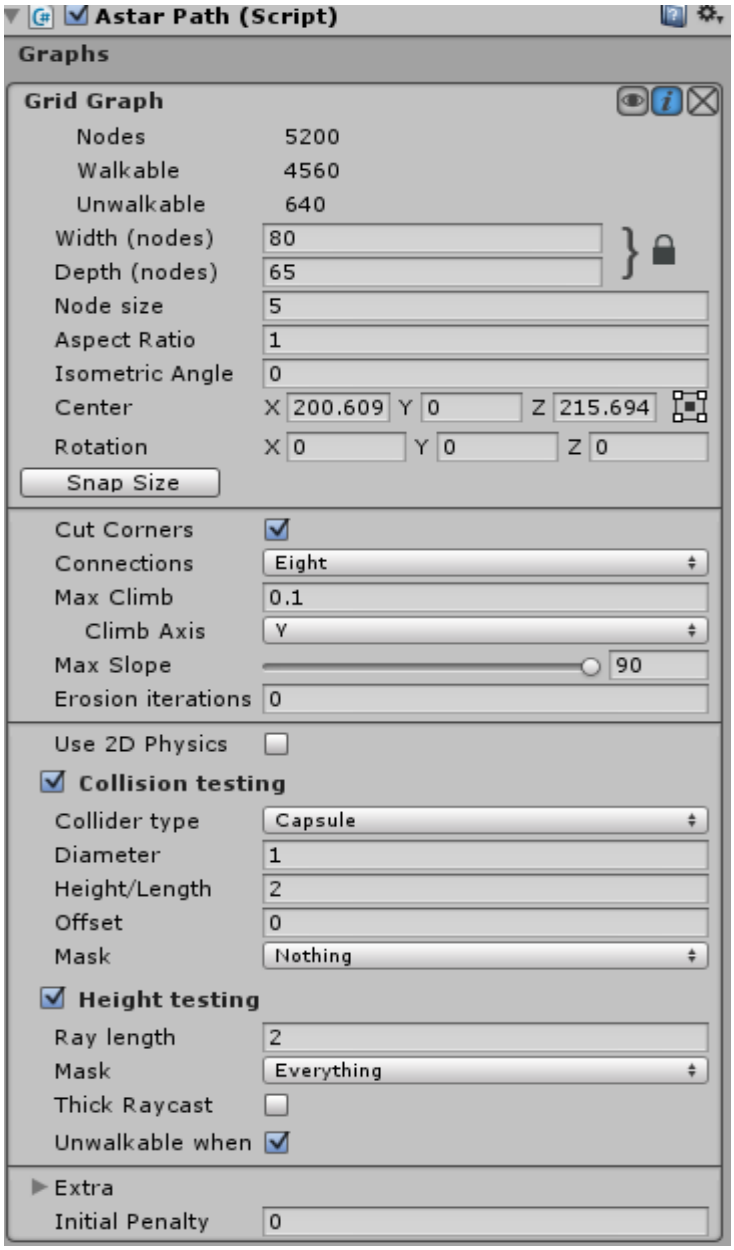
Unity tukee C#, JavaScript ja Boo-ohjelmointikieliä. Projektin skriptit on ohjelmoitu käyttämällä C#-ohjelmointikieltä. Unityn mukana tulee MonoDevelop (IDE, integrated development environment) ohjelmointiympäristö. MonoDevelop sisältää tekstieditorin, virheenkorjauksen ja projektin hallintatehtäviä.

5.2 Unityn lisäosat

Unityn voi ladata lisäosia Asset Storesta. Lisäosia on paljon erilaisia mm. animaatioita, tekstuureita ja ääniä. Osa lisäosista on maksullisia.

5.2.1 A* Pathfinding Project-Unity-lisäosa

A* Pahtfinding Project on Unityn tekoäly-lisäosa, jonka voi ladata http://arongranberg.com/astar/ -sivulta (A* Pathfinding Project). Siitä on saatavilla ilmainen ja maksullinen versio. Projektissa käytettiin ilmaista versiota, koska oman reitinhakusysteemin kehittäminen olisi saattanut viedä liikaa aikaa. Lisäosan tarkoituksena on hakea reitti, jota pitkin voi liikkua. Reitin hakuun käytettiin Grid graph -reitinhakua, jossa liikkumakelpoinen alue määritellään ylhäältä päin laukaistavilla säteillä. Säteillä katsotaan korkeuserot, joiden avulla voidaan määrittää kävelykelpoinen alue määritetyllä ruudukolla. Hierarchy-ikkunassa voidaan määritellä paremetrejä esimerkiksi, kuinka korkealle peliobjekti pystyy maksimissaan kiipeämään. Kuvassa 25 on A* Pathfinding Project asetuksia.



Kuva 25. A* Pahtfinding Project

5.2.2 Terrain assets -tekstuuripaketti

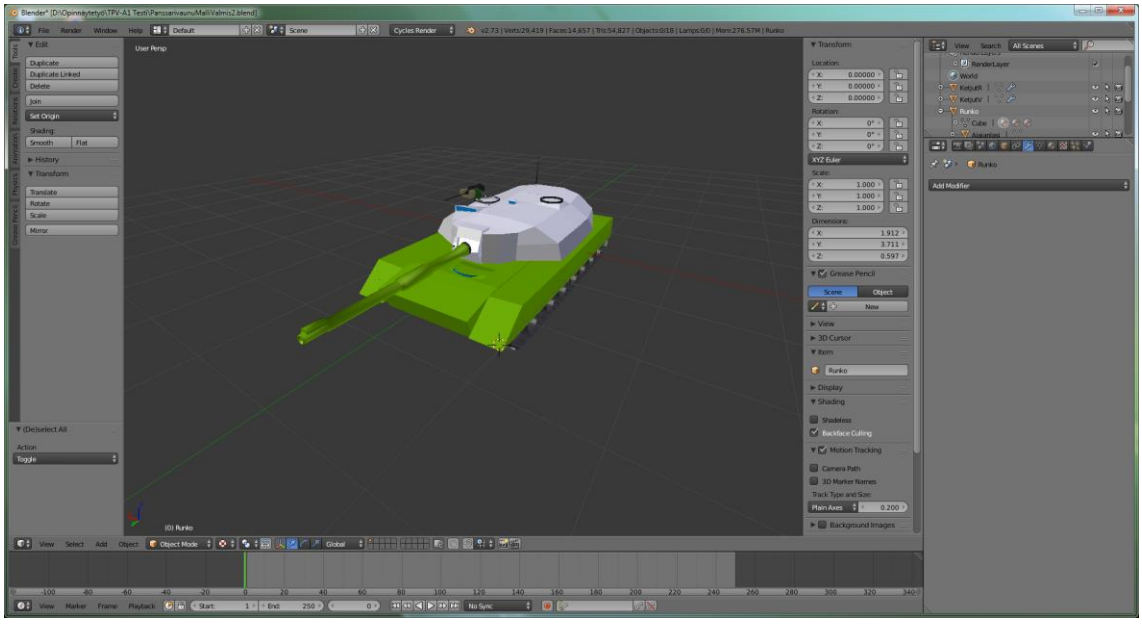
Terrain assets on ilmainen tekstuuripaketti, jonka voi ladata Unityn assetstoresta. Paketin on julkaissut Unity Technologies. Tekstuuripaketti sisältää 12-puu-, 12-puska-, 6-ruoho-3D-mallia ja 22-maatekstuuria (Asset store).

5.3 Blender

Blender on ilmainen ja suosittu 3D-grafiikan mallinnusohjelma, jonka on luonut hollantilainen NeoGeo ja Not A Number Technologies. Not a Number technologies sulkeutui 2002. Blenderin kehittäjä aloitti kampanjan, jonka tarkoituksena oli saada Blender ilmaiseksi, koska Not a Number technologies oli kaatunut ja Blenderin kehitys olisi keskeytynyt kokonaan. Blenderin perustaja perusti voittoa tavoittelemattoman yrityksen Blender Foundationin, jonka tarkoituksena oli jatkaa Blenderin kehitystä yhteisöprojektina. Blender kehitystä jatkaa Blender Foundation. (Blender 2013.)

Blenderillä toteutettiin pelin 3D-mallit. Blenderillä voi luoda useita erilaisia malleja, animaatioita, grafiikkaa ja pelejä.

Blenderin opetteluun meni aikaa, koska ohjelman käyttöliittymä on monimutkainen aloittelijoille. Käyttöliittymän navigoiminen ja pikanäppäinten oppimiseen meni oma aikansa. Kuvassa 26 näytetään Blenderin käyttöliittymä.



Kuva 26. Blenderin käyttöliittymä

Blenderin käyttöliittymä kuvassa näkyy selvästi malli, jota muokataan. Hierarkialista on oikealla ja siinä näkyvät käyttäjälle kaikki mallit, jotka ovat kentällä, mallien näkyvyys ja hierarkia. Aivan alhaalla näkyy aikajana, jota käytetään animoinnissa ja liikkeiden luomisessa ja sen työkalurivi. Vasemmalla

on työkalu-paneeli, jota käytetään mallin tekemiseen. Paneelissa on useita vaihtoehtoja käyttäjän eri tarpeisiin. Oikealla hierarkian vieressä vasemmalla puolella on navigointiin liittyvä paneeli, josta voi muokata kameran sijaintia, mallin sijaintia ja näkymää.

Blenderin ominaisuudet tulevat esille hyvin nopeasti. Alkuvaiheessa pitää opetella navigointi ja erilaiset työkalut, joita on tarjolla. Blenderissä kameraa ohjataan hiiren kolmannella painikkeella, joka sijaitsee hiiren rullapainikkeessa. Malli valitaan ja siirretään hiiren oikealla painikkeella ja 3D osoitinta siirretään hiiren vasemmalla painikkeella. Mallit poistetaan hierarkiasta X-näppäimestä. Mallista päästetään irti painamalla A-näppäintä. Tämän takia Blenderi voi tuntua hieman omituiselta alussa.

Blenderillä voi tehdä useita erilaisia 3D-malleja ja animaatioita. Blenderissä luotiin pari animaatiota, kuten esimerkiksi tykkien ampuminen ja robotin käveleminen. 3D-malleja tehtiin yhteensä 16 kappaletta. Blenderillä voi tehdä myös maastoja digitaalisista mittauksista tai luoda niitä itse käyttämällä Blenderin työkaluja. Opinnäytetyössä käytettiin maastoon kuitenkin Unityn kenttän muokkaustyökalua. Kaikki tehdyt mallit tuotiin ulos .fbx-tiedostoina, jotka siirrettiin Unityyn.

Robotin 3D-malli piti animoida Blenderissä, ennen kuin se siirrettiin Unityyn. Animoinnissa käytettiin luurankoa, jonka luita liikuttamalla luotiin kävelyanimaatio. Luun vaikutusalue 3D-mallissa tehtiin käyttäen “weight paint”-työkalua. Näin pystyttiin tarkasti määrittämään, mikä luu vaikuttaa mihinkin robotin osaan. Robotissa käytettiin seitsemää eri luuta, kuten kuvassa 27 näkyy. Luut ovat runkoluu, reisiluut, sääriluut ja nilkkaluut. Pose modessa pystyttiin tekemään animaatio liikuttamalla luita haluttuihin asentoihin ja lukitsemalla ruudussa oleva asento. Kävelyanimaatiossa käytettiin viittä eri asentoa, joita toistamalla robotti kävelee.

Kuva 27. Robotin luuranko

Malleja luotiin 12 tornia ja neljä vihollista. Konekivääritornit, ohjustornit, tykkitornit ja kranaatinheittimet versioista Mk I versioon Mk III asti. Vihollisiksi luotiin kävelevä robotti, maastoauto, taistelupanssarivaunumalli yksi ja raskaampi versio taistelupanssarivaunusta malli kaksi. Mallien teko aloitettiin vihollisista ja kummatkin tekivät oman mallin. Jussi Juuti aloitti panssarivaunulla ja Julian Parman aloitti maastoautolla. Mallintaessa vihollista alkoi oppia Blenderin käyttöä ja sen työkaluja. Ensimmäisten mallien tekemiseen meni paljon aikaa.

5.4 Gimp 2

Gimp 2 on ilmainen kuvankäsittelyohjelma. GIMP 2:sella tehdään pelin tekstuurit ja muut tarvittavat kuvamateriaalit. Sana GIMP tulee akronyymistä

GNU Image Manipulator Program. Ohjelma on avointa lähdekoodia ja on internetissä jaossa ilmaiseksi. (GIMP 2014.)

GIMP oli entuudestaan tuttu ohjelma, koska sitä on käytetty ennenkin pelien tekstuureihin, grafiikkaan ja animaatioihin. GIMPillä tehtiin Mechanical Warfare -projektissa tekstuurit käytössä olleisiin 3D-malleihin ja käyttöliittymään kuvapohjat. Esimerkiksi Ostovalikon tausta on tehty GIMPillä.

Ohjelma on monimutkainen käyttää ja vaatii kohtalaisen paljon totuttelua, ennen kuin ohjelmaa osaa käyttää. Käyttöliittymä on alussa sekava. Työkalujen toiminta ja asetukset ovat myös vaikea oppia.

5.5 Audacity

Audacity on ilmainen äänenmuokkausohjelma, jonka on kehittänyt joukko vapaaehtoisia henkilöitä, ja sitä jaetaan GNU General Public lisensillä (Audacity About). Audacityllä äänitiedostot muokattiin peliin sopiviksi, eli muokkaamalla äänenvoimakkuutta ja äänen kestoa.

Pelin äänet on haettu www.findsounds.com-sivustolta. Tarpeen mukaan ääniä muokattiin tarkoitukseen sopivaksi. Tämä tapahtui avaamalla musiikkitiedosto ja muokkaamalla ääntä. Äänitiedostoa joko pienennettiin tai sen bassoa lisättiin. Lopuksi tiedosto tallennettiin .mp3-tiedostoksi ja siirrettiin Unityyn.

6 Yhteenveto ja pohdinta

Opinnäytetyön tavoitteena oli suunnitella ja luoda demoversio Mechanical Warfare -tornipuolustuspelistä. Mechanical Warfaren kehitys jatkuu opinnäytetyön jälkeen ja kokoversio on tarkoitus julkaista mobiilialustalle. Mechanical Warfaren kehitys oli haastavaa, koska kummallakaan tekijöistä ei ollut paljoa kokemusta pelin kehityksestä tai 3D-mallintamisesta. Pelin sopivan tasapainoituksen löytäminen on haastavaa. Tavoitteeseen kuitenkin päästiin ja demoversio saattiin valmiiksi.

Yhtenä opinnäytetyön tavoitteena oli opetella käyttämään Blender-ohjelmistoa, Unityä ja tuottaa toimivia 3D-malleja. Malleja luotiin neljä tornimallia ja neljä

vihollista. Jokaisella tornimallilla on kaksi päivitystä, joten malleja tornia kohti on kolme kappaletta. GIMP-kuvan mallinnusohjelmaa käytettiin mallien tekstuureihin ja käyttöliittymän pohjiin. Audacity-äänenmuokkausohjelmalla saavutettiin sopivat äänet demoversioon. Tavoite saavutettiin, koska demoversioon saatiin tarvittavat 3D-mallit tekstuureineen ja äänineen.

Pelin kehitys on haastavaa, koska siinä pitää osata monta erilaista osa-aluetta. Jos kehitystiimi on pienikokoinen, niin osaamisalueen laajentaminen on tarpeellista. Pitää osata ohjelmoida, 3D-mallintaa ja tehdä musiikkia, tekstuureita ja ääniä. Toisin kuin isoissa yrityksissä, pienessä porukassa on mahdotonta tehdä ryhmiä, jotka keskittyvät omaan osaamisalueeseensa. Esimerkiksi artistit keskittyvät luomaan tekstuureita ja 3D-malleja ja ohjelmoijat keskittyvät ohjelmoimaan skriptejä ja pelimekaniikkaa. Jokaisen ryhmän jäsenen on keskityttävä eri tehtäviin tarpeen mukaan ja siirryttävä muihin tehtäviin, kun tarve on täytetty. Tämä voi olla stressaavaa etenkin, kun töitä on paljon ja alkaa olla kiire. Pienellä miehistöllä on kuitenkin omat etunsa. Päätöksenteko on helpompaa, tekijöillä on luomisen vapaus, laajoja dokumentaatioita ei tarvita ja budjetit ovat pieniä.

Kuvat

Lähteet

A* Pathfinding Project. A* Pathfinding Project Documentation. http://arongranberg.com/astar/docs/. Luettu 09.02.2015.

Asset store. Terrain assets. https://www.assetstore.unity3d.com/en/#!/content/6. Luettu 09.02.2015.

Audacity About. http://audacity.sourceforge.net/about/. Luettu 27.3.2015.

Blender. Blender history. http://www.blender.org/foundation/history/. Luettu 25.2.2015.

Damage per second. http://en.wikipedia.org/wiki/Damage_per_second. Luettu 09.02.2015.

GIMP. About GIMP. http://www.gimp.org/about/introduction.html. Luettu 09.02.2015.

LudoCraft. Suppean pelisuunnitelman pohja. www.ludocraft.com/pelisuunnittelija/Suppean_pelisuunnitelman_pohja. Luettu 03.09.2014.

Tonino Blasi 1997, 21. Step by Step to Success: THE ART OF SKEET SHOOTING.http://ampumaurheiluliitto-fi-bin.directo.fi/@Bin/7d2e1063513441f6a0b6663691e9c652/1424335605/application/pdf/163562/Step_to_step_fi.pdf. Luettu 30.01.2015.

Unity 3D store. Products. https://store.unity3d.com/products. Luettu 09.02.2015.

Unity (game engine). http://en.wikipedia.org/wiki/Unity_%28game_engine%29. Luettu 09.02.2015.

Unity Technologies. Wikipedia. http://en.wikipedia.org/wiki/Unity_Technologies. Luettu 09.02.2015.