jamk.fi

Selainpeli ideasta toteutukseen

Päivi Niinikangas

Opinnäytetyö
Joulukuu 2014

Mediatekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

KUVAILULEHTI

Tekijä(t) Niinikangas, Päivi	Julkaisun laji Opinnäytetyö	Päivämäärä 2.12.2014
	Sivumäärä 48	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty (X)

Työn nimi
Selainpeli ideasta toteutukseen

Koulutusohjelma
Mediatekniikan koulutusohjelma

Työn ohjaaja(t)
Manninen, Pasi

Toimeksiantaja(t)
Nianas

Tiivistelmä

Opinnäytetyön tavoitteena oli toteuttaa toimiva pelikokonaisuus HTML5- ja JavaScript-tekniikoita hyödyntäen. Pelin toteutuksessa on painotettu erityisesti suunnittelua. Opinnäytetyössä on pyritty johdonmukaisesti selvittämään pelin kehityksen vaiheet alkaen ideoinnista ja päättyen valmiiseen kokonaisuuteen. Työn tarkoituksena on antaa lukijalle kuva pelikehityksen prosessista, sekä selvittää selainpohjaisten tekniikoiden tuomia mahdollisuuksia pelien kannalta. Opinnäytetyö toteutettiin tekijän yritystoiminnan kehittämiseksi.

Opinnäytetyössä toteutetun pelin kehitys eteni vaiheittain siten, että idean pohjalta rakennettiin suunnitelma, jossa eriteltiin peliin tarvittavat resurssit, selostettiin pelin kulku sekä listattiin tarvittavat ääni- ja grafiikkaresurssit. Pelin suunnitelma pyrittiin laatimaan käytettävissä olevat resurssit huomioon ottaen siten, että peli oli mahdollista myös toteuttaa sellaisenaan.

Suunniteltu peli pohjaa ideansa suomalaisen kansanperinteen metsänpeitto-tarinoihin. Peli on mekaniikaltaan pulmapeli, jossa interaktiiviset pulmaosiot vuorottelevat kerronnallisten välinäytösten kanssa.

Suunnitelman pohjalta valittiin työssä hyödynnettävät teknologiat, minkä jälkeen aloitettiin varsinainen toteutus. Toteutusvaihe jakaantui ohjelmointiin, grafiikan toteutukseen sekä äänisuunnitteluun.

Opinnäytetyössä päästiin haluttuun tavoitteeseen. Työn tuloksena syntyi suunnitelman mukainen peli. Opinnäytetyössä toteutettua peliä ei ole tarkoitus jatkokehittää, mutta työn kuluessa opitut tekniikat ja tietoperusta toimivat tukena tulevissa projekteissa.

Avainsanat (asiasanat)
Pelisuunnittelu, HTML5, JavaScript

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Description

Author(s)	Type of publication	Date
Niinikangas, Päivi	Bachelor's/~~Master's~~ thesis	2.12.2014
		Language of publication: Finnish
	Number of pages 48	Permission for web publication: x

Title of publication

Browser game - from idea to execution

Degree programme

Media Engineering

Tutor(s)

Manninen, Pasi

Assigned by

Nianas

Abstract

The goal of the thesis was to create a completed, functioning game using HTML5 and JavaScript technologies. The work focused heavily on game design. The thesis documents the steps of the chain of events starting from the idea and ending with the completed game in a logically structured manner. The intent of the thesis is to give the reader an outlook on the process of game design, and examine the possibilities browser based technologies bring to games. The thesis was created to further the entrepreneurship of the author.

The development of the game created in the thesis progressed in phases, so that a game design plan specifying the needed resources, outlining the story of the game, and listing the needed sound and graphics resources was created based on the initial idea. The plan was created by taking into account the available resources, so that the game could be completed based on it.

The idea of the game was based on the stories of being spirited away common in Finnish folklore. The game is a puzzle game, where interactive puzzle segments alternate with narrative interludes.

The execution of the game began in earnest after the utilized technologies were chosen based on the plan. The execution phase was divided into three parts: programming, graphics, and sound design.

The thesis reached its intended goal. A game outlined in the design plan was created as the result of the work. The game created in the thesis will not be developed further, but the techniques learnt in the process and the acquired knowledge base will offer a solid support in future projects.

Keywords/tags (subjects)

Game design, HTML5, JavaScript

Miscellaneous

SISÄLTÖ

KUVIOT

TAULUKOT

KÄSITTEET

AIR	Adoben kehittämä ajoympäristö, joka mahdollistaa verkkosovellusten rakentamisen siten, että niitä on mahdollista käyttää työpöytäsovelluksina.
API	Ohjelmointirajapinta, mahdollistaa eri ohjelmien välisen tiedon vaihdon ja pyynnöt.
Canvas	HTML5:n elementti, jota hyödynnetään grafiikan dynaamiseen esittämiseen, usein JavaScriptin avulla.
Gamification	Pelillistäminen, tarkoittaa pelimäisten elementtien tuomista eipelilliseen ympäristöön käyttäjän osallistumisen lisäämiseksi.
JavaScript	Pääasiassa verkkosovelluksissa käytetty oliopohjainen skriptikieli, jota käytetään tuomaan verkkosivuille dynaamisia ominaisuuksia. Modernit verkkoselaimet tukevat JavaScriptiä oletusarvoisesti.
HTML5	HTML-merkintäkielen uusin versio. Lisäsi aiempaan uusia ominaisuuksia, kuten canvas-, audio- ja video-elementit.
Osoita ja klikkaa	Pelimekaniikka, jossa eteneminen tapahtuu hiirellä osoittamalla ja klikkaamalla.
Pelimoottori	Pelinkehityksen ohjelmistokehys, joka mahdollistaa erilaisten ominaisuuksien, kuten fysiikkamoottorin tai grafiikanmallinnuksen, hyödyntämisen ilman, että pelinkehittäjän on niitä itse ohjelmoitava.
Piirrosanimaatio	Animaation tekniikka, jossa liikkeen illuusio luodaan nopeasti vaihtuvilla yksittäisillä kuvilla.
Preload	Alustava lataus, joka mahdollistaa sovelluksen käyttämien resurssien lataamisen ennen varsinaisen sovelluksen käynnistämistä. Tällä saadaan käyttökokemus saumattomammaksi ja estetään lataustauot kesken sovelluksen käytön.

Puzzle- tai pulmapeli Peli, jonka mekaniikka koostuu erilaisista pulmista, joita pelaajan on ratkottava edetäkseen.

SDK Software development kit, kokoelma ohjelmiston rakennukseen tarkoittavia työkaluja.

Seikkailupeli Peligenre, joka painottaa tarinaa, ongelmaratkaisua sekä hahmojen kehitystä toiminnan kustannuksella.

Sprite Peligrafiikan elementti, tyypillisesti hahmomalli. Tässä opinnäytetyössä sprite-termiä käytetään myös muista dynaamisista pelielementeistä.

Spritesheet Kuvakartta, joka sisältää sprite-elementin animaation. Tyypillisesti yksi kuvatiedosto, joka sisältää piirrosanimaation ruudut vierekkäin. Kuvakarttaa luetaan koordinaattien perusteella.

SWF Adobe Flash -ohjelmiston tiedostomuoto.

TypeScript Microsoftin kehittämä JavaScriptin ylijoukkoon kuuluva ohjelmointikieli.

Tween Animaatiotekniikka, jossa kahden ruudun välinen siirtymä generoidaan automaattisesti.

Visual novel Peligenre, jossa painotus on voimakkaasti tarinalla. Visual novel genren pelin sisältävät tyypillisesti varsinaisesti interaktiivisia pelielementtejä karsitusti.

WebGL Web Graphics Library, JavaScriptin ohjelmointirajapinta grafiikan mallintamiseen.

1 TYÖN TAUSTAT

Selainpohjaiset pelit ovat pelinkehityksen nykypäivää, ja tekniikat niiden toteutukseen kehittyvät koko ajan. Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa selainpohjainen peli hyödyntäen HTML5:sta ja JavaScriptiä. Tässä työssä painotus oli selainpohjaisilla teknologioilla, eikä palvelinpuolen ratkaisuja ole otettu huomioon.

Työn tarkoituksena oli erityisesti rakentaa pohja, joka jatkossa mahdollistaa opittujen tekniikoiden hyödyntämisen kaupallisissa ja taiteellisissa projekteissa. Tarkoituksena oli testata samalla sarjakuvamaisen ilmaisun sekä pelillisten elementtien yhdistämistä. Lisäksi erityistä huomiota kiinnitettiin pelin visuaaliseen ilmeeseen ja ulkoasun suunnitteluun. Selainpohjaiset verkkotekniikat mahdollistavat sarjakuvan työstön printtimedian rajoituksista vapaana, ja pelillisyyden tuominen tuo sarjakuvaan uutta ulottuvuutta.

Visuaalisen suunnittelun lisäksi työnkulussa painotettiin itse pelin huolellista suunnittelua. Vaikka suunnitteluvaihe söi huomattavan osan resursseista, oli siihen panostamisesta toteutusvaiheessa korvaamaton apu. Ilman huolellista suunnitelmaa pelin johdonmukainen työstäminen olisi ollut haastavaa.

Tämän opinnäytetyön tarkoituksena on selventää lukijalle pelin suunnittelun sekä toteutuksen prosessia ja antaa kuva siitä, millaista pelinkehitys on selainympäristöön.

2 TAUSTATUTKIMUS JA VALMISTELU

2.1 Tutkimuksen taustoista

Tässä opinnäytetyössä toteutettu peli oli tarkoitettu toimimaan selaimessa ilman lisäosien asentamista. Lisäksi työn tarkoituksena oli tutustua modernien JavaScript-kirjastojen tuomiin mahdollisuuksiin. Näiden kriteerien pohjalta tekniikoiksi valikoitiin HTML5, CSS ja JavaScript.

Taustatutkimus painottui JavaScript-moottorin valintaan. Moottorin valinnassa huomio kiinnittyi sen soveltuvuuteen suunnitellun pelin vaatimuksiin. Lisäksi moottorin valintaan vaikuttivat maksuttomuus sekä dokumentaation laajuus.

Pelin asettamat vaatimukset pelimoottorille olivat tuki 2D-grafiikalle, animaatioille, äänille ja käyttäjän palautteen tunnistukselle. Jälkimmäisellä tässä tarkoitetaan hiiren painallusten tunnistusta. Animaation osalta tuki tuli löytyä sekä piirrosanimaatiolle että tween-siirtymille.

2.2 Testausympäristö

Testausympäristöksi koneelle asennettiin WampServer-ympäristö. WampServer on Windowsille tarkoitettu verkkokehitysympäristö. Se mahdollistaa verkkosovellusten luomisen Apache2:n, PHP:n ja MySQL-tietokannan avulla. Näiden ohessa PhpMyAdmin-ohjelmisto mahdollistaa tietokantojen helpon hallinnoinnin. (Bourdon 2014.) Ilman palvelinympäristöä kirjastojen testaus olisi jäänyt vajaaksi, sillä selaimet rajoittavat JavaScriptin käyttöä paikallisesti tietoturvasyistä.

2.3 Pelimoottorien vertailu

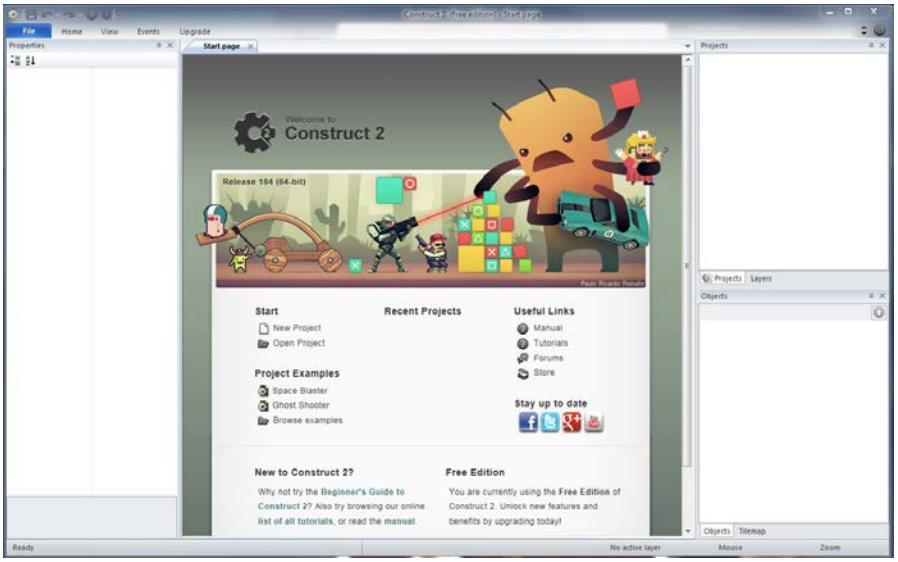
Vertailuun valittiin viisi vaihtelevan laajuista pelimoottoria/kirjastoa. Joukosta karsittiin pois maksulliset versiot. Pois jätettiin myös kirjastot, jotka eivät tukeneet peliin tarvittavia tekniikoita, sekä ne, joiden dokumentaatio oli puutteellinen tai hankalasti tulkittava.

2.3.1 **Construct 2**

Construct 2 on 2D peleihin suunniteltu HTML5-pelimoottori, jonka kehittäjä on englantilainen Scirra Ltd. (Create Games with Construct 2 – Scirra.com 2014). Pelimoottoria markkinoidaan sen erityisellä aloittelijaystävällisyydellä. Construct 2 on pelinteko-ohjelma, mikä tarkoittaa sitä, ettei käyttäjä suoraan kirjoita JavaScriptiä, vaan käyttää ohjelman toimintoja (actions), tapahtumia (events) ja ehtoja (conditions) pelinrakennukseen. Pelimoottori on paljolti testattu ja käytetty ja sillä on aktiivinen kehittäjäyhteisö. (Construct 2 details, reviews, and important links 2014.)

Pelimoottorista on saatavilla myös maksullinen versio, mutta tähän työhön testattiin vain ilmaisversiota. Vertailuun valituista moottoreista Construct 2 oli selvästi pisimmälle kehitetty, vaikkakaan ei välttämättä laajin ominaisuuksiltaan.

Construct 2 asentuu koneelle vaivattomasti .exe-tiedostosta. Opinnäytetyön kirjoittamisen hetkellä tuettuja käyttöjärjestelmiä ovat Windows 8, Windows 7, Vista ja XP. Construct 2:n käyttöönotto nopeaa. Ohjelma toimii sellaisenaan. Se tarjoaa graafisen kehitysympäristön, jossa pelinrakennus on mahdollista vedä ja pudota –logiikalla. (Ks. kuvio 1.)



Kuvio 1. Construct 2:n graafinen käyttöliittymä

Construct 2:n käyttö ja tapahtumapohjainen logiikka on nopea omaksua eikä vaadi ohjelmointitaitoja. Tämä mahdollistaa pelin toteuttamisen myös esimerkiksi ulkoasun suunnittelijalle. Toisaalta tässä helppoudessa piilevät myös sen ongelmat – pelkistetty, palikkamainen pelinrakennus asettaa rajoitteita, joita puhtaasti koodaamalla

ei ole. Palikkamaisessa helppoudessa on toki myös hyvät puolensa. Esimerkiksi nopeaan prototyypin rakennukseen ja peli-idean testaukseen Construct 2 voi olla erinomainen työkalu. (Sidhion 2012).

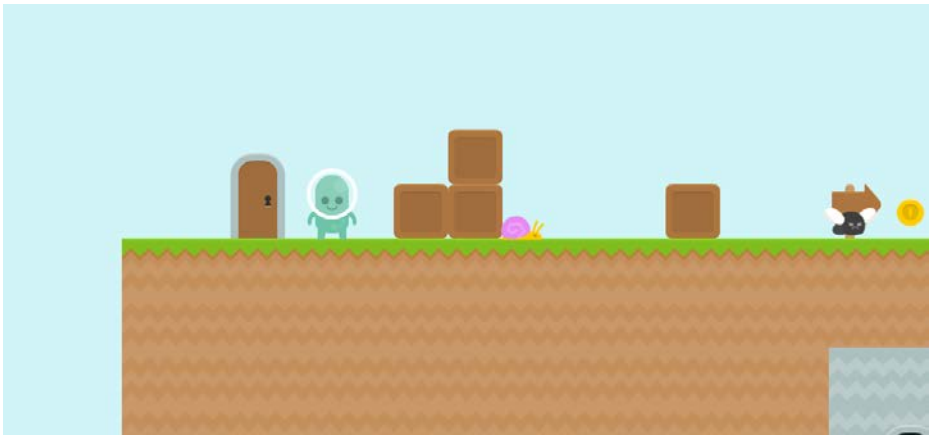
Construct 2 on ladattavissa osoitteesta https://www.scirra.com/construct2.

2.3.2 **Quintus**

Quintus on HTML5-pelimoottori, joka on suunniteltu modulaariseksi ja kevyeksi ytimekkäällä JavaScript-ystävällisellä syntaksilla. Pelimoottori on suhteellisen uusi, mutta aktiivisesti päivitetty. (Quintus details, reviews, and important links 2014.) Se on tarkoitettu sekä mobiili- että työpöytäsovelluksiin.

Quintus pyrkii minimoimaan koodin määrän. Se on rakenteeltaan modulaarinen ja tukee tapahtumapohjaista lähestymistapaa ohjelmointiin. (Munogee 2012). Quintus pyrkii tuomaan olio-ohjelmoinnin periaatteet HTML5-pelinkehitykseen.

Pelimoottorin nuori ikä näkyy vielä toistaiseksi melko suppeassa dokumentaatiossa sekä esimerkkien vähäisyydessä. Toisaalta jo olemassa olevien esimerkkien valossa pelimoottorilta on lupa odottaa paljon, jos kehitystyö vain jatkuu tulevaisuudessakin aktiivisena. Esimerkkinä pelikirjaston mukana tulee muun muassa kuviossa 2 kuvattu tasoloikka.



Kuvio 2. Ruudunkaappaus Quintus-pelimoottorin esimerkkipelistä

Quintus on ladattavissa osoitteesta http://www.html5quintus.com/.

2.3.3 **CreateJS**

CreateJS on yksittäisen kirjaston sijaan kokoelma useammasta modulaarisesta kirjastosta, jotka työskentelevät yhdessä mahdollistaen interaktiivisen sisällön luomisen verkkosovelluksiin HTML5:n avulla. Kirjastot on suunniteltu toimimaan itsenäisesti, mutta ne ovat myös vapaasti yhdisteltävissä tarpeiden mukaan. (CreateJS | A suite of Javascript libraries and tools for building rich, interactive experiences with HTML5 2014.)

CreateJS-kirjastokokonaisuuden muodostavat EaselJS, TweenJS, SoundJS, PreloadJS ja Zoë. EaselJS vastaa HTML5:n canvas-elementin hallinnasta. TweenJS nimensä mukaan siirtymistä. SoundJS sekä PreloadJS hoitavat niin ikään nimensä mukaisia tehtäviä. Zoë puolestaan on AIR sovellus, jonka avulla on mahdollista luoda SWF-animaatioista EaselJS-hahmoarkkeja canvas-elementillä käytettäväksi.

CreateJS-paketin sponsoreista löytyvät muun muassa Adobe ja Mozilla. Kokonaisuus on laajasti dokumentoitu, ja sen esimerkit antavat hyvän kuvan moottorin kyvyistä. CreateJS on hyvä valinta Flash-pohjaiselle pelinkehittäjälle. CreateJS:n esimerkkikirjastosta löytyy muun muassa lyhyitä pelejä, joiden toiminnasta saa hyvän kuvan moottorin suorituskyvystä. (Ks. kuvio 3.)

Kuvio 3. Ruudunkaappaus CreateJS:n Space Rocks -esimerkkipelistä

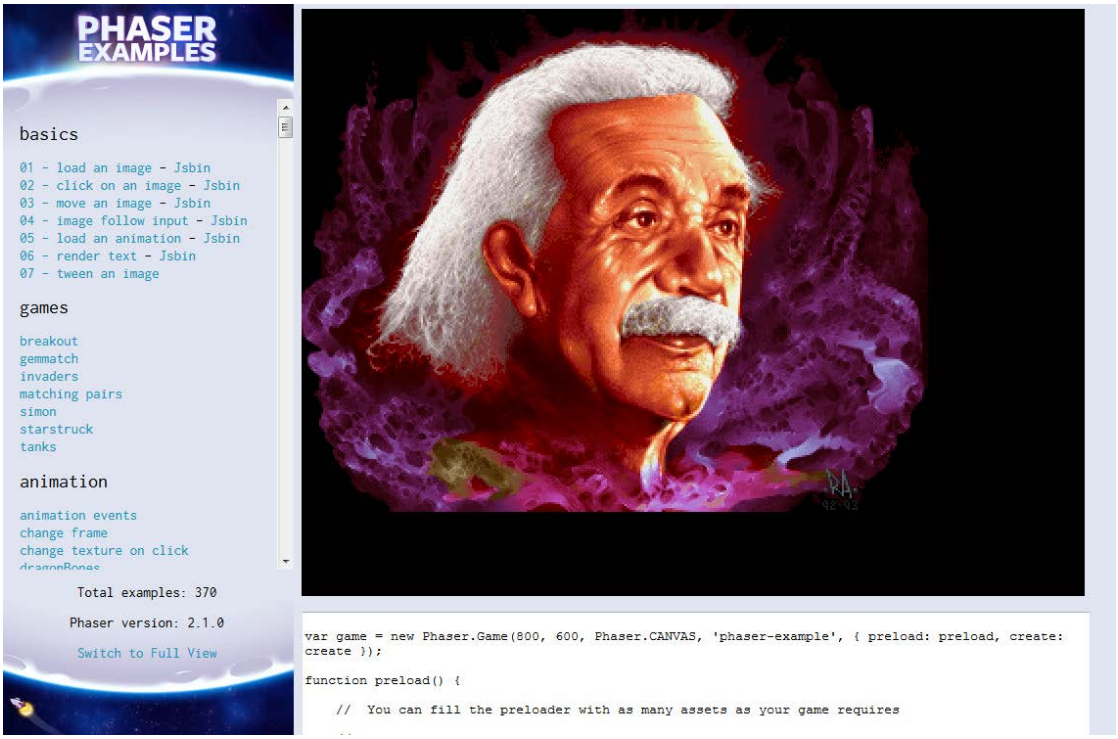
CreateJS-paketti on ladattavissa osoitteesta http://www.createjs.com/. Paketin kirjastot on mahdollista ladata erikseen, ja niitä voi vapaasti yhdistellä tarpeensa mukaan.

2.3.4 **Phaser**

Phaser on HTML5-pohjainen pelinkehityksen viitekehys, joka soveltuu niin mobiili- kuin työpöytäkäyttöön. Se pohjautuu vahvasti Flixel-kirjastoon, ja sen ylläpitäjä ja kehittäjä on Richard Davey. Phaser on vielä kohtalaisen tuore tulokas, mutta se on aktiivisesti ylläpidetty. (Phaser details, reviews, and important links 2014.)

Phaser tukee JavaScriptin lisäksi myös TypeScriptiä. TypeScript on Microsoftin kehittämä JavaScriptiin pohjaava ohjelmointikieli. Se kääntyy Javascriptiksi, mutta on kehittäjän mukaan parempi suurien sovellusten kehittämiseen. (Bright 2012.)

Phaserin dokumentaatio on kattava ja hyvin ylläpidetty, ja sen esimerkkikirjasto on erinomaisesti jäsennelty, mikä helpottaa sen käytön omaksumista huomattavasti. (Ks. kuvio 4.)



Kuvio 4. Ruudunkaappaus Phaser-pelimoottorin esimerkkikirjastosta

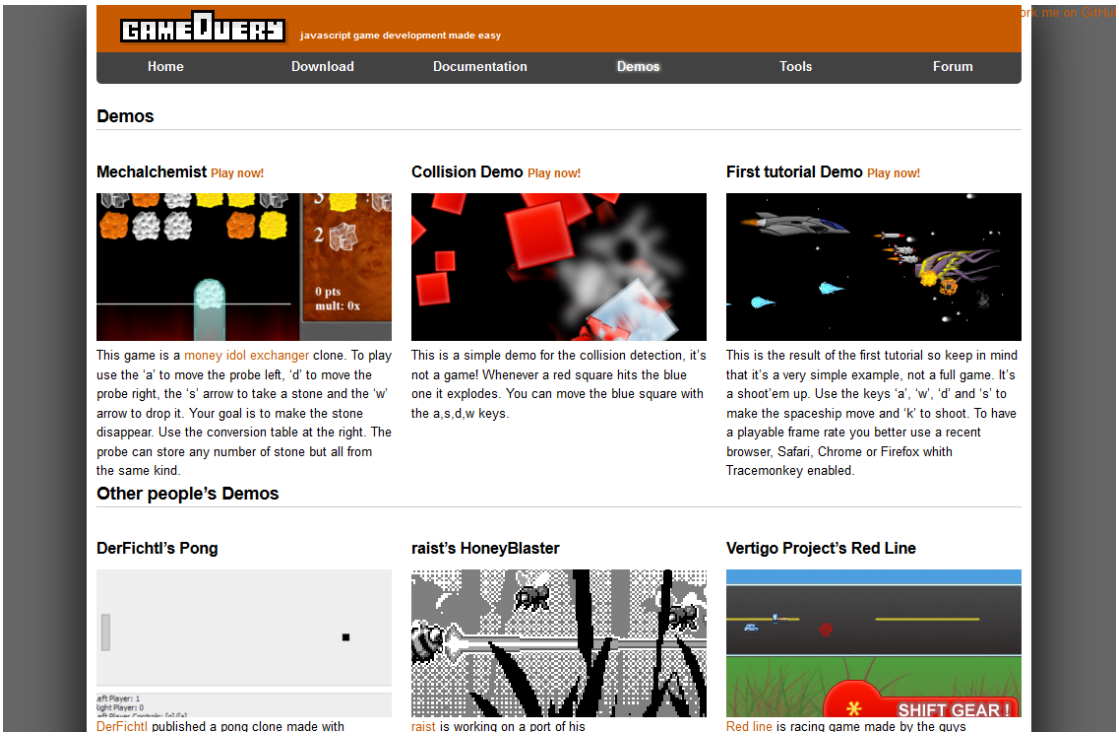
Grafiikan mallintamiseen Phaser käyttää Pixi.js-kirjastoa. Pixi.js on 2D-grafiikkaan erikoistunut moottori webGL-tuella. Tarvittaessa Pixi.js hyödyntää HTML5:n canvas-elementtiä.

Phaseria on mahdollista käyttää myös graafisessa kehitysympäristössä selainpohjaisen Mighty Editorin avulla.

Phaser löytyy ladattavaksi osoitteesta http://phaser.io/. Mighty Editor puolestaan löytyy osoitteesta http://mightyfingers.com/.

2.3.5 GameQuery

GameQuery on, kuten nimikin vihjaa, jQueryn liitännäinen. Se on suppein kaikista tähän vertailuun valituista kirjastoista, ja otettiin mukaan lähinnä jQueryn tunnettuuden vuoksi. GameQueryn mukana ei tule erillistä esimerkkikirjastoa, mutta sen verkkosivuilta löytyy lista sillä toteutetuista demoista. (Ks. kuvio 5.)



Kuvio 5. Ruudunkaappaus GameQuery-kirjaston verkkosivuilta

GameQuery on kelvollinen valinta pientä ja näppärää kirjastoa etsivälle, jQueryyn tottuneelle kehittäjälle. Sen dokumentaatio on kuitenkin suppea, eikä kehitystyö kovin aktiivista.

GameQueryn kotisivut löytyvät osoitteesta http://gamequeryjs.com/.

2.4 Vertailun tulokset

Vertailun pohjalta pelin toteutukseen valittiin Phaser-kirjasto. Sen tarjoama esimerkkikirjasto oli vertailun laajin, ja se suoriutui vaivattomasti kaikista pelin mekaniikan vaatimuksista. Lisäksi sen käyttöönotto ja käytön opettelu onnistuvat luontevasti.

Muista kirjastoista CreateJS-paketti nousi kärkeen, mutta hävisi lopulta Phaserille dokumentaation laajuudessa. Tulevaisuudessa CreateJS voi mahdollisesti tarjota paremmin viritetyn kokonaisuuden, mutta vielä toistaiseksi se kaipaa tuekseen enemmän esimerkkejä moottorin mahdollisuuksista. Lisäksi paketin laajuus tekee siitä osittain hankalasti lähestyttävän. CreateJS:n tulevaisuuden puolesta puhuvat vakuuttavat sponsorit.

Myös Quintus ansaitsee maininnan. Sillä on mahdollisuuksia kehittyä pitkälle, jos kehitystyö vain jatkuu, mutta toistaiseksi dokumentaation puutteellisuus aiheuttaa haasteita erityisesti aloittelevalle pelinkehittäjälle. Construct 2 sen sijaan sopii erinomaisesti aloittelijalle, mutta karsiutui pois palikkamaisen kankeutensa vuoksi. Construct 2 sopii hyvin graafiseen kehitysympäristöön tottuneelle.

GameQuery suoriutui mukisematta pelimoottorille asetetuista vaatimuksista, mutta sen dokumentaation suppeus ja kehitystyön epävarmuus tiputtivat sen harkinnasta.

Vertailun tueksi kirjastot pisteytettiin taulukon 1 mukaisesti vaadittujen ominaisuuksien osalta. Pisteet jaettiin asteikolla 1-5 siten, että 5 on paras ja 1 huonoin.

Taulukko 1. Pelikirjastojen pisteytys

	Construct 2	**Quintus**	**CreateJS**	**Phaser**	**GameQuery**
2D-grafiikka	4	4	5	5	3
Animaatio	3	3	4	4	3
Audio	4	3	4	3	2
Käyttäjän palaute	4	3	3	5	5
Dokumentaatio	4	2	3	5	1
Yhteensä	19	15	19	22	14

3 PELI-IDEASTA SUUNNITELMAKSI

3.1 Lähtökohdat suunnitteluun

Pelin tai minkä tahansa muun luovan prosessin voi tiivistää kolmeen vaiheeseen: idea, suunnittelu ja toteutus. Näistä prosessin suunnittelu on kriittisin vaihe. Suunnittelu on tehtävä huolellisesti, mutta kuitenkin niin, että suunniteltu tuote on mahdollista toteuttaa. Paisuessaan kohtuuttomaksi suunnitteluvaihe uhkaa näivettää koko pelin, ja erinomainenkin idea saattaa jäädä pelkäksi puheeksi. (Boyer 2014.)

Pelisuunnittelussa huomiota on kiinnitettävä käytössä oleviin resursseihin. Niiden pohjalta voidaan rakentaa suunnitelma, joka on mahdollista myös toteuttaa ilman, että lopputuloksesta joudutaan tinkimään.

Suunnitelma kannattaa kuitenkin toteuttaa huolellisesti, sillä mitä huolellisemmin suunnittelutyö on tehty, sitä helpompi on edetä toteutuksessa. Tässä opinnäytetyössä toteutetun pelin suunnittelussa pyrittiin huomioimaan interaktiivisen tarinankerronnan viisi periaatetta:

1) Keskity tarinankerrontaan.
2) Pelillisten elementtien tulisi käyttää suurin osa peliajasta
3) Interaktiivisuuden tulee tukea tarinaa
4) Ei turhaa toiminnallista toistoa
5) Ei ylitsepääsemättömiä esteitä pelissä etenemiselle.

(Grip 2013.)

Edellä listattujen periaatteiden lisäksi tässä opinnäytetyössä toteutetun pelin suunnittelussa painotettiin erityisesti pelin rakennetta ja toiminnallisuutta sekä visuaalista ilmettä.

Logiikaltaan tässä opinnäytetyössä käsitelty peli on osoita ja klikkaa -tyylinen pulmapeli. Pelaaja ohjaa pelin toimintoja hiiren oikealla painikkeella klikkailemalla. Pelin kulku on jaoteltu dialogia sisältäviin sarjakuvamaisesti eteneviin välinäytöksiin sekä pelaajan toimintaa vaativiin pulmiin. Tarkoitus oli sisällyttää peliin tarinan kaarta ja samalla tutkia, kuinka pelillisten elementtien ja sarjakuvallisen ilmaisun yhdistäminen

onnistuu. Tarinalliset osuuden vertautuvat osittain visual novel -genren pelimekaniikkoihin.

Peli on suunniteltu hallituksi kokonaisuudeksi, joka on tarkoitettu näkymään pelaajalle tietyllä tavalla. Tästä johtuen peli toimii tarkoituksenmukaisesti vain määrätyllä kokoonpanolla, eikä sitä ole optimoitu esimerkiksi mobiilikäyttöön.

3.2 **Idea**

Pelin taustalla oli suomalaisen kansanperinteen ajatus metsänpeitosta. Ajateltiin, että metsä oli metsänväen valtakuntaa ja erinäisistä rikkomuksista metsänhenkeä kohtaan saattoi joutua metsänpeittoon. Metsänpeitossa oleva ei pystynyt löytämään tietä takaisin kotiin ilman apua. Eksyneen etsijät eivät nähneet häntä, vaikka olisivat kulkeneet hänen vierestään. (Kimmo 2006.) Tähän pohjaten, pelin ideana on avustaa päähahmoa pakenemaan metsänpeitosta.

Peli-idean työstäminen alkoi hahmojen sekä pelin rakenteen ja pulmien mekaniikan luonnostelusta. Pelin idea eli jonkin verran alkuperäisestä ajatuksesta, mutta suurin osa elementeistä pysyi samana ideoinnista toteutukseen. (Ks. kuvio 6.)

Kuvio 6. Alustava suunnitelma pelin kulusta ja hahmoista

Pelin nimeksi valikoitui Henki. Vaikka peli on muuten englanninkielinen, tuntui suomenkielinen nimi luontevalta sen aihepiiri huomioon ottaen. Varsinaisesti peli ei kuitenkaan pyri erityiseen suomalaisen kulttuurin kuvaukseen vaan pikemminkin fantasiamaiseen ilmaisuun.

3.3 **Rakenne**

Pelin rakenne pyrittiin pitämään mahdollisimman yksinkertaisena. Tarkoituksena oli toteuttaa lyhyt, hallittu ja yhtenäinen kokonaisuus, joka jaoteltiin selkeisiin osioihin: pulmiin ja välinäytöksiin. (Ks. taulukko 2). Välinäytöksiin on luettu myös pelin prologi ja loppunäytös.

Taulukko 2. Pelin rakenne

Vaihe	**Kuvaus**
Prologi	Lyhyt johdanto peliin
Pulma 1	Pelaajan on käänneltävä pelinäkymään jaotellut paneelit oikeaan järjestykseen
Välinäytös 1	Henki saapuu paikalle, lyhyt dialogi hengen ja paimenen välillä
Pulma 2	Pelaajan tulee etsiä pelinäkymään piilotetut kappaleet hengen naamiosta
Välinäytös 2	Lyhyt dialogi hengen ja paimenen välillä
Pulma 3	Pelaajan tulee järjestellä pelinäkymä siten, että sen keskelle muodostuu polku
Loppunäytös	Paimen pääsee metsänpeitosta, pohtii oliko kaikki vain unta

Pelin osioista pulmat ovat varsinaista interaktiivista sisältöä, ja välinäytökset puolestaan passiivisempia. Pulmissa pelaajan tehtävänä on löytää ratkaisu esitettyyn ongelmaan. Välinäytöksissä pelaajan rooli jää tarkkailijaksi, jonka tehtävänä on ainoastaan seurata dialogin kautta esitettyä tarinaa. Välinäytösten tarkoituksena on myös tarjota pelaajalle rauhallisempi tauko aktiivisten osioiden välissä.

Peli alkaa sarjakuvatyylisellä johdannolla, jossa pelin hahmot ja lähtökohta esitellään pelaajalle viitteellisesti. Johdannossa pelin päähenkilö, pelaajan avustettava paimenpoika, nukahtaa vahtiessaan lammaslaumaa laitumella metsässä. Paikalle saapuu

metsänhaltija, pelin nimen henki. Nukkuvan paimenen huomattuaan henki kuljettaa tämän mukanaan metsänpeittoon.

Johdannon jälkeen pelaajan on ratkaistava pelin ensimmäinen pulmatehtävä. Tehtävässä pelialue on jaettu kolmeen paneeliin, jotka pelaajan on käänneltävä oikeaan järjestykseen. Kun pelaaja on selvittänyt pulman, alkaa ensimmäinen välinäytös.

Ensimmäisessä välinäytöksessä henki ilmestyy paikalle. Henki ja paimen keskustelevat lyhyesti. Hengen naamio hajoaa kolmeen palaseen, minkä jälkeen peli siirtyy toiseen pulmaan.

Toisessa pulmassa pelaajan tehtävänä on löytää hengen naamion pelialueelle ripotellut palaset. Kun palaset ovat kasassa, peli etenee toiseen välinäytökseen, jossa henki ja paimen keskustelevat jälleen.

Pelin kolmas pulma aukeaa toisen välinäytöksen jälkeen. Pulmassa pelaajan tehtävä on järjestellä pelinäkymään puut siten, että näkymän keskelle aukeaa polku. Kun pelaaja on selvittänyt kolmannen pulman, seuraa loppunäytös.

Loppunäytöksessä paimen pääsee ulos metsänpeitosta pohtien, mahtoiko kaikki olla vain unta. Paimenen takana erottuu hengen hahmo metsän reunassa. Varsinainen pelinäkymä siirtyy tämän jälkeen pois, ja peli päättyy listaukseen pelintekijätiedoista.

3.4 **Hahmot**

3.4.1 **Hahmosuunnittelu**

Pelissä on kaksi päähahmoa, nimettömät henki ja paimen. Hahmosuunnitteluun haettiin vaikutteita erityisesti animaation maailmasta. Hahmomallit pyrittiin pitämään yksinkertaisina ja tunnistettavina. Hahmojen ulkoasun suunnittelussa haluttiin ilmentää hahmojen olemusta ja luonnetta, sillä pelin tarinan kesto rajoitti osaltaan hahmojen syventämistä. Hahmomallien oli erotuttava selkeästi toisistaan, mutta kuitenkin tyylin tuli olla siten yhtenäinen, että hahmot toimivat saman tarinan sisällä.

3.4.2 **Henki**

Henki on toinen pelin hahmoista. Henki piilottaa paimenen metsänpeittoon ilkikurisuuttaan. Varsinaisesti henki ei ole pahantahtoinen vaan pikemminkin leikkisä. Henki on ulkoasultaan pitkä ja hoikka, repaleisiin housuihin pukeutunut hahmo, joka on peittänyt kasvonsa sarvekkaalla naamiolla. (Ks. kuvio 7.)

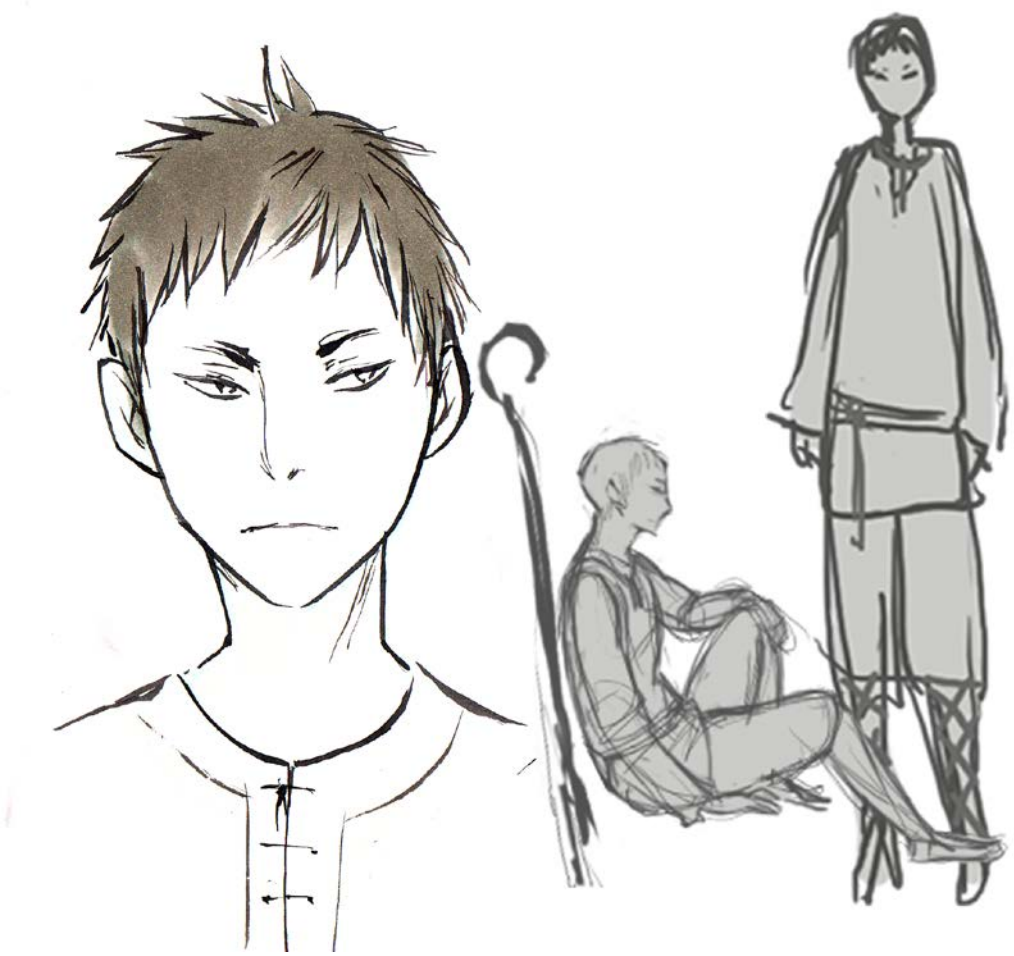
Kuvio 7. Luonnoksia henki-hahmosta

Hengen hahmoon pyrittiin tuomaan salaperäisyyttä, ja visuaalisesti ilmentämään hahmon yliluonnollista alkuperää. Hahmon suunnittelussa lähtökohtana oli ajatus epäinhimilllisyydestä, mutta kuitenkin siten, ettei hahmoa voi pitää hirviömäisenä tai vastenmielisenä.

Hengen hahmon olennaisin ominaispiirre on hahmon kasvoja peittävä sarvekas naamio, jota käytetään myös pelillisenä elementtinä pelin toisessa pulmassa.

3.4.3 **Paimen**

Paimen on hengen lisäksi toinen pelin hahmoista. Paimen on pelissä pelaajan pelastettavana. Ratkaisemalla pelin pulmat paimen vapautuu metsänpeitosta. Paimen on nuori mies, jolla on lyhyet harmaanruskeat hiukset sekä yksinkertainen, fantasiavaikutteinen vaatetus. (Ks. kuvio 8.)



Kuvio 8. Luonnoksia paimen-hahmosta

Paimenen hahmon suunnittelussa lähtökohtana oli maanläheisyys. Tätä pyrittiin ilmentämään erityisesti hahmon värityksen kautta. Vaikka peli ei varsinaisesti ole leimallisen suomalainen, pyrittiin paimen-hahmon olemukseen tuomaan viitteitä suomalaisuudesta. Vastapainona henki-hahmon yliluonnolliselle olemukselle paimenesta pyrittiin suunnittelemaan vakavampi ja selkeästi pelin reaalimaailmaan sidottu.

3.5 **Visuaalinen ilme**

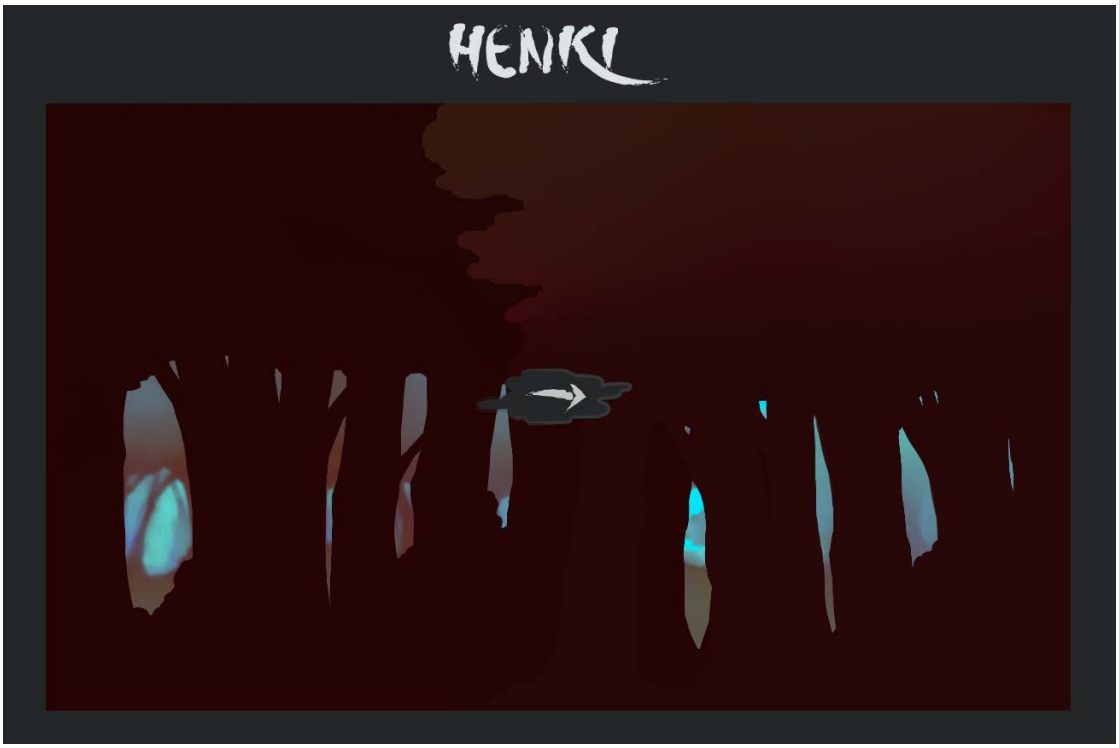
Pelin visuaalinen ilme on suunniteltu selkeäksi ja esteettisesti miellyttäväksi. Grafiikan toteutustapa on digitaalinen. Taustoissa on käytetty maalausmaista tekniikkaa ja hahmomalleissa puolestaan pelkistettyä vektorigrafiikkaa.

Visuaalisen ilmeen suunnittelussa vaikutteita on haettu animaation ja sarjakuvien kuvakielestä. Eritysti vaikutteita on haettu Disneyn Bambi-elokuvasta ja japanilaisen Ghibli-animaatiostudion Prinsessa Mononoke -elokuvasta. Piirroselokuvamaisuutta on pyritty tuomaan mukaan yhdistämällä pelkistetyt hahmomallit realistisempaan taustaan. Sarjakuvallista ilmettä peliin pyrittiin tuomaan typografisten valintojen kautta. Suunnittelussa on pyritty luomaan tasapaino eri elementtien välillä ja pitämään visuaalinen ilme yhteneväisenä.

Visuaalisen ilmeen suunnittelussa on pyritty erottamaan toisistaan pelimaailman todellinen ulottuvuus sekä metsänpeiton henkimaailma. Henkimaailman väriskaalassa toistuvat kylmät, erityisesti siniset sävyt. Todellisen maailman värit taas ovat sävyiltään lämpimämmät. Värimaailmojen eroja on havainnollistettu kuviossa 9.

Kuvio 9. Ylhäällä pelin reaalimaailman väritystä, alhaalla esimerkki henkimaailmasta

Pelimaailman lisäksi suunnittelussa on otettu huomioon myös verkkosivu, jolla peli pyörii. Sivu on pidetty mahdollisimman yksinkertaisena, ja huomio siinä on keskittynyt pelialueeseen. Värisävyiltään sivu on tumma. (Ks. kuvio 10.)

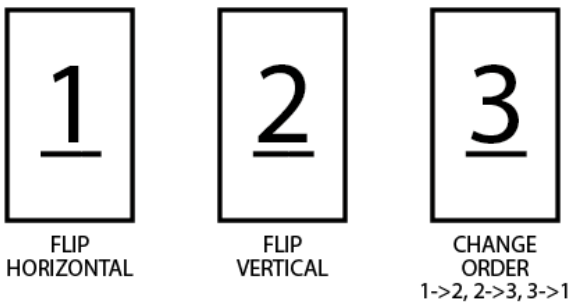


Kuvio 10. Ruudunkaappaus pelin verkkosivulta

3.6 **Pulmat**

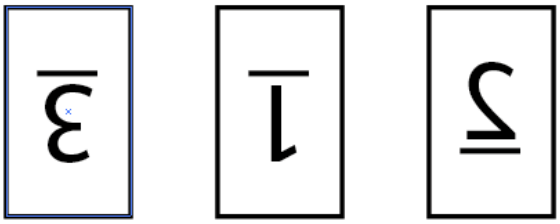
3.6.1 **Pulma 1**

Pelin ensimmäisessä pulmassa pelialue on jakautunut kolmeen paneeliin, jotka liikkuvat niitä klikkaamalla kuviossa 11 määritellyn logiikan mukaan.



Kuvio 11. Ensimmäisen pulman siirtymien logiikka

Paneelit on pulman alkuun käännelty väärään järjestykseen (ks. kuvio 12). Pelaajan tehtävä on selvittää pulman logiikka ja kääntää paneelit oikeaan järjestykseen.

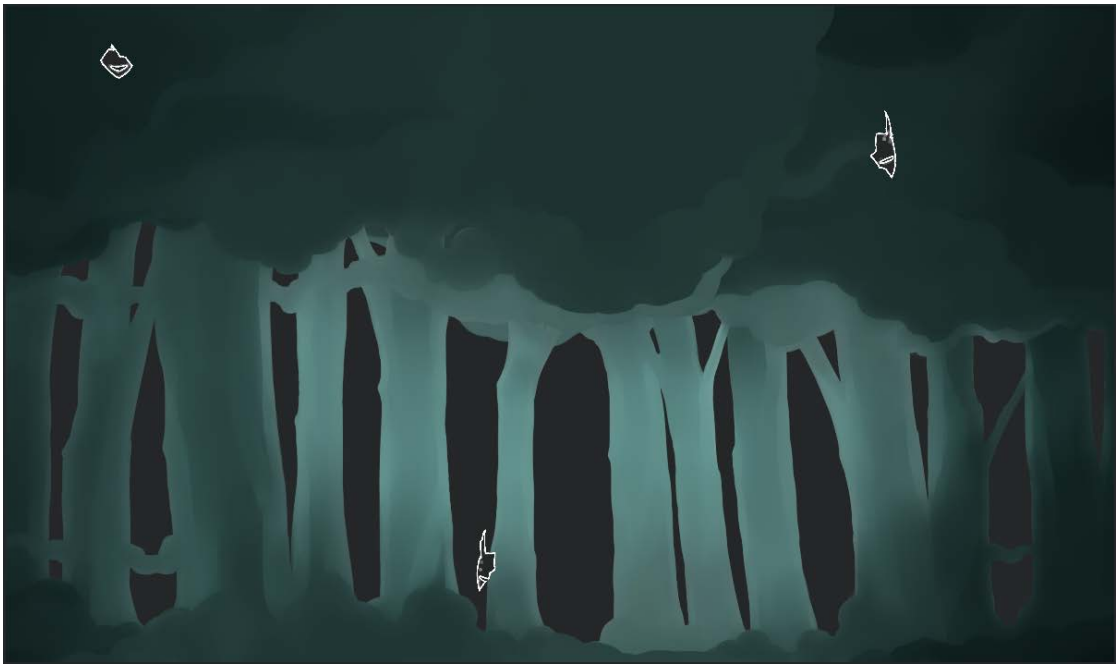


Kuvio 12. Ensimmäisen pulman alkuasetelma

Pulman taustalla on idea kansanperinteestä – metsänpeitosta päästäkseen oli käännettävä jotain nurinpäin. Pelissä idea tosin on päinvastainen. Samoin ajatuksena on järjestää pelimaailma paimenen näkökulmasta järkevään muotoon.

3.6.2 **Pulma 2**

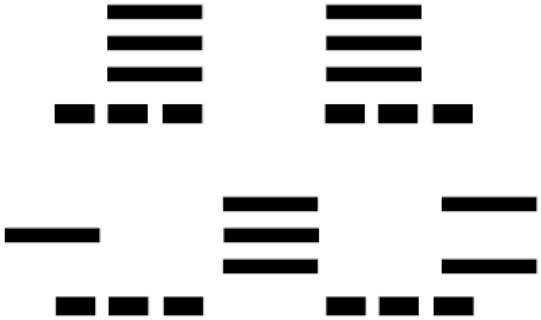
Toisen pulman ideana on löytää pelialueelta hengen käyttämän naamion palaset. Henki antaa pelaajalle tehtäväksi selvittää olemuksensa kokoamalla palat kasaan. Naamion palaset on piilotettu maisemaan. (Ks. kuvio 13.) Pelaajan tehtävä on kerätä niitä hiirellä klikkailemalla.



Kuvio 13. Naamion palaset sijoiteltuna taustan päälle pulmassa kaksi

3.6.3 **Pulma 3**

Kolmannessa pulmassa pelaajan tehtävä on järjestellä pelimaailman puut siten, että niiden keskelle muodostuu polku. Pulman ajatuksena on selvittää siten tie ulos metsänpeitosta. Pulman alkuasetelmassa polku on peitossa, kuten kuviossa 14 on havainnollistettu.



Kuvio 14. Pelin kolmannen pulman ratkaisu- ja alkuasetelma

Puut on jaoteltu neljään riviin – ensimmäisen rivin kolmea vasemmanpuoleista puuta klikkaamalla taaempien rivien puut vaihtavat paikkoja kaksi riviä kerrallaan. Ensimmäisen rivin oikeanpuoleiset puita klikkaamalla puolestaan rivien puut vaihtavat paikkaa yksi rivi kerrallaan. Pulma on mahdollista selvittää useammalla eri tavalla, mutta ratkaisu vaatii molempien puolien puiden hyödyntämistä.

3.7 **Dialogi**

Pelin dialogi on rajoitettu esi-, väli- ja loppunäytöksiin. Pulmat on tarkoituksella jätetty keskustelusta vapaaksi pelimekaniikan rauhoittamiseksi. Dialogissa on pyritty luomaan tunnelmaa sekä avaamaan pelin juonta. Kaikki dialogi käydään pelin kahden hahmon, paimenen ja hengen, välillä.

Tässä opinnäytetyössä käsitellyssä pelissä pelaajan ei ole mahdollista vaikuttaa pelin vuorosanoihin.

Pelissä ei ole ääninäyttelyä, vaan dialogi etenee puhtaasti tekstipohjaisena. Kulloinkin puhuva hahmo on pyritty erottelemaan typografian keinoin.

3.8 **Äänimaailma**

Pelin äänimaailmassa pyrittiin visuaalisen ilmeen tapaan erottamaan metsänpeiton maailma sekä todellinen maailma toisistaan. Äänien valinnassa kiinnitettiin huomiota erityisesti kokonaisuuteen. Äänien tehtävänä pelissä on luoda tunnelmaa.

Vaikutteita äänisuunnitteluun haettiin kauhuelokuvista sekä japanilaisesta kabuki-teatterista. Nämä näkyvät erityisesti pelin henkimaailmaosiossa. Pelin reaalimaail-massa äänillä pyrittiin antamaan viitteitä metsäisestä ympäristöstä valikoimalla käyt-töön nauhoituksia todellisesta luonnosta.

4 SUUNNITELMASTA PELIKSI

4.1 Työn kulku

Suunnittelun ja pelimoottorin käyttöönoton jälkeen varsinainen työstö alkoi väliaikaisten tuotantografiikoiden luomisella. Tuotantografiikat mahdollistavat koodirakenteen testauksen ja nopean prototyypin rakennuksen, ennen kuin lopulliset grafiikat on viimeistelty. Tämä mahdollistaa nopeamman työskentelyn tiimissä, jossa ohjelmoinnista ja grafiikoista vastaavat eri henkilöt, mutta nopeuttaa työtä myös projekteissa, joissa yksi ihminen vastaa molemmista.

Tuotantografiikoilla voidaan testata koodin lisäksi myös grafiikoiden toimivuus ja karsia ja täydentää resursseja niiden osalta tarpeen mukaan. Tuotantografiikkoina voivat toimia yksinkertaiset luonnokset, joita on mahdollista hyödyntää myöhemmin lopullisten grafiikkojen luomisessa. (Ks. kuvio 15.)

Kuvio 15. Esimerkki tuotantografiikoista

Tuotantografiikoiden jälkeen työ eteni HTML-sivupohjan ja CSS-tyylitiedoston sekä JavaScript-tiedostorakenteen luomiseen. HTML-rakenne on yksinkertainen, sillä käytännössä kaikki pelin elementit tulevat JavaScriptin kautta. JavaScript-rakennetta pyrittiin selkiyttämään hajauttamalla pelin eri osiot eri tiedostoihin. Tämä helpotti

tuotantovaiheessa työskentelyä merkittävästi, kun tiedostokohtainen rivimäärä jäi kohtuulliseksi.

Työn varsinainen ohjelmointi tapahtui JavaScriptillä. Kehityksessä käytettiin Notepad++-editoria ja työtä testattiin Firefox- sekä Chrome-selaimilla. Ohjelmointi vei suunnittelun jälkeen suurimman osan resursseista.

Kun koodipohja oli saatu kuntoon, viimeisteltiin grafiikka ja testattiin sen toimivuus käytännössä. Grafiikan muutosten pohjalta koodirakennetta oli säädettävä erityisesti animaatioiden osalta.

Grafiikan viimeistelyn jälkeen keskityttiin äänimaailmaan. Äänistä ei käytetty tuotantoversiota, sillä ääniä valitessa on lähes mahdoton saada ajoitus kuntoon, jos testiääni ja lopullinen ääni poikkeavat toisistaan esimerkiksi kestoltaan. Tuotantografiikoissa tätä ongelmaa ei ole, sillä on vaivaton tehdä lopullisesta grafiikasta mittasuhteiltaan luonnoksen kokoinen. Tässä opinnäytetyössä toteutetussa pelissä äänien valinta jätettiin pelin viimeistelyyn, mikä tarkoitti sitä, että koodia oli muokattava vielä sen jälkeen. Tämä oli mahdollista, sillä äänien lisääminen Phaser-kirjaston avulla sujui vaivattomasti, eikä vaatinut kohtuuttomia muutoksia.

Äänien lisääminen oli työn viimeinen suurempi vaihe. Sen jälkeen työssä keskityttiin viimeistelyyn ja rakenteen siistimiseen.

4.2 HTML ja CSS

Pelin HTML ja CSS-rakenteet ovat pelkistetyt. HTML-tiedostossa on perusmäärittelyjen lisäksi lähinnä header-elementti, joka sisältää pelin logon, ja div-elementin pelille. Lisäksi HTML sisältää kutsut JavaScript-tiedostoihin ja tyylitiedostoon.

Phaser ei vaadi nimettyä div-elementtiä pelille, mutta id- tai class-määritteen lisääminen mahdollistaa CSS:n tarttumisen elementtiin.

Tyylitiedosto sisältää taustavärin ja määrittelyn, jolla pelialue ja logo keskitetään sivulle. Pelin sisäiset tyylittelyt on hoidettu JavaScriptissä tai kuvina.

4.3 JavaScript

4.3.1 Rakenne

JavaScript-rakenne jaettiin useaan eri tiedostoon tuotannon helpottamiseksi taulukon 3 mukaisesti. Lopulliseen versioon tiedostot olisi voinut yhdistää, mutta ne jätettiin erillisiksi ulkopuolisen tarkastelun helpottamiseksi.

Taulukko 3. Pelin tiedostorakenne

Tiedostonimi	**Kuvaus**
henki.js	Vastaa muiden tiedostojen kutsumisesta oikeassa järjestyksessä
manager.js	Vastaa siirtymistä eri pelivaiheiden välillä hallinnoimalla seuraava-napin painalluksia
start.js	Vastaa pelin aloitusnäkymän luomisesta
prologue.js	Vastaa pelin esinäytöksen toiminnoista
puzzle1.js	Vastaa pelin ensimmäisen pulman toiminnoista
interlude1.js	Vastaa pelin ensimmäisen välinäytöksen toiminnoista
puzzle2.js	Vastaa pelin toisen pulman toiminnoista
interlude2.js	Vastaa pelin toisen välinäytöksen toiminnoista
puzzle3.js	Vastaa pelin kolmannen pulman toiminnoista
epilogue.js	Vastaa pelin jälkinäytöksen toiminnoista

Muiden tiedostojen kutsujen lisäksi henki.js sisältää funktion, jolla tuhotaan pois käytöstä siirtyvät resurssit pelin aikana. Lisäksi se sisältää pelin luonnin määrittelyllä

```
var game = new Phaser.Game(1200, 800, Phaser.AUTO, 'gamearea',
{
      preload: preload,
      create: create,
      update: update
});
```

Tässä määritellään pelialueen koko, 1200 kertaa 800 pikseliä. Phaser.AUTO-määrittelyn avulla Phaser valitsee automaattisesti käyttöön joko webGL:n tai canvas-elementin, riippuen selaimen tarjoamasta tuesta. Gamearea viittaa HTML-tiedostossa määritettyyn div-elementtiin, johon Phaser sijoittaa pelin. Preload, crea-

te ja update ovat Phaserin oletusarvoisia peliosioita. Preload lataa nimensä mukaan käytetyt resurssit etukäteen, create vastaa niiden luomisesta pelimaailmaan ja update puolestaan huolehtii näkymän päivityksestä. Niiden tilalle voi halutessaan määritellä omat osiot, mikä helpottaa peliosioiden hallinnointia. Peliosiot huolehtivat myös resurssien tuhoamisesta niiden välillä. Tässä opinnäytetyössä peliosioita ei voitu hyödyntää, sillä pelin rakenteen vuoksi resursseja hyödynnetään useammassa kuin yhdessä vaiheessa. Resurssien lataus aina erikseen olisi lisännyt muistinkulutusta tarpeettomasti.

Manager.js sisältää switch case-rakenteen, joka hallinnoi seuraava-napin painamista pelin sisällä. Seuraava-napilla siirrytään pelissä vaiheista toiseen switch case-rakenteeseen annetun parametrin mukaisesti. Parametrinä käytetään automaattisesti päivittyvää muuttujaa, joka laskee klikkausten määrää, ja määrittää niiden mukaan seuraavan pelivaiheen.

Start.js-tiedosto sisältää pelin alkunäkymän elementit ja käynnistää varsinaisen pelin. Peli käynnistyy pelaajan klikattua aloitus-painiketta, joka käynnistää pelin esinäytöksen kutsuvaa funktiota startbuttonPressed.

```
function startbuttonPressed () {

      if (startbuttonClickCheck === false) {
            startbuttonClickCheck = true;

            kling.play();

            startbuttontween.to(
            {alpha:0}, 3000, Phaser.Easing.Linear.None, true)
            .onComplete.add(function() {
            destroySprite(startbutton);
            });

            lefttween.to(
            {x: '-1500' }, 3000, Phaser.Easing.Linear.None,
            true)
            .onComplete.add(function() {
            destroySprite(treesleft);}
            );

            righttween.to( { x: '+1000' }, 3000, Phas-
            er.Easing.Linear.None, true)
            .onComplete.add(function() {
            destroySprite(treesright);});

            startPrologue();
      }

}
```

Funktiossa startbuttonClickCheck-muuttuja tarkastelee onko painiketta jo painettu. Tällä estetään tuplaklikkaukset. Startbuttontween, lefttween ja righttween huolehtivat eri elementtien siirtymistä. Siirtymien lopuksi elementit tuhotaan destroySprite-funktiota kutsumalla, sillä niitä ei tarvita seuraavissa pelivaiheissa. Pelin esinäytös käynnistetään kutsumalla funktiota startPrologue, joka sijaitsee tiedostossa henki.js.

Prologue.js ja epilogue.js ovat rakenteellisesti melko samanlaiset, samoin interlude1.js ja interlude2.js. Ne vastaavat pelin alku-, loppu- ja välinäytöksiin liittyvien toimintojen hallinnoinnista. Nämä peliosiot sisältävät enimmäkseen animaatiota ja dialogia.

Puzzle1.js-, puzzle2.js- ja puzzle3.js-tiedostot sisältävät pelin pulmatehtäviin liittyvät toiminnot. Näissä paino on pelaajan toiminnan käsittelyllä. Pulmat ovat pelin aktiivisia osioita. Muun pelin tapaan pulmissa edetään klikkailemalla hiirellä pelin eri elementtejä. Klikkausten käsittely määräytyy pulman logiikan mukaisesti.

4.3.2 **Pelielementit**

Phaser hyödyntää useita erityyppisiä pelielementtejä. Näistä tavallisesti käytettyjä ovat muun muassa image-, sprite-, audio- ja teksti-elementit. Näiden lisäksi käytettävissä on myös esimerkiksi koodillisesti piirrettäviä grafiikkaelementtejä, mutta niitä ei ole tässä opinnäytetyössä käsitellyssä pelissä otettu käyttöön.

Image on nimensä mukaan kuvaelementti. Se on tyypiltään staattinen, ja toimii esimerkiksi taustakuvana. Toimintoja siihen ei voida kohdistaa. Sprite-elementti puolestaan on kuva, johon voidaan kohdistaa toimintoja. Esimerkiksi animaatioita varten elementit on tyypitettävä spritenä. Audio-elementti on nimensä mukaisesti äänitiedostoja varten. Phaser tukee useita erilaisia äänitiedostotyyppejä, mutta niiden käytössä on huomioitava mahdolliset selaimen tuomat rajoitukset.

Ulkopuolista tiedostoa hyödyntävät kuvaelementit pohjustetaan preload-funktiossa se seuraavasti:

```
game.load.image('frame', 'img/frame.png');
```

Tässä frame on tunniste, jolla kuvaan viitataan myöhemmin, ja frame.png puolestaan tiedostonimi, jolla kuva löydetään img-hakemistosta. Äänitiedostojen lataaminen onnistuu samaan tapaan, vaihtamalla image-määritteen tilalle audio.

Elementit lisätään peliin seuraavasti:

```
sheepbg = game.add.sprite(-5000, 710, 'sheepbg');
```

Tässä spritelle määritellään nimi ja sijainti, sekä kuva joka siihen yhdistetään. Tässä esimerkissä elementti on sijoitettu x-akselin suhteen koordinaatistossa 5000 pikseliä pelialueen vasemmalle puolelle. Sijainti johtuu siitä, että elementti liikutetaan myöhemmin tween-siirtymällä pelialueelle, jolloin se liukuu kuvaan vasemmasta reunasta. Audio-elementille sijaintia ei määritellä. Alustuksen lisäksi audio-elementit on käynnistettävä koodissa halutussa kohtaa.

Tekstiä voidaan lisätä peliin useammalla tavalla. Näistä tässä opinnäytetyössä käsitellyssä pelissä on käytetty kahta tapaa.

Ensimmäinen tekstityyppi toimiin samaan tapaan kuin HTML-määritelty teksti ja sille voidaan antaa tyylimäärittelyt CSS-tyylien tapaan seuraavasti:

```
pzl1InfoText1 = game.add.text(game.world.centerX, 740, 'Click
on the panels to move them.', {font: "12px 'Trebuchet MS'",
fill: "#e1e9e9", align: "center"});
```

Tämä tapa ei tarvitse resurssien pohjustusta preload-funktiossa, mutta fonttivalikoima on rajoitettu.

Toinen tekstityyppi hyödyntää bittikarttaa fontin luomisessa. Fontista luodaan kuvatiedosto, josta kaikki halutut kirjaimet löytyvät, ja xml-tiedosto, joka kertoo Phaserille kunkin kirjaimen sijainnin ja välistyksen. Tämä tekstityyppi ladataan preload-funktiossa seuraavasti:

```
game.load.bitmapFont('spirittext', 'fonts/spirittext.png',
'fonts/spirittext.xml');
```

Teksti lisätään peliin halutussa vaiheessa seuraavalla tavalla:

```
int1Dia9 = game.add.bitmapText(80, 350, 'spirittext',"Who
knows?", 22);
```

Bittikarttatekstien hyödyntäminen mahdollistaa laajemman valikoiman fontteja, ja mahdollistaa tekstin muotoilun halutulla tavalla.

Bittikarttaa ei tarvitse generoida käsin, vaan sen luomiseen löytyy valmiita työkaluja. Tässä opinnäytetyössä bittikarttojen generointiin käytettiin selainpohjaista Littera-palvelua. Siinä käyttöön haluttu fontti ladataan TrueType-tiedostona palveluun, missä kirjasinten ulkonäköä on mahdollista rajoitetusti säätää. Kun halutut säädökset on tehty, palvelu generoi fontista png-tiedostomuotoisen kuvakartan ja fnt-muotoisen tiedoston. Fnt-tiedosto on mahdollista muuttaa suoraan tiedostopäätettä muokkaamalla xml-tiedostoksi, jolloin se on hyödynnettävissä Phaserin kanssa. Littera löytyy osoitteesta http://kvazars.com/littera/.

4.3.3 **Animaatio**

Opinnäytyössä toteutetusta pelistä löytyy kahdentyyppistä animaatiota: piirrosanimaatiota sekä tween-siirtymiä. Piirrosanimaatiossa animaatio tapahtuu vaihtamalla nopeaan tahtiin yksittäisiä kuvia, jolloin syntyy illuusio liikkeestä.

Perinteinen piirrosanimaatio on tekniikkana todella työläs, sillä jokainen ruutu on piirrettävä erikseen. Tietokoneella toteutettuna animaatiossa etuna on se, että hahmomallit on helposti mahdollista kopioida ruudusta toiseen ja siten hyödyntää osia jo piirretyistä malleista uudelleen. Kuitenkin uskottavan liikkeen aikaansaamiseksi ruutuja on piirrettävä suuri määrä. Tässä opinnäytetyössä käsitellyssä pelissä animaatiot on jätetty tarkoituksella typistetyiksi ja hidasta ruudunpäivitystä käytetään tehokeinona.

Phaser toteuttaa piirrosanimaation hyödyntämällä spritesheet-kuvakarttoja, jossa jokainen animaation ruutu on kuvattu samaan kuvatiedostoon. Kuviossa 16 on nähtävissä esimerkki neljän ruudun hahmoanimaatiosta yksinkertaisella liikkeellä.

Kuvio 16. Esimerkki spritesheet-kuvakartasta

Koodillisesti piirrosanimaatio tapahtuu seuraavasti:

```
game.load.spritesheet('spirit-behind-tree',
'img/sp-behind-tree.png', 249, 612);
```

Spritesheet-kuva ladataan preload-funktiossa staattisten kuvaelementtien tapaan. Kuvan latauksessa sille määritellään id, jolla kuvaan viitataan (tässä `spirit-behind-tree`), annetaan tiedostonimi ja määritellään yhden animaatioruudun koko (tässä 249 kertaa 612 pikseliä). Phaser osaa ruudun mittasuhteiden perusteella ladata automaattisesti kaikki spritesheetillä olevat ruudut. Jos spritesheetin koko ei ole jaollinen ruudun koolla, tulee kutsussa määritellä ladattavien ruutujen lukumäärä.

```
sp1 = game.add.sprite(-250, 80, 'spirit-behind-tree', 0);
```

Animoitava sprite alustetaan muuttujaksi create-funktiossa. Viimeinen parametri (0) määrittää mikä ruutu spritesheetistä ladataan kun sprite asetetaan näkymään.

```
sp1.animations.add('headtilt', [0,1]);
```

Animaatiot alustetaan antamalla animaatiolle nimi, ja määrittämällä animaatioruudut, jotka halutaan ottaa käyttöön. Animaatioruutujen numerointi alkaa ohjelmoinnista tutun periaatteen mukaisesti nollasta.

```
sp1.animations.play('headtilt', 1, false);
```

Animaatiota kutsutaan koodissa halutussa kohtaa yllä olevaan tapaan. Yhteen spriteen voidaan yhdistää useampia animaatioita, joten kutsussa ilmoitetaan mitä niistä tarkoitetaan. Numero 1 viittaa siihen kuinka monta ruutua sekunnissa näytetään. Tässä tapauksessa ruutuja näytetään yksi sekunnissa. False-parametri viittaa animaation toistoon. True-arvolla animaatio toistuu yhä uudelleen siihen asti, että se pysäytetään.

Tween-siirtymissä kuvaa siirretään, tai sen arvoja muutetaan laskennallisesti ilman, että välivaiheen ruutuja tarvitsee piirtää käsin. Tween-animaatio on mahdollista toteuttaa Phaserissa seuraavasti:

```
game.load.image('puzzle1-trees-left', 'img/pzl1-forestbg-
left.png');
```

Tässä tween-siirtymä on toteutettu staattiseen kuvaan, mutta myös animoituun spriteen on mahdollista yhdistää siirtymiä.

```
pzl1treesleft = game.add.sprite(game.world.centerX + 300, 0,
'puzzle1-trees-left');
```

Spriten määrittely tapahtuu samaan tapaan kuin aiemmassa esimerkissä. Game.world.centerX-määrittely hyödyntää pelimaailman koordinaatiston x-akselin suuntaista keskipistettä. Game.world.centerY puolestaan käyttäisi y-akselin suuntaista keskipistettä. Näitä arvoja hyödyntämällä voidaan elementit keskittää ilman tarkkojen koordinaattien laskemista. Tässä esimerkissä elementti on sijoitettu 300 pikseliä x-akselin mukaisesta keskipisteestä oikealle. Kyseinen elementti on toinen puolisko kaksiosaisesta tausta elementistä, ja täysi keskittäminen estäisi elementtejä istumasta saumattomasti.

```
game.add.tween(pzl1treesleft)
.to({alpha:1}, 2000, Phaser.Easing.Linear.None, true)
.onComplete.add(function() {bigdrum.play();});
```

Tässä tween-siirtymä on toteutettu kuvan läpinäkyvyyttä hallinnoivaan alpha-arvoon. Arvo 1 tarkoittaa kuvan täydellistä peittävyyttä, kun taas arvon muuttaminen nollaan tekee kuvasta täysin läpinäkyvän. Alpha-arvon manipulointia voidaan käyttää elementtien häivyttämiseen kuvaan ja kuvasta. Yllä mainitussa esimerkissä arvo 2000 viittaa siirtymän kestoon millisekunteina ja Phaser.Easing.Linear.None puolestaan sen tyyppiin. Siirtymätyypillä voidaan määrittää mihin tapaan elementti käyttäytyy siirtymän edetessä. Tässä käytetty siirtymä etenee tasaista nopeutta siirtymän alusta loppuun. Yllä mainitussa esimerkissä tween-siirtymään on lisäksi kytketty funktio joka toistaa äänitiedoston siirtymän loputtua.

Alpha-arvon lisäksi tween on mahdollista toteuttaa esimerkiksi siirtymänä koordinaatistossa tai koon skaalaukseen. Lisäksi tween-siirtymiä on mahdollista ketjuttaa seuraavasti:

```
game.add.tween(nightsky)
.to({alpha: 1}, 4000, Phaser.Easing.Linear.None, true, 7000)
.to({ y: 800}, 3000, Phaser.Easing.Linear.None, true);
```

Tässä pelielementin alpha-arvo muutetaan arvoon 1 jonka jälkeen elementti siirretään koordinaatistossa y:n arvoon 800.

4.3.4 **Pulma 1**

Tässä opinnäytetyössä käsitellyn pelin ensimmäisessä pulmassa pelaajan tehtävä on järjestellä kehysten sisällä oleviin paneeleihin jaettu pelialue oikeaan järjestykseen. (Ks. kuvio 17.)

Kuvio 17. Pelialueen jaottelu pelin ensimmäisessä pulmassa.

Pulman toiminnallisuus tapahtuu pääasiassa kahden funktion avulla. Ensimmäinen niistä reagoi hiiren klikkauksiin ja muokkaa pelialuetta sen mukaan, mitä elementtiä siitä on klikattu. Klikkauksen tunnistus on kytketty paneelien kehyksiin, sillä itse paneelit vaihtavat paikkaa. Pelaajan ei ole tarve kuitenkaan tähdätä klikkausta ohueen kehykseen, vaan Phaser reagoi hiirisyötteen tunnistuksessa myös läpinäkyviin pikseleihin, ellei toisin erikseen määritellä. Klikkaukseen reagointi haetaan frameClick-funktion switch case -rakenteesta kehyksen tunnuksen mukaisesti. Vasemmalta katsottuna ensimmäisen kehyksen klikkaaminen käsitellään seuraavasti:

```
smalldrum.play();
if (panOrder[0].scale.x === -1) {
      panOrder[0].scale.x = 1;
      panOriX[0] = 1;
}
else {
      panOrder[0].scale.x = -1;
      panOriX[0] = -1;
}
checkOrder();
```

Rakenne peilaa kehyksen sisällä olevan paneelin x-akselin ympäri. Lisäksi se soittaa äänitiedoston ja kutsuu funktiota checkOrder.

Toisen kehyksen klikkaaminen käsitellään käytännössä samoin kuin ensimmäisen kehyksen tapauksessa, mutta x-akselin sijaan se peilaa kehyksen sisällä olevan paneelin y-akselin suuntaisesti.

Kolmannen kehyksen klikkaaminen käsitellään kahdesta aiemmasta tapauksesta hieman poiketen seuraavasti:

```
smalldrum.play();
panOrder[0].x = game.world.centerX;
panOrder[1].x = game.world.centerX + 260;
panOrder[2].x = game.world.centerX - 260;

helper = panOrder.slice(0);
helperX = panOriX.slice(0);
helperY = panOriY.slice(0);

panOrder[0] = helper[2];
panOrder[1] = helper[0];
panOrder[2] = helper[1];

panOriX[0] = helperX[2];
panOriX[1] = helperX[0];
panOriX[2] = helperX[1];

panOriY[0] = helperY[2];
panOriY[1] = helperY[0];
panOriY[2] = helperY[1];

checkOrder();
```

Rakenne vaihtaa paneelien järjestyksen. Ensimmäisen ja toisen kehyksen tapaan kolmannen kehyksen klikkaaminen lisäksi toistaa äänitiedoston ja kutsuu funktiota checkOrder. Paneelien asennot on tallennettu apumuuttujiin, joiden arvoja verrokkimuuttujiin vertailemalla voidaan testata niiden järjestystä.

Funktio checkOrder on pulman rakenteen toinen pääfunktio. Rakenteeltaan se on yksinkertainen if-lause, mutta pelimekaniikan kannalta funktio on oleellinen. CheckOrder-funktion tehtävä on testata paneelien järjestys ja kutsua pelin seuraavaa vaihetta kun järjestys on saatu oikein. Funktio tutkii paneeleista niiden sijainnin, sekä asennon x- ja y-akselin suhteen.

4.3.5 **Pulma 2**

Pelin toisessa pulmassa pelaajan tehtävänä on kerätä pelialueelta kolme henkihahmon maskin kappaletta. Ne on pyritty piilottamaan pelialueelle siten, ettei niiden sijainti ole ilmeinen. (Ks. kuvio 18)

Kuvio 18. Pelinäkymä pelin toisessa pulmassa.

Maskin palat reagoivat hiiren klikkaukseen siirtymällä pelialueen keskelle, sekä muuttamalla väriä ja kokoa. Tämän jälkeen palat katoavat pelialueelta. Pelialueen alalaidassa on ääriviivat maskin paloista, jotka täyttyvät vaalealla värillä merkiksi palan löydyttyä. Koodillisesti peli tutkii palojen klikkausta ja reagoi niihin etsimällä palan tunnistetta vastaavan kohdan switch case-rakenteesta. Klikkausten käsittely tapahtuu seuraavasti:

```
tap.play();
foundCounter++;
maskpieceHidden1.inputEnabled = false;
maskpieceHidden1.frame = 1;
maskpieceHidden1.scale.setTo(0.5, 0.5);
maskpieceHidden1.x = game.world.centerX;
maskpieceHidden1.y = game.world.centerY;
maskoutline1.frame = 1;
game.add.tween(maskpieceHidden1)
.to({alpha: 0}, 50, Phaser.Easing.Linear.None, true, 1000)
.onComplete.add(function() {foundCheck(maskpieceHidden1);});
            break;
```

Aiemmin mainittujen lisäksi koodi poistaa palasta klikkauksen tunnistuksen. Tällä poistetaan mahdollisuus tuplaklikkauksiin. Lisäksi kasvatetaan foundCounter-muuttujaa ja kutsutaan foundCheck-funktiota. Funktiolle lähetetään parametrinä palan tunniste:

```
function foundCheck (destroyThis) {

      destroySprite(destroyThis);

      if (foundCounter === 3) {

            kling.play();

            game.add.tween(maskoutline1)
            .to({alpha: 0}, 3000, Phaser.Easing.Linear.None,
            true, 1000);
            game.add.tween(maskoutline2)
            .to({alpha: 0}, 3000, Phaser.Easing.Linear.None,
            true, 1000);
            game.add.tween(maskoutline3)
            .to({alpha: 0}, 3000, Phaser.Easing.Linear.None,
            true, 1000);

            puzzle2End();
      }
}
```

FoundCheck-funktio tuhoaa löydetyn palan kutsumalla destroySprite-funktiota. Funktiolle lähetetään parametrinä löydetyn palan tunniste. FoundCheck-funktio tarkistaa myös, onko foundCounter-muuttujan arvo yhtä kuin 3. Jos arvo täsmää, funktio toistaa lyhyen äänitiedoston ja häivyttää pelialueen alalaidassa olleet maskinpalasten kuviot näkyvistä tween-siirtymällä alphan arvoon 0.

Lopuksi kutsutaan puzzle2End-funktiota. Se huolehtii niiden pulmassa käytettyjen resurssien tuhoamisesta, joita ei käytetä enää myöhemmissä vaiheissa. Tämän jälkeen funktio kutsuu seuraavaa pelivaihetta.

4.3.6 **Pulma 3**

Pelin kolmannessa pulmassa pelaajan tehtävä on järjestellä pelialueen metsämaisema siten, että sen keskelle muodostuu polku.

Pelialueen metsä on jaettu neljään riviin. Näistä ensimmäisessä on kuusi yksittäistä puuta, ryhmiteltynä pelialueen kumpaankin laitaan. Kolmessa jälkimmäisessä rivissä on joukko puita, jotka on jaoteltu siten, että jokaisen rivin keskellä on aukko. Polku

pelialueelle muodostetaan saattamalla nämä aukot samaan linjaan keskenään. Pulman alkuasetelmassa polku on peitossa. (Ks. kuvio 19.)

Kuvio 19. Pelin kolmannen pulman alkuasetelma.

Puut liikkuvat riveittäin etualan yksittäisiä puita klikkaamalla. Aiempien pulmien tapaan, myös kolmannen pulman logiikan pohjana toimii switch case-rakenne, josta klikkaukseen reagointi etsitään. Vasemman puoleiset puut liikuttavat kahta taaemmista riveistä kerralla. Tästä esimerkkinä vasemman laidan ensimmäisen puun klikkauksen käsittely:

```
lowklank.play();
if (row1Check === false) {
      game.add.tween(row1)
      .to({x: game.world.centerX}, 1000,
      Phaser.Easing.Linear.None, true);
      row1Check = true;
      }
else {
      game.add.tween(row1)
      .to({x: game.world.centerX + 500}, 1000,
      Phaser.Easing.Linear.None, true);
      row1Check = false;
      }

if (row2Check === false ) {
      game.add.tween(row2)
      .to({x: game.world.centerX}, 1000,
      Phaser.Easing.Linear.None, true);
      row2Check = true;
      }
else {
      game.add.tween(row2)
      .to({x: game.world.centerX - 500}, 1000,
```

```
        Phaser.Easing.Linear.None, true);
        row2Check = false;
        }
        checkPath();
```

Koodi tarkistaa, ovatko rivit auki. Rivin ollessa auki ne suljetaan ja suljetut rivit puolestaan avataan. Row1Check-muuttuja kertoo onko ensimmäinen rivi auki, row2Check puolestaan kertoo saman toisesta rivistä. Muuttujan arvon ollessa tosi, polku on sen rivin kohdalta avattu. Lisäksi kutsutaan checkPath-funktiota, joka tarkistaa onko polku avattu rowCheck-muuttujien arvoista.

Oikean laidan puiden klikkaukset käsitellään seuraavan esimerkin mukaisesti:

```
hiklank.play();
if (row1.x === game.world.centerX - 500) {
      game.add.tween(row1)
      .to({x: game.world.centerX + 500}, 1000,
      Phaser.Easing.Linear.None, true);
      row1Check = false;
      }
else {
      game.add.tween(row1)
      .to({x: game.world.centerX - 500}, 1000,
      Phaser.Easing.Linear.None, true);
      row1Check = false;
      }
```

Oikean puolen puiden klikkaus sulkee aina yhden rivin kerrallaan, ja muuttaa samalla rivin rowCheck-muuttujan arvon epätodeksi. Klikkauksen käsittely testaa rivin sijainnin x-koordinaatin, ja liikuttaa riviä sen mukaisesti joko oikealle tai vasemmalle. Liikuttamalla riviä, vaikka se olisi ollut jo valmiiksi suljettuna, viestitään pelaajalle, että klikkaukseen on reagoitu. Vasemman puoleisten puiden klikkaus ei kutsu checkPath-funktiota, sillä se muuttaa rowCheck-arvot ainoastaan epätosiksi.

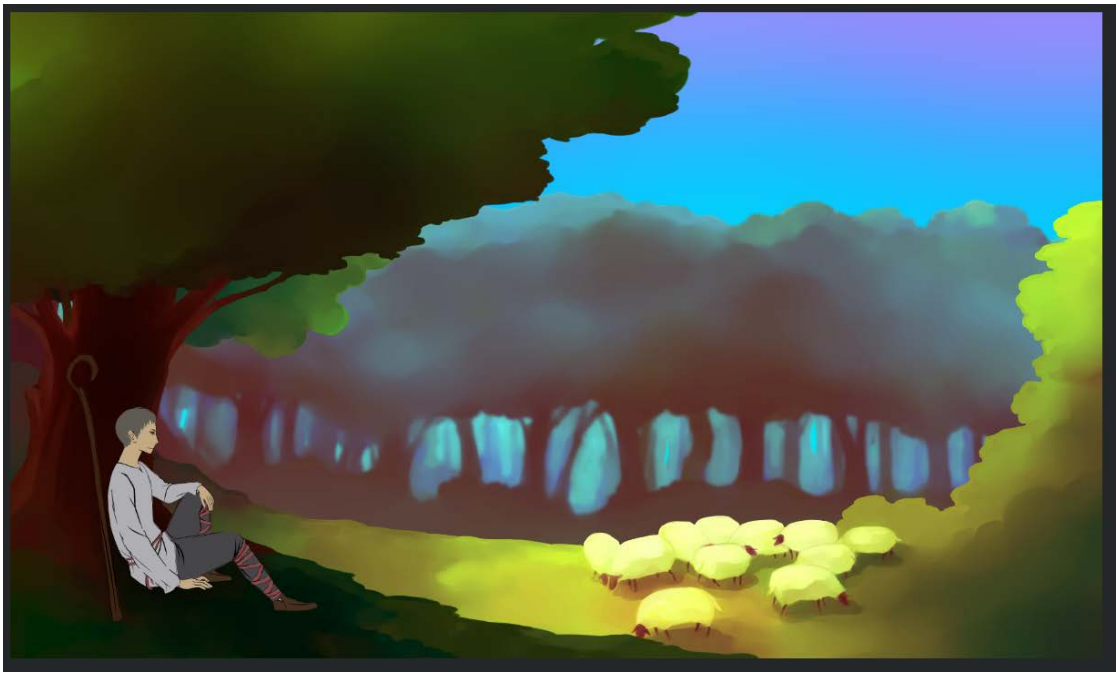
Peli siirtyy checkPath-funktiosta seuraavaan pelivaiheeseen kun rowCheck-muuttujien arvot on kaikki muutettu tosiksi.

4.4 **Grafiikka**

Pelin grafiikka on toteutettu digitaalisesti. Grafiikan luomisessa on hyödynnetty Adobe Photoshop CC-, Adobe Illustrator CC- ja Easy PaintTool SAI-ohjelmistoja.

Grafiikka on toteutettu osittain vektoroituna ja osittain maalattuna pikseligrafiikkana. Grafiikan tyylien eroilla on pyritty tavoittelemaan piirroselokuvamaista tunnelmaa. Yksinkertaisen hahmomallit yhdistettynä realistisempaan taustaan tuo grafiikkaan

kontrastia ja hahmojen pelkistetty tyyli auttaa niitä nousemaan selkeämmin pääosaan. (Ks. kuvio 20.)



Kuvio 20. Esimerkki vektoroidun ja maalauksellisen grafiikan yhdistämisestä.

Vaikka osa grafiikasta on toteutettu vektoroimalla, on se peliin muunnettu pikseligrafiikaksi, sillä Phaser-kirjasto ei tue vektorimuotoista svg-grafiikka. Kuvat on tallennettu png-tiedostomuodossa, sillä se tarjoaa mahdollisuuden hyödyntää grafiikassa läpinäkyvyyttä.

4.4.1 **UI ja typografia**

Pelin käyttöliittymän suunnittelussa on pyritty selkeyteen. Rakenteeltaan pelin käyttöliittymä on yksinkertainen ja pelkistetty. Kaikki käyttöliittymän toiminnot liittyvät pelissä etenemiseen, ja ne suoritetaan hiirellä klikkaamalla.

Pelistä tekstiä löytyy dialogista, pelin ohjeistuksesta ja tekijätietoista, sekä pelin logosta. Näistä logossa ei ole käytetty fonttia, vaan se on toteutettu luonnoksen pohjalta vektoroimalla.

Pelin ohjeistukseen ja tekijätietoihin pyrittiin löytämään selkeä ja helppolukuinen fontti, joka toimisi myös pienemmällä pistekoolla.

Hahmojen dialogin typografiaan kiinnitettiin erityistä huomiota. Fonttivalinnoilla pyrittiin selkeyttämään kulloinkin puhuvaa hahmoa dialogien aikana. Tätä havainnollistaa kuvio 21, jossa vasemmalla henki-hahmon vuorosanat ja oikealla paimen-hahmon dialogi.

Kuvio 21. Esimerkki hahmojen dialogien erottelusta fonttivalinnoilla

Fonttivalinnoissa pyrittiin selkeyteen ja helppoon luettavuuteen. Lisäksi hahmoille valittujen fonttien piti osaltaan heijastella hahmon persoonaa. Fonttivalinnat käyttötarkoituksineen on listattu taulukkoon 4.

Taulukko 4. Listaus pelin fonttivalinnoista

Fontti	**Käyttötarkoitus**
TrashHand	Paimen-hahmon vuorosanat
Wahroonga	Henki-hahmon vuorosanat
Trebuchet MS	Pulmien ohjeistus, lopputekstit

4.4.2 **Hahmot**

Pelissä on kaksi hahmoa, paimen ja henki. Hahmojen mallinnus peliin on toteutettu vektoroimalla. Hahmojen grafiikan tyyli on pelkistetty animaatioiden helpottamiseksi. Tästä syystä hahmomallit jätettiin pääosin varjostamatta, ja niiden väritys on pidetty tasaisina pintoina.

Hahmojen tyyliin haluttiin piirroselokuvista tuttua selkeää kuvakieltä, ja vaikutteita haettiin erityisesti japanilaisesta animaatiosta. Kuviossa 22 on nähtävillä esimerkit molempien hahmojen hahmomalleista.

Kuvio 22. Esimerkki pelin hahmomalleista.

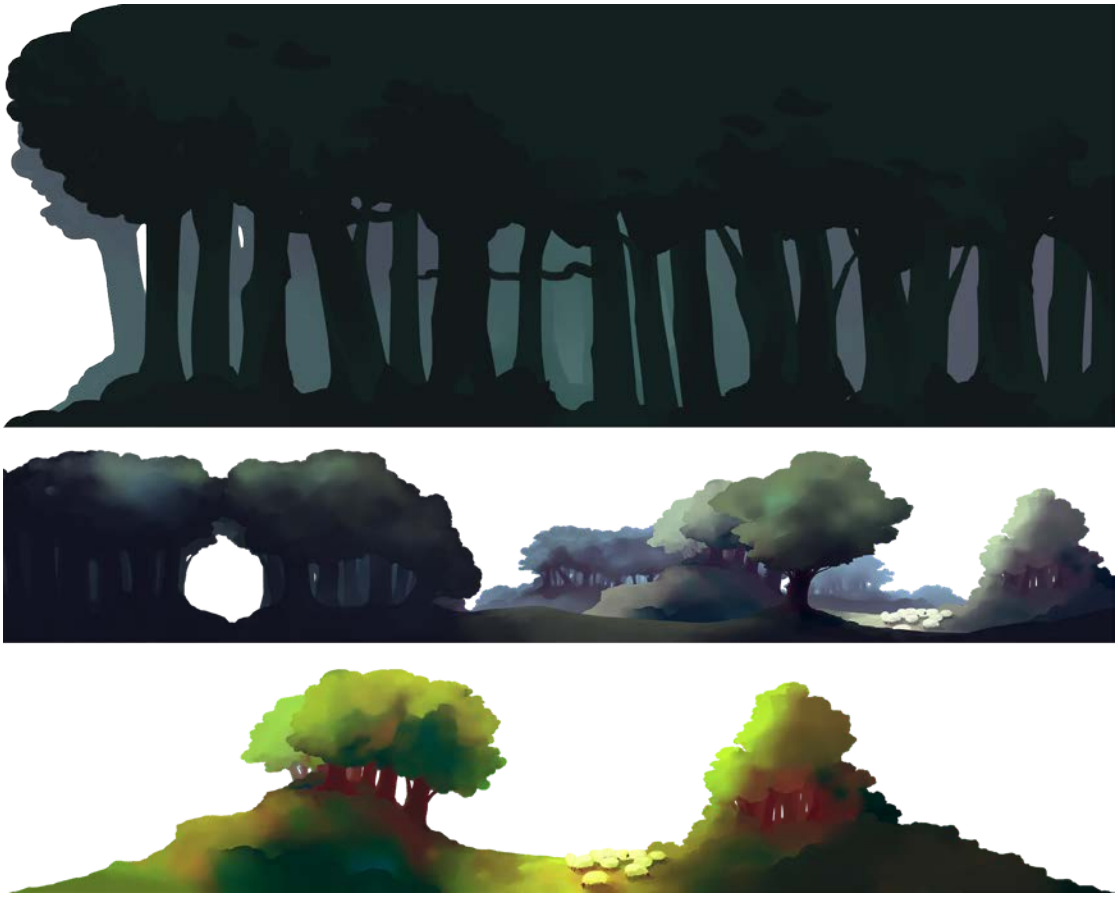
Hahmoihin liittyvä animaatio on enimmäkseen toteutettu spritesheet-tekniikalla. Hahmojen liikkeet pyrittiin pitämään mahdollisimman yksinkertaisina, sillä ruutujen lisääminen kasvattaa aina spritesheetin kokoa.

Hahmojen värityksessä pyrittiin antamaan kuvaa hahmon olemuksesta. Henki-hahmon värityksessä käytettiin etupäässä sinisen sävyjä, kun taas paimen-hahmon värityksessä pyrittiin maanläheisempään otteseen, ja käytössä on enimmäkseen harmaan ja ruskean sävyjä. Henki-hahmon värimaailmalla pyrittiin myös heijastelemaan pelin henkimaailman osioiden tunnelmaa ja väritystä. Samoin paimenen hahmossa pyrittiin heijastelemaan pelin reaalimaailman väritystä, mutta murretumpana ja hillitympänä.

4.4.3 **Taustat**

Taustat on toteutettu enimmäkseen pikseligrafiikkana maalaamalla Easy PaintTool SAI-ohjelmassa. Taustoissa on osittain käytetty myös tekniikkaa, jossa elementin päämuoto vektoroidaan pohjaksi jonka päälle maalataan.

Pelin taustoissa on enimmäkseen kuvattu metsämaisemaa, eri tunnelmissa. Niiden avulla on pyritty luomaan kuvaa pelin ympäristöstä. Taustoilla on pyritty lisäksi antamaan eri peliosioille omanlaisensa, selkeästi erottuva ilme. Pelin henkimaailmaan sijoittuvissa osioissa taustojen tyyli on pyritty pitämään yksinkertaisena, kun taas reaalimaailman osioissa taustojen tyylissä on pyritty yksityiskohtaisempaan ja hieman realistisempaan ilmaisuun. (Ks. kuvio 23.)



Kuvio 23. Esimerkki taustatyylien eroista eri peliosioissa. Yllä esimerkki pelin henkimaailmasta, alla esimerkit reaalimaailman taustoista yöllä ja päivällä.

Reaalimaailman taustoissa tavoitteena oli saada erottumaan eri vuorokauden ajat luomaan illuusiota ajan kulumisesta henkimaailman osioiden aikana. Oman hankaluutensa tähän toi väritys, sillä yöllisen maiseman värimaailma on tyypillisesti melko lähellä henkimaailman osioihin valittua tummaa ja sinisävyistä väritystä. Tätä pyrittiin selkeyttämään mallintamalla grafiikat eri tyyleillä.

4.5 Äänet

Pelin äänimaailma on koottu useasta eri äänitiedostossa, joita yhdistämällä eri peliosioiden äänimaisemaan on voitu varioida. Taulukossa 5 on listattuna äänet ja niiden kuvaukset.

Taulukko 5. Pelissä käytetyt ääniresurssit

Äänitiedosto	**Tekijä**	**Kuvaus**
COPLAND DRUM 01, TAIKO DRUM 001	sandyrb	Rummunlyöntejä
Dark Ambience	Patrick Lieberkind	Henkimaailman taustaambient
Drum taiko	Toaru Otoya	Rummunlyönti
Etude	TheWorkingBamboo	Pelin lopun pianokappale
Forest thunder	electroviolence	Ukkosen jyrähdys metsässä
Hammering woodpecker	Reinsamba	Luonnon ääniä, linnunlaulua
Hi Woodblock, Low Woodblock	Francisco Padilla	Puukalikan kalahduksia
jf Victorious Trees	cmusounddesign	Puiden narinaa ja paukahduksia tuulessa
Movie Trailer Boom	hykenfreak	Matala kumahdus
NightSounds3	tonant	Öisen metsän ääniä
slidingdoor opening	joedeshon	Avautuva liukuovi
Tiny bell strike	Herbert Boland	Tiu'un helähdys
Wind Chimes	littleboot	Tuulikellojen kilinää
Wind through trees	spoonbender	Puiden narinaa tuulessa

Äänien valinnassa pyrittiin tunnelman luomiseen. Pelin reaalimaailman kohtauksiin pyrittiin tuomaan rauhallista ja osittain melankolista tunnelmaa. Henkimaailmaan puolestaan pyrittiin luomaan painostava, hieman uhkaava tunnelma.

Kaikki pelin äänitiedostot on julkaistu sivustolta http://freesound.org/ Creative Commons 0- tai Creative Commons Attribution-lisenssillä. Näistä ensimmäinen sallii tiedoston vapaan käytön ilman rajoituksia, ja toinen sallii käytön epäkaupallisissa töissä siten, että tiedoston tuottaja mainitaan teoksen yhteydessä.

5 TULOKSET

Opinnäytetyön tuloksena valmistui suunnitelman mukainen peli. Pulmien rakenteen ja grafiikan kanssa tapahtui jonkun verran elämistä, mutta kaikki halutut ominaisuudet ja pelivaiheet saatiin toteutettua. Enimmäkseen eläminen johtui uuden oppimisesta ja siitä, ettei osa alun perin suunnitelluista metodeista toiminut käytännössä toivotulla tavalla Phaser-kirjaston puitteissa.

Grafiikan osalta suurin muutos suunnitelmasta oli siirtyminen yksittäisten kuvien käytöstä spritesheet-kuvakarttoihin. Pelin taustat oli alun perin tarkoitus vektoroida hahmojen tapaan ajan säästämiseksi, mutta lopulta päädyttiin kuitenkin maalausmaiseen tekniikkaan. Muutos ei vienyt kohtuuttomasti enempää resursseja, ja lopputuloksena taustoista saatiin visuaalisesti näyttävämpiä kuin vektoroimalla.

Huolellisesti toteutettu pelisuunnitelma osoittautui toteutuksen aikana ensiarvoisen tärkeäksi. Työ eteni hallitusti, kun seuraavan vaiheen pystyi tarkistamaan suunnitelmasta. Samoin suunnitelma helpotti kokonaisuuden hahmottamista ja auttoi pysymään aikataulussa. Aikataulun seuraamisessa avuksi oli myös se, että suunnitelma oli alun perin luotu käytössä olevat resurssit huomioon ottaen.

Valmiista pelistä saatu palaute on ollut pääasiassa positiivista. Pelin pulmien haastetasapainon säätäminen tuotti työn aikana päänvaivaa, mutta pelaajilta saadun palautteen mukaan tasapaino onnistuttiin saavuttamaan kohtalaisesti. Suurin osa pelaajista selviytyi pulmista ilman merkittävää katkoa pelikokemuksessa.

Phaser-kirjasto oli miellyttävä käyttää, ja se tuki pelin toteutusta hyvin. Kirjaston selkeästä dokumentaatiosta oli työskentelyssä korvaamaton apu. Käytön sujuvuudesta huolimatta kirjasto aiheutti pelin työstämisessä omia hankaluuksiaan. Näistä merkittävimpänä oli kirjaston tapa bufferoida resursseja. Tämä aiheuttaa sen, että jos pelin kuluessa vaihtaa pois pelin välilehdeltä tai ikkunasta, peli saattaa jumittua, eivätkä kaikki resurssit jatkossa lataudu oikein.

Lopullinen peli löytyy osoitteesta http://nianas.net/henki/. Peli on testattu Firefox 33.1- ja Google Chrome 39-selaimilla.

6 JOHTOPÄÄTÖKSET

Opinnäytetyön tavoitteena oli pelinteon kautta tarkastella JavaScriptin ja HTML5:n tuomia mahdollisuuksia pelinkehitykseen sekä tutkia sarjakuvamaisen ilmaisun yhdistelyä pelilliseen kerrontaan. Erityisesti pelinkehitykseen tarkoitettuja JavaScript-kirjastoja löytyy nykyään runsaasti ja niistä löytyy vaihtoehtoja jo hyvin monentyyppisten pelien toteuttamiseen. Tässä opinnäytetyössä painotettiin kirjastoissa 2D-grafiikan tukea, mutta myös esimerkiksi 3D-grafiikkaa tukevia kirjastoja löytyy jo useita. Pelinkehityksessä on kuitenkin muistettava selainten suorituskyvyn tuomat rajoitukset. Tässä opinnäytetyössä toteutettu peli optimoitiin pitäen silmällä uusimpia selaimia työpöytäympäristössä. Jos tarkoituksena on rakentaa peli mahdollisimman laajalle pelaajakunnalle, on myös vanhemmat selaimet ja mobiililaitteet otettava huomioon. Tämä tuo kehitykseen omat haasteensa, erityisesti mobiililaitteiden laajan kirjon huomioon ottaen, mutta erityisesti kaupallisissa projekteissa tähän on ensiarvoisen tärkeää kiinnittää huomiota.

Selainpohjaisen pelinkehityksen mahdollisuudet eivät rajoitu pelkästään peleihin, vaan pelillisiä elementtejä voidaan niiden kautta hyödyntää myös perinteisissä verkkopalveluissa. Pelillisen interaktiivisuuden kautta voidaan tuoda verkkosivuille uutta sisältöä, ja auttaa lisäämään käyttäjän kiinnostusta. Pelillistäminen eli gamification onkin tuonut viime vuosina verkkosivujen suunnitteluun uusia ulottuvuuksia ja modernit JavaScript-pelimoottorit tukevat sisältöä monipuolisesti. Pelimoottorin valinnassa on kuitenkin muistettava pitää mielessä kehitettävän pelin tai pelillisen elementin asettamat vaatimukset.

Phaser-kirjasto toimi pienistä puutteistaan huolimatta erinomaisena tukena pelinkehitykselle, ja jos sen kehitys jatkuu yhtä aktiivisena kuin tähän asti, on se jatkossakin varteenotettava vaihtoehto tässä opinnäytetyössä toteutettua peliä vastaavissa projekteissa. Phaser on ominaisuuksiltaan laaja pelimoottori, ja se tarjoaa lukuisasti erilaisia ominaisuuksia, joista tässä opinnäytetyössä raapaistiin vain pintaa.

Tässä opinnäytetyössä toteutettua peliä ei ole tarkoitus tämän jälkeen jatkokehittää, vaan se on valmis kokonaisuus sellaisenaan. Aivan suoraan tässä opinnäytetyössä toteutettua peliä ei voi hyödyntää kaupallisissa tarkoituksissa, mutta työssä opittuja tekniikoita voidaan hyödyntää tulevaisuudessa samankaltaisten projektien parissa.

Opinnäytetyön toteutus opetti paljon, ja saavutetulta tietopohjalta on osaamista helppo lähteä laajentamaan. Opittavaa riittää varmasti myös tulevaisuudessa, mutta nyt luotu tietoperusta varmistaa, ettei jatkossa tarvitse rakentaa tyhjän päälle. Myös tältä osin tämän opinnäytetyön voidaan katsoa saavuttaneen tavoitteensa.

Tulevaisuudessa on tarkoitus syventää opinnäytetyön toteutuksen aikana opittujen tekniikoiden tuntemusta ja laajentaa osaamista erityisesti interaktiivisten elementtien hyödyntämisessä myös peliympäristöjen ulkopuolella.

LÄHTEET

Bourdon, R. 2014. WampServer, the web development platform on Windows – Apache, MySQL, PHP. Viitattu 25.10.2014 http://www.wampserver.com/en/

Boyer, B. 2014. Less Talk More Rock - Boing Boing. Viitattu 20.11.2014. http://boingboing.net/features/morerock.html

Bright, P. 2012. Microsoft TypeScript: the JavaScript we need, or a solution looking for a problem? Viitattu 25.10.2014. http://arstechnica.com/information-technology/2012/10/microsoft-typescript-the-javascript-we-need-or-a-solution-looking-for-a-problem/

Construct 2 details, reviews, and important links. 2014. Viitattu 15.10.2014. http://html5gameengine.com/details/4/construct-2

Create Games with Construct 2 – Scirra.com. 2014. Viitattu 14.10.2014. https://www.scirra.com/construct2

CreateJS | A suite of Javascript libraries and tools for building rich, interactive experiences with HTML5. 2014. Viitattu 30.10.2014. http://www.createjs.com/#!/CreateJS

Grip, T. 2013. In The Games Of Madness. Viitattu 20.11.2014. http://frictionalgames.blogspot.fi/2013/08/5-core-elements-of-interactive.html

Kimmo, K. 2006. Varo, metsänhaltija voi eksyttää sinut metsään. Suomen Metsäyhdistys ry. Viitattu 25.10.2014. http://www.forest.fi/smyforest/forest.nsf/tiedotteetlookup/58068487CC999CFFC225724B003CC1B0

Munogee, H. 2012. Quintus - An Easy-To-Learn, Fun-To-Use HTML5 Game Engine For Mobile, Desktop & Beyond. Viitattu 24.10.2014. http://css.dzone.com/articles/quintus-easy-learn-fun-use

Phaser details, reviews, and important links. 2014. Viitattu 23.10.2014. http://html5gameengine.com/details/25/phaser 15.10.2014

Quintus details, reviews, and important links. 2014. Viitattu 15.10.2014. http://html5gameengine.com/details/24/quintus

Sidhion, D. 2012. Review: Construct 2, a Drag and Drop HTML5 Game Maker. Viitattu 27.10.2014. http://code.tutsplus.com/articles/review-construct-2-a-drag-and-drop-html5-game-maker--active-10825