jamk.fi

Selainpohjainen 3D-peli

Case: Maagiset Madot

Jesse Friman

Opinnäytetyö
Marraskuu 2016
Luonnontieteiden ala
Tradenomi (AMK), tietojenkäsittelyn tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Friman, Jesse	Opinnäytetyö, AMK	Marraskuu 2016
	Sivumäärä 44	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Selainpohjainen 3D peli
Case: Maagiset Madot

Tutkinto-ohjelma
Tietojenkäsittelyn tutkinto-ohjelma

Työn ohjaaja(t)
Jarkko Immonen

Toimeksiantaja(t)
Kauko Sihvonen

Tiivistelmä

Web-teknologiat sekä web-selaimet pelinkehityksessä ovat kehittyneet merkittävästi, mikä mahdollistaa monimutkaisten ja näyttävien sekä raskaiden pelien toteuttamisen web-selainympäristössä. Selainpohjaisiin peleihin, etenkin WebGL-teknologiaan on pitkään liittynyt erinäisiä ongelmia. Tietokoneiden sekä web-tekniikoiden kehittyessä ovat nämä ongelmat siirtymässä pois, vieden pelinkehitystä enemmän selainpainotteiseen suuntaan. Selainpohjaisten pelien kehittämisessä käytetään lukuisia ohjelmakirjastoja, mitkä voivat toimia myös sovelluskehyksenä. Näiden tarkoitus on helpottaa pelinkehittämistä ja tuomaan monimutkaisia ominaisuuksia, joita peleihin nykyisin toivotaan, kuten pelimoottorit. HTML5-pelit voi olla perinteisten 2D-pelien lisäksi myös 3D-pelejä sekä WebGL-tekniikkaan perustuvia pelejä.

Maagiset Madot on toimeksiantaja Kauko Sihvosen kehittelemä puinen, fyysinen pulmapeli lapsille ja aikuisille. Toimeksiantajalle oli aiemmin tehty 3D-mainosvideo tukemaan mainontaa. Myöhemmin hän halusi keksimästään tuotteestaan nykyaikaisen digitaalisen pelin lisäämään omalle tuotteelleen lisäarvoa sekä tuomaan markkinoille uuden tuotteen. Tehtäväksi muodostui tutkia, minkälaisia teknologioita selainpohjaisiin peleihin kuuluu ja kuinka 3D-peli toteutetaan web-selaimelle. Tutkimus toteutettiin kehittämistutkimuksen muodossa, ja tavoitteena oli saada toimeksiantajalle suunnitelman mukainen selainpohjainen 3D-peli. Toteutustapa oli etsiä oikeat työkalut ja tekniikat, joiden avulla peli saatiin toteutettua. Työssä myös hankitaan tietoa ja teoriaa selainpohjaisten pelien kehityksestä, tukemaan laadullista osuutta. Peliin liittyi ominaisuuksia, jotka vaativat monipuolisen ohjelmakirjaston. Pelinkehityksen vaiheisiin liittyi pieniä, mutta aikaa vieviä ongelmia. Ongelmakohdat oli kuitenkin ratkaistavissa, prosessi eteni suunnitelman mukaisesti, joten peli saatiin toteutettua odotetulla tavalla.

Avainsanat (asiasanat)
3D, pelinkehitys, WebGL, HTML5

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Friman, Jesse	Bachelor's thesis	November 2016
		Language of publication: Finnish
	Number of pages 44	Permission for web publication: x

Title of publication
Browser based 3D game
Case: Maagiset Madot

Degree programme
Business Information Systems

Supervisor(s)
Immonen, Jarkko

Assigned by
Sihvonen, Kauko

Abstract

Web technologies and web browsers in game development have developed significantly, which enables the implementation of complex and ambitious as well as huge games in web browsers. There have been particular issues for long time in browser based games and in particular with the WebGL technology. The issues are disappearing during the progression of computers and web technologies, leading game development towards more browser based orientation. Numerous libraries have been used on browser based game development which could also be used as a framework. The purpose of these libraries is to make game development easier and offer complex features such as game engines, which are expected to be included in games today. HTML5 games could be 3D games as well as WebGL based games along with the traditional 2D games. Maagiset Madot is a physical, wooden puzzle game for children and adults invented by the thesis client Kauko Sihvonen who had previously ordered 3D commercial animation to support advertising. Later, he a wanted modern digital game for his invented product to increase the value of his original product as well as get a new product on the markets. The assignment consisted of a research on technologies used in browser based game development and implementation of 3D games on a web browser. The research was carried out in a form of development research and the goal was to accomplish a browser based 3D game for the client. The method was to search proper tools and technologies, using which the game could be accomplished. The information and theoretical material about development of browser based games were gathered to support the qualitative side of the project. The game included features requiring a versatile library. Small but time consuming issues were associated in the phases of game development, however, they were resolved at the end, and the process proceeded as planned, therefore, the game was accomplished as expected.

Keywords/tags (subjects)
3D, game development, WebGL, HTML5

Miscellaneous

Sisältö

Kuviot

Määritelmät ja sanasto

API	Ohjelmointirajapinta
Canvas	HTML-elementti grafiikan piirtoa varten
Edge	Verteksien liitos
Face	Verteksien muodostama pinta
Framework	Sovelluskehys mikä toimii runkona ohjelmistolle
GLSL	Varjostinohjelmakieli
HTML5	Kieli verkkosivujen toteutukseen
JavaScript	Ohjelmointikieli
JSON	Tiedostomuoto
Library	Ohjelmakirjasto
Polygon mesh	Polygoneista muodostuva kokonaisuus
OpenGL	Ohjelmointirajapinta tietokonegrafiikan toteutukselle
Plugin	Liitännäinen
Polygon	Verteksien muodostama alue
Puzzle	Pulmapeli sekä ongelmanratkaisu peligenre
User Interface(UI)	Käyttöliittymä
UV -map	Kaksiulotteinen levityskuva polygonista

Vertex/vertice	Tietokonegrafiikassa esiintyvä avaruudellisia arvoja kuvastava tietorakenne
WebGL	JavaScript API interaktiivisen 2D- ja 3D-grafiikan renderöintiin web-selaimella

1 Johdanto

Selainpohjaisia pelejä on ollut olemassa jo pidemmän aikaa, ja näitä on toteutettu perinteisesti 2D-peleinä sekä myöhemmin 3D-peleinä. Peleistä tehdään kokoajan näyttävämpiä, laajempia ja raskaampia, joten ne vaativat alalla kehittyviltä teknologioilta yhä enemmän. Uusin teknologia on mahdollistanut pelien toteutuksen myös mobiililaitteille perinteisten konsoleiden sekä tietokoneiden lisäksi. Pelejä on saatavilla joko asentamalla laitteeseen tai pelaten suoraan web-selaimella. Web-selaimella toimivien pelien etu on siinä, että niitä voidaan pelata millä laitteella tahansa, sijainnista riippumatta. Näitä web-selaimelle toteutettavia pelejä tuottavat peliyhtiöt, joilla on henkilöstöä ja resursseja toteuttaa isompiakin hankkeita. Tämän päivän kehittyneiden tekniikoiden myötä on mahdollista, että pienetkin yritykset ja jopa vain yhden hengen yritykset voisivat toteuttaa pelejä, toimimalla joko pääurakoitsijana tai alihankkijana.

Tässä kehittämistutkimuksessa tutkitaan, miten toimeksiantajan ehdottama digitaalinen versio hänen pulmapelistään voitaisiin toteuttaa. Peli halutaan saada kaikkien ulottuville, joten siitä on tarkoitus toteuttaa web-selaimella toimiva 3D-peli tietokoneille. Tutkimuksesta pyritään nostamaan esille, kuinka yrittäjän näkökulmasta tällainen projekti on mahdollista toteuttaa kohtuullisen ajankäytön sekä matalien resurssien sisällä. Tämän lisäksi itse pelin on tarkoitus tuoda lisäarvoa sekä tunnettavuutta toimeksiantajan jo olemassa olevalle, hänen keksimälleen pulmapelille.

Tutkimusasetelmassa käydään läpi toimeksiantaja, hänen Maagiset Madot tuotteensa sekä lähtötilanne. Siinä mietitään tavoitteet sekä rajataan tutkimusalue ja selvitetään työn tutkimusmentelmät ja kysymykset. Asetelmassa käydään myös läpi projektin tarkoitus. Työ aloitetaan tutkimalla pelinkehitystä etenkin selainpohjaisten pelien saralta. Itse pelin toteutukseen määritellään sopivat työkalut ja tekniikat vertailemalla niitä toisiinsa sekä tutkimalla näiden teoreettista taustaa. Työn jälkeen pohditaan tuloksia, miltä peli näyttää, kuinka se toimii ja vastasiko se toimeksiantajan määritelmiä sekä tutkimuskysymyksiä.

2 Tutkimusasetelma

2.1 Maagiset Madot ja toimeksiantaja

Toimeksiantajana toimiii henkilö Kauko Sihvonen, joka on keksinyt ja patentoinut Maagiset Madot-pulmapelin.

Kuvio 1. Maagiset Madot

Pulmapeli on fyysinen, usein useasta palasta koostuva tuote, ja pelissä tehtävänä on koota näistä paloista tietynlainen muoto, joko kaksi- tai kolmiulotteisena (Allan 2015). Kauko Sihvosen Maagiset Madot pulmapeli on juuri sellainen. (Ks. Kuvio 1.) Tässä pelissä on neljä palaa ja niistä saadaan kolme erilaista muotoa yhdistelemällä ne toisiinsa oikein. Pelin idea on saada muodot mahdollisimman nopeasti, jolloin voidaan kilpailla siitä, kuka on nopein. Haastavuutta voidaan lisätä hankkimalla useampi pakkaus ja sekoittamalla ne keskenään, sillä jokaisen pakkauksen palat ovat uniikkeja, eikä niitä pysty yhdistämään toisten pakkausten paloihin. Peli on suunnattu niin aikuisille kuin myös lapsille.

Hänen tuotettaan on myyty messuilla ja muissa erilaisissa tapahtumissa. Myyntiä pyritään myös jalkauttamaan jälleenmyyntiin eri kauppoihin ympäri Suomea. Tuotetta on myös markkinoitu muissa Euroopan maissa sekä pian myös Aasiassa.

Toimeksiantaja toivoo saavansa lisää myyntiä ja näkyvyyttä omalle tuotteelleen saamalla siitä digitaalisen version markkinoille. Tämän odotuksen pohjalta pelin tyyli määräytyy osittain mainostavaksi sekä toimeksiantajan alkuperäistä tuotetta esitteleväksi.

2.2 Tausta, tavoitteet ja rajaus

Toimeksiantajalle on tarve saada digitaalinen versio hänen olemassa olevasta pulma-pelistään, jotta saisi näkyvyyttä ja lisäarvoa hänen tuotteelleen. Pelin digitaalisen version avulla toimeksiantaja myös pyrkii saavuttamaan uuden markkina-alueen. Tavoitteena on saada digitaalinen pulmapeli toimimaan web-selaimella, jotta käyttäjillä olisi esteetön pääsy pelaamaan peliä millä tahansa internetyhteyden omaavalla laitteella. Toinen seikka on selainpohjaisten pelien suuri suosio ja helppokäyttöisyys. Tässä työssä toteutuva kehittämistutkimus on hyvin tärkeässä asemassa pienille yrityksille ja yksityisille toimijoille, sillä usein yritysten lelut, fyysiset pelit ja lautapelit sekä muut harrastusvälineet jäävät toteuttamatta digitaalisessa muodossa. Työssä käsitellään vain pelin toteuttamisen vaiheita ja siihen liittyviä teknologioita. Koska tavoite on selkeä ja toteutus on virtaviivainen, saadaan tarkka rajaus toteutettua.

Maagiset Madot pulmapelin digitaalisen version idea, rakenne ja visuaalinen ilme toteutetaan toimeksiantajan ja tekijän yhdessä tehdyn suunnitelman mukaan. Toimeksiantaja toivoo pelin muistuttavan mahdollisimman tarkasti tuota olemassa olevaa pulmapeliä, joten se on yksi tavoitteista. Työn keskeinen tavoite on toteuttaa selainpohjainen peli nimeltään Maagiset Madot, viitaten toimeksiantajan samannimiseen pulmapeliin. Työssä on tarkoitus toteuttaa lyhyehkö 3D-pelidemo, mikä on itsessään kokonainen tuote. Peliä on tarkoitus laajentaa sekä jatkokehittää tämän työn jälkeen.

Pelinkehitys ja koko työn toteutus rajataan siten, että pelin tulee toimia vain Chrome-selaimella, sen versiosta 50 ylöspäin. Käytettävän laitteen on oltava nykypäiväinen tietokone ja näytönohjaimen on tuettava OpenGL 2.0-tekniikkaa. Muita web-selaimia, kuten Internet Explorer, Mozilla Firefox ja Safari sekä mobiililaitteita ei ole luvattu toimimaan pelin kanssa. Tavoitteellinen osuus itse

pelistä on toteuttaa sen graafinen sisältö ja pelin mekanismi kokonaisuudessaan, mutta äänimaailma muodostuu vasta jatkokehityksen myötä. Tämän lisäksi pelin vienti palvelimelle ja domainin hankkiminen eivät kuulu tämän projektin alle.

Tekniikoista käydään läpi vain tutkimuksen työhön vaikuttavia asioista, joten sovelluskehyksistä tai 3D-mallintamisesta tutkitaan vain pelinkehitykseen liittyvät kohdat.

2.3 Tutkimusmenetelmät

Opinnäytetyö toteutetaan kehittämistutkimuksena. Kyseinen tutkimusmenetelmä soveltuu parhaiten tämän tutkimuksen aiheeseen, sillä tutkimuksessa pyritään löytämään olemassa olevaan ongelmaan ratkaisu ja päätyä lähtötilanteesta parempaan lopputulokseen. Lisäksi tutkimuksessa kartoitetaan aiheeseen liittyviä teorioita, joiden pohjalta tutkimuksen työ lähdetään toteuttamaan. (Kananen 2012, 19-21.)

Työssä esiintyy myös Case-tutkimuksen piirteitä, sillä tutkimuksessa on saatava syvä ymmärrys tutkittavasta aiheesta, mikä on toimeksiantajan digitaaliseksi kehitettävä tuote. Tutkija toimii työssä aktiivisena toimijana ja ainoana tekijänä, jonka muutoksien pohjalta havainnoidaan tutkimuksen ja työn tulokset. Näin tutkimus on osalti myös toimintatutkimusta. (Kananen 2012, 25-27.) Case-tutkimus päättyy ja siirrtyminen toimintatutkimuksen muotoon tapahtuu siinä vaiheessa, kun käsiteltävää kohdetta aletaan muuttamaan, eikä se jää vain toteamisvaiheeseen (Kananen 2012, 37). Tutkimus siis etenee Case-tutkimuksesta toimintatutkimukseen, mutta kehittämistutkimus on kuitenkin tutkimuksen vallitseva menetelmä, sillä kyseessä on tuotteeseen liittyvä ongelma, johon pyritään löytämään ratkaisu. Tuotteelle suunnitellaan kehitystoimenpiteet ratkaisemaan kyseinen ongelma ja parantamaan nykyinen tilanne. Nämä ovat kehittämistutkimuksen tunnusomaisia piirteitä (Kananen 2012, 41).

Tällä hetkellä toimeksiantajalla on ainoastaan Maagiset Madot-pulmapeli sekä aiemmin tehty 3D-animaatio mainosvideo tukemaan näkyvyyttä ja mainontaa. Pelin digitalisointia on lähdetty ideoimaan kyseisen animaation pohjalta. Tämän työn tarkoitus on toteuttaa toimeksiantajan olemassa olevan pulmapelin digitaalinen

versio hänen sekä työn toteuttajan suunnitelmien, vaatimusten ja rajausten mukaisesti.

Maagiset Madot-pulmapelin digitaalisen version voi toteuttaa usealla eri tavalla, sillä pelinkehityksen teknologioita on lukematon määrä. Se minkälainen totetustapa valitaan määräytyy projektin lyhyestä ajasta, tekniikan soveltuvuudesta web-selaimelle sekä yhteensopivuudesta mobiililaitteille. Yhteensopivuus mobiililaitteille on otettava huomioon projektin jälkeisen jatkokehityksen vuoksi.

Tutkimuksessa hyödynnetään alan kirjallisuutta, dokumentaatiota, artikkeleita sekä tarkastellaan samankaltaisia web-selaimella toimivia pelejä. Peliin liittyvä tieto tutkimuksessa kerätään haastattelemalla toimeksiantajaa. Näitä tietoja ovat hänen keksimänsä tuotteen tilanne ja lähtökohta. Haastattelun pohjalta muodostetaan pelin vaatimukset, sisältö ja toiminnallisuudet ideoiden, havaintojen sekä tarpeiden myötä. Opinnäytetyön toteuttaja osallistuu vahvasti pelin suunnitteluun sekä ideointiin toimeksinantajan kanssa. Pelistä esitellään näytteitä, otantaa sekä demoja toimeksiantajalle. Näiden pohjalta toimeksiantaja antaa palautetta, niin mahdollisia muutosideoita kuin myös visuaalisen sisällön ja toiminnallisuuksien muutoksia.

Projekti toteutetaan alustamalla vahva teoreettinen pohja ja käsitys teknologioista, selainpohjaisten pelien käytettävyydestä ja yleisilmeestä sekä hankkimalla oppia ja tietoa pelinkehityksen vaiheista. Teorian jälkeen, projektin alkuvaiheessa tehdään kokeiluja ja karkeita testejä vaadittavien kriteerien tavoittamiseksi. Näitä ovat siis ne tekijät, joista on mainittu kohdassa 2.2. Suurin osa ominaisuuksista todennetaan toimivaksi ja hyväksi testaamalla näitä piirteitä. Testaamiseen liittyy suorituskyvyn yksinkertainen arviointi, ja tämä tapahtuu kokeilemalla toteutettuja toimintoja ja pelaamalla peliä. Testaus voidaan todeta läpäistyksi, kun tarkasteltava toiminnallisuus toimii halutulla tavalla sekä peli ei visuaalisesti ilmennä ongelmia. Tämän jälkeen voidaan toimeksiantajalle suunniteltua peliä lähteä toteuttamaan.

Peliä esitellään toimeksiantajalle siinä vaiheessa, kun pelistä on jotain näytettävää. Tämä tapahtuu, kun pelissä on visuaalista sisältöä ja yleisilmeestä saadaan ymmärrys. Esittelyjen jälkeen peli voidaan viedä valmiiksi asti, jonka jälkeen seuraa viimeistely ja lopulliset muutokset, mikäli näitä ilmenee.

2.4 Tutkimuskysymykset

Tavoitteista ja tehtävästä nousi tutkimukselle merkittäviä kysymyksiä, joihin oli saatava ratkaisu osoittamaan tutkimuksen työn tuloksia. Nämä tutkimuskysymykset muodostuivat seuraavanlaisiksi.

Millaisilla tekniikoilla selainpohjainen 3D-peli voidaan toteuttaa?

Kuinka toteutetaan digitaalinen versio Maagiset Madot pelistä?

3 Pelinkehitys web-selaimille

3.1 Historia ja tulevaisuus

Web-selaimella toimivia pelejä on tehty lukuisia. Ensimmäisiä pelejä esiintyi 1990-luvun alkupuolella. Kehitys sekä pelien suosio on ollut huomattava tuosta ajasta tähän päivään. Ensimmäiset pelit toteutettiin DHTML-, eli Dynamic HTML-kieltä käyttäen. Kyseinen tekniikka omasi dynaamisia toimintoja, kuten alasvetovalikoita ja yksinkertaisia animaatioita. Tämä tarkoitti myös sitä, että pelit olivat yksinkertaisia ja teksipainotteisia verraten nykypäiväisiin, visuaalisiin ja tehostetäytteisiin peleihin. Tunnettuja teknologioita, joita käytettiin ja käytetään liitännäisinä web-selaimiin ovat Flash, Unity, Java sekä Shockwave. Myös palvelinpään kielet Java, Python, Perl, ASP, Ruby sekä PHP liittyivät osaksi selainpelien rakennetta, mahdollistamaan moninpeli ominaisuuden ja käyttäjätunnuksen omaavia pelejä. Tänäpäivänä niin sanotut taitopelit (*engl. skill games*) sekä uhkapelit (*engl. gambling games*) ovat suuressa suosiossa, koukuttavuuden vuoksi. Peleissä esiintyy myös pelinsisäisiä maksuja (*engl. in-game purchases*) tuomaan pelaajalle parempia ominaisuuksia ja mahdollisuuksia. (The History of Online Browser Games, 2012.)

Edelleen suurin osa peleistä on toteutettu 2D-ympäristöön, ilman WebGL-tekniikkaa, mutta 3D-pelien toteutus web-selaimille kasvaa ja on ajatuksia jopa AAA -luokan peleistä tapahtuvan web-selaimella, perustuen JavaScript-ohjelmointirajapintoihin (*engl. Application Programming Interface, eli API*), kuten WebGL. (Cocilowa 2013.) Web-selainten takana olevat yhtiöt, kuten Google ja Mozilla ovat suuntaamassa ajatukseen, että nykyiset konsolipelit voitaisiin toteuttaa web-selaimille ja pelejä

pääsisi pelaamaan vain aukaisemalla sivuston web-selaimella. Tällä hetkellä resursseja käytetään huomattavasti vähemmän selainpohjaisten- kuin konsoli- ja pc-pelien kehitykseen, mutta edellä mainitut yhtiöt ovat jo tekemässä tähän murrosta. Ongelmakohtia ovat nykypelien vaatimat tehot ja ominaisuudet, joihin ollaan kehittelemässä ratkaisuja. (Rodriguez 2013.)

HTML5 toi mukanaan uudistuksia helpottamaan web-selamiin toteutettavia ominaisuuskia, kuten videoiden tai muun multimedian lisäämistä. HTML5:n mukana tuli myös pelinkehitykselle merkittävä canvas-elementti. Se on tarkoitettu suorittamaan 2D- ja 3D-grafiikan piirtoa ja tukemaan videoiden ja muun median suorittamista web-sivustolla. Tämän uudistuksen myötä pystyttiin JavaScript API:lla manipuloidessa canvas-elementtiä toteuttaa dynaamista grafiikkaa sekä animaatioita, jonka pohjalta toteutettiin ensimmäisiä 3D-ympäristön kokeiluja. (Hawkes 2011.)

Alkujaan canvas-elementtiä ei otettu vakavasti pelinkehityksen osalta, sillä vuosia sitten JavaScriptin suorituskyky ja tehokkuus sekä korketasoinen grafiikan renderöinti eivät vastanneet pelien vaatimuksia. Vuodesta 2008 kehitys muuttui parempaan suuntaan ja JavaScript herätti huomiota esimerkiksi Googlen Gmail-sovelluksen käytössä. Myöhemmin Google kehitti Chrome-selaimen omalla V8 JavaScript-moottorilla varustettuna, ja samoja piirteitä toteutti Safari sekä Firefox. JavaScript on tuosta hetkestä noussut hiljaista vauhtia suositummaksi ja uskottavammaksi ohjelmointikieleksi. Pelinkehityksen piireissä heräsi mielenkiinto canvas-elementistä potentiaaliseksi työkaluksi sen ilmaantuessa yhä useammalle web-selaimelle. Pian HTML5- ja canvas-tekniikan ympärille alkoi muodostumaan useita kirjastoja sekä pelimoottoreita tukemaan niin pelinkehitystä kuin myös muun visuaalisen ja toiminnallisen sisällön näyttöä. Mainittakoon, että tässä vaiheessa myös WebGL on tullut osaksi tunnettuja teknologioita. (Rettig 2012.)

Esimerkki selainpohjaisen pelinkehityksen historiasta on Runescape. Runescape on Jagex pelitalon kehittämä ilmainen roolipeli missä kohdataan muita pelaajia fantasiamaailmassa (RuneScape Review, Sullivan n.d.). Peli on ollut useamman vuoden Java -kieleen pohjautuva ja clientillä toimiva. Myöhemmin Jagex on julkaissut uuden version pelistä, joka pohjautuu WebGL-teknologiaan, mahdollistamaan nopeamman ja raskaamman pelimekansimin, grafiikanpiirtoa myöten. (Rose 2013.)

3.2 Selainpohjainen HTML5 peli

Kuvio 2. Kuvakaappaus julkaisemattomasta HTML5-pelidemosta

Selainpohjainen peli (*engl. browser based game*) on lyhyesti peli mitä voidaan pelata web-selaimella ilman liitännäisiä tai muita asennuksia. HTML5-tekniikalla toteutettu peli on useimmiten toteutettu JavaScript-ohjelmointikielellä ja näyttäytyy web-selaimen sivustolla canvas-elementin kautta. HTML5 tarjoaa pelien yhteensopivuuden myös mobiililaitteille ja useille eri web-selaimille. Tällä tavalla toteutettu peli soveltuu siis eri alustoille ja laitteille, vaikka se ohjelmoidaan vain kertaalleen. (Kuryanovich ym. 2012.) Tämän hyödyn lisäksi web-selaimelle toteutettu peli on kaikkien ulottuvilla, jos käyttäjällä vain on laite missä internet yhteys.

Yleiset API:t

HTML5-tekniikkaan sovelletaan useita eri teknologioita kuten edellä mainittu canvas, SVG ja WebGL. Canvas-elementille on toteutettu usein 2D-pelejä ja myöhemmin myös WebGL-ohjelmointirajapintaan pohjautuvia 3D-pelejä. (van der Spuy 2015.) SVG:llä taas vektoripohjaista 2D-sisältöä mikä omaa skaalautuvuuden ja sisäänrakennetun animaatioiden toteutuksen. Silti SVG ei ole niin suosittu kuin canvas eikä sillä ole täydellistä pikselitarkkuutta (*engl. pixel-perfect precision*), jota vaaditaan pelin törmäyksien ja klikattavien toimintojen havainnointiin. WebGL on tarkoitettu 2D- ja 3D-ympäristön toteutukseen web-selaimella. Se johtaa teknologiansa OpenGL ES 2.0-tekniikasta, josta kerrotaan seuraavassa luvussa.

HTML5:een voidaan toteuttaa kaikki oleellinen peliin liittyvä ja näitä ovat äänet, animaatiot, grafiikat, tekstit sekä toiminnallisuudet. (Kuryanovich ym. 2012.)

Haasteet

Selainpohjaiseen pelin toteuttamiseen liittyy myös huonoja puolia. Ehkä suurimpia ovat suorituskykyyn vaikuttavat seikat kuten web-selainten väliset eroavaisuudet sekä laitteiden eritasoiset tehot ja niiden suorituskykyyn liittyvät rajoitukset. On siis päätettävä jo pelinkehityksen suunnitteluvaiheessa, että minkälaisille alustoille ja minkälaisilla vaatimuksilla peli toteutetaan. Joka tapauksessa teknologiat selainpohjaisiin peleihin kehittyvät kokoajan ja WebGL on yksi näistä. Väitetään, että selainpohjaista peliä ei ole järkevä toteuttaa, ellei jokainen web-selain ja laite ole sille tuettu. (Kuryanovich ym. 2012.) Tässä työssä kyseinen väite ei kuitenkaan ole relevantti sillä Chrome-selain mille peli suunnataan on tietoinen valinta ja kyseinen web-selain on todettu hyväksi, niin kehitysympäristönä kuin myös peliä testatessa. Lisäksi Chrome-selain on arvioitu yhdeksi parhaista sen kehitystyökalujen sekä monipuolisen yhteensopivuutensa vuoksi (Cox 2016).

3.3 Tekniikat

3.3.1 WebGL

OpenGL ja OpenGL ES

OpenGL on ollut olemassa vuodesta 1992 ja varjostuksia sekä valaistuksia toteuttava GLSL-tekniikka tuli mukaan OpenGL version 2.0 mukana. OpenGL suorittaa toimintoja laitteistopohjaisesti (*engl. hardware accelerated*), eli monimutkaiset ja raskaat toiminnot, kuten visuaaliset tehosteet sekä fysiikkaa simulaatiot tapahtuvat grafiikkaprosessorin (*engl. Graphic Processing Unit, eli GPU*) kiihdyttämänä. OpenGL ES 2.0 on erillinen ja kevyempi toteutus OpenGL 2.0:sta sulautettuja järjestelmiä, kuten mobiililaitteita varten. Siinä on matalampi suoritusteho sekä joitain ominaisuuksia on karsittu. Kokonaisuutena se ei eroa merkittävästi, sillä se mihin OpenGL pystyy, niin pystyy myös OpenGL ES. (Kuryanovich ym. 2012.)

WebGL

WebGL perustuu OpenGL-teknologiasta polveutuvaan OpenGL ES 2.0-teknologiaan.

```
<script>
    window.onload       =       setupWebGL;
    var gl = null;

    function        setupWebGL()
    {
        var         canvas      =       document.getElementById("my-canvas");
        try{
                gl        =     canvas.getContext("experimental-webgl");
        }catch(e){
        }

        if(gl)
        {
                //set the clear color to red
                gl.clearColor(1.0, 0.0, 0.0, 1.0);
                gl.clear(gl.COLOR_BUFFER_BIT);
        }else{
                alert( "Error: Your browser does not appear to support
        WebGL.");
        }
    }
</script>
```

Kuvio 3. WebGL:n käyttö HTML-sivulla

Se on JavaScript API web-selaimelle ja yhteydessä tietokoneen grafiikkaprosessoriin renderöityessä HTML canvas-elementin sisällä (Kuryanovich ym. 2016). Canvasilla määritellään WebGL:n käyttö ja tämän jälkeen WebGL:n ominaisuuksia, kuten esimerkiksi värien renderöintiä voidaan käyttää. (Ks. Kuvio 3.)

WebGL API:a käyttäessä tulee otttaa joitain asioita huomioon, kuten sen yhteensopimattomuus web-selaimien sekä grafiikkaprosessoreiden välillä. Osa web-selaimista erityyppisillä tietokoneilla tai laitteilla tuottaa virheilmoituksia yhteensopivuudesta WebGL:n kanssa. Useimmiten kyseinen ilmoitus on virheellinen, koska web-selaimen WebGL-ominaisuus on oletuksena otettu web-selaimesta pois käytöstä, eikä siis yhteensopivuusongelmaa välttämättä ole. WebGL-ominaisuuksien käyttöönotto Chrome-selaimeen tapahtuu menemällä osoitteeseen chrome://flags/ ja ottamalla WebGL-luonnoslaajennukset käyttöön. Mikäli ongelma ei korjaannu, voi ongelma olla näytönohjaimessa ja sen asetuksissa. Suorituskyky ongelmat voivat johtua myös tietokoneen tai laitteen alhaisista tehoista. (Danchilla 2012.)

Kuvio 4. Kuvakaappaus WebGL-pelidemosta (https://jbouny.github.io/fft-ocean/#night)

WebGL itsessään pystyy teknisestä näkökulmasta katsottuna toteuttamaan kaiken tarvittavan pelien ja visuaalisten sovellusten suorittamiseksi. (Ks. Kuvio 4.) Näitä ovat siis kuvasuhteet, eli resoluutiot ja skaalautuvuus, valaistukset ja varjostukset, vektoreiden, värien ja tekstuureiden piirto, fysiikat ja animaatiot sekä monia muita ominaisuuksia mitä peleissä useimmiten esiintyy. (Danchilla 2012.) WebGL eroaa pelimoottorista siten, että se on puhdasta 2D- ja 3D-renderöintiä sekä siitä puuttuu valmiit määritelmät kuten törmäysmoottori, partikkeli-efekti ja fysiikka simulaatio. Kaikki edellä mainittu on mahdollista toteuttaa WebGL-tekniikalla, mutta suurin osa asioista on toteutettava itse, eli valmiita, esimerkiksi animaatioon liittyviä asetuksia ei löydy. WebGL on hyvä valinta, jos pyritään toteuttamaan jotain laajempaa, kuten esimerkiksi oman pelimoottorin toteutus. (Arora 2014.)

WebGL omaa muutamia niin sanottuja sudenkuoppia ja näitä ovat välimuistiin tallentuminen, yhteensopimattomuus ilman vaihtoehtoa tai virheilmoitusta, puutteellinen optimointi mobiililaitteille, kameran asennot, rikkonaiset tai vääristyneet tekstuurit sekä web-selain eroavaisuudet. (Danchilla 2012.) Kyseisiä ongelmakohtia esiintyy oletettavasti myös myöhemmin esiteltävissä WebGL-

sovelluskehyksissä. Voidaan tulkita, että sudenkuopat ovat väistettävissä tekemällä sovellus eheillä ominaisuuksilla ja toiminnoilla sekä poissulkemalla edella mainitut ongelmat. Tämä ei tarkoita kuitenkaan sitä, etteikö joitain ominaisuuksia voisi vielä käyttää, vaan että halutut ominaisuudet saavutetaan toteuttamalla ohjelmiston rakenne toisella tavalla.

GLSL

WebGL:n renderöinnissä tapahtuu useampi vaihe ennen kuin lopullinen tulos nähdään. GLSL ohjaa GPU:ta toteuttamaan objektien ja grafiikan piirtämisen näytölle käyttäen kahta suoritinta yhdessä. Suorittimet ovat verteksi-varjostin (*engl. vertex shader, eli VS*) sekä fragmentti-varjostin (*engl. fragment shader, eli FS*). Ensimmäisenä 3D-mallin verteksit siirtyy omiksi taulukoiksi verteksipuskureihin (*engl. vertex buffer object, eli VBO*), jonka jälkeen tieto siityy verteksi-varjostimeen. VS toteuttaa verteksi-arvojen laskelmoinnit ja verteksien sijoittumisen piirtyvään kuvaan, jonka jälkeen tieto siirtyy fragmentti-varjostimeen. Sillä voi olla myös roolina toteuttaa värimäärittely, teksturointi sekä valaistus. Tuossa siirtymässä tapahtuu rasterointi, jotta FS voi tehdä lopullisen pikselikohtaisen värimäärityksen ja mahdollisesti suorittamaan tekstuureiden hakuja. GLSL:n prosessointi on hyvä huomioida, sillä jos peliin on tuotu liian raskaita 3D-malleja, efektejä ja muita visuaalisia tehosteita, niin peli voi hidastua merkittävästi. (Danchilla 2012.)

3.3.2 Sovelluskehykset

Useamman lähteen mukaan Babylon.js sekä Three.js ovat vakuuttavimmat WebGL-ohjelmakirjastot 3D-pelille web-selaimeen. Noeticforce-nettisivulla kyseiset ohjelmakirjastot ovat ensimmäisten listalla (noeticsunil 2015.) Myös Darrenin ja Scottin kirjoittamissa blogeissa esiteltiin nämä ohjelmakirjastot.

Darren kävi läpi kolme ohjelmakirjastoa, jotka ovat Three, Babylon sekä Superpowers, mutta artikkelin mukaan hänen aikomus on soveltaa teknologioita muuhun kuin pelehin (Cook 2016.) Scott käsitteli blogissaan vain Three- ja Babylon-ohjelmakirjastojen ominaisuuksia (Hanselman 2014.)

Kyseiset lähteet ovat yleinen katsaus tekniikoihin, joten näiden pohjalta tehdään varovainen tulkinta tekniikoista. Tietoperustan selvityksen myötä Three.js-

ohjelmakirjastoon viittaavia kirjoja on kirjoitettu useampi, kuten esimerkiksi Jos Dirksenin LearningThree.js: The JavaScript 3D library for WebGL sekä Isaac Sucinin Game Development with Three.js, mutta Babylonista ei löytynyt viitattauksia, lukuun ottamatta lyhyitä mainintoja kyseisestä ohjelmakirjastosta ja syy voi olla, että Babylon on julkaistu vasta vuonna 2014.

Babylon.js

Babylon tarjoaa laajan paketin, niin perustoimintoja kuin myös monimutkaisempia toimenpiteitä sovelluksen toimintoihin. Peliohjelmointi Babylonilla on pääsääntöisesti JavaScriptia, lukuunottamatta pakollista HTML-sivua. Edellä mainitun noeticforcen artikkelin mukaan Babylon on paras JavaScript-kielinen 3D-pelimoottori, mutta ilman perusteluja. Babylonin merkittävimpiä ominaisuuksia ovat näkymän muokkausta varten tarkoitettu tietorakenne, valaistuistukset, kamerat, materiaalit, 3D-mallit, törmäys-, ääni- ja fysiikkamoottorit sekä optimointi. Babylon tarjoaa käyttöliittymän, jossa voi kokeilla erilaisia demoja sekä muokata näitä. (noeticsunil 2015.) Babylon sisältää myös muita tyypillisiä ominaisuuksia kuten animaatiomoottorin, partikkeli-funktiot sekä suuren määrän varjostin- ja fx-ominaisuuksia. (Specifications n.d.) Babylonilla on laaja ja aktiivinen yhteisö sekä dokumentoituja demoja tarjoamaan katsausta koodiin sekä apua pelinkehityksen vaiheisiin. (Cook 2016.)

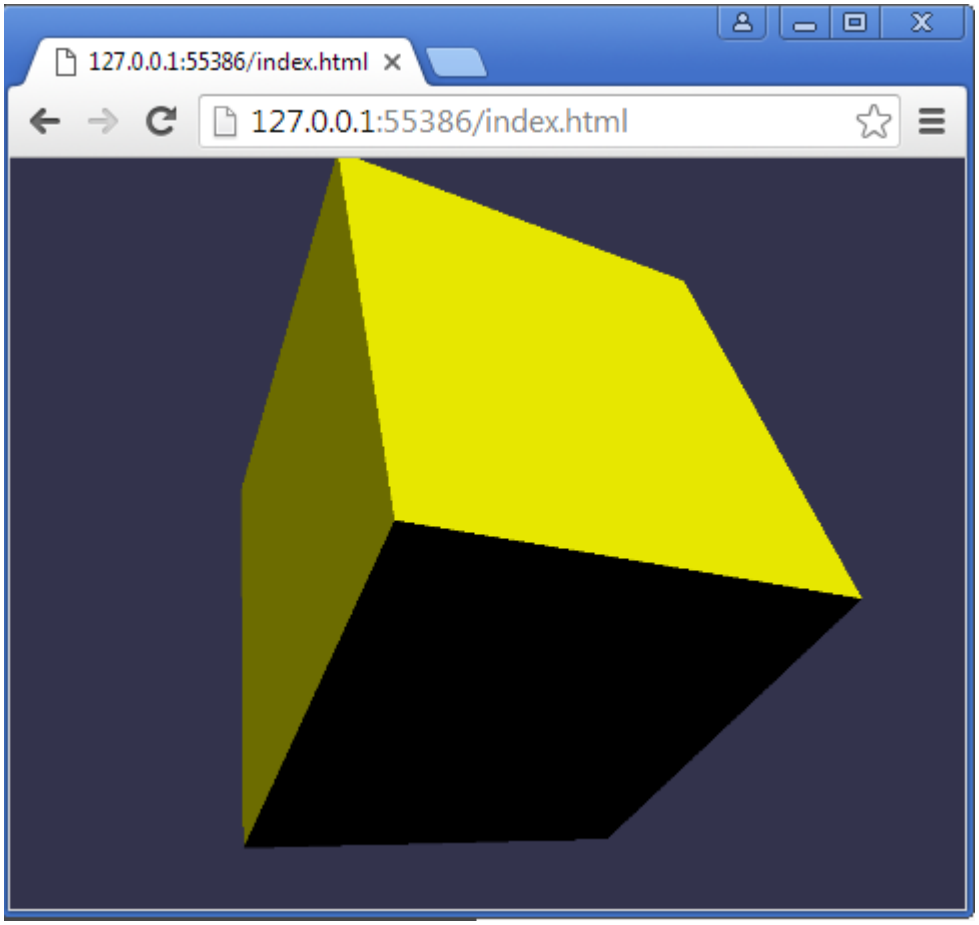
```html
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

        <title></title>
        <script src="lib/babylon.2.4.js"></script>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
        <script src="lib/game.js"></script>
        <style>
            html, body {
                overflow: hidden;
                width: 100%;
                height: 100%;
                margin: 0;
                padding: 0;
            }

            #renderCanvas {
                width: 100%;
                height: 100%;
            }
        </style>
    </head>
    <body>
        <canvas id="renderCanvas"></canvas>
    </body>
</html>
```

```javascript
var canvas;
var engine;
var scene;

document.addEventListener("DOMContentLoaded", startGame, false);

function startGame () {
    canvas = document.getElementById("renderCanvas");
    engine = new BABYLON.Engine(canvas, true);
    sceneB = new BABYLON.Scene(engine);

    var camera = new BABYLON.ArcRotateCamera("camera", 1, 0.8, 10, new BABYLON.Vector3(0, 0, 0), sceneB);

    var light = new BABYLON.DirectionalLight("light", new BABYLON.Vector3(0, -1, 0), sceneB);
    //light.diffuse = new BABYLON.Color3(1, 0, 0);
    //light.specular = new BABYLON.Color3(1, 1, 1);

    var box = BABYLON.Mesh.CreateBox("box", 5.0, sceneB);

    var material = new BABYLON.StandardMaterial("texture", sceneB);
    material.diffuseColor = new BABYLON.Color3(1, 1, 0);
//    material.diffuseTexture = new BABYLON.Texture("./sampleTexture.jpg", sceneB);
    box.material = material;

    sceneB.activeCamera.attachControl(canvas);

    engine.runRenderLoop(function () {
        box.rotation.x += 0.005;
        box.rotation.y += 0.01;
        sceneB.render();
    });

}
```

Kuvio 5. Perusnäkymän lähdekoodi

Babylon tarjoaa nopean mahdollisuuden päästä kokeilemaan luonnosvaiheessa olevaa peliä monenlaisilla ominaisuuksilla. Esimerkiksi kuution luominen ja sen teksturointi sekä valaistuksen ja kameran määrittely näkymään tapahtuu muutamalla koodi rivillä. (Ks. Kuvio 5.) HTML-tiedosto näyttäytyy edellä esitellyn kuvion alussa, jossa canvas-elementti nimeltään renderCanvas. Kyseinen canvas-elementti otetaan

käyttöön alampana näkyvässä koodiosuudessa. Pelimekanismin ja sen visuaalisen sisällön määrittely tapahtuu tuossa koodissa, joka renderöityy canvas-elementissä.



Kuvio 6. Perusnäkymä

Tuossa näkymässä pystyy katsella objektia eri suunnista ja zoomata sitä lähemmäksi. Babylon on varteenotettava tulevaisuutta ajatellen, sillä siihen on pelikonsoli-tuen lisäksi virtuaalitodellisuus-liittymä (engl. virtual reality, eli VR) mahdollistamaan VR-lasien, kuten Oculus Rift-lasien käytön. (Hanselman 2014.)

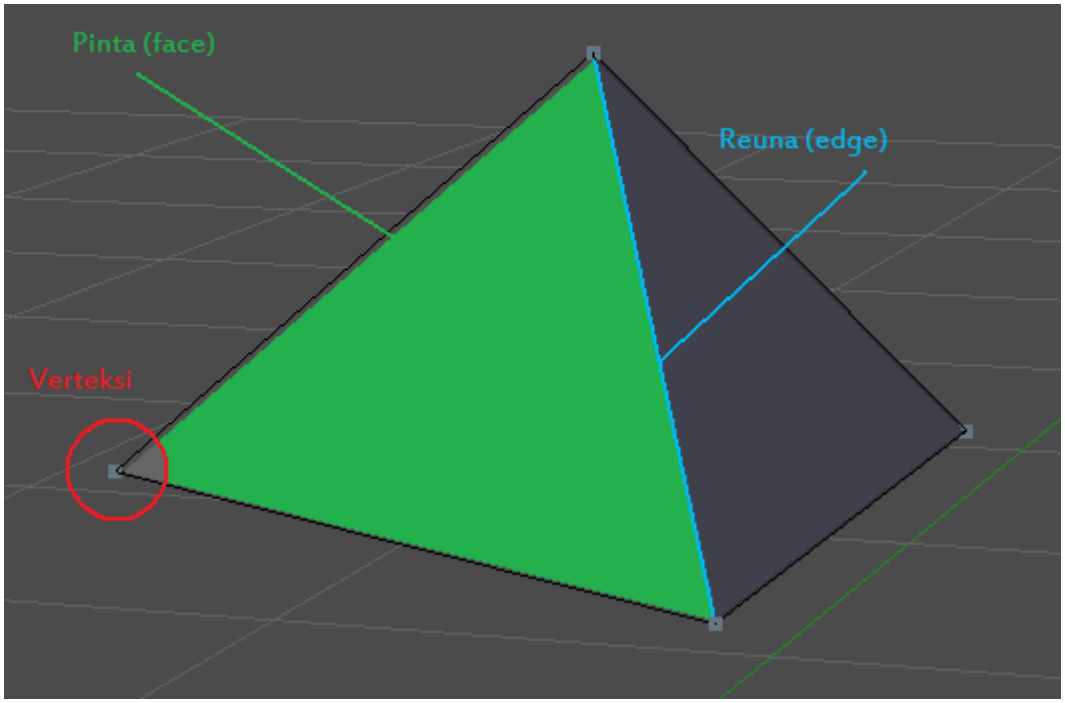
Tällä hetkellä Babylonilla ei ole HTML-pohjaisen käyttöliittymän toteutukseen omaa komponenttia, joten se on rakennettava alusta asti itse. Vaihtoehto tähän on Babylon-yhteisön jäsenien tarjoama käyttöliittymä-laajennus. (Weber 2016.) Käyttöliittymän pystyisi toteuttamaan luomalla painikkeet sekä ulkoasu 2D- tai 3D-objekteilla. Komponentilla tarkoitetaan tässä tapauksessa sisäänrakennettuja rakenteita Babylon-ohjelmakirjastossa, joiden avulla käyttöliittymä pystytään muodostamaan erilaisilla asetuksilla koodissa.

Three.js

Three on kokonaisuudeltaan hyvin lähellä Babylonia ja sillä voi tehdä paljon muitakin asioita kuin vain pelejä. Three:n yhteisö on laajempi kuin Babylonin ja siitä löytyy paljon demoja, opetusvideoita, lähdekoodeja sekä kirjallisuutta. (Hanselman 2014.) Lähteitä tutkiessa havaittiin, että Three.js ei sisällä pelimekansimiin liittyviä asioita, kuten pelimoottoria tai valmista näppäinkomentoja ymmärtävää toteutusta. Näppäimien toiminnot pystytään kuitenkin toteuttamaan puhtaalla JavaScriptillä (Sukin 2013, 37-39).

3.3.3 3D-mallinnus

3D-malli



Kuvio 7. 3D-mallin perusrakenne

Peleissä 3D-mallia kutsutaan usein meshiksi ja se koostuu pinnoista (*engl. face*), jotka muodostuvat vähintään kolmen verteksin yhdistämisestä. Meshin pintaa kutsutaan myös polygoniksi (Arora 2014). Verteksi on X-, Y- ja Z-arvoja sisältävä piste 3D-ympäristössä. Kahden verteksin välille muodostuu reuna (*engl. edge*) yhdistämään

verteksit osaksi 2D- tai 3D-muotoa. Pinta on kolmen tai useamman verteksin tulos, mutta useimmiten kolmen. (Ks. Kuvio 7.) 3D-malli on useamman pinnan muodostama kokonaisuus. 3D-mallin tekstuuri, varjostukset ja valaistus määräytyvät sen pintojen mukaan sekä sen rakenne ja liikeradat verteksien mukaan. (Kuryanovich ym. 2016.)

3D-malliin toteutetaan monia erilaisia asioita 3D-grafiikkaohjelmalla. 3D-malli tai niin kutsuttu mesh viedään johonkin tiedostomuotoon, kuten esimerkiksi .obj ja tätä tiedostoa voidaan käyttää peliohjelmoinnissa. Obj-tiedostomuoto ei ole kovin yhteensopiva JavaScript-kielisiin ohjelmakehyksiin hitauden vuoksi, joten tiedostomuodolle tehdään konversio toiseen tiedostomuotoon, kuten JSON. JavaScript pystyy parsimaan (*engl. parsing*) JSON-tiedostoa nopealukuisesti. (Arora 2014, 52-57.)


```
"diffuseTexture":{
    "name":"TexturesCom_WoodPlanksClean0061_1_seamless_S.jpg",
    "level":1,"hasAlpha":1,"coordinatesMode":0,"uOffset":0,
    "vOffset":0,"uScale":1,"vScale":1,"uAng":0,"vAng":0,
    "wAng":0,"wrapU":1,"wrapV":1,"coordinatesIndex":0},
{"name":"Pieces.WoodText.001","id":"Pieces.WoodText.001","ambient"
    "diffuse":[0.9258,0.7583,0.5718], "specular":[0.0559,0.0559,0.
    "specularPower":14,"alpha":1,"backFaceCulling":true,"checkReady
```


Kuvio 8. JSON-koodi

3D-grafiikkaohjelmissa voi olla valmiina tai näihin on saatavilla liitännäisiä viemään tehty 3D-malli suoraan JSON-tiedostomuotoon. 3D-mallin JSON-tiedoista löytyy kaikki tieto mitä kyseinen mesh sisältää. Näitä ovat siis nimi, id, materiaalit, tekstuurit, läpinäkyvyysarvo, verteksi- ja pinta-taulukot sekä monia muita tietoja. (Ks. Kuvio 8.) Sovelluskehyksillä on omat metodinsa 3D-mallin lataamiseen ja renderöintiin. WebGL voisi käyttää JavaScript-funktiota noutamaan JSON-tiedoston, jonka jälkeen haluttuja arvoja voidaan tallentaa omiin JavaScript-muuttujiin. Muuttujaan tallennettuja arvoja voidaan jälkeenpäin käsitellä, kuten esimerkiksi värejä ja varjostin-arvoja. (Arora 2014, 59-66.)

Blender

3D-grafiikkaohjelmista Blender osoittautuu parhaaksi ilmaisena vaihtoehtona, jolla pystyy samaan lopputulokseen kuin alan suosituimmilla Autodeskin 3ds Max ja Maya

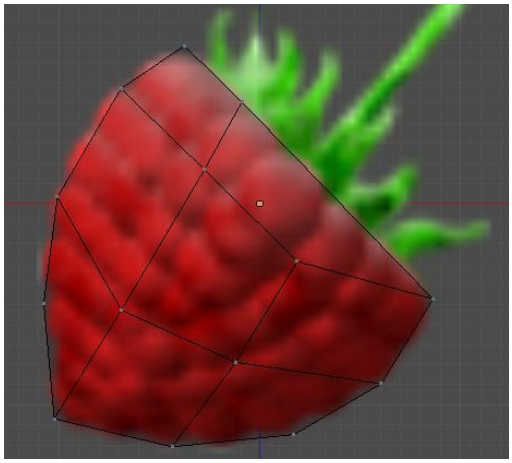
ohjelmilla. Autodeskin ohjelmat ovat suuria ja näillä on melko kova hinta. Blenderin suosio on kasvanut viime vuosina ja tavoittanut suuren käyttäjäkunnan. Ohjelman GNU-lisenssi mahdollistaa ohjelman muokkauksen ja osoittaa ohjelman käytön vapaaksi. (Fronczak 2011.)

Blender toimii hyvin teknisesti, eli toiminnot kuten objektin koon muuttaminen tai muiden muotoilutyökalujen käyttö tapahtuu näppäinkomennoilla. Tämä eroaa kuvankäsittelyohjemista, joissa toiminnot tapahtuvat työkaluikonia klikkaamalla. Blenderistä löytyy työkaluikonit, mutta näitä käyttäen muokkaus voi olla hitaampaa ja haastavampaa. Ohjelmasta löytyy esiasetettuja perusmuotoja, muokkaustyökaluja, tekstuureiden ja materiaalien lisääminen sekä kaikki mitä 3D-animaation tai pelin sisältöön vaaditaan. Se tukee lukuisia tiedostomuotoja ja sillä pystyy viemään 3D-mallin useaan eri tiedostomuotoon. (Brandt 2011.)

3D-mallintaminen ja grafiikka

Mallintamiseen liittyvät toiminnot eroavat ohjelmittain ja tässä esitellään mallintamisen vaiheita Blender-ohjelmaa käyttäen.

Mallintaminen aloitetaan usein yksinkertaisesta muodosta, kuten esimerkiksi kuutiosta. Muotoa voidaan tarkastella useasta eri kulmasta navigaatiotyökalua käyttäen. 3D-malli voidaan tehdä valituksi objektina ja sitä voidaan skaalata, liikutella ja pyöritellä ohjelman näkymässä. Objektin verteksejä voidaan muokata menemällä objektin muokkaustilaan. Perinteisen muokkauksen lisäksi on olemassa objektin veistäminen, partikkeleiden muokkaaminen sekä tekstuurin maalaus. (Simonds 2013.)



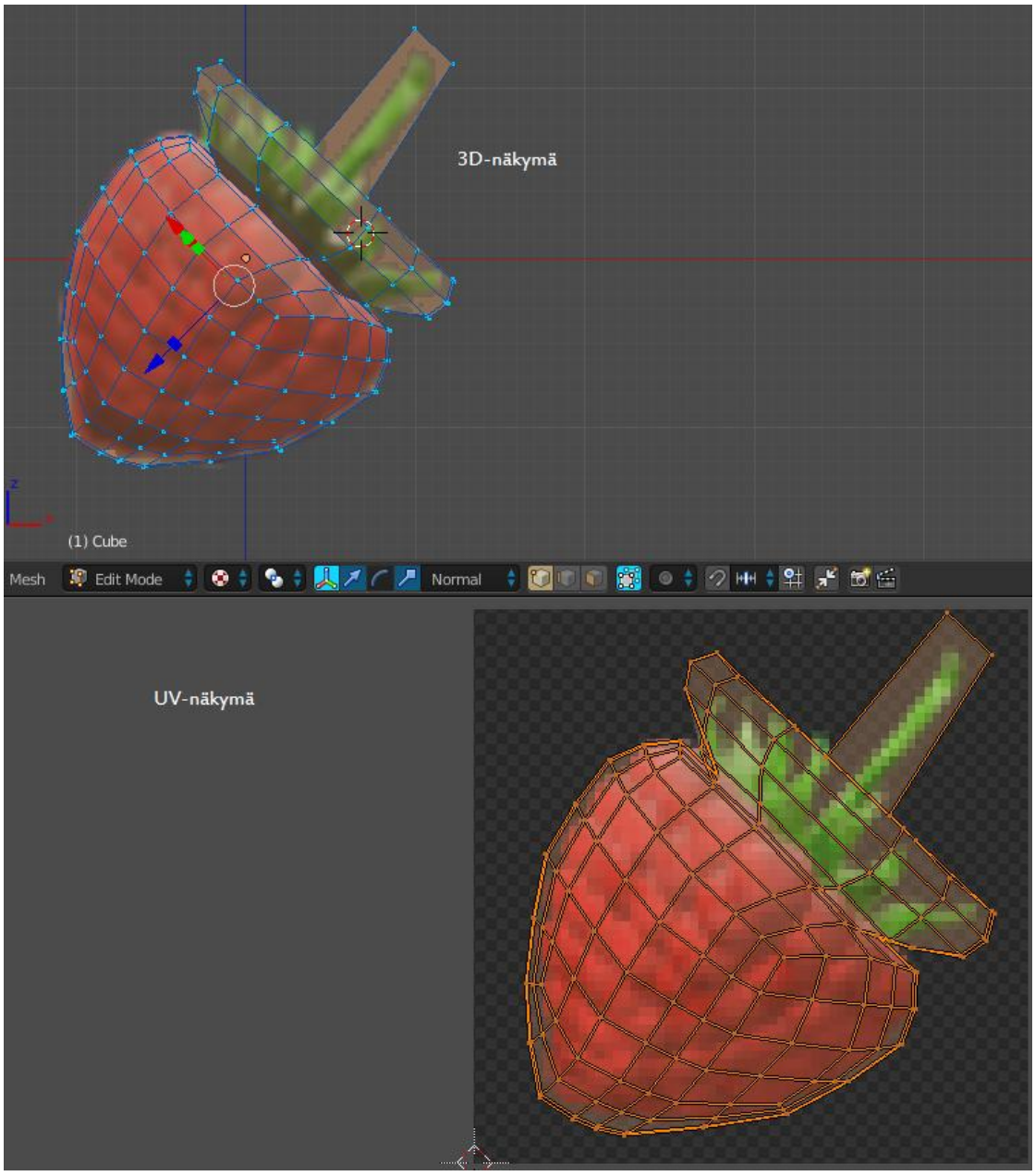
Kuvio 9. Taustakuva mallintamiseen

Tietyn muodon mallintaminen voi olla hankalaa ilman kuvaa, josta ottaa mallia. Kuva voidaan etsiä internetistä tai sellaisen voi tehdä itse. Kuvaa käytetään ohjelman taustalla, niin että se on oikeassa mittasuhteessa mallinnettavaan objektiin. (Ks. Kuvio 9.) Kuvia on syytä olla useampi ja mielellään eri kuvakulmista hahmottamaan mallinettavan objektin 3D-muotoa. Mikäli kuvasta halutaan mallintaa suora kopio, niin kuvan objektilla on oltava tarkat ääriviivat, jotta verteksien linjaa voidaan mallintaa ääriviivoja myöten. Mallintaminen voi tapahtua perspektiivi- tai otrografinäkymässä, mutta taustakuva näkyy vain ortografinäkymässä. (Simonds 2013.)

Blender ja monet muut 3D-grafiikkaohjelmat käyttävät 3D-malleihin modifikaattoreita nopeuttamaan mallintamisen vaiheita. Esimerkkejä näistä on ovat 3D-mallia monistava array-modifikaattori ja eri suuntiin peilaava mirror-modifikaattori. Mirror-modifikaattorilla voidaan siis luoda esimerkiksi ihminen toteuttaen siitä vain toisen puolen. Modifikaattorit sekä muut ominaisuudet on asetettava lopullisesti ennen 3D-mallin vientiä JSON- tai muuhun tiedostomuotoon. Asettaminen on tehtävä, sillä moni vaikuttava tekijä on vain esittävässä tilassa, eikä todellisesti vielä tallentunut 3D-mallin tietoihin. Osa liitännäisistä kuitenkin havaitsee modifikaattorit ja osaa asettaa ne ennen vientiä. (Simonds 2013.)

Mallintamiseen liittyy tiettyjä sääntöjä, joita on hyvä noudattaa, jotta lopullinen tulos on vaikuttava sekä virheetön. Verteksejä yhdistäessä tulisi ottaa huomioon, että ne muodostavat aina nelikulmioita ja verteksiin liittyvien reunojen määrä olisi pieni. Mikäli 3D-malliin on joutunut kolmio voi se aiheuttaa varjostuksissa ja valaistuksissa

vääristymiä, mutta vähäisesti näkyvät pienet kolmiot ei oletettavasti aiheuta ongelmia. (Simonds 2013.)



Kuvio 10. UV-muokkaustila

3D-malliin asetetaan tekstuuri käyttämällä ohjelman UV-muokkaustyökalua. Tekstuuri voidaan sijoittaa avaamalla se ohjelmaan 3D-mallin polygonien ollessa UV-kartassa (*engl. UV-map*). (Ks. Kuvio 10.) UV-muokkaustilassa polygoneja tai verteksejä voidaan siirellä, jotta tekstuuri asettuu 3D-malliin oikein. (Simonds 2013.)

3D-malliin toteutetaan pääsääntöisesti kolmea erilaista tekstuuria. Yleisin on 2D-tekstuuri ja tämän lisäksi on kuutionmuotoiselle mallille tarkoitettu Cubemap-tekstuuri sekä erikoistehosteissa proseduraaleina käytetty 3D-tekstuuri, jota ei yleensä esiinny peleissä. 2D-tekstuuri on kaksiulotteinen ja tallennetaan useimmiten jpg- tai png-tiedostomuotoon pelin 3D-mallia varten. Tekstuurin koordinaatteja tarvitaan, kun se sijoitetaan 3D-malliin näkymään oikein. Tekstuurin pikselit kohdistetaan 3D-mallin vertekseihin, eli jokaisella verteksillä on oma tekstuurikoordinaatti vastaamaan tekstuurin sijoittumista. (Arora 2014, 117-119.)

Polygonin määrä vaikuttaa 3D-mallin renderöinti nopeuteen ja peleissä määrällä on merkitystä. Polygonien määrä on hyvä pudottaa niin alhaiseksi kuin mahdollista ja määrä vaikuttaa 3D-mallin laatuun. Mitä vähemmän, sitä heikompi ja karkeampi 3D-malli on. Modifikaattoreilla ja veistämisellä voidaan tehdä monen miljoonan verteksin 3D-malli, josta renderöidään väriarvoja sisältävä kuva, jota kutsutaan nimellä normal-map. Tuo kuva liitetään alhaisen polygon-määrän omaavaan 3D-malliin UV-muokkaustyökalulla, joka luo illuusion syvyysvaikutelmasta sen pinnalle. Varjostimet osaavat lukea noita väriarvoja ja pystyvät näin myös heijastamaan valoja sekä varjoja kuvan myönteisesti. (Simonds 2013.)

4 Toteutus

Tässä luvussa käydään läpi, kuinka Maagiset Madot-peli toteutettiin.

4.1 Aloitus

Loppukesästä 2016 aloitettiin tämän työn toteuttaminen. Toimeksiantajalle oli aiemmin toteutettu 3D-animaatiomainosvideo hänen tuotteestaan, minkä jälkeen heräsi ajatus toteuttaa hänen tuotteestaan digitaalinen versio hänen tuotteestaan, jonka tulisi toimia web-selaimilla. Ajatuksia oli myös pelin toteutuksesta mobiilialustoille, mutta pienen mietinnän jälkeen tultiin johtopäätökseen, että peli toteutetaan ensin yksinkertaisemmin web-selaimelle ja jalostetaan myöhemmin muille alustoille.

Itse työn toteuttajalla on ollut mielenkiinto toteuttaa 3D-pelejä, ja tämän tilaisuuden myötä toimeksiantajan kanssa aloitettiin pelin suunnittelu. Lähtökohtainen kokemus WebGL-teknologiaan oli olematon, mutta tietämystä tekniikasta ja kyseisen teknolo-

gian omaavista peleistä löytyi ja näin oli mahdollista lähteä soveltamaan WebGL- ja Babylon.js-tekniikoita 3D-pelin toteuttamiseksi.

Kokemusta 3D-mallintamiseen sekä graafiseen suunnitteluun löytyi kattavasti, joten priorisointi painottui näiden uusien teknologioiden opettelussa sekä projektin vaiheiden suunnittelussa. Työn aloitusvaiheessa kartoitettiin mitä työvälineitä tarvitaan ja missä työ toteutetaan. Työn toteuttaja näki parhaaksi tehdä projekti omalla tietokoneella ja ilmaisilla ohjelmistoilla, välttääkseen mahdolliset lisenssikustannukset.

4.2 Suunnittelu

Tutkimuksen ja kokonaisen pelin suunnittelu oli jaettava useampaan osaan. Tähän kuului teoreettisen aineiston hankinta ja siihen perehtyminen, jotta saavutetaan käsitys aiheen teknologioista sekä siihen liittyvistä prosesseista. Taustateorian jälkeen aloitettiin aiheeseen liittyvän pelin suunnittelu ja toteutus. Peliin oli toteutettava käyttöliittymän ja aloitusvalikon suunnittelu, pelimekanismin ja idean suunnittelu sekä teknisen ja graafisen toteutuksen suunnittelu.

Käyttöliittymän ja aloitusvalikon suunnittelu

Kuvio 11. Käyttöliittymä

Käyttöliittymä on osa visuaalista yleisilmettä sekä tarkoitettu toimimaan informatiivisena osana pelaajalle. Tähän oli toteutettava ohjauspaneeli, jossa HTML-elementeillä toteutetut painikkeet. Painikkeiden avulla aloitetaan peli alusta, poistutaan takaisin aloitusvalikkoon sekä ohjataan pelissä esiintyviä kappaleita. (Ks. Kuvio 11.) Pelissä tuli näkyä myös ajanoton tila. Kentän alkaessa tulisi pelaajalle näkyä ilmoitus kentästä ja toteutettavasta matojen muodosta sekä ilmoitus kentän läpäisystä. Kummassakin ilmoituksessa tulisi olla klikattava hyväksyntä-painike, jota klikattaessa peli etenee.

Kuvio 12. Aloitusvalikko

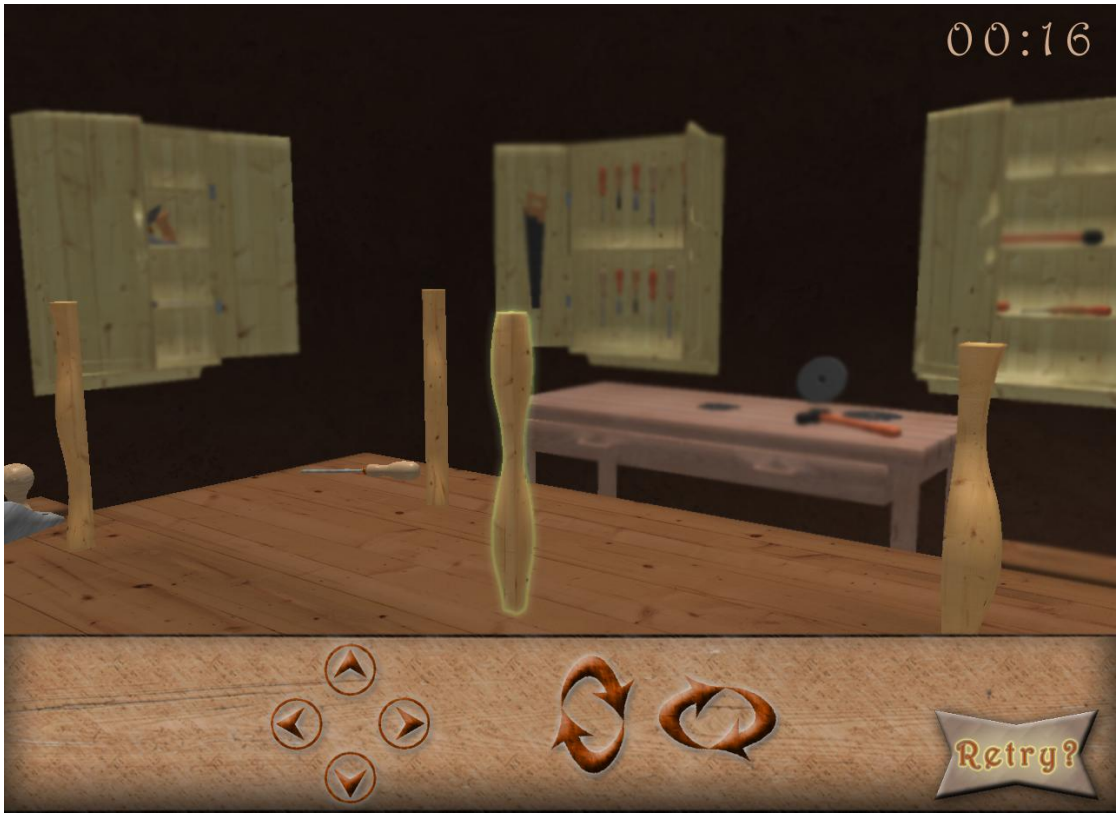
Käyttöliittymän lisäksi peliin tulee toteuttaa aloitusvalikko, joka on pelin aloitustila. Aloitusvalikosta voidaan valita peliohjeet, tietoa Maagiset Madot-pulmapelistä, lopputekstit tai pelin pelaaminen. (Ks. Kuvio 12.) Aloitusvalikon vaihtoehdot ovat painikkeita ja näiden takana on toiminnallisuuksia. Kaikista painikkeista avautuu tekstiikkuna aloitusvalikkoon, lukuun ottamatta painiketta, josta siirrytään pelaamaan peliä.

Idean suunnittelu

Pelille oli määriteltävä sen keskeinen idea, jonka pohjalta peliä lähdetään toteuttamaan. Idea on sama kuin toimeksiantajan oikeassa pelissä. Pelissä on neljä esinettä, joita kutsutaan madoiksi.

Kuvio 13. Pelinäkymän luonnos

Jokainen mato on erilainen muodoltaan ja pyörittelemällä ne oikeaan asentoon sekä yhdistämällä toisiinsa saadaan niistä kolme erilaista muotoa. Tämän alkuperäisen pulmapelin madot ovat tehty puusta ja ne ovat n. 20 cm korkeat, alta 2 cm leveät ja alta 2 cm syvät. Nämä mittasuhteet pyrittiin toteuttamaan peliin kuin myös näiden puinen materiaali.

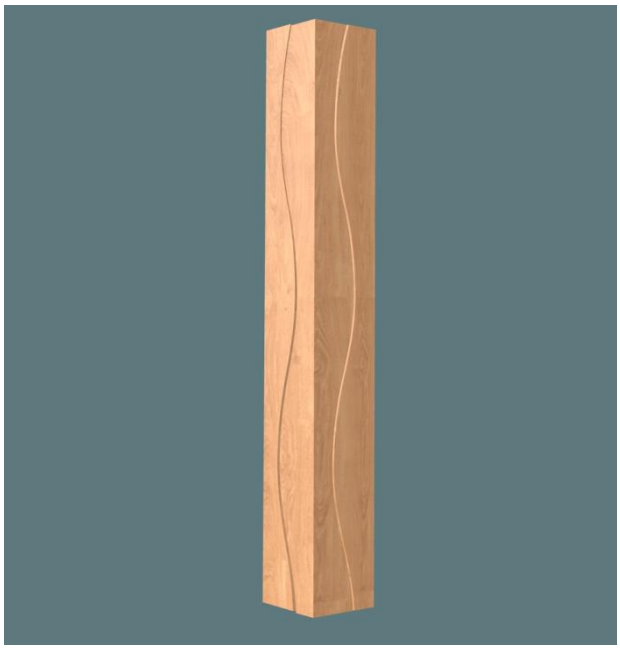


Kuvio 14. Pelinäkymä ja luonnos käyttöliittymästä

Virtuaalisen pelin logiikka on hieman erilainen, sillä matoja pyöritellään ja liikutellaan painikkeita painaen, mutta kokonaisuus on muuten sama. Ideana on siis muodostaa nuo kaikki kolme muotoa mahdollisimman lyhyessä ajassa. Peliä hankaloittaa se, että madot ilmestyvät jokaisen kentän alussa sattumanvaraiseen asentoon, x- ja y-akselin suuntaisesti sekä sijoittuvat myös sattumanvaraisiin kohtiin x- ja z-koordinaatistossa. Tämän jälkeen yksi mato kerrallaan siirretään toista matoa vasten, jolloin peli ilmoittaa onko madot oikeassa asennossa. Digitaalinen versio eroaa alkuperäiseen peliin verrattuna siten, että siinä on kolme kenttää (*engl. level*) ja jokaiselle kentälle on osoitettu yksi näistä kolmesta muodosta. Alkuperäisessä nuo kolme muotoa toteutetaan samalla kertaa. Kenttä on läpäisty, kun odotettu muoto on saavutettu ja silloin pelin sisäinen ajanotto pysähtyy, ilmoittaen tästä pelaajalle. Tämän jälkeen pelaaja siirtyy seuraavalle kentälle, jossa tehtävänä on toteuttaa toinen muoto. Edellisen kentän aika summataan aina seuraavan kentän läpäisyaikaan ja kolmannen kentän jälkeen peli ilmoittaa pelaajalle kokonaisajan.

Pelimekanismin suunnittelu

Klikkaamalla ja pitämällä hiiren vasenta näppäintä pohjassa sekä raahaamalla sitä haluamaansa suuntaan, pystyy kentällä pyörimään liikuteltavien matojen ympäri. Rullaamalla hiiren rulla osaa, pelaaja pystyy lähestymään ja tarkentamaan kuvaa madoista. Klikkaamalla hiiren vasemmalla painikkeella liikuteltavasta madosta, muuttuu se valituksi. Tällöin käyttöliittymän ohjauspaneelin painikkeet ovat aktiivisia ja näitä klikattaessa, matoa voi liikutella sekä pyörittää y- ja z-koordinaatistossa itsensä ympäri. Madon törmätessä toiseen matoon se pysähtyy, joten tähän oli toteutettava havainnointi törmäykselle. Mikäli madot ovat oikeissa asennoissa, lukittuvat ne toisiinsa ja peli ilmoittaa onnistuneesta liitoksesta.



Kuvio 15. Maagiset Madot - Muoto 1

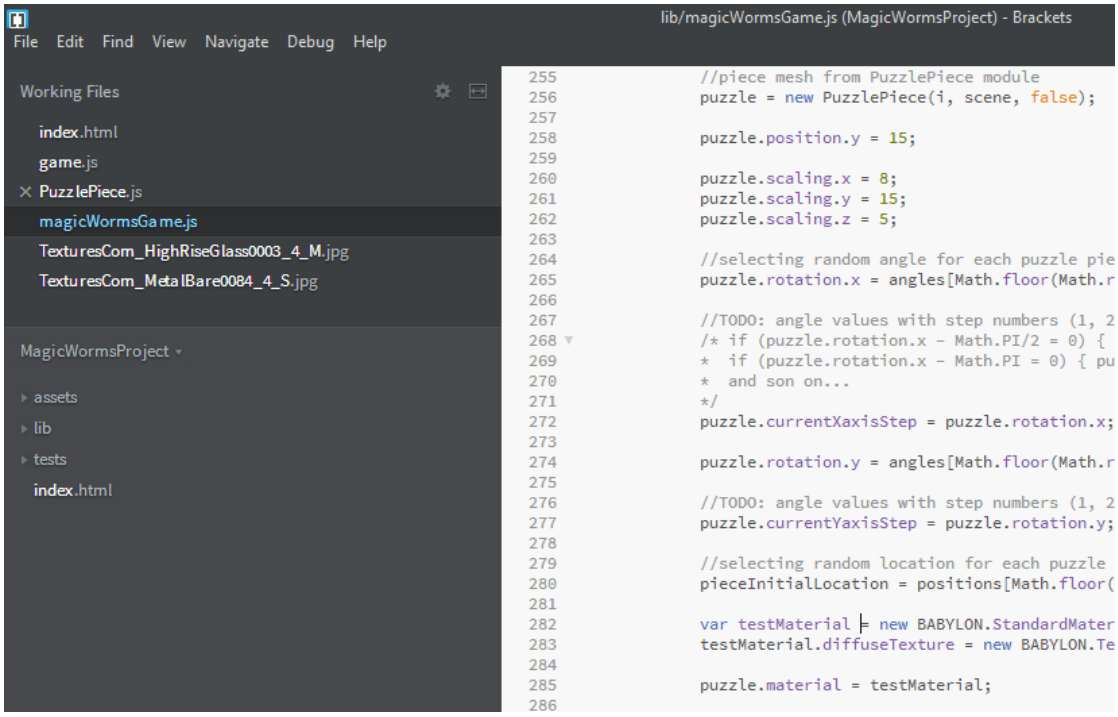
Aiemmin mainittu kenttä päättyy, kun madot ovat liitetty toisiinsa oikein. Ensimmäinen kenttä päättyy, matojen muodostaessa yksinkertaisen pötkön. (Ks. Kuvio 15.) Ensimmäisen kentän päätyttyä peli ilmoittaa pelaajalle kentän läpäisystä ja ehdottaa jatkamaan kentälle kaksi. Kenttä kaksi alkaa samalla tavalla kuin kenttä yksi, eli madot ilmestyvät kentälle sattumanvaraisesti ja kenttäkohtainen ajanotto käynnistyy. Tämän jälkeen madot ovat taas liikuteltavia ja toinen kenttä päättyy, kun niistä on muodostettu yhteen suuntaan mutkitteleva muoto. Toinen kenttä päättyy ja siirtyminen seuraavaan kenttään tapahtuu samalla tavalla kuin kentältä yksi.

Kuvio 16. Maagiset Madot - Muoto 3

Kolmas kenttä alkaa samalla tavalla kuin edelliset kentät. Se päättyy, kun madoista on muodostettu kahteen suuntaan mutkitteleva muoto (Ks. Kuvio 16), jolloin peli päättyy ja kokonaisaika pysähtyy. Kokonaisaika muodostuu kenttien yhteenlasketusta läpäisyajasta.

Tekninen ja graafinen suunnittelu

Pelin oli toimittava web-selaimella ja sen oli oltava 3D-muodossa, joten peli tuli toteuttaa HTML5- ja WebGL-tekniikoilla. Teknisen esiselvityksen pohjalta web-selaimeksi valikoitui Chrome, sen WebGL- ja HTML5-yhteensopivuuden vuoksi. Tässä työssä web-selain toimii pelin alustana, jolla peliä pelataan. Babylon.js valikoitui sovelluskehykseksi sen hyvien ominaisuuksien sekä uusien teknologioiden vuoksi. Ohjelmointikielet ovat JavaScript ja HTML5. Pelin ohjelmointi voidaan toteuttaa millä tahansa tekstinkäsittelyohjelmalla, jossa on koodikieliin tuki.



Kuvio 17. Brackets-ohjelma ja tiedostorakenne

Brackets-teksinkäsittelyohjelma on kevyt ja nopea, joten se valikoitui soveltuvaksi tämän työn pelin toteutukseen. Koodi kommentoidaan ja pelille tehdään kevyt tiedostorakenne työn helpottamiseksi. Tiedostorakenne muodostuu useammasta kansiosta sekä koodeja jaetaan omiin tiedostoihin. (Ks. Kuvio 17.)

Babylonilla toteutettuun peliin pystyi tuomaan JSON-tiedoston, josta voi muodostaa kokonaisen näkymän tai objektin. Pelin sisältö päätettiin toteuttaa Blender-ohjelmalla, sillä se on ilmainen ja siinä on Babylon-liitäntä, jonka avulla Blenderillä mallinnettu sisältö voidaan viedä JSON-tiedostoksi ja käyttää sellaisenaan Babylon-pelissä.

Tämän työn peligrafiikan ei tarvinut olla nykypelien tasoa, mutta tarkoitus on saada pelistä näyttävä ja miellyttävä. Graafinen ulkoasu määräytyy toimeksiantajan toiveiden ja toteuttajan mahdollisuuksien mukaan. Työhön päätettiin valita ilmaisia tekstuureja, joita muokataan ja asetetaan 3D-malleihn. Graafinen osuus tekstuureiden osalta jää pieneksi, mutta 3D-mallinnus on yksi suurimmista vaiheista. Kaikki mallinnettu sisältö viedään lopuksi Babylon-projektiin, jonka jälkeen aloitetaan ohjelmoinnin osuus.

Peli on suhteellisen pienikokoinen, joten suorituskykyyn vaikuttavia asioita ei ole kovin paljon. Ohjelmoinnissa tulee ottaa huomioon hyviä käytänteitä ja väistää

WebGL:n ongelmakohdat. Mikäli sovelluskehystä ei olisi valittu, tulisi ohjelmoinnin käytänteitä nojata puhtaaseen JavaScript-ohjelmointtiin liittyvään kirjallisuuteen. Tässä työssä käytänteitä on otettava suoraan Babylon-ohjelmakirjaston menetelmistä. Babylon-yhteisön esimerkeistä rakennetaan pelin perusta, jonka jälkeen peliä lähdetään toteuttamaan omanlaiseksi. Esimerkeistä otettaessa mallia, voidaan peli todeta toimivaksi sekä koodi eheäksi.

4.3 Prosessi

4.3.1 Tekninen osuus

Peliä lähdettiin toteuttamaan rakentamalla perusnäkymä. Näkymään sijoitettiin kamera sekä valaistus. Ennen kuin pelin sisältöä tuotiin näkymään, niin valaistus säädettiin sopivaksi ja kameralle toteutettiin rajatut liikeradat. Kameran oli pyörittävä näkymässä, kiertäen ja kohdentaen suuntansa juuri kyseiseen pisteeseen. Myös kameran vertikaalinen suunta haluttiin rajata, sillä ei ollut olennaista nähdä taivasta tai kattoa vaan ainoastaan liikuteltavat madot. Peliin oli syytä kuitenkin toteuttaa kameran zoomaus lähemmäksi tai loitommaksi esineitä hiiren rullaa pyörittämällä.

```
//random X and Y axis angles
var angles = [Math.PI/2, Math.PI, 0, -Math.PI/2, -Math.PI];

//random positions
var positionsX = [-20, 20];
var positionsZ = [-120, -80];

var positions = [[-20, -120],[-20, -80],[20, -120],[20, -80]];

//array where meshes are added
var puzzlePieces = [];

//takes tempLocation
var pieceInitialLocation = [];

//create as many as requested in initial phase
for (var i = 0; i < numberOfPieces; i++) {
    //piece mesh from PuzzlePiece module
    puzzle = new PuzzlePiece(i, scene, false);

    puzzle.position.y = 15;

    puzzle.scaling.x = 8;
    puzzle.scaling.y = 15;
    puzzle.scaling.z = 5;

    //selecting random angle for each puzzle piece
    puzzle.rotation.x = angles[Math.floor(Math.random()*angles.length)];
```

Kuvio 18. Objektin asennon ja koon määritys

Näkymän luonti

Perusnäkymän luonnin jälkeen tuotiin peliin myös madot sekä kaikki muu sisältö ja tausta, luomaan yleisilme. Tuodut objektit eivät ole suoraan halutussa mittakaavassa, joten näitä skaalattiin haluttuun kokoon. Valaistusta oli säädettävä uudestaan sisällön ollessa näkymässä, sillä objektien tekstuurit näkyivät huonosti ja yleinen valoisuusaste oli alhainen. Valaistuksiin kuului erityyppisiä valoja, joista yksi oli aurinkoa jäljittelevä ja loput spottivaloja. Kaikki pelin sisältö sijaitsee x-, y- ja z-koordinaatistossa, joten osa näistä objekteista oli sijoitettava näkymään, määrittäen niille halutut koordinaatit. (Ks. Kuvio 18.) Tausta ja ei-toiminnalliset objektit tuotiin yhtenä JSON-tiedostona ja ne toteutettiin peliin yksinkertaisella koodilla. Liikuteltavat madot tuotiin omana tiedostona, mutta tiedostosta otettiin luontivaiheessa objekti kerrallaan, sillä jokaiselle asetettiin yksilöllisiä arvoja sekä oma muoto ja tekstuuri. Tällä toteutusmallilla voitiin estää duplikaattien syntyminen, mitä pelin alkuvaiheessa havaittiin. Jotta 3D-mallin omaavaa objektia voitaisiin liikutella tuli muodottomalle objektille määritellä tuon kyseisen mallin ulkomuoto ja tekstuuri tuodusta JSON-tiedostosta.

Objektien valinta

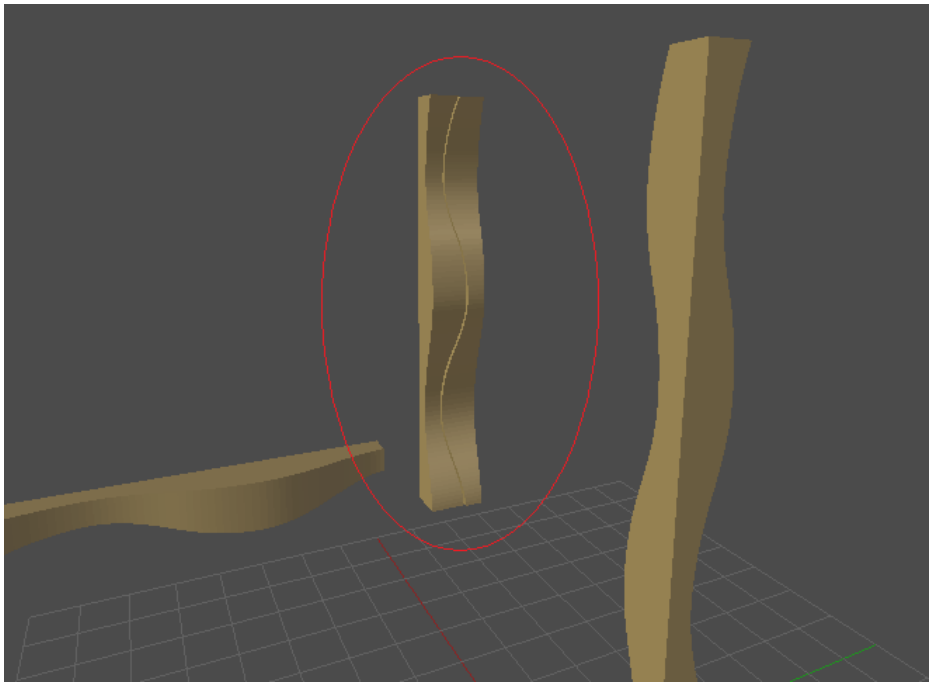
3D-malleista vain madoille oli toteutettava funktionaalisuutta sekä lukuisia asetuksia. Jokainen mato tarvitsi oman id-arvon erottuakseen ryhmästä, johon nämä sijoitettiin. Lisäksi madoilla oli oltava totuusarvomuuttuja (*engl. Boolean*) tunnistamaan mikä mato on milloinkin valittuna tuosta kyseisestä ryhmästä. Boolean-arvoksi voitiin määritellä joko luku 1 tai luku 0 sekä kirjaimellisesti myös true- tai false-arvo. JavaScript tunnisti arvon yksi todeksi ja nolla-arvon epätodeksi. Madot sijoitettiin ryhmään, sillä tämän avulla jokaiselle madolle voitiin antaa sama nimi, mutta eri id- sekä Boolean-arvo, niin että matojen valinta muuttui halutulla tavalla. Mato muuttui valituksi, kun sitä klikattiin hiiren kursorilla. Tässä tilanteessa kyseisen madon Boolean-arvo muuttui luvuksi yksi ja muiden matojen Boolean-arvo asetettiin nolla-arvoiseksi.

Babylonin omista toiminnoista määriteltiin funktioita, joita pelin renderöinti silmukasta (*engl. render loop*) kutsuttiin aina, kun hiiren vasenta näppäintä tai pelin sisäisiä painikkeita klikattiin. Funktiot toteuttivat madon valinnan ja valinnan poiston muista

madoista. Madon ollessa aktiivinen, eli Boolean-arvoltaan yksi ja painikkeita klikattaessa, mato liikkui kyseisten painikkeiden toimintojen mukaisesti.

Törmäykset

Madoille oli määritettävä törmäyksen havainnointi (*engl. collision detection*) estämään matojen siirtyminen toisten matojen, että kentän rajojen (*engl. boundary*) läpi. Babylonilla on omat funktiot törmäyksen havainnointiin sekä ominaisuudet objektin törmäys raja-arvojen (*engl. colllision bound*) määrittämiseen. Kentän rajat toteutettiin luomalla laatikko-objekti ja määrittämällä se näkymättömäksi. Madoille luotiin myös karkea nelikulmio törmäykselle. Törmäyksen havainnointi tapahtui renderöinti silmukan avulla, sillä pelin on kuunneltava tulevia törmäyksiä jatkuvasti. Matojen väliseen törmäykseen toteutettiin näiden yhteensopivuus ja kiinnittyminen. Törmäyksen havainnointi funktion parametreiksi (*engl. parameter*) vietiin kaikkien törmäyksessä esiintyvien matojen id- sekä kylki-arvo ja näille arvoille tehtiin yhteensopivuudelle vertailu. Vertailussa testattiin sopiiko madon id- ja kylki-arvo toisen madon id- ja kylki-arvon kanssa.



Kuvio 19. Kahden objektin liitos

Esimerkiksi tilanne missä mato, jonka id-arvo on yksi ja sen kylki-arvo on yksi törmää matoon, jonka id-arvo on kaksi ja kylki-arvo on kolme tuottaa onnistuneen liitoksen. (Ks. Kuvio 19.) Jos ensimmäisellä madolla olisi kylki-arvo joku muu kuin yksi, olisi liitos

epäonnistunut. Onnistuneen tai epäonnistuneen liitoksen jälkeen oli madoille määritettävä arvo ilmaisemaan näiden ollessa törmäystilassa. Tämä siksi, ettei yhteensopivuuden vertailua tapahtuisi kuin yhden kerran. Tarkastelu tapahtuu uudestaan vasta silloin, kun yhtä madoista on liikutettu erilleen ja takaisin kosketukseen.

Kentän läpäisy

Jokaisen madon kiinnittyessä toiseen, peliin määritelty funktio käy kiinnitykseen liittyneiden matojen arvoja läpi ja vertailee näitä ennalta asetettuihin arvoihin ja tarkistaa näin onko kenttä läpäisty vai ei. Esimerkkinä ensimmäisen kentän neljä arvoa on oltava true ja nämä arvot on tallennettu kentän aloitusvaiheessa taulukkoon false-arvoisiksi. Funktio lähettää true-arvon jokaisesta erillisestä matojen välisestä kiinnittymisestä, ja kun kaikki madot ovat kiinnittyneet, niin neljä true-arvoa on saavutettu. Peli ilmoittaa kentän suoritetuksi, kun funktion vertailusta on palautunut neljä true-arvoa.

Pelin mekanismiin määriteltiin ilmoitus pelaajalle sekä painike seuraavaan kenttään siirtymiselle.

4.3.2 Graafinen osuus

Mallintaminen aloitettiin pyytämällä kuvia Maagiset Madot-pelin paloista, jonka jälkeen aloitettiin 3D-mallien toteuttaminen ottamalla kuvista mallia. Muiden esineiden kohdalla mallia otettiin internetistä haetuista kuvista. Esimerkiksi vasaran tai taltan mallintaminen helpottui huomattavasti, kun ohjelman editointitilan taustalla on siitä kuva. Työssä pyrittiin pitämään alhainen polygonien määrä, mutta pelin ollessa todella pieni, ei polygonien määrä voinut kasvaa merkittävälle tasolle.

Pelissä esiintyvä tausta ja toiminnalliset objektit ovat 3D-malleja ja näillä on tekstuurit. Ilmaisia ja laadukkaita tekstuureja oli tarjolla erillisellä web-sivulla, josta tekstuurit haettiin ja käytettiin sellaisenaan. Tekstuurit olivat sellaisilla lisensseillä, jotka soveltuivat markkinoitavaan peliin ja sen sisällöksi. Tekstuureja oli muokattava peliin sopivaksi ja muokkaus tapahtui grafiikkaohjelmia käyttäen. Grafiikan toteuttaminen aloitettiin kenttien suunnitellun sisällön perusteella. Pelin kolme kenttää ovat samannäköisiä, eli vain matojen tavoiteltava muoto eroaa kenttäkohtaisesti. Pelin

kenttien sisältö on puuverstas, joten kentässä oli oltava puuverstaaseen liittyviä asioita.

Tekstuurin lisääminen 3D-mallin pintaan tapahtui monivaiheisesti. Ensin oli valittava 3D-malli aktiiviseksi, minkä jälkeen mentiin muokkaus-tilaan ja valittiin kaikki vertekset. Seuraavaksi valittiin Blenderin UV-map-toiminto ja 3D-mallin vertekseistä muodostui 2D-levityskuva. Tuohon 2D-levityskuvaan avattiin tekstuuri jpeg-tiedostona ja asetettiin se halutulla tavalla x- ja y-koordinaattien mukaisesti. UV-map-toiminnon jälkeen muutokset olivat toteutuneet siten, että tekstuuri piirtyi 3D-mallille odotetusti, kun se vietiin Babylon-liitännällä JSON-muotoon ja tuotiin peliin.

Kuvio 20. Mallinnus huonekaluista

Esineet mallinnettiin ensin yksi kerrallaan ilman tekstuureja ja muita ominaisuuksia. (Ks. Kuvio 21.)

Kuvio 21. Mallinnus ja teksturointi työkaluista

Mallintamisen jälkeen objekteille toteutettiin UV-map-toiminto ja tekstuurit määriteltiin halutulla tavalla. (Ks. Kuvio 22.)

Kuvio 22. Pelin puuverstas

Sisällöntuottaminen määriteltiin listaamalla asioita, jota kyseisellä puuverstaalla haluttiin näkyvän. Näitä olivat verstaan oven sekä ikkunoiden lisäksi esimerkiksi tuolit, pöydät, ruuvipenkki, kaapit, vasarat, höylät, taltat, käsiporat sekä muita vastaavia työkaluja. (Ks. Kuvio 23.) Ruuvipenkki nousi jo suunnittelun alkuvaiheessa ideaaliksi

kohteeksi, missä pelin kaikki toiminta tapahtuisi, eli matojen liikuttaminen ja yhdistäminen.

Esineet tuotiin verstaan muokkaustilaan ja tässä vaiheessa kokonaisuudelle tehtiin tarvittavia muutoksia. Tämän jälkeen verstas muutettiin Babylon-liitännällä JSON-muotoon ja vietiin peliin.

Grafiikkaa toteutettiin myös käyttöliittymän eri osille. Adoben Photoshop-kuvankäsittelyohjelmalla toteutettiin käyttöliittymän painikkeet sekä toimintopaneeli, johon painikkeet muodostettiin. Myös päävalikko ja sen painikkeet toteutettiin Photoshopilla.

4.4 Tulokset

Projekti kesti alle kolme kuukautta ja vaatimusten sekä suunitelman mukainen pelidemo saatiin osittain toteutettua. Toimeksiantaja sai ensimmäisen version tuotteestaan, jota tullaan jatkokehittämään tämän työn toteuttajan kanssa.

Tutkimuksesta saatiin jokseenkin syvä ymmärrys ja oppi selainpohjaisen 3D-pelin toteutuksen vaiheista, siihen liittyvistä teknologioista sekä ongelmakohdista.

Tutkimuksen pelin toteutuksen vaiheisiin liittyi joitain ongelmia. Oman 3D-mallin vienti peliin ja sen käyttö objektina ei ollut yksinkertainen. Mallin käyttöönotto sekä toiminnallisuuksien määrittäminen tuotti hankaluuksia. Objektien pyörittäminen x-, y- ja z-akselilla ei toiminut luonnollisesti vaan sen luonnolliseen pyörittämiseen oli toteutettava erilainen pyöritys-metodi. Babylon-ohjelmakirjasto ei tukenut HTML-elementtejä joissain tapauksissa, mikä aiheutti ongelmia menun sekä painikkeiden käytössä, mutta tämä saatiin liitännäisen myötä toteutettua. Tekniikoiden selvitys tuotti haasteita, sillä vaihtoehtoja oli niin monia ja tämän tutkimuksen aiheen alle muodostui useita vaiheita eri tekniikoista.

Tavoitteena oli saada peli vaatimustenmukaiseksi sekä vastaamaan suunnitelmaa. Peli vastasi suunnitelmia kokonaisuutena, mutta muutoksia oli tehtävä alkuperäiseen suunitelmaan pelinkehityksen vaiheissa.

Tavoitteet ja vaatimukset täyttyivät lähes odotetusti. Tavoitteesta jäi uupumaan kenttien välinen siirtyminen sekä muita pieniä ominaisuuksia, jotka kuitenkin päätettiin siirtää jatkokehityksen puolelle.

Pelin julkaisu ja arvosteluista saatuja tietoja ajateltiin merkittävinä tulosten kannalta. Tuloksia ei kuitenkaan saavutettu, sillä pelin olisi oltava palvelimella ja käyttövalmiina sekä pelin olisi oltava pelaajien tavoitettavissa vähintään kuukauden ajan, jotta haluttuja tulosksia oltaisiin saavutettu. Pelin sisältö ja mekanismi sekä aiheesta saavutettu tietämys osoittautui kuitenkin tärkeäksi jatkokehityksen kannalta.

5 Pohdinta

Toteutukseen liittyi ajatus pelin toteuttamisesta täysin itsenäisesti. Lähtökohtaisesti toteutus oli hyvin hankalaa, mutta mahdollista. Ehkä vähintään kaksi henkilöä voisi toteuttaa virtaviivaisen pelinkehityksen. Pelistä jäi uupumaan joitain osia, jotka siirrettiin jatkokehitykseen. Peli olisi vastannut täysin odotettua tulosta, jos projektin pituutta olisi jatkettu noin kuukaudella tai kahdella.

Useasta lähteestä selvisi, että pelin olisi voinut toteuttaa myös puhtaasti WebGL-tekniikalla ja näin olisi yksi asia vähemmän tutkittavaa ja näin tutkimusta olisi voitu rajata entisestään. Peli on kuitenkin vasta demo ja sitä tullaan jatkokehittämään pidemmälle, joten pelimoottorin sisältävä sovelluskehys oli hyvä ja varma valinta.

Jokaista pelinkehitykseen kuuluvaa asiaa ei esitelty täysin, joten ei voida olla varmoja onko tutkimuksen tulosten pohjalta mahdollista lähteä soveltamaan täysin valmiin pelin toteutusta. Selainpohjaisen WebGL-pelin voi toteuttaa, jos aiheen lähteitä käytetään tutkimuksen lisäksi.

5.1 Vastauksia tutkimuskysymyksiin

Millaisilla tekniikoilla selainpohjainen 3D-peli voidaan toteuttaa?

Selainpohjaisen peli oli web-selaimella toimiva sovellus, jolla on peliin viittaavia elementtejä, eli peliin täytyi määritellä selaimeen soveltuvat tekniikat. Pelkällä WebGL-tekniikalla pystyy toteuttamaan täysin toimivan 3D-pelin, mutta tämän lisäksi myös siihen pohjautuvilla sovelluskehyksillä. Jotta 3D-peli toimisi selaimella ilman WebGL-

tekniikka, niin vaati se jonkinlaisen liitännän, kuten Flash tai Shockwave. Joillain pelimoottoreilla voi olla omat liitännät selaintueksi. Selainpohjainen peli voi joidenkin lähteiden mukaan olla myös työpöytä sovellus, joka käyttää runkona selaimeen liittyviä ominaisuuksia. Työpöytä sovellus, jota kutsutaan clientiksi, ottaa yhteyttä peliä tarjoavaan palveluun ja käyttäjä pääsee tuolloin pelaamaan peliä. (Kuryanovich ym. 2012.)

Kuinka toteutetaan digitaalinen versio Maagiset Madot pelistä?

Toimeksiantajan yhdessä laaditun suunnitelman mukaan pelistä toteutetaan 3D-peli web-selaimelle. Projektissa käytettään ilmaisia ohjelmistoja, joiden avulla pelin sisältö ja ohjelmointi saadaan toteutettua ilman kuluja. Tekniset valinnat määräytyvät aiheen teorian pohjalta. Maagiset Madot-pelin madoista muodostettaan 3D-mallit ja kentän visuaalinen ympäristö luodaan yhteisen mielipiteen mukaisesti. Kenttä toteutetaan 3D-ympäristönä, mutta ympäristön voi luoda 3D-mallinnettuna tai vain 2D-taustakuvana.

Peliin ei toteuteta merkittävää määrää funktioita, jotta peli pystytään toteuttamaan hyvin pienillä resursseilla ja osa teknisistä haasteista pystytään välttämään. 3D-mallintamiseen sekä käyttöliittymän toteutukseen käytetään paljon aikaa, sillä pelin rooli on toimia mainostavana.

5.2 Pelin jatkokehitys

Peliin tullaan tekemään suunnitelma jatkokehitystä varten. Seuraavat toimenpiteet tulee olemaan äänien lisääminen, erinäköisten matojen toteutus, käyttäjätilin luonti-ominaisuus sekä pelin vienti palvelimelle. Pelin voidaan sisällyttää pistetilasto, käyttäjän rekisteröinti ja sen pelituloksen vieminen pistetilastoon, toisten käyttäjien nähtäväksi, pelin ollessa palvelimella.

Lähteet

Allan, S. 2015. Interlocking Puzzles: Get The 101 Here. Viitattu 22.10.2016. https://www.siammandalay.com/blogs/puzzles/53444867-interlocking-puzzles-get-the-101-here.

Arora S. 2014. WebGL Game Development. Viitattu 29.10.2016. http://site.ebrary.com.ezproxy.jamk.fi:2048/lib/jypoly/home.action.

Brandt, C. 2011. Blender: Free, Open-Source 3D Software With a Steep Learning Curve. Viitattu 14.11.2016. http://www.pcworld.com/article/244442/blender.html.

Cook, D. 2016. Comparison Of Three WebGL Libraries. Viitattu 13.11.2016. http://darrendev.blogspot.fi/2016/03/comparison-of-three-webgl-libraries.html.

Cocilowa, A. 2013. The future of video games will be in your browser. Viitattu 15.10.2016. http://www.pcworld.com/article/2048967/the-future-of-video-games-will-be-in-your-browser.html.

Cox, A. 2016. The best web browser 2016. Viitattu 06.11.2016. http://www.techradar.com/news/software/applications/best-browser-which-should-you-be-using-932466.

Danchilla, B. 2012. Beginning with WebGL for HTML5. Viitattu 09.10.2016. http://www.books24x7.com/books24x7.asp.

Fronczak, T. 2011. Blender: Animation Features Worth Knowing. Viitattu 14.11.2016. http://www.animationcareerreview.com/articles/blender-animation-features-worth-knowing.

Hanselman, S. 2014. Easy accelerated 3D Games in a browser with JavaScript and WebGL using Three.js or Babylon.js. Viitattu 13.11.2016. http://www.hanselman.com/blog/EasyAccelerated3DGamesInABrowserWithJavaScriptAndWebGLUsingThreejsOrBabylonjs.aspx.

Hawkes, R. 2011. Foundation HTML5 Canvas: For Games and Entertainment. Viitattu 10.11.2016. http://www.books24x7.com/books24x7.asp.

Kuryanovich, E. Shalom S., Goldenberg, R., Paumgarten, M., Strauß, D., Lee-Delisle, S., Renaudeau, G., Wagner J., Bergknoff, J., Danchilla B. & Hawkes R. 2012. HTML5 Games Most Wanted: Build the Best HTML5 Games. Viitattu 10.11.2016. http://www.books24x7.com/books24x7.asp.

noeticsunil. 2015. Top 10 HTML5, JavaScript 3D Game Engines and Frameworks. Viitattu 14.11.2016. http://noeticforce.com/best-3d-javascript-game-engines-frameworks-webgl-html5.

Ramtal, D. & Dobre, A. 2014. Physics for JavaScript Games, Animation, and Simulations: with HTML5 Canvas. Viitattu 10.11.2016. http://www.books24x7.com/books24x7.asp.

Rettig, P. 2012. Professional HTML5 Mobile Game Development. Viitattu 10.11.2016. http://www.books24x7.com/books24x7.asp.

Rodriguez, S. 2013. Google, Mozilla are tweaking browsers for Web-based games. Viitattu 16.10.2016. http://www.latimes.com/business/la-fi-browser-games-20131115-story.html.

Rose, M. 2013. Jagex turns to HTML5 for RuneScape 3. Viitattu 14.10.2016. http://www.gamasutra.com/view/news/189228/Jagex_turns_to_HTML5_for_RuneScape_3.php.

Simonds, B. 2013. Blender Master Class: A Hands-On Guide to Modeling, Sculpting, Materials, and Rendering. Viitattu 10.11.2016. http://www.books24x7.com/books24x7.asp.

Specifications. BabylonJS:n internetsivut. Viitattu 11.11.2016. http://www.babylonjs.com/#specifications.

van der Spuy, R. 2015. Advanced Game Design with HTML5 and JavaScript. Viitattu 17.10.2016. http://www.books24x7.com/books24x7.asp.

Sukin, I. 2013. Game Development with Three.js. Viitattu 14.10.2016. http://site.ebrary.com.ezproxy.jamk.fi:2048/lib/jypoly/home.action.

Sullivan, S. Runescape Review. Viitattu 15.10.2016. https://mmos.com/review/runescape.

Weber, R. Game Development - Babylon.js: Advanced Features for Enhancing Your First Web Game. Viitattu 12.11.2016. https://msdn.microsoft.com/en-us/magazine/mt614269.aspx.

The History of Online Browser Games. 2012. Wowin internetsivut. Viitattu 15.10.2016. http://www.wowin.com/news/category/the-history-of-online-browser-games.