jamk.fi

# Kenttäeditorin kehittämistutkimus

## Case: Kinahmi Games Oy

Kim Lindström

Opinnäytetyö
Toukokuu 2016
Luonnontieteiden ala
Tradenomi (Amk), tietojenkäsittelyn tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

**Kuvailulehti**

| Tekijä(t) | Julkaisun laji | Päivämäärä |
|---|---|---|
| Lindström, Kim | Opinnäytetyö, AMK | Toukokuu 2016 |
| | Sivumäärä<br>35 | Julkaisun kieli<br>Suomi |
| | | Verkkojulkaisulupa myönnetty: x |

Työn nimi

**Kenttäeditorin kehittämistutkimus**

Case: Kinahmi Games Oy

Tutkinto-ohjelma

Tietojenkäsittelyn tutkinto-ohjelma

Työn ohjaaja(t)

Tommi Tuikka

Toimeksiantaja(t)

Kinahmi Games Oy


Tiivistelmä

Kinahmi Games Oy julkaisi elokuussa 2015 pelin nimeltä Galactic Conquerors. Pelin kenttien tekemiseen ei ollut erillistä ohjelmaa, minkä takia kenttien tekeminen oli hankalaa. Valmiin kenttäeditorin tehtävä oli helpottaa toimeksiantajaa kenttien suunnitellussa ja toteuttamisessa sekä mahdollistaa pelaajille työkalu, jolla he voivat tehdä omia kenttiä peliin. Tavoitteena oli saada peliin parempia kenttiä ja lisää sisältöä.

Opinnäytetyö toteutettiin ohjelmiston kehittämistutkimuksena. Projektin aikana käydyt palaverit ja keskustelut toimeksiantajan kanssa, pelin ominaisuuksien tutkiminen ja muiden pelien kenttäeditorien tarkastelu auttoivat kenttäeditorin määrittelyssä, suunnittelussa ja toteutuksessa.

Kenttäeditorin kehittäminen aloitettiin tarvittavien ja haluttujen ominaisuuksien määrittelyllä, minkä jälkeen valittiin toteuttamiseen sopivimmat ohjelmistot ja tekniikat. Kenttäeditori ohjelmoitiin vaatimusten mukaiseksi. Kaikki ominaisuudet testattiin ja korjattiin löydettyjä ongelmakohtia, jotka liittyivät sekä käytettävyyteen että konkreettisiin virheisiin. Lopuksi tuote esiteltiin toimeksiantajalle ja projekti päätettiin.

Projektista saatiin tulokseksi valmis kenttäeditori toimeksiantajalle. Projekti oli onnistunut lukuisista ongelmista huolimatta. Etenkin ajankäytössä ja projektin ominaisuuksien rajaamisessa oli haasteita.



Avainsanat (asiasanat)

Unity, kenttäeditori, pelikehitys, ohjelmointi


Muut tiedot

jamk.fi

**Description**

| Author(s) | Type of publication | Date |
|---|---|---|
| Lindström, Kim | Bachelor's thesis | May 2016 |
| | | Language of publication: Finnish |
| | Number of pages: 35 | Permission for web publication: x |

Title of publication

**Development Research of Map Editor**

Case: Kinahmi Games Oy

Degree programme

Business Information Systems

Supervisor(s)

Tuikka, Tommi

Assigned by

Kinahmi Games Oy


Abstract

Kinahmi Games Oy released a game called Galactic Conquerors in August 2015. There was no dedicated tool for creating maps, which made them difficult to make. The purpose of the map editor was to make it easier for the company to design and implement maps and also to enable players to create their own maps for the game. The goal was to provide the players with better maps and more content in the game.

The research was performed as a development study. The meetings and conversations with the assigner, studying the game's mechanics and map editors from other games helped in defining, designing and implementing the editor.

The development of the map editor started by defining the necessary and desired features, after which the most suitable applications and techniques to accomplish the project were chosen. The map editor was programmed according to the specifications. Every feature was tested and any problems found with usability and mistakes were fixed.

The end result was a complete map editor for the assigning company. Despite multiple problems the project was a success. Problems arose especially in time management and definition of the project's features.



Keywords/tags (subjects)

Unity, map editor, game development, programming


Miscellaneous

## Sisältö

Käsitteet ..... 4

1 **Johdanto** ..... 5

2 **Tutkimusasetelma** ..... 6

2.1 Opinnäytetyön taustateoriaa, rajaukset ja tavoitteet ..... 6

2.2 Tutkimusmenetelmät ..... 7

2.3 Tutkimuskysymykset ..... 7

3 **Kenttäeditori** ..... 8

3.1 Historia ..... 8

3.2 Ominaisuudet ..... 9

3.3 Editorin hyödyt ..... 10

4 **Unity** ..... 11

4.1 Historia ..... 12

4.2 Ohjelmointi ..... 14

4.3 Unity Asset Store ..... 15

5 **Toteutus** ..... 15

5.1 Projektin aloittaminen ..... 16

5.2 Määrittely ja suunnittelu ..... 16

5.2.1 Valmis editori vai omatekemä ..... 17

5.2.2 Ohjelmointikielen valinta ..... 19

5.2.3 Vaatimusmäärittely ..... 19

5.2.4 Käyttöliittymän suunnittelu ..... 19

5.3 Kehittäminen ..... 20

5.3.1 Kehitys- ja toteutusympäristö ..... 20

5.3.2 Kehitysympäristön pystyttäminen ..... 21

5.3.3 Ohjelmointi ..... 21

5.4 Testaus ..... 22

6 **Tulokset** ..... 23

6.1 Tutkimuskysymykset ..... 23

6.2 Jatkokehitys ..... 25

7 **Pohdinta** ..... 26

Lähteet ..... 28

Liitteet ..... 30

Liite 1. Vaatimusmäärittelyn dokumentti ..... 30

**Kuviot**

Kuvio 1. Yksinkertaistettu vuokaavio kenttäsuunnittelusta ..... 11
Kuvio 2. Unityn käyttöliittymä..... 12
Kuvio 3. Unityssa käytetyt ohjelmointikielet ..... 14
Kuvio 4. Käyttöliittymä objektin muokkaamiseen ..... 20

**Taulukot**

Taulukko 1. Valmiin editorin hyvät ja huonot puolet ..... 18
Taulukko 2. Omatekemän editorin hyvät ja huonot puolet..... 18

## Käsitteet

**Asset**: Yksittäinen 3d-malli, skripti, äänitiedosto yms. Assettien avulla luodaan pelin audiovisuaalinen ulkoasu.

**Modaus**: pelin muokkaamista siten, että saadaan ominaisuuksia, joita alkuperäinen valmistaja ei ole suunnitellut.

**Mono**: avoimeen lähdekoodiin perustuva ohjelmistokehitysympäristö, joka on alusta- ja käyttöjärjestelmäriippumaton pohja .NET-arkkitehtuurille.

**Pelimoottori:** ohjelmistokehys, joka sisältää tärkeimmät pelin tekemiseen tarvittavat ominaisuudet: renderointimoottorin, fysiikkamoottorin, animoinnin, äänet, tekoälyn yms.

**SDK:** Software Development Kit on pelinkehittäjien luoma työkalupaketti, jonka avulla kuka tahansa voi luoda uutta sisältöä pelimoottorille ja/tai pelille.

**XML:** Extensible Markup Language on merkintäkielien standardi, jolla määritellään niiden rakenteita

# 1 Johdanto

Peliteollisuus on nopeasti kasvava toimiala niin Suomessa kuin maailmanlaajuisestikin. Varsinkin viime vuosien aikana peliteollisuudesta on tullut trendi. Nykypäivänä monet suuretkin yritykset tarjoavat pelimoottoreitaan jopa ilmaiseksi, mikä on vauhdittanut peliteollisuuden kehitystä. Pelien tekeminen ja jakelu ovat helpottuneet niin paljon, että lähes kuka tahansa pystyy tekemään pelin ja jakamaan sitä internetin välityksellä.

Toimeksiantaja huomasi tehdessään kenttiä peliin, että niiden tekemiseen olisi paljon helpompikin keino kuin nykyinen toimintamalli. Nykyisellään kenttien tekeminen ja muokkaaminen tapahtuvat pelkän tekstinkäsittelyohjelman avulla. Kyseinen toimintamalli on hidas ja vaikea. Lisäksi se tuotti lukuisia virheitä, ja esimerkiksi objektit saattoivat sijaita päällekkäin ja näin ollen kentät eivät toimineet halutulla tavalla. Valmis kenttäeditori helpottaa toimeksiantajaa kenttien tekemisessä ja mahdollistaa pelaajien luoda omia kenttiä peliin. Paremmat ja monipuolisemmat kentät tuovat pelille lisäarvoa. Pelialan kasvavilla markkinoilla kilpailu on erittäin kovaa, ja pienetkin asiat voivat vaikuttaa merkittävästi pelin myyntilukuihin.

Opinnäytetyön tarkoitus on kuvata kenttäeditorin kehitys aloituspalaverista projektin päättämiseen asti. Teoriaosuudessa käydään läpi kenttäeditoreja ja Unitya yleisesti. Lopuksi vastataan tutkimuskysymyksiin ja pohditaan opinnäytetyön teoriaosuutta sekä projektia.

# 2 Tutkimusasetelma

Tässä luvussa käydään läpi tutkimuksen tausta ja tavoitteet, kuvataan tutkimusmenetelmät ja esitellään tutkimuskysymykset.

## 2.1 Opinnäytetyön taustateoriaa, rajaukset ja tavoitteet

**Tausta**

Tutkimuksen tarkoituksena on kartoittaa yrityksen nykyinen tilanne kenttien tekemisen ja muokkaamisen suhteen sekä selvittää tulevaisuuden tarpeet. Tutkimustulosten perusteella on tarkoitus tehdä graafinen kenttien muokkaamiseen ja luomiseen tarkoitettu sovellus.

**Näkökulma ja perustelut**

Opinnäytetyön aihetta lähdetään tarkastelemaan yrityksen tarpeiden kannalta. Tällä hetkellä yrityksessä kenttien muokkaaminen ja tekeminen tapahtuvat pelkän tekstinkäsittelyohjelman avulla. Tämän takia kenttien tekeminen vie paljon aikaa. Haluttuun lopputulokseen pääseminen on lähes mahdotonta, koska objektien sijainnin koordinaatit kirjoitetaan käsin XML-tiedostoon. Lisäksi kenttätiedostoihin on saattanut tulla virheitä, jotka ovat aiheuttaneet ongelmia pelin toiminnollisuuksiin.

Kenttäeditori antaisi mahdollisuuden tavallisille pelaajille tehdä omia kenttiä. Tämä on käytännössä mahdotonta tällä hetkellä, mikä on pelin kannalta huono asia. Jos pelaajat pystyvät tekemään omaa sisältöä, pelin elinikä on yleensä pidempi ja kentät ovat parempia. Yrityksellä ei ole aikaisempaa sovellusta kenttien tekemiseen, joten kehittäminen ja suunnittelu aloitetaan tyhjästä. Vain pieniä suuntaviivoja on tiedossa esimerkiksi käyttöliittymän tyylin suhteen.

**Tutkittavan alueen rajaus**

Tutkimuksen rajat asetetaan tutkimuksen ja ohjelmistosuunnittelun osalta niin, että kehitettävä työkalu pystytään luomaan niin, että jatkokehitys sekä skaalautuvuus yrityksen mahdollisille seuraaville tuotteille mahdollistetaan.

**Toimeksiantaja**

Kinahmi Games Oy. Osoite Kynnystie 1 F 15, 40640 Jyväskylä. Toimiala Ohjelmistojen suunnittelu ja valmistus (TOL: 62010), Y-tunnus 26224555. Kinahmi Games Oy on perustettu vuonna 2014.

**Tehtävä ja tavoite**

Toteutuessaan sovellus helpottaa ja nopeuttaa kenttien tekemistä sekä muokkaamista. Sovellus auttaa myös kenttien suunnitteluvaiheessa hahmottamaan lopputulosta. Lisäksi se ehkäisee käyttäjästä johtuvia syntaksivirheitä sekä semanttisia virheitä. Tavoitteena on saada peliin parempia kenttiä ja lisää sisältöä.

## 2.2 Tutkimusmenetelmät

Opinnäytetyö tulee olemaan ohjelmiston kehittämistutkimus. Aluksi kartoitetaan yrityksen nykyinen toimintamalli havaintojen ja haastattelujen pohjalta. Havaintojen ja haastattelujen pohjalta kartoitetaan tarvittavat ominaisuudet ohjelmistolle.

Toimivia kehitysversioita annetaan testikäyttöön toimeksiantajalle, josta toivottavasti saadaan käyttäjäpalautetta. Käyttäjäpalautteen perusteella tehdään tarvittaessa muutoksia käyttöliittymään, visuaaliseen ulkonäköön sekä ohjelmiston rakenteeseen.

## 2.3 Tutkimuskysymykset

Tutkimuskysymyksiksi muodostuivat seuraavat kysymykset:

- Mitä ominaisuuksia kenttäeditorissa tulee olla?
- Kuinka kenttäeditori kannattaa toteuttaa?
  - Toteutetaanko alusta alkaen itse vai valmis editori?
  - Millä toteutustekniikoilla?

# 3 Kenttäeditori

Kenttäeditori on työkalu, jolla tehdään kenttiä tietylle pelimoottorille ja pelille. Pelit sisältävät tuhansia assetteja, jotka ovat koottu kenttäeditoriin. Kenttäeditori mahdollistaa kenttäsuunnittelijalle ympäristön, jossa hän pystyy luomaan uusia kenttiä käyttämällä näitä tai luomalla omia assetteja. Eikä hänen tarvitse tehdä mitään muutoksia pelin tai pelimoottorin koodiin. Kenttäeditorin käyttö on yleensä rajoittunut siihen peliin, jota varten se on tehty. (McShaffry & Graham 2013, 747; What Level Editor and Game Engine Should You Use n.d.)

Jotkin kenttäeditorit on tehty hyvinkin joustaviksi. Näissä editoreissa on annettu käyttäjille paljon erilaisia työkaluja, joilla he voivat luoda uutta sisältöä peliin. Käyttäjät ovat voineet tehdä uusia pelimuotoja ja toimintoja, joita ei alkuperäisessä pelissä ole ollut. Kun peliin tehdään uusia toiminnallisuuksia, syntyy siitä modi. Esimerkiksi Counter-Strike on Half-Life-peliin tehty modi, josta tuli myöhemmin kokonaan oma peli.(What Level Editor and Game Engine Should You Use n.d.)

Kenttäeditorit julkaistaan yleensä SDK:n (Software Development Kit) yhteydessä. SDK tulee usein joko pelin mukana tai ladattavana lisäosana. On myös mahdollista, että sitä ei julkaista ollenkaan. Jotkut peliyhtiöt pitävät kenttäeditorit vain omien työntekijöiden käytössä eivätkä anna pelin pelaajille tai yhteisölle mahdollisuutta luoda omaa sisältöä peliin. (What Level Editor and Game Engine Should You Use n.d.)

## 3.1 Historia

Vuonna 1983 Broderbund Software julkaisi tasohyppelypelin nimeltään Lode Runner, joka sisälsi pelin lisäksi kenttäeditorin. Tämä on yksi ensimmäisistä peleistä, joihin julkaistiin kenttäeditori mahdollistaen käyttäjille kenttien tekemisen ilman ohjelmointitaitoja. Lode Runnerin kenttäeditori on hyvin yksinkertainen kenttien tekemiseen tarkoitetun työkalu. Käytännössä se on pelkästään tiilien ja tikapuiden asettamista kenttään. Tämän lisäksi se oli myös loistava markkinointiväline ja kantava voima pelin suosioon nousemisessa. Computer Gaming World -lehti järjesti kilpailun siitä, kuka pystyy tekemään parhaimman kentän kyseiseen peliin. Tämän ansiosta

pelin myynti kasvoi ja syntyi maailman ensimmäinen peliyhteisö, joka tuotti lisäsisältöä peliin. (Top 10 level editors; Lode Runner's History n.d.)

Samana vuonna julkaistiin myös yksi ensimmäisistä modeista, Castle Smurfenstein, joka on parodia Silas Warnerin pelistä Castle Wolfenstein. Castle Wolfenstein oli natsiaiheinen peli, minkä takia se kiellettiin. Tämän takia peliyhtiö antoi pelaajille luvan muokata peliä. Castle Smurfenstein -peli näytti muuten samalta kuin alkuperäinen peli, mutta musiikki oli vaihdettu Barney-nimisen lastenohjelman tunnusmusiikkiin. Lisäksi kentässä olevat viholliset olivat muutettu smurffeiksi, ja pelin päävastus oli violetti hymyilevä dinosaurus. (Kushner 2004, 115−116.)

Tammikuussa 1994 ladattiin internetiin ohjelma nimeltä Doom Editor Utility, joka on kenttäeditori Doom-pelille. Tämä ohjelma oli ensimmäisiä 3d-grafiikkaa sisältävälle pelille tehtyjä kenttäeditoreja. Doom Editor Utility ei ollut pelintekijöiden vaan pelin fanien valmistama ohjelma. Tästä huolimatta Doom Editor Utilityllä pystyi muokkaamaan lähes kaikkia kenttien ominaisuuksia ja luomaan kokonaan uusia kenttiä. (Kushner 2004, 167−168.)

## 3.2 Ominaisuudet

Kenttäeditorin keskeisimmät tehtävät ovat objektien lisääminen karttaan, niiden ominaisuuksien muokkaaminen ja kentän tallentaminen tiedostoksi. Kenttäeditorin ominaisuuden ovat täysin riippuvaisia pelimoottorista ja pelistä, jolle se on tarkoitettu. Joihinkin editoreihin on myös integroitu muitakin kuin kentän muokkaamiseen tarkoitettuja työkaluja, esimerkiksi animaatioiden muokkaamiseen tarkoitettu työkalu. Yleensä animaatioiden muokkaamiseen on kokonaan erillinen ohjelma. (McShaffry & Graham 2013, 747.)

Hutchison (2009, 51−53) mukaan suositun Far Cry 2 -pelin kenttäeditorissa käyttäjällä on mahdollisuus tehdä seuraavia asioita:

- Maaston muokkaaminen
- Objektien lisääminen
- Tekstuurien maalaaminen
- Ympäristön asetukset (sää, vuorokauden aika, vedenpinnan korkeus jne.)

- Pelialueen rajat
- Teiden tekeminen.

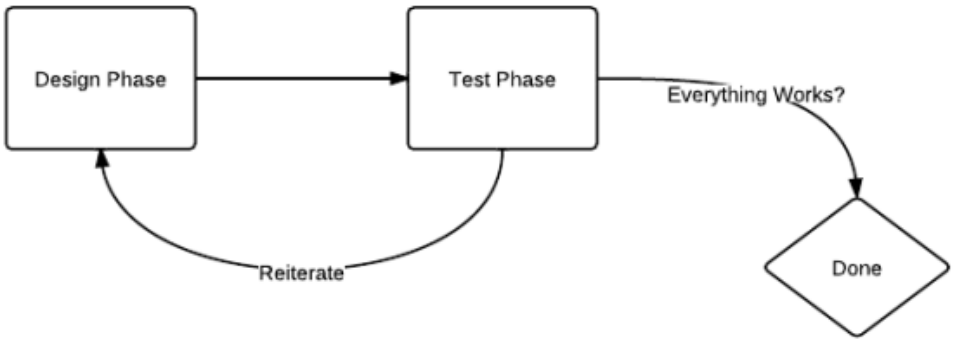
Valve Hammer Editorissa, joka on Source-pelimoottoria käyttäville peleille tarkoitettu kenttäeditori, voi käyttäjä tehdä muun muassa seuraavaa (Hammer n.d.):

- Luoda kentän arkkitehtuuria: geometria, tekstuurit ja valaistus
- Modelien lisääminen
- Entiteettien lisääminen
- Entiteettien skriptien kirjoittaminen
- Tekoälyn liikkumisen määrittäminen ja
- Kenttä-tiedostojen kääntäminen ja avaaminen.

Näiden kahden editorien ominaisuuksien vertailu osoittaa, että editoreja on hyvin erilaisia. Far Cry 2:n kenttäeditorissa pääpainona on ollut käyttäjäystävällisyys ja helppokäyttöisyys, kun taas Valve Hammer Editorissa on pyritty antamaan käyttäjälle mahdollisen monipuoliset työkalut toteuttaa kenttiä.

## 3.3 Editorin hyödyt

Miksi käyttää kenttäeditoria, kun teoriassa kaikki kenttätiedostot pystytään näyttämään tekstimuodossa? Visuaalisen sisällön tuottaminen ja muokkaaminen tekstinä on todella hidasta ja haastavaa. Kenttäeditorin tuomista hyödyistä suurin on niin sanottu WYSIWYG (What You See Is What You Get) eli mitä näet, sitä saat. Mitä editorissa näkyy tulee olemaan myös itse pelissä. Toinen suuri editorin tuoma etu on kentän suunnittelu- ja testausvaiheiden iteroinnin läpiviemisen nopeutuminen. (Pemmaraju 2012.)




Kuvio 1. Yksinkertaistettu vuokaavio kenttäsuunnittelusta (Pemmaraju 2012)

Editori tekee kenttien tekemisestä vähemmän koodimaista, helpommin ymmärrettävää. Kenttäeditori voi helpottaa kommunikointia kehittäjätiimin sisällä, etenkin ohjelmoijien ja graafikoiden välillä, koska graafikot eivät välttämättä ymmärrä ohjelmoinnista juurikaan. Visuaalinen tuki kommunikoinnissa helpottaa asian ymmärtämistä. (Pemmaraju 2012.)

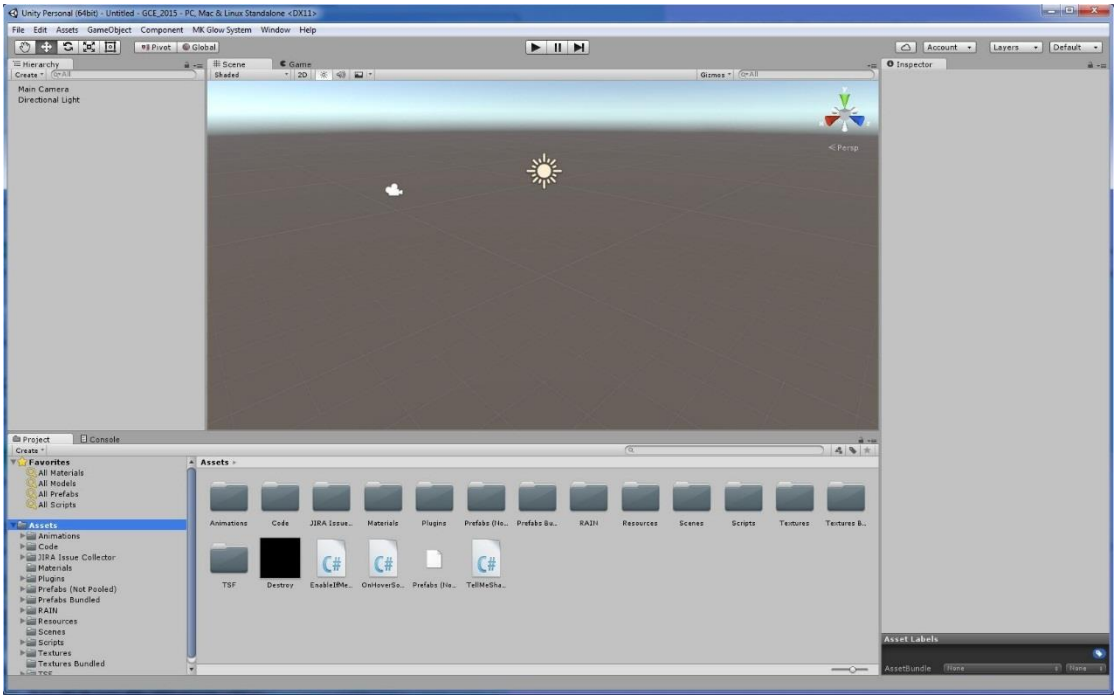
Kenttäeditori on myös markkinointiväline pelille. Editorin julkaiseminen yleiseen jakoon voi tuoda pelille paljon lisäsisältöä ilmaiseksi peliyhteisöltä. Tämä lisää pelin suosiota ja elinikää. Parhaita esimerkkejä, jossa peliyhteisö on ollut suuressa osassa pelin suosioon, ovat Doom, Counter-Strike ja Cities Skylines. Näistä viimeisin on julkaistu reilu vuosi sitten, ja peliyhteisö on tehnyt siihen jo yli 17 000 kenttää, 600 modia ja useita tuhansia rakennuksia sekä kulkuneuvoja. (Steam Community, Cities: Skilines n.d.)

# 4 Unity

Unity on Unity Technologiesin kehittämä pelimoottori ja editori, jonka avulla voidaan luoda pelejä ja muuta interaktiivista sisältöä. Unity on saavuttanut suuren suosion sekä helppokäyttöisyyden että monipuolisuuden ansiosta. Vuonna 2015 Unitylla oli 4,5 miljoonaa rekisteröitynyttä kehittäjää. Yli 600 miljoonaa ihmistä on pelannut pelejä, jotka ovat tehty Unitylla. (Unity n.d.)

Unityn editorissa voidaan luoda itse objekteja, liittää valmiita assetteja ja yhdistää näitä koodilla. Editori on rakennettu visuaalispohjaisella periaatteella, jotta kaikkea pystyy käyttämään raahaa ja pudota -menetelmällä, muuttujien määrittämisestä moniosaisten assettien luomiseen. Unity on saatavilla Windowsille, Linuxille ja Mac Os

X:lle, ja sillä voidaan tehdä tuotteita 21:lle eri alustalle kattaen mobiililaitteet, tietokoneet, konsolit, nettiselaimet, älytelevisiot ja virtuaalilasit. (Menard & Wagstaff 2014, 3; Unity n.d.)



Kuvio 2. Unityn käyttöliittymä

Unitysta on saatavilla ilmainen sekä maksullinen pro-versio. Ennen ilmaisversiossa oli rajoitettu osia pelimoottorin ominaisuuksista, mutta nykyään molemmissa versioissa on samat ominaisuudet. Pro-version mukana tulee lukuisia lisäpalveluita kuten pilvitallennus, analysointi- ja suorituskykytyökaluja. (Unity n.d.)

## 4.1 Historia

Ensimmäisen versio Unitystä julkaistiin kesäkuun 6. päivä vuonna 2005. Tämän tekivät kolme tanskalaista: David Helgason, Joachim Ante ja Nicholas Francis. Heidän tavoitteensa oli tehdä edullinen mutta ammattimainen pelimoottori amatööreille. Unityn ensimmäinen versio oli saatavilla vain Mac OS X:lle, ja sillä pystyi tekemään tuotteita vain muutamalle eri alustalle. Alussa Unityn käyttäjät muodostuivat lähinnä harrastajista ja itsenäisistä kehittäjistä. Potentiaalisilta asiakkailta meni muutama vuosi vakuuttua, että Unitya tullaan asianmukaisesti päivittämään. (Fear 2009; Brodkin 2013.)

Unityn versio 2 julkaistiin syksyllä 2007. Päivitys painoittui suurimmaksi osaksi suorituskyvyn parantamiseen, kuten tuki Microsoft DirextX:lle, joka nopeutti pelejä noin 30 prosenttia Windows-alustoilla. Päivitykseen kuuluivat myös muun muassa maastonmuokkaus työkalu, reaaliaikaiset varjot ja Unity Asset Server. (Unite 2007 – Keynote 2007.)

Joulukuussa 2008 julkaistiin erillinen tuote Unity iPhone, joka mahdollisti julkaisun iPhonelle. Tämän jälkeen Unity Technologies huomasi, että uudet käyttäjät olivat ostaneet Macintosh tietokoneen käyttääkseen Unitya. He aloittivat koodin uudelleenkirjoittamisen, jotta saisivat alustariippumattoman editorin. Vuonna 2009 julkaistiinkin versio 2.5, joka oli ensimmäinen myös Windowsia tukeva editori. (Fear 2009.)

Seuraava suuri julkaisu tapahtui syyskuussa 2010, jolloin tuli versio 3. Tässä versiossa Unityn erilliset editorit yhdistyivät kaikki saman ohjelman sisään. Muita tärkeitä lisäyksiä olivat: virheenkorjaus työkalu, FMOD-äänisuodatin sekä lukuisia grafiikkaa ja suorituskykyä parantavia ominaisuuksia. Version 3 julkaisuhetkellä Unitysta oli tullut maailman suosituin mobiililialustoille tarkoitettu työkalu, ja sillä oli yli 200 000 rekisteröitynyttä kehittäjää. Reilu vuosi myöhemmin julkaistiin Unity 3.5 ensiksi kokeiluversiona ja myöhemmin varsinaisena julkaisuna. Unity 3.5 versiossa lisättiin tuki Flash:lle, mikä mahdollisti pelien tekemisen verkkoselaimille. (Unity Technologies Delivers Unity 3. 2010; Unity Technologies Begins Unity 3.5 Open Beta With Flash Player Deployment 2011.)

Marraskuussa 2012 julkaistiin versio 4, joka mahdollisti pelien tekemisen linux-ympäristöön. Muita suuria muutoksia olivat Mecanim-animaatiojärjestelmä ja Shuriken-partikkelijärjetelmä. Mecanimin avulla pelinkehittäjät pystyivät tekemään korkeatasoisempia ja monimutkaisempia animaatioita. Shuriken mahdollisti aidomman näköiset partikkeliefektit. (The Next Generation of the Unity Game Engine Unveiled 2012.)

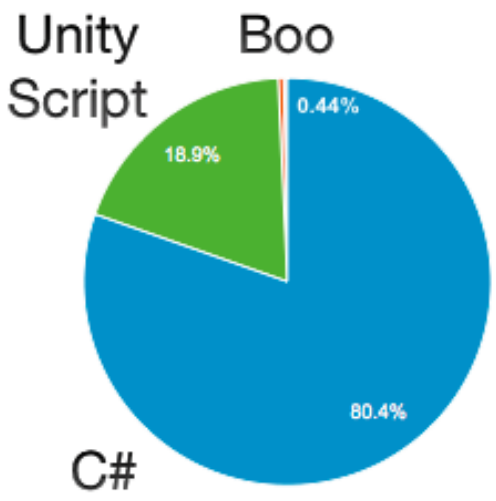
Tällä hetkellä viimeisin versio on 5, joka julkaistiin maaliskuussa 2015. Unity toi ensimmäistä kertaa kaikki editorin ominaisuudet myös ilmaisversion käyttäjille. Aiemmin vain pro-lisenssin ostaneet käyttäjät olivat voineet käyttää kaikkia ominaisuuksia. Tämä versio toi kehittäjille myös muun muassa editorin sisäisen

äänimikserin, PhysX 3.3-fysiikkamoottorin ja animaatiojärjestelmän parannuksia. Unity myös lopetti tuen Flash:lle, jonka tilalle tuli uusi WebGL-teknologia, joka mahdollisti sisällön käyttämisen uusimmilla verkkoselaimilla ilman lisäosia. (Unity 5 Is Here n.d.; Unity 5 n.d.)

Unitysta julkaistiin virallinen kokeiluversio Linux -käyttöjärjestelmille elokuussa 2015. Tässä versiossa oli tuki vain osalle alustoista, ja esimerkiksi kaikki konsolit ja iPhone olivat jääneet pois. Myöskään jatkokehittämisestä ei ollut takuita. (Bard 2015.)

## 4.2 Ohjelmointi

Unityssa voidaan käyttää kolmea ohjelmointikieltä: C#, UnityScript ja Boo. Suosituin ja tuetuin kieli on C#, jota käytetään yli 80 % ajasta. Vähiten käytetty kieli on Boo, jota käyttää vain muutama ohjelmoija. Vähäisen käytön takia Unity lopetti dokumentaatiotuen Boo-kielelle versiosta 5 eteenpäin.(Documentation, Unity scripting languages and you 2014.)




Kuvio 3. Unityssa käytetyt ohjelmointikielet (Documentation, Unity scripting languages and you 2014)

Unityssa käytetään Mono-ohjelmistokehitysympäristöä, jossa kaikkia ohjelmointikieliä ajetaan. Mono mahdollistaa kaikkien kolmen kielen käyttää Unityyn sisäänrakennettuja .NET-kirjastoja, joita edellytetään esimerkiksi tiedostojen käyttöön, verkkotoimintoihin ja XML-tiedostojen lukemiseen. Mono sisältää JIT-tulkin (just-in-time), joka kääntää lähdekoodin konekielelle ohjelman ajon aikana. Tämä

takia kaikki ohjelmointikielet ovat yhtä nopeita keskenään. (Menard & Wagstaff 2014, 144.)

Unity ei rajoita ohjelmointikielien käyttöä projekteissa. Yhdessä projektissa voidaan käyttää kaikkia kolmea kieltä. Funktioita voidaan kutsua toisella kielellä kirjoitetusta tiedostosta, ja GameObjectiin voidaan liittää eri kielellä kirjoitettuja kooditiedostoja.(Menard & Wagstaff 2014, 143−144.)

C# on Microsoftin kehittämä vahvasti tyypitetty ohjelmointikieli. Eniten käytetty ohjelmointikieli Unityssa. Unityn kaikki tutoriaalit ja dokumentaatioiden esimerkki käyttötapaukset kirjoitetaan ensimmäiseksi C#-kielellä. UnityScript, käytetään myös nimeä Unityn JavaScript, on Unityn kehitämä ohjelmointikieli. UnityScript on .NET-pohjainen muunnos JavaScriptista. Syntaksi muistuttaa hyvin paljon JavaScriptia, mutta semantiikaltaan nämä ovat hyvin erilaisia. (UnityScript versus JavaScript n.d.)

Boo on Rodrigo Barreto de Oliveiran kehittämä ohjelmointikieli. Se on .NET-pohjainen kieli, jonka syntaksi muistuttaa hyvin paljon Python-ohjelmointikieltä.

### 4.3 Unity Asset Store

Unity Asset Store on kolmannen osapuolen kauppapaikka, joka avattiin marraskuussa 2010. Asset Storessa pelinkehittäjät voivat myydä ja ostaa toistensa tekemiä tuotteita. Unity Technologies pitää 30 % myyntihinnasta itsellään ja antaa loput 70 % tuotteen tekijälle. Asset Store sisältää jo yli 15 000 tuotetta, osa näistä on ilmaisia ja osa maksullisia. (Sell Assets n.d.)

Asset Storessa myytävät tuotteet ovat kaikille pelikehityksen osa-alueille, muun muassa skriptejä, 3d-malleja, äänipaketteja ja editorilaajennuksia. Periaatteessa mitä tahansa, jota voidaan hyödyntään pelikehityksessä Unityssa. Jokainen tuote tarkistetaan laadun ja toimivuuden takaamiseksi, ennen kuin ne pääsevät kaupan sivuille. (Sell Assets n.d.)

## 5 Toteutus

Tässä luvussa käydään läpi, kuinka kenttäeditori toteutettiin.

## 5.1 Projektin aloittaminen

Kenttäeditoriprojekti lähti käyntiin alkuvuodesta 2015, kun toimeksiantaja Kinahmi Games Oy ehdotti sitä opinnäytetyön aiheeksi. Tuolloin olin vielä suorittamassa harjoittelua samassa yrityksessä. Alkuvaiheessa kaikki tapahtui vain epävirallisena jutteluna, eikä mitään virallista kirjattu ylös talteen. Keräilin ja pohdiskelin ideoita, joita tuli vastaan harjoittelun ohessa.

Varsinaisesti projekti käynnistyi kesällä saatuani harjoittelun päätökseen. Ensimmäiseksi pidimme aloituspalaverin toimeksiantajan kanssa, missä kävimme läpi keväällä syntyneitä ideoita ja ajatuksia. Palaverin edetessä alkoi projektin kokonaisuus ja laajuus hahmottua selkeämmäksi. Pääajatuksena oli tehdä editorista selkeä ja helppokäyttöinen.

Toimeksiantajan puolelta ei tullut rajoitteita pelimoottorin, ohjelmointikielen tai muidenkaan tekniikoiden suhteen. Pelimoottorin valinta kuitenkin tapahtui muutamassa sekunnissa. Valinta oli Unity, koska itse peli, jolle kenttäeditori tehdään, on tehty Unitylla. Ohjelmointikieli sekä muut kenttäeditoriin liittyvät ominaisuudet vaativat lisää tutkimista, joten niistä ei päätetty aloituspalaverissa.

Toimeksiantaja myös näytti aloituspalaverissa vanhaa kenttäeditoria, joka oli tehty pelin kehittämisvaiheessa. Editori oli pahasti keskeneräinen eikä se myöskään toiminut oikein, koska peli oli muuttunut paljon tämän jälkeen. Loppujen lopuksi vanhasta kenttäeditorista ei saanut mitään käyttökelpoista itse projektiin.

Aloituspalaverissa päätimme myös, että kenttäeditori tullaan julkaisemaan yleiseen jakoon eikä pelkästään pelin kehittäjille. Julkaisukanavaa ei pystytty lyömään lukkoon. Vaihtoehtoina oli integroida editori varsinaiseen peliin, julkaista editori erillisenä sovelluksena Steamissa tai yrityksen verkkosivuilla.

## 5.2 Määrittely ja suunnittelu

Aloituspalaverista saatujen tietojen pohjalta aloitin määrittelemään kenttäeditoria ja tutkimaan, millä tekniikoilla toteuttaisin editorin tekemisen. Avoimia kysymyksiä oli

tässä vaiheessa vielä monia. Suurin kysymys oli: käytetäänkö editorissa valmista pohjaa, josta räätälöitäisiin peliin sopiva editori, vai tehdäänkö editori alusta loppuun itse. Myös ohjelmointikieli oli avoin; vaihtoehtoina oli joko C# tai UnityScript.

### 5.2.1 Valmis editori vai omatekemä

Kenttäeditoreja on tehty ja julkaistu lukuisia kappaleita. Osa näistä on geneerisiä editoreja, eli ne eivät ole sidottuja yhdelle pelimoottorille tai pelille. Vaikka editoreja on tehty todella monia, niistä harvoin löytää täysin omalle pelille sopivaa, koska jokainen peli on erilainen.

Päädyin lopulta tekemään kokonaan omatekemän kenttäeditorin tutkittuani molempien vaihtoehtojen hyötyjä ja haittoja. Suurena syynä oli, että valmiista editoreista ei löytynyt täysin sopivaa. Vaikka olisi valinnut jonkin valmiin editorin, niin sitä olisi täytynyt muokata hyvin paljon, jotta se sopisi kyseiselle pelille. Valmiin editorin antama hyöty olisi jäänyt vähäiseksi.

Tutkin Unity Asset Storen valikoimaa: olisiko sieltä löytynyt kenttäeditoriin sopivia tuotteita. Suurin osa näistä tuotteista oli tarkoitettu maaston muokkaamiseen, jota ei Galactic Conquerors -pelissä ole ollenkaan. Kenttäeditoriin mahdollisesti sopivat Asset Storen tuotteet maksoivat 10–75 dollaria kappaleelta, joten niiden soveltuvuuden testaaminen olisi tullut kalliiksi. Yksi tuote, Runtime Level Editor, vaikutti lupaavalta, joten siihen perehdyin syvemmin.

**Valmis editori**

Runtime Level Editor on Unity Asset Storessa myynnissä oleva tuote, jolla on hintaa 30 dollaria. Se on geneerinen kenttäeditori projekti, jossa kaikki lähdekoodi on saatavilla. Tuotteen kehittäjän mukaan se sopii kaikenlaisten pelien kenttäeditoriksi, ja se on helppo konfiguroida. Ominaisuuksien puolesta vaikutti muuten hyvältä, mutta se on tehty vanhalle Unity versiolle, ja edellisestä päivityksestäkin on yli 1,5 vuotta.

Valmis editori kuulosti aluksi paremmalta idealta. Ei olisi tarvinnut ihan jokaista ominaisuutta tehdä itse. Jos yleisiä ominaisuuksia, kuten kentän tallentaminen tiedostoksi ja lataaminen tiedostosta, olisi valmiina, jäisi muiden ominaisuuksien tekemiseen enemmän aikaa. Kuitenkin lähes kaikki valmiista editoreista oli tarkoitettu 2D-

peleillä, joten niiden käyttö olisi ollut vaikeaa Galactic Conquerorsin kohdalla, mikä on 3D-peli.

Valmis editori on varmasti toimiva, ja siinä on osa ominaisuuksista valmiina. Suurimpina haittapuolina ovat mahdolliset ongelmatilanteet, joiden ratkaiseminen vaatisi lähdekoodin muokkaamista, sekä kenttäeditorin asettamat rajoitteet. Alla olevassa taulukossa on listattuna kaikki hyvät sekä huonot puolet valmiin editorin käyttämisestä.

Taulukko 1. Valmiin editorin hyvät ja huonot puolet

| Hyvät puolet | Huonot puolet |
|---|---|
| Luotettava ja testattu toimivaksi | Voi rajoittaa pelikohtaisia muokkausmahdollisuuksia |
| Vie vähemmän aikaa ohjelmoida | Huonommat jatkokehitysmahdollisuudet |
| | Lähdekoodi on vierasta ja ei välttämättä saatavilla |

**Omatekemä editori**

Alussa kuulosti työläältä tehdä koko editori alusta asti, mutta lopulta tämä oli varmasti nopeampi tapa saada projekti valmiiksi. Ongelmatilanteiden ratkominen oli helpompaa, kun lähdekoodi ei ollut vieraan ihmisen tekemää.

Tekemällä editorin kokonaan itse varmistutaan, että editorissa on kaikki ominaisuudet, jotka halutaan. Lisäksi kenttäeditori on juuri sellainen, kuin itse haluaa käyttöliittymän ja käytettävyyden suhteen. Taulukko 2 kaikista hyvistä ja huonoista puolista, jotka liittyvät omatekemään editoriin.

Taulukko 2. Omatekemän editorin hyvät ja huonot puolet

| Hyvät puolet | Huonot puolet |
|---|---|
| Räätälöity sopivaksi juuri kyseiselle pelille | Vie enemmän aikaa ohjelmoida |
| Paremmat jatkokehitysmahdollisuudet | Täytyy testata toimintoja enemmän |
| Lähdekoodi on tuttu | |

### 5.2.2 Ohjelmointikielen valinta

Unity mahdollistaa käyttämään kolmea ohjelmointikieltä, Boo, C# ja UnityScript. Tiputin Boon ensimmäisenä pois sen vähäisen käytön ja dokumentoinnin puuttumisen takia. Jäljelle jääneistä kielistä valinta olikin vaikeampi. UnityScriptin käytöstä oli kokemusta, ja näin ollen helpompi ja nopeampi aloittaa, kun taas peli on kirjoitettu C#-kielellä.

Valitsin lopulta toteutukseen C#-kielen, koska peli on kirjoitettu kyseisellä kielellä. Unityssa on mahdollista kutsua UnityScriptillä kirjoitetusta koodista C#-kielellä kirjoitettuja funktioita, mutta se ei ole suotavaa ja lisää mahdollisia ongelmatilanteita.

### 5.2.3 Vaatimusmäärittely

Teknisten vaatimuksien ja toteutustapojen määrittelyn jälkeen aloitin laatimaan toiminnallisia vaatimuksia. Kirjoitin vaatimuksia lyhyesti ylös palavereissa ilmi tulleiden ja itse kerättyjen taustatietojen perusteella, minkä jälkeen kävimme läpi ne toimeksiantajan kanssa. Tarvittavien korjauksien ja lisäyksien jälkeen laadin vaatimukset käyttäjäkertomuksiksi vaatimusmäärittely-dokumenttiin (liite 1).

### 5.2.4 Käyttöliittymän suunnittelu

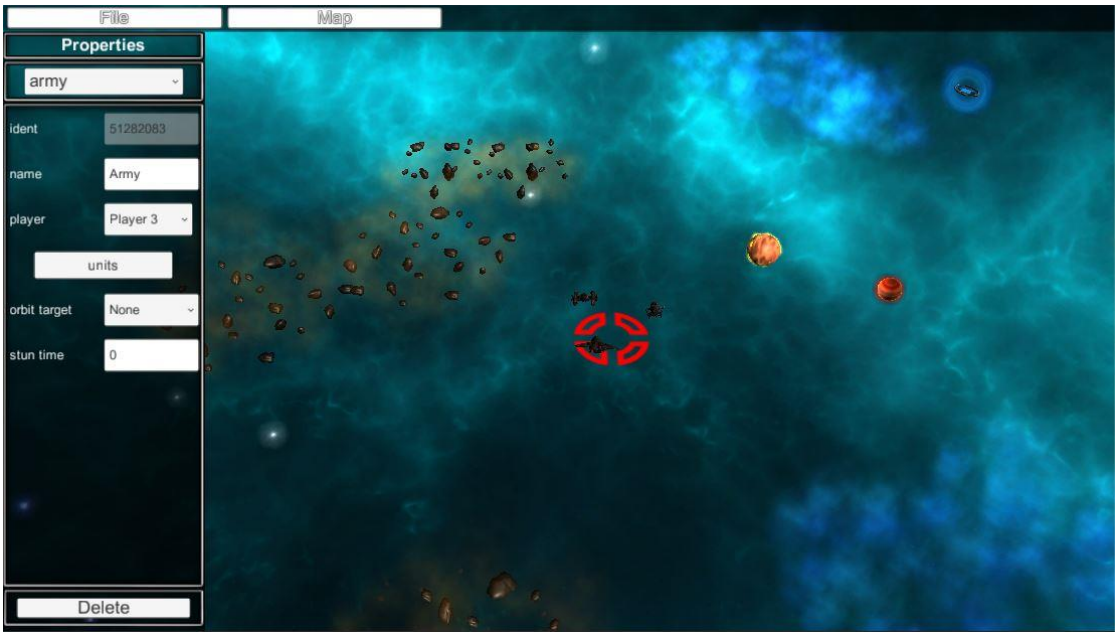
Käyttöliittymäsuunnitteluvaiheessa suunniteltiin käyttöliittymä kenttäeditorille sekä aloitusvalikolle. Toimeksiantajan toiveena oli saada käyttöliittymästä selkeä, helppokäyttöinen ja mahdollisimman intuitiivinen. Visuaalisen yleisilmeen pitäisi muistuttaa varsinaisen pelin käyttöliittymää.

Ennen käyttöliittymän suunnittelemisen aloittamista tutkin, millaisia käyttöliittymiä on tehty muihin kenttäeditoreihin ja editoreihin yleensäkin. Huomasin, että työkaluvalikot ovat yleensä joko vasemmalla tai ylhäällä. Joissakin tapauksissa molemmissa, mutta tämä johtui työkalujen suuresta määrästä. Myös valikkojen yhtenevä ulkoasu sekä neutraalit värit lisäävät käyttömukavuutta.

Kenttäeditorin käyttöliittymään suunniteltiin seuraavat näkymät:

- aloitusvalikko
- editorin yleisnäkymä (ei ole objektia valittuna)

- editorin objektin muokkausnäkymä (objekti valittu)
- objektin muokkaamisen lisävalikot (planeetan resurssit, armeijan yksiköt) ja
- editorin valikkojen näkymät (pelaaja-asetukset, kentän yleistietojen muokkaaminen).




Kuvio 4: Käyttöliittymä objektin muokkaamiseen

## 5.3 Kehittäminen

Määrittely ja suunnittelu -vaiheiden jälkeen siirryttiin kehittämisvaiheeseen, joka aloitettiin valitun kehitysympäristön pystyttämisellä.

### 5.3.1 Kehitys- ja toteutusympäristö

Kehitysympäristöksi valikoitui Unity 5 ja Visual Studio 2015:n yhdistelmä. Visual Studion tarjoamat työkalut auttoivat ja nopeuttivat huomattavasti työskentelyä C#-kielellä. Visual Studion suurimmat hyödyt itselle olivat: funktioiden kutsuhierarkian katsominen, virheiden etsimisen helpottuminen sekä ennakoiva komentosyöttö (muutaman kirjaimen syöttämisen jälkeen näkyy lista, josta voi valita haluamansa metodin tai muuttujan). Lisäksi Visual Studio voidaan integroida Unityyn, jolloin siitä tulee vieläkin tehokkaampi työkalu. Esimerkiksi Unityn puolella virheilmoitusta klikkaamalla aukeaa kooditiedosto Visual Studiossa kohdasta, jossa virhe on tapahtunut.

Versionhallintaan käytettiin yrityksen omaa palvelinta ja ohjelmaa nimeltä SourceTree. Versionhallintatyökalusta oli silti apua, vaikka tein projektia yksin, koska se mahdollisti projektin tekemisen kahdella eri koneella. Oli myös muutaman kerran tilanteita kehittämisvaiheessa, jossa kokeilemalla testattiin, voisiko asioita tehdä jollakin tavalla. Huomattuani, että kokeilu ei tuottanut haluttua lopputulosta, oli monia koodeitiedostoja ehditty muokkaamaan. Helpoin tapa palata takaisin alkupisteeseen oli versionhallintatyökalu, josta muutaman napin painallus riitti. Muiden projektinhallintatyökalujen käyttöä en kokenut tarpeelliseksi, koska tein projektia yksin.

Toimeksiantaja työskenteli samoissa tiloissa suurimman osan aikaa projektin kehitysvaiheesta, minkä ansiosta hän oli hyvin perillä projektin kehityksestä. Helppo ja nopea kommunikointikanava myös mahdollisti, että projekti menee koko ajan oikeaan suuntaan. Ei tullut tehtyä turhaa työtä, kun epävarmoissa tilanteissa sai nopeasti vastauksen.

### 5.3.2 Kehitysympäristön pystyttäminen

Kehitysympäristön pystyttäminen oli melko yksinkertaista, koska se vaati vain kolmen ohjelman asentamisen: Visual Studio, Unity ja SourceTree. Jopa Visual Studion integrointiin vaadittavat työkalut tulivat Unityn asennuspaketin mukana. Visual Studio ja Unity olivat käyttövalmiita heti asennuksen jälkeen. SourceTree vaati konfigurointia, jotta pystyttiin muodostamaan yhteys toimeksiantajan palvelimelle.

### 5.3.3 Ohjelmointi

Ohjelmointi aloitettiin, kun ohjelmistot oltiin saatu asennettua ja asetukset laitettua kuntoon. Kooditiedostojen kirjoittamista ei käydä yksityiskohtaisesti läpi. Ne tehtiin parhaaksi katsotulla tavalla siten, että vaatimusmäärittely-dokumentin vaatimukset täyttyivät. Alla käydään kuitenkin pääpiirteittäin läpi järjestys, jossa ohjelmointi toteutettiin.

Koko ohjelmoinnin ajan otettiin huomioon jatkokehitysmahdollisuudet. Tämä näkyi muun muassa pelin objektien muokkaamisen toteutuksessa. Esimerkiksi armeijalle tai planeetalle halutaan lisätä yksi muokattava ominaisuus. Tämä onnistuu lisäämällä

yksi sarake tauluun, jossa määritetään muokattavat ominaisuudet, minkä jälkeen ohjelma hakee automaattisesti käyttöliittymään tarvittavat tekstikentät ja nappulat sekä hoitaa muutosten tallentamisen.

Ohjelma aloitettiin luomalla omat osiot (Scenet) aloitusvalikolle ja itse kenttäeditorille. Tämän jälkeen haettiin kaikki pelin käyttämät kooditiedostot, joiden avulla pystyttiin lataamaan kentät tiedostoista. Ongelmilta ei kuitenkaan vältytty, koska peli on tehty pelaamiseen ja kenttäeditori muokkaamiseen. Joten jouduttiin tekemään paljon lisäyksiä, jotta objektit saatiin ladattua ja näytettyä oikein kenttäeditorissa.

Kun objektien lataaminen toimi oikein siirryttiin seuraavaan vaiheeseen. Tässä vaiheessa tehtiin objektien ja kameran liikuttamiseen tarvittavat funktiot. Myös objektien lisääminen, objektien ominaisuuksien muokkaaminen, pelaajien sekä kentän yleisasetuksen muokkaaminen ja kentän tallentaminen tiedostoon kuuluivat tähän vaiheeseen.

Viimeisessä vaiheessa tehtiin käyttöliittymien ulkoasut ja korjailtiin ilmenneitä ongelmia. Käyttöliittymät tehtiin suunnitelmien mukaisesti sekä toimeksiantajan toiveita kuunnellen. Käyttöliittymät saatiin yhtenäisen näköiseksi niin keskenään kuin pelin kanssa käyttämällä peliin tehtyjä tekstuureita.

## 5.4 Testaus

Kenttäeditori-projektiin ei tehty varsinaista testaussuunnitelmaa, koska kyseessä oli yhden ihmisen projekti ja käytettävä aika oli rajallinen. Testaussuunnitelman tekemiseen kulunut aika olisi ollut pois joltakin muulta osa-alueelta. Testaussuunnitelman puuttuminen ei kuitenkaan tarkoittanut, etteikö ohjelmaa olisi testattu. Vaatimusmäärittely-dokumentissa määritetyt toiminnallisuudet testattiin niiden toimivuuden varmistamiseksi.

Testausta tehtiin koko projektin ajan. Jokaisen toiminnallisuuden jälkeen testattiin, että kyseinen ominaisuus toimii niin kuin oli suunniteltu. Jokaisen isomman kokonaisuuden valmistuttua, esimerkiksi objektien lisääminen tai muokkaaminen, testattiin, että se täyttää vaatimusmäärittely-dokumentissa asetetut vaatimukset.

Projektin lopussa, kun kaikki vaatimusmäärittelyssä olevat toiminnallisuudet olivat valmistuneet, tehtiin vielä käyttöönottotestaus. Käyttöönottotestauksessa ilmenneet muutamat pienet virheet korjattiin ennen projektin päättämistä.

# 6 Tulokset

Projekti kesti arviolta noin puoli vuotta, jonka aikana saatiin tuotettua toimiva kenttäeditori, joka sisältää kaikki tarvittavat ominaisuudet kenttien tekemiseen. Kenttäeditori tehostaa toimintaa kenttien suunnittelussa sekä toteuttamisessa ja vähentää käyttäjästä johtuvia semanttisia virheitä.

Tarkempia tuloksia editorin tuomista hyödyistä saadaan julkaisun jälkeen käyttäjiltä tulevan palautteen sekä heidän tekemien kenttien tutkimisen myötä. Tällä hetkellä kaikki palaute on tullut pelkästään toimeksiantajalta, mikä on ollut sävyltään enimmäkseen positiivista.

Toimeksiantaja on tutkimuksen ensisijainen hyödyn saaja, sillä kenttäeditori kehitettiin tehostamaan juuri heidän liiketoimintaansa. Koko projektin ajan pidettiin mielessä editorin jatkokehittäminen ja muunneltavuus toimeksiantajan mahdollisille seuraaville peleille.

## 6.1 Tutkimuskysymykset

Tässä luvussa käydään läpi raportin alussa esitetyt tutkimuskysymykset ja vastataan näihin kysymyksiin.

**Mitä ominaisuuksia kenttäeditorissa tulee olla?**

Kenttäeditorien ominaisuudet vaihtelevat hyvin paljon riippuen pelistä, jolle kenttäeditori tehdään. Voidaan kuitenkin sanoa, että kenttäeditorin pakollisia ominaisuuksia pelistä riippumatta ovat:

- kentän avaaminen
- objektien lisääminen kentälle ja
- kentän tallentaminen.

Nämä ominaisuudet eivät kuitenkaan riitä edes kohtuulliseen kenttäeditoriin. Tutkimuksen kohteena olevaan kenttäeditoriin katsottiin pakolliseksi myös seuraavat ominaisuudet:

- kentän lataaminen tiedostosta
- objektien muokkaaminen
- kameran liikuttaminen ja
- kentän perustietojen muokkaaminen (nimi, pelaajamäärä, yms.)

**Kuinka kenttäeditori kannattaa toteuttaa?**

Kenttäeditorin toteuttamisessa kannattaa noudattaa yleistä ohjelmistokehityksen kaavaa: määrittely, suunnittelu, toteutus, testaus ja ylläpito. Näiden päävaiheiden läpikäyminen on sitä tärkeämpää, mitä suurempi ja monimutkaisempi kenttäeditori on kyseessä. Pienemmissä projekteissa usein määrittely ja suunnittelu jäävät vähäiseksi, mikä kostautuu toteutuksen aikana.

Alla olevien kahden alakysymysten vastaukset antavat lisää tietoa tähän tutkimuskysymykseen.

**Toteutetaanko alusta alkaen itse vai valmis editori?**

Kannattaako editori toteuttaa alusta asti itse vai käyttää valmista editoria pohjana on täysin riippuvainen pelistä, käytettävissä olevasta ajasta ja tekijästä? Pieniin peliprojekteihin on helpompaa löytää valmis editori kuin suuriin projekteihin. On myös mahdollista, että otetaan valmiita komponentteja projektiin, esimerkiksi maastonmuokkaustyökalu, ja tehdään loput ominaisuudet itse.

Valmiita editoreja löytyy lähes pelkästään 2D-peleille. Näistä editoreista voidaan saada pienellä muokkauksella sopivia omalle pelille. Tämä on vähemmän aikaa vievä tapa valmistaa kenttäeditori. Valmiit editorit kuitenkin saattavat rajoittaa muokattavia ominaisuuksia, jolloin niiden avulla tuotettujen kenttien taso laskee.

Tekemällä editorin alusta asti itse voidaan varmistua, että editori on juuri sellainen kuin halutaan. Tämä vie enemmän aikaa kuin valmiin editorin muokkaaminen sopivaksi. Itse tehdyssä editorissa on kuitenkin lähdekoodi tuttua, mikä helpottaa ongelmatilanteissa ja jatkokehityksessä. Valmiiden komponenttien käyttö editorissa nopeuttaa projektia, eivätkä ne tuota lisäongelmia, ellei niiden tarjoamia ominaisuuksia tarvitse lähteä muokkaamaan.

Tässä projektissa valittiin editorin tekeminen alusta asti itse. Syynä oli valmiiden editorien heikko sopivuus pelin ominaisuuksien kanssa. Lisäksi haluttiin, että editori on mahdollisimman helposti muokattavissa ja jatkokehitettävissä. Valmiita komponentteja otettiin käyttöön vain varsinaisesta pelistä, kolmannen osapuolen tekemiä komponentteja ei käytetty.

**Millä toteutustekniikoilla?**

Mikäli kenttäeditorin tekee kokonaan omaksi sovellukseksi, ei toteutukseen käytettäviä tekniikoita ole rajoitettu. Kuitenkin kannattaa käyttää samaa pelimoottoria ja ohjelmointikieltä kuin itse pelissä. Vaikka peli on tehty pelaamiseen, löytyy siitä paljon samoja elementtejä kuin kenttäeditorista. Jos kenttäeditori ja peli on tehty samoilla tekniikoilla, voidaan pelissä olevia komponentteja käyttää kenttäeditorissa, mikä vähentää työmäärää huomattavasti.

Tämä projekti toteutettiin täysin samoilla tekniikoilla kuin itse pelikin. Nämä tekniikat olivat Unity-pelikehitysympäristö ja C#-ohjelmointikieli. Samojen tekniikoiden käyttö mahdollisti koodin lainaamisen pelin puolelta, mikä nopeutti kenttäeditorin valmistumista.

## 6.2 Jatkokehitys

Koska projekti rajattiin siten, että yksi ihminen pystyy tekemään sen kohtuullisessa ajassa, jäi jatkokehitykseen paljon mahdollisuuksia. Esimerkiksi voidaan lisätä ominaisuuksia, jotka on listattu vaatimusmäärittely-dokumentin lopussa, sekä parannella ulkoasua. Parannusehdotuksia saadaan julkaisun jälkeen käyttäjäpalautteena ja heuristisiten arviointien perusteella tarvittaessa.

Kenttäeditoriin voisi myös integroida Steamin Workshopin, jolloin käyttäjien tekemät kentät pystyisi lataamaan suoraan internettiin muiden pelaajien saataville. Steamin Workshoppiin saisi kenttien lisäksi ladattua myös mahdolliset modit ja modelit, mikäli käyttäjät niitä tekevät. Kaikki lisäsisältö olisi yhdessä paikassa, mikä helpottaisi niin käyttäjiä kuin toimeksiantajaakin.

Käytettävyyden parantaminen on myös jatkokehitettävien asioiden listalla. Vaikka käytettävyyteen kiinnitettiin huomiota projektin aikana, voidaan sitä parantaa huomattavasti. Käytettävyysongelmia voidaan etsiä ja korjata heurististen eli kokemukseen perustuvien arviointien avulla (Käyttötuotteen heuristinen arviointi n.d.).

Jatkokehittäminen jäi projektin aikana avoimeksi. Jos jatkokehittämistä tapahtuu, niin se, kuka sen tekee ja milloin se tapahtuu, ovat toistaiseksi täysin avoinna. Lupauduin tekemään pieniä korjauksia, mikäli ongelmia ilmenee jo valmiina olevissa ominaisuuksissa. Täydellisestä ylläpidosta en kuitenkaan suostunut ottamaan vastuuta.

# 7 Pohdinta

Projekti oli kokonaisuudessaan erittäin haastava, vaikka kaikki käytettävät ohjelmistot ja tekniikat olivat tuttuja pois lukien C#-ohjelmointikieli. Galactic Conquerors on toimeksiantajan ensimmäinen peli, minkä takia lähdekoodi ei ole kaikista helppokäyttöisintä. Vaikka kenttäeditori on tehty kokonaan erilliseksi ohjelmaksi, käyttää se osittain samoja komponentteja pelin kanssa. Näiden komponenttien saaminen toimimaan editorissa oli suuri haaste. Ongelmia muodostui myös Unityn ohjelmistopäivitysten myötä, jotka saattoivat tuoda haluttujen ominaisuuksien lisäksi myös ei-toivottuja ominaisuuksia.

Vaikka projektissa tuli vastaan ongelmia, saatiin projekti valmiiksi ja lopputulos oli hyvä ottaen huomioon tämän olleen ensimmäinen suurempi Unity-projekti allekirjoittaneelle. Aikataulu oli suurin heikkous, sillä vaikka toimeksiantaja ei antanutkaan mitään määräaikoja, ei projekti valmistunut itsemääritellyn aikataulun puitteissa. Kuitenkaan suurta myöhästymistä ei tapahtunut.

Projektin käyttöliittymien graafinen ulkoasu ei onnistunut aivan täydellisesti. Tekijän taidot Photoshopin tai vastaavan työkalun käyttämiseen ovat hyvin rajoittuneet. Onneksi pelin puolelle tehdyistä grafiikoista löytyi lähes sopivia joka tilanteeseen, joten ulkoasusta tuli siedettävän näköinen.

Opinnäytetyön teoriaosuuden kirjoittaminen osoittautui vaikeammaksi kuin alkujaan oletettiin. Syynä tähän oli kattavien ja laadukkaiden lähteiden puuttuminen. Vaikka Unitysta on kirjoitettu kymmenittäin kirjoja, jokainen näistä käsittelee ainoastaan sitä, kuinka Unitylla tehdään pelejä. Kaikissa kirjoissa oli vain muutamalla lauseella esitelty itse pelimoottoria ja editoria. Kenttäeditoreista oli vieläkin haastavampaa löytää hyviä lähteitä, ja kaikki löytämäni kenttäeditoria käsittelevät kirjat olivat käyttöoppaita, eivätkä näin ollen käsitelleet editoreja yleisellä tasolla.

Kokonaisuudessaan projekti oli erittäin opettavainen. Vaikka aikaisempaa kokemusta oli projektityöskentelystä sekä Unityn käyttämisestä, oli yksin tehtäessä kuitenkin monta uutta asiaa, joihin ei ollut aikaisemmin kiinnittänyt huomiota. Tällaisia asioita olivat muun muassa ajankäyttö ja projektin kokonaisuuden hahmottaminen. Myös ohjelmointikieli oli kokonaan uusi tekijälle.

Projektin aihe oli mielenkiintoinen ja auttoi tekijää ymmärtämään kenttäeditorien hyödyllisyyden pelille. Tästä sekä muista projektin aikana opituista asioista on varmasti hyötyä tulevaisuuden projekteissa, varsinkin pelialalle liittyvissä projekteissa, mutta myös muutenkin.

# Lähteet


Bard, N. 2015. Unity Comes to Linux: Experimental Build Now Available. Viitattu 11.3.2016 http://blogs.unity3d.com/2015/08/26/unity-comes-to-linux-experimental-build-now-available/.

Brodkin, J. 2013. How Unity3D Became a Game-Development Beast. Viitattu: 3.3.2016. http://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/.

Documentation, Unity scripting languages and you. 2014. Viitattu: 15.3.2016. http://blogs.unity3d.com/2014/09/03/documentation-unity-scripting-languages-and-you/.

Fear, E. 2009. United they stand. Viitattu: 3.3.2016. http://www.develop-online.net/analysis/united-they-stand/0116643.

Hammer. N.d. Viitattu 6.5.2016. https://developer.valvesoftware.com/wiki/Category:Hammer.

Hutchinson, D. 2009. From Gamer to Game Designer: The Official Far Cry 2 Map Editing Guide. Boston: Course Technology PTR.

Kananen, J. 2010. Opinnäytetyön kirjoittamisen käytännön opas. Jyväskylän ammattikorkeakoulun julkaisuja -sarja.

Kananen, J. 2012. Kehittämistutkimus opinnäytetyönä. Jyväskylän ammattikorkeakoulun julkaisuja -sarja.

Kushner, D. 2004. Masters of Doom: how two guys created an empire and transformed pop culture. Yhdysvallat: Random House.

Lode Runner History. N.d. Viitattu: 24.2.2016. http://www.tozaigames.com/loderunner/history.

McSgaffry, M. & Graham, D. 2013. Game Coding Complete, Fourth Edition. Boston: Course Technology PTR.

Media lab Helsinki. N.d. Käyttötuotteen heuristinen arviointi. Viitattu 17.4.2016. http://mlab.uiah.fi/polut/Design/tyokalu_heuristinen_arvio.html.

Menard, M. & Wagstaff, B. 2014. Game Development with Unity, Second Edition. Boston: Gengage Learning PTR.

Pemmaraju, V. 2012. Make Your Life Easier: Build a Level Editor. Viitattu: 29.2.2016. http://gamedevelopment.tutsplus.com/articles/make-your-life-easier-build-a-level-editor--gamedev-356.

Sell Assets. N.d. Viitattu 7.5.2016. http://unity3d.com/asset-store/sell-assets.

Steam Community, Cities: Skilines. N.d. Viitattu: 15.4.2016. https://steamcommunity.com/app/255710/workshop/.

The Next Generation of the Unity Game Engine Unveiled. 2012. Viitattu: 7.3.2016. http://www.marketwired.com/press-release/the-next-generation-of-the-unity-game-engine-unveiled-1670512.htm.

Top 10 level editors. N.d. Viitattu: 24.2.2016. http://www.gamestm.co.uk/features/top-10-level-editors/2/.

Unite 2007 – Keynote. 2007. Video. Viitattu: 4.3.2016. http://unite.unity.com/archive/2007.

Unity – Game Engine. N.d. Viitattu: 20.3.2016. http://unity3d.com/.

Unity 5. N.d. Viitattu: 10.3.2016. http://unity3d.com/5.

Unity 5 Is Here. 2015. Viitattu: 10.3.2016. http://www.marketwired.com/press-release/unity-5-is-here-1997038.htm.

Unity Technologies Begins Unity 3.5 Open Beta With Flash Player Deployment. 2011. Viitattu: 7.3.2016. http://www.marketwired.com/press-release/unity-technologies-begins-unity-35-open-beta-with-flash-player-deployment-1601525.htm.

Unity Technologies Delivers Unity 3. 2010. Viitattu 6.3.2016. http://www.marketwired.com/press-release/Unity-Technologies-Delivers-Unity-3-1325564.htm.

UnityScript versus JavaScript. N.d. Viitattu 15.3.2016. http://wiki.unity3d.com/index.php/UnityScript_versus_JavaScript.

# Liitteet

Liite 1. Vaatimusmäärittelyn dokumentti

# Johdanto

Kinahmi Games Oy teki vuoropohjainen strategiapelin nimeltä Galactic Conquerors. Kyseisen pelin kartat olivat tehty muokkaamalla xml-dokumentteja tekstieditorilla. Tällä tavalla kenttien tekeminen oli hidasta sekä haastavaa, koska objektit saattoivat mennä päällekkäin sekä identit saattoivat olla vahingossa samoja, jolloin syntyi konflikteja kenttää pelatessa. Tämän ohjelmiston tarkoitus onkin visualisoida kenttien tekeminen ja muokkaaminen sekä vähentää käyttäjästä johtuvia semanttisia virheitä. Samalla helpottamaan sekä nopeuttamaan pelintekijöiden työtä kenttien suunnitellussa ja luomisessa. Tarkoituksena on myös julkaista ohjelmisto pelin pelaajille, jotta he voivat tehdä haluamansa näköisiä kenttiä, mikäli valmiit kentät eivät miellytä.

# Yleiskuvaus

Kenttäeditori on kokonaan erillinen ohjelmisto, eikä näin ollen vaadi pelin omistamista. Ensimmäisessä versiossa ohjelman on tarkoitus sisältää kaikki perusominaisuudet kenttien tekemiseen alusta loppuun. Käyttöliittymän tulee olla helppokäyttöinen ja selkeä, mutta sillä tulee kuitenkin pystyä hallitsemaan kaikkia ominaisuuksia.

# Tämänhetkinen tilanne

Tällä hetkellä kenttien tekeminen tapahtuu kokonaan tekstieditorilla. Käyttäjä kirjoittaa xml-tiedostoon jokaisen objektin koordinaatit sekä ominaisuudet erikseen. Tämän jälkeen käyttäjä avaa pelin, lataa kentän ja katsoo miltä se näyttää varsinaisessa pelissä. Jonka jälkeen käyttäjä suorittaa haluamansa muokkaukset, käynnistää pelin uudestaan ja katsoo kenttäänsä uudestaan. Tätä jatketaan niin kauan, kunnes kenttä on valmis.

## Toiminnalliset vaatimukset

1. **Käyttäjä pystyy luomaan uuden kentän.**

   **Kuvaus:** Käyttäjä voi aloittaa uuden kentän tekemisen.

   **Hyväksymiskriteerit:**

   - Kenttäeditori aukeaa tyhjän kentän kanssa.

2. **Käyttäjä pystyy muokkaamaan tallennettuja kenttiä.**

   **Kuvaus:** Käyttäjä pystyy valitsemaan listasta haluamansa kentän, jota lähtee muokkaamaan.

   **Hyväksymiskriteerit:**

   - Kentät pitää olla listattuna valikkoon
   - Listasta pystytään valitsemaan kenttä
   - kenttäeditori aukeaa ja lataa valitun kenttätiedoston

3. **Käyttäjä pystyy lisäämään objekteja kenttään.**

   **Kuvaus:** Käyttäjä voi lisätä haluamansa objektin kenttään.

   - Prefabit pitää olla listattuna valikkoon
   - Listasta voidaan valita haluttu prefabbi
   - Objekti luodaan valitulle prefabille ja sille alustetaan vakioarvot sekä ident

4. **Käyttäjä pystyy siirtämään objekteja kentällä.**

   **Kuvaus:** Käyttäjä voi siirtää objekteja kentällä raahaamalla niitä hiirellä.

   **Hyväksymiskriteerit:**

   - Objektit siirtyvät kursorin mukana, kun niitä raahataan

5. **Käyttäjä pystyy liikuttamaan kameraa.**

   **Kuvaus:** Käyttäjä voi siirtää kameraa kentässä. Käyttäjä pystyy zoomaamaan kameraa lähemmäs ja kauemmas. Käyttäjä voi pyörittää kameraa 360 astetta.

   **Hyväksymiskriteerit:**

   - Kameraa voidaan liikuttaa haluttuun kohtaan kentällä
   - Kamera liikkuu lähemmäs ja loitommas annetuilla pikanäppäimillä
   - Kamera pyörii annetulla pikanäppäimellä ja hiiren liikkeellä

6. **Käyttäjä pystyy poistamaan objektin kentältä**

   **Kuvaus:** Käyttäjä voi poistaa objektin kentältä painamalla poista-nappulaa

**Hyväksymiskriteerit:**

- Poista-nappula löytyy käyttöliittymästä
- Valittu objekti poistuu kentältä nappulaa painettaessa

7. **Käyttäjä pystyy valitsemaan objektin kentältä**

   **Kuvaus:** Käyttäjä voi valita objektin kentältä klikkaamalla sitä hiirellä. Valitessaan objektin aukeaa vasempaan laitaan näyttöä käyttöliittymä kyseiselle objektille.

   **Hyväksymiskriteerit:**

   - Objekti voidaan valita kentältä klikkaamalla sitä hiirellä
   - Käyttöliittymä aukeaa näytön vasempaan laitaan

8. **Käyttäjä pystyy muokkaamaan objektien ominaisuuksia**

   **Kuvaus:** Käyttäjä on valinnut objektin. Käyttöliittymästä käyttäjä voi muokata objektin ominaisuuksia. Muokattavat ominaisuudet vaihtuvat objektin mukaan.

   **Hyväksymiskriteerit:**

   - Objektin ominaisuuksia voidaan vaihtaa seuraavasti:
   - Planeetta:
     - Nimi
     - Kiertorata
     - Kiertoradan nopeus
     - Pyörimisnopeus
     - Prefab
     - Resurssit (aloitus, jäljellä, per vuoro)
     - Omistaja (pelaaja/neutraali)
     - Planeetalla oleva armeija yksiköineen
   - Armeija:
     - Nimi
     - Omistaja (pelaaja/neutraali)
     - Yksiköt
     - Stunniaika vuoroissa
   - Portaali:
     - Nimi
     - Prefab
     - Kohde portaali
   - Tähti:
     - Nimi
     - Prefab

- Muun tyyppiset objektit:
    - Prefab

9. **Käyttäjä pystyy vaihtamaan skybox-materiaalia.**

   **Kuvaus:** Käyttäjä voi valita haluamansa skybox materiaalin pudotusvalikosta.

   **Hyväksymiskriteerit:**

   - Skybox-materiaalit listataan pudotusvalikkoon
   - Pudotusvalikosta voidaan valita haluttu skybox

10. **Käyttäjä pystyy lisäämään ja poistamaan tähtijärjestelmiä(starsystem).**

    **Kuvaus:** Käyttäjä voi lisätä tähtijärjestelmiä enintään kahdeksan kappaletta. Käyttäjä pystyy poistamaan tähtijärjestelmiä, kuitenkin yksi tähtijärjestelmä on pakollinen.

    **Hyväksymiskriteerit:**

    - Tähtijärjestelmiä voidaan lisätä nappia painamalla
    - Tähtijärjestelmiä voidaan poistaa nappia painamalla
    - Tähtijärjestelmien maksimimäärä on kahdeksan (8)
    - Tähtijärjestelmien minimimäärä on yksi (1)
    - Yritettäessä poistaa viimeistä tähtijärjestelmää ohjelmisto vain tyhjentää tähtijärjestelmän poistamatta sitä.

11. **Käyttäjä pystyy vaihtamaan pelaajien aloitusresursseja ja liittoumaa.**

    **Kuvaus:** Käyttäjä voi avata valikon, josta näkyy pelaajien ominaisuudet. Käyttäjä voi vaihtaa jokaiselle pelaajalle eri aloitusresurssit ja liittouman.

    **Hyväksymiskriteerit:**

    - Pelaajat listataan valikkoon, josta nähdään kaikkien pelaajien aloitusresurssit, liittouma ja omistaako pelaaja planeettoja tai armeijoita.
    - Pelaajien aloitusresurssit voidaan vaihtaa
    - Pelaajien liittoumat voidaan vaihtaa

12. **Käyttäjä pystyy muokkaamaan kentän nimeä, kuvausta ja aloitusviestiä**

    **Kuvaus:** Käyttäjä voi avata valikon, josta näkee kentän yleistiedot. Käyttäjä voi muuttaa näitä tietoja.

    **Hyväksymiskriteerit:**

    - Kentän yleistiedot listataan valikkoon.
    - Kentän nimeä voidaan muuttaa
    - Kentän kuvausta voidaan muuttaa
    - Kentän aloitusviestiä voidaan muuttaa

13. **Käyttäjä pystyy tallentamaan kentän tiedostoon**

**Kuvaus:** Käyttäjä voi avata tallennusvalikon. Käyttäjä voi tallentaa kentän tiedostoon.

**Hyväksymiskriteerit:**

- Tallennusvalikko voidaan avata
- Kenttä voidaan tallentaa tiedostoon
- Tallennettaessa ohjelma ottaa huomioon vain ne pelaajat, joilla on vähintään yksi planeetta tai armeija

14. **Käyttäjä pystyy palaamaan aloitusvalikkoon editorista.**

    **Kuvaus:** Käyttäjä voi palata aloitusvalikkoon kenttäeditorista.

    **Hyväksymiskriteerit:**

    - Editorista pystyy palaamaan aloitusvalikkoon

15. **Käyttäjä pystyy sulkemaan kenttäeditorin.**

    **Kuvaus:** Käyttäjä voi sammuttaa kenttäeditorin.

    **Hyväksymiskriteeri:**

    - Kenttäeditori voidaan sammuttaa

16. **Käyttäjä pystyy käynnistämään pelin.**

    **Kuvaus:** Käyttäjä voi käynnistää pelin kenttäeditorista.

    **Hyväksymiskriteerit:**

    - Peli käynnistyy normaalisti

## Ei toiminnalliset vaatimukset

17. **Tulee toimia samoilla minimivaatimuksilla kuin itse pelikin.**
    - Käyttöjärjestelmä: Windows XP SP2, Mac OS X 10.8, Ubuntu 12.04
    - Prosessori: 2 Ghz Dual-Core
    - Muisti: 1 GB RAM
    - Näytönohjain: Shader Model 2.0 yhteensopiva, sekä 64 MB VRAM
    - Direct X version 9.0
18. **Tulee toimia ilman Galactic Conquerors -pelin asentamista.**
19. **Ohjelma tulee tehdä Unity3D-pelimoottorilla.**
20. **Ohjelma tulee kirjoittaa C#-ohjelmointikielellä.**

# Jatkokehitys

Ensimmäiseen versioon ei rajallisen ajan takia pystytä tekemään kaikkia ominaisuuksia, tässä luvussa listataan ominaisuuksia, jotka karsittiin pois.

- Pelialueen rajat säädettäviksi
- Planeetoille valmiita rakennuksia
- Muokkauksen kumoaminen ja uudelleentekeminen (undo/redo).