KARELIA-AMMATTIKORKEAKOULU
Media-alan koulutus

Olli Saarinen

MOBIILIPELI ILMAN KOODAUSTAITOJA

Opinnäytetyö
Kesäkuu 2019

Karelia AMMATTIKORKEAKOULU	**OPINNÄYTETYÖ** **Kesäkuu 2019** **Media-alan koulutus** Tikkarinne 9 80200 JOENSUU +358 13 260 600 (vaihde)

Tekijä
Olli Saarinen

Nimeke
Mobiilipeli ilman koodaustaitoja

Toimeksiantaja

Tiivistelmä

Tämä opinnäytetyö käsittelee sitä, miten mobiilipelin voi nykyään suunnitella täysin ilman koodaustaitoja. Opinnäytetyön alussa kerrotaan videopelien historiasta ja siitä, millainen on pelialan nykyinen tilanne Suomessa. Opinnäytetyössä käsitellään visuaalista ohjelmointia, miten se eroaa perinteisestä ohjelmoinnista ja millaisia ohjelmistoja 2D-pelien suunnitteluun visuaalisen ohjelmoinnin avulla on olemassa.

Opinnäytetyön toiminnallisessa osuudessa suunnitellaan mobiilipelin ja kuvataan koko suunnitteluprosessin läpi vaihe vaiheelta. Projekti aloitettiin tekemällä tarkat suunnitelmat projektin etenemisestä. Hahmoteltiin pelin sisältö tarkasti etukäteen, mikä nopeutti esimerkiksi varsinaista ohjelmointityötä huomattavasti. Pelin ohjelmointiin valittiin GameSalad-niminen ohjelmisto. Työssä käydään sen teknisiä ominaisuuksia, toimintaperiaatetta, vahvuuksia ja heikkouksia. Pelin graafinen suunnittelu on tehty Adobe Illustratorilla. Opinnäytetyöstä selviää myös esimerkiksi kuvatiedostoilta vaaditut tekniset ominaisuudet.

Opinnäytetyön liitteissä on yksinkertainen ohjelmointiesimerkki kuvineen. Liitteistä löytyy myös esimerkiksi opinnäytetyön toiminnallisessa osuudessa suunnitellun pelin ohjelmointia tukemaan tehty pelin pohjapiirustus, ohjelmointi vuokaavioineen, sekä kuvakaappauksia.

Kieli suomi	Sivuja 48 Liitteet 4 Liitesivumäärä 14

Asiasanat
mobiilipeli, visuaalinen ohjelmointi, peligrafiikka, pelisuunnittelu

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
June 2019
Degree Programme in Media

Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600 (switchboard)

Author
Olli Saarinen

Title
How to Design Mobile Games Without Coding Skills

Commissioned by

Abstract

The purpose of this thesis was to find out how mobile games can be designed without coding and what kind of skills and computer programs are needed to do so. At the beginning of this thesis the history of video games and how things are going in the gaming industry in Finland right now are discussed. What actually visual programming actually means and how does it differ from traditional coding? What kind of alternative softwares are available for 2D game design?

The goal of the functional part of this thesis was to design a small mobile game and to document the whole process step by step from the beginning to the finished game. The project started by setting the limits, so the project wouldn't grow too big. A program called GameSalad was used in the programming and the graphic design was done with Adobe Illustrator.

The appendices contain mobile game's plot drawing, visualized GameSalad programming with flow diagrams and some screenshots.

Language
Finnish

Pages 48
Appendices 4
Pages of Appendices 14

Keywords
mobile game, visual programming, game graphic, game design

Sisältö

1 Johdanto 5
2 Pelit ja niiden suunnittelu 6
2.1 Peli 6
2.2 Videopelit 6
2.3 Mobiilipelit 10
2.4 Pelisuunnittelun roolijako 11
3 Visuaalinen ohjelmointi ja pelisuunnittelu 12
3.1 Mitä visuaalinen ohjelmointi tarkoittaa? 12
3.2 Visuaalisen ohjelmoinnin historiaa 12
3.3 2D-mobiilipelin suunnittelu visuaalisen ohjelmoinnin avulla 13
4 GameSaladin käyttäminen 15
4.1 Toimintaperiaate, keskeiset ominaisuudet ja toiminnot 15
4.2 GameSaladin käyttäminen 20
4.3 GameSaladin vahvuudet ja heikkoudet 26
5 Esimerkkipelin suunnittelu 27
5.1 Peliprojektin hallinta 27
5.2 Pelin suunnittelussa käytetyt ohjelmistot 28
5.3 Pelin ideointi 28
5.4 Grafiikka 31
5.4.1 Kuvitusprosessi 31
5.4.2 Kuvatiedostoilta vaaditut tekniset ominaisuudet 34
5.5 Ohjelmointi 36
5.5.1 Esivalmistelut ennen ohjelmointia 36
5.5.2 GameSaladin asetukset 36
5.5.3 Päävalikon ja Asetukset -näkymän nappien suunnittelu 37
5.5.4 Kenttien hahmotyyppien suunnittelu 41
5.6 Pelin testaaminen ja valmistelu sovelluskauppoja varten 43
6 Pohdinta 45
Lähteet 47

Liitteet

Liite 1 Ohjelmointiesimerkki ”Hello world!”
Liite 2 Esimerkkipelin pohjapiirustus
Liite 3 Kuvakaappauksia esimerkkipelistä
Liite 4 Esimerkkipelin ohjelmointi ja vuokaaviot

1 Johdanto

Tämä opinnäytetyöni käsittelee mobiilipelien suunnittelua ja sitä miten niiden tekeminen on nykyään mahdollista melko helposti myös ilman ohjelmointikoodin näkemistä käyttämällä visuaalista ohjelmointia. Opinnäytetyöni tulee sisältämään GameSalad-ohjelmiston käyttämistä ja myös pelikäyttöön soveltuvan 2D-grafiikan tekemistä Adobe Illustrator -ohjelmistolla. Tämän lisäksi tulen lyhyesti käsittelemään mobiilipelin testaustapoja.

Janne Monosen (2016) opinnäytetyössä ”Viiden 3D-pelimoottorin vertailu uuden kehittäjän näkökulmasta” vertailtiin nimensä mukaisesti 3D-pelimoottoreita ja pyrittiin antamaan kokemattomille käyttäjille apua sopivan pelimoottorin valinnan suhteen. Niko Nevalainen käsitteli vuonna 2014 julkaistussa opinnäytetyössään käytettävyyttä ja sen soveltamista videopeleihin. Eetu Mahonen (2018) käsitteli opinnäytetyössään ”Prototyypin kehitys Unreal Enginen Blueprinteillä” visuaalista ohjelmointia 3D-hahmon prototyypin kehittämisessä. Mainita täytyy myös Arhi Makkosen (2015) opinnäytetyö, jossa hän pureutui esimerkiksi pelisuunnittelun rooleihin. Tämän opinnäytetyön toiminnallisessa osassa tavoitteena saada aikaan tietynlainen mobiilipelin suunnitteluopas ohjelmointikieliä taitamattomille henkilöille. Opinnäytetyöni pyrkii vastaamaan siihen, että miten mobiilipeliprojekti on mahdollista toteuttaa ilman koodin näkemistä, millaisia tietoja ja taitoja tekijällä tulisi olla ja millaisia ohjelmistoja pelin tekemiseen tarvitaan.

Tavoitteenani oli kehittää omia taitojani pelisuunnittelussa. Pyrin parantamaan taitojani ohjelmoinnin, graafisen suunnittelun ja projektinhallinnan parissa. Tarkoitukseni oli tehdä mahdollisimman tarkat etukäteissuunnitelmat siitä, miten peliprojekti etenisi, millaisia elementtejä peliin kuuluisi ja mitä varsinainen ohjelmointi pitäisi sisällään. Pyrin siis pitämään projektin hallussa alusta alkaen ja etenemään suunnitelmallisesti.

2 Pelit ja niiden suunnittelu

2.1 Peli

Peli ja leikki muistuttavat toisiaan paljon, Suomen kielessä niiden merkitykset ovat hyvin lähellä toisiaan (Räty 1999, 9). Hollantilainen historioitsija Johan Huizinga määritteli pelin siten, että siihen kuuluu ehdottomasti vapaus, tiukat säännöt ja järjestys. Leikki eroaa hänen mukaansa normaalista arkielämästä suuresti, se on kuin oma kuplansa, josta voi hakea iloa ja jännitystä. Leikki voi alkaa ja loppua silmänräpäyksessä, se on ihmisille ja eläimille normaalia ja mielekästä toimintaa, jota halutaan vapaaehtoisesti tehdä. Leikissä eivät päde normaalin elämän tavat ja lait. (Huizinga 1967, 9-21.)

Pelien tekemisestä puhutaan varsinaisesti siinä vaiheessa, kun peli tuotteistetaan. Tuotteistuksella tarkoitetaan tässä sitä, että peli nimetään, sille luodaan oma identiteetti, sekä säännöt tai muut ohjeet. (Vuorela 2007, 14.) Pelityyppien luokittelussa esiintyy jonkin verran vaihtelua, mutta usein ne luokitellaan seuraavasti: Roolipelit, keräilykorttipelit, lautapelit, korttipelit, miniatyyripelit ja tietenkin videopelit, joihin luetaan kuuluvaksi esimerkiksi tietokoneella pelattavat pelit, konsolipelit, käsikonsolipelit ja mobiilipelit. (Vuorela 2007, 19.)

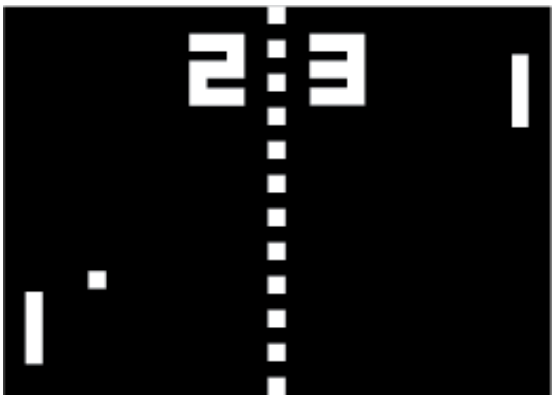
2.2 Videopelit

Videopeli on siis yläkäsite, jonka alle kuuluu joukko erilaisia pelityyppejä. Niko Nirvi määrittelee videopelin seuraavasti: ”videopeli on yleistermi elektronisille peleille, joihin liittyy vuorovaikutus käyttöliittymällä, joka generoi visuaalista palautetta videolaitteelle” (Nirvi 2009). Videopelien historian alku ulottuu viime vuosisadan puoliväliin saakka. Pelialan tutkijoilla ei ole kuitenkaan olemassa yhteisymmärrystä siitä, että mikä oli oikeastaan ensimmäinen videopeli. Pelejä tehtiin useassa eri paikassa, eikä kaikkea ole varmasti dokumentoitu. (Herman 2008a, 54.) Esimerkiksi jo vuonna 1952 englantilainen Alexander Douglas

kehitti yliopistossa opiskellessaan OXO-nimisen ristinollatyyppisen pelin EDSAC-tietokoneelle. Toinen varhainen peli oli yhdysvaltalaisen William Higinbothamin suunnittelema peli nimeltään Tennis For Two (1958). Tuota peliä pidetään ensimmäisenä fysiikkaa simuloineena pelinä. Pelin teki erikoiseksi myös se, että sitä pelattiin oskillaattorin näytön kautta. (Barton & Loguidice 2009, 2.)

Ensimmäinen reaaliaikainen kaksinpeli oli yhdysvaltalaisen Steve Russellin ohjelmoima Spacewar (1962). Siinä pelaajat taistelivat toisiaan vastaan erilaisilla avaruusaluksilla, vältellen samalla pelinkentän keskellä sijainnutta Aurinkoa, jonka painovoima veti puoleensa niin avaruusaluksia kuin ammuksiakin. Vaikka kyseessä ei ollutkaan videopeleistä ensimmäinen, pidetään sitä laajalti yhtenä alkuaikojen merkittävimmistä. (Barton & Loguidice 2008, 1–2.)

Ensimmäisen videopelin keksijää on hankala saada selville, sillä kehitystyötä tehtiin useissa eri paikoissa suunnilleen samoihin aikoihin. Toisin on konsolipelien ja -laitteiden kohdalla. Vuonna 1966 saksalaissyntyinen yhdysvaltalainen televisioinsinööri Ralph Baer halusi kehittää televisiolle muutakin käyttöä ohjelmien katselun lisäksi. Hän sai idean laitteesta, joka kytkettäisiin televisioon kiinni ja se pystyisi kommunikoimaan käyttäjän kanssa. Laite lisensoitiin Magnavox-nimiselle elektroniikka-alan yritykselle, joka antoi sille nimeksi Odyssey. Odyssey sisälsi 12 peliä mukaan lukien alkeellisen pöytätennispelin. Laite oli hankittavissa ainoastaan Magnavoxin omista myymälöistä, eivätkä monet tuon aikakauden ihmisistä olleet edes kuulleet sen olemassaolosta. Kaikki kuitenkin muuttui vuonna 1972, jolloin Atari julkaisi huippusuosituksi nousseen kolikkopelin nimeltään Pong (1972). Peli muistutti Odysseyn mukana tullutta pöytätennispeliä ja se innosti ihmisiä hankkimaan Odysseyn koteihinsa. Sitä myytiin tämän jälkeen kolmessa vuodessa yli 300000 kappaletta. Atarin Pong oli samankaltaisesta ulkonäöstään huolimatta kuitenkin paljon kehittyneempi peli sisältäen muun muassa pistelaskurin (kuva 1). (Herman 2008a, 53–54.)



Kuva 1. Kuvituskuva Atarin Pong –pelistä.

Atarilla huomattin Odysseyn menestys ja siellä aloitettiin suunnittelemaan kotikäyttöön tarkoitettua Pong-pelilaitetta vuonna 1974. Sen suunnittelussa pyrittiin välttämään Odysseyn suunnittelussa havaitut virheet ja puutteet. Siinä oli esimerkiksi suurempi resoluutio, parempi käytettävyys ja myös värit, mikäli se liitettiin väritelevisioon. Laite saatiin nopeasti valmiiksi ja se oli jo seuraavana vuonna valmis kauppojen hyllyille. Jakelijaksi löytyi lopulta Sears-tavarataloketju, jolla oli yli 900 myymälää ympäri Yhdysvaltoja. Yhteistyö Searsin kanssa kannatti, sillä laitetta myytiin vuonna 1975 yli 40 miljoonan dollarin arvosta. (Herman 2008a, 54–55.)

Vaikka yhdysvaltalaiset olivatkin Atarin johdolla hallinneet maailmanlaajuisesti pelimarkkinoita, oli myös esimerkiksi Japanissa tapahtunut kehitystä. Vuonna 1983 Sega julkaisi kaksi pelikonsolia, mallinimiltään SG-1000 ja Mark III. Samana vuonna Nintendo julkaisi Famicom-pelikonsolin. Kummankin valmistajan konsolit oli alkuperäisten suunnitelmien mukaan tarkoitettu vain Japanin markkinoille, mutta Famicomin hurja menestys Japanissa sai Nintendon tavoittelemaan jalansijaa myös Pohjois-Amerikasta. Nintendo onnistuikin lopulta Amerikan valloituksessaan NES-pelikonsolillaan (1985). Osa kunniasta kuuluu sen Super Mario Bros.-pelille (1985), joka on jatko-osineen noussut yhdeksi kautta aikain menestyneimmistä videopelisarjoista. Nintendon menestyksestä innostuneena myös Sega julkaisi Mark III-konsolistaan uuden, erityisesti Yhdysvaltojen markkinoille tarkoitetun version nimeltään Sega Master System (1985) eli SMS. Vaikka monet olivatkin sitä mieltä, että SMS oli teknisesti huomattavasti parempi kuin Nintendon NES, ei se noussut kuluttajien suosioon. Syynä tähän pidettiin esimerkiksi sitä, että Segalla ei ollut tarjota

laitteelleen Super Mario Brosin kaltaista hittipeliä. Erilaisia pelikonsoleita on julkaistu vuosien mittaan erittäin paljon ja ne ovat vielä nykyäänkin suosittuja pelaajien keskuudessa. (Herman 2008b, 116–118.)

Niin sanottujen käsikonsoleiden historia sai alkunsa 1970-luvun puolivälissä, kun maailman suurin lelujen valmistaja Mattel halusi päästä mukaan videopelimarkkinoille. Se kehitti Auto Race (1976) -nimisen pelin, joka oli samalla maailman ensimmäinen käsikonsolipeli. Laitteessa ei ollut lainkaan varsinaista näyttöä, vaan siinä käytettiin hyväksi LED-lamppuja (Light-emitting diode), jotka esittivät kuvaannollisesti autoja. Pelaajan tehtävänä oli väistellä omalla autollaan pelin yläreunasta vastaan tulevia autoja. Samaa LED-tekniikkaa käyttäviä pelejä tuli markkinoille tämän jälkeen muitakin, esimerkiksi Mattel julkaisi seuraavana vuonna jalkapallopelin nimeltään Football (1977). Käsikonsoleiden etuna oli jo tuolloin se, että niitä oli helppo kantaa mukana ja ne sisälsivät kaiken pelaamiseen tarvittavan. (Herman 2008c, 143.)

Ensimmäiset kannettavat elektroniikkapelit oli rakennettu näyttöä myöten siten, että niillä oli yleensä mahdollista pelata vain yhtä sisäänrakennettua peliä. Esimerkki tällaisesta on Nintendon Game & Watch –elektroniikkapelisarja (1980), johon kuuluu kaikkiaan 59 eri peliä (Gameberry 2019). Varsinaiset käsipelikonsolit sisältävät paremman näytön ja niihin voi vaihtaa pelejä. Tällaisia ovat esimerkiksi Nintendon Game Boy (1989), Game Boy Color (1998) ja Game Boy Advance (2001). Nintendolla on ollut luonnollisesti myös haastajia, kuten Sony omalla PlayStation Portable (2004) -laitteellaan. Myös esimerkiksi Sega ja Atari ovat julkaisseet omia käsikonsoleitaan, mutta niistä ei muodostunut Nintendon kaltaista menestystarinaa. (Herman 2008c, 144–146.) Käsikonsolilaitteista puhuttaessa voidaan mainita myös 1980-luvun alkupuolella julkaistut rannekellot, joiden nestekidenäyttö mahdollisti pelien pelaamisen. Yksi ensimmäisistä oli Casio Game-10 (1980), jolla pystyi pelaamaan Space Invaders -pelistä tehtyä miniatyyriversiota. Myös Nintendo oli aktiivinen rannekellojen ja pelien yhteensovittamisessa tuoden markkinoille esimerkiksi Pac-Man ja Super Mario Bros. -peleillä varustettuja kelloja. Nintendo ei tosin itse valmistanut näitä pelikelloja, vaan siitä vastasi Nelsonic-niminen yritys. (Crecente 2015.)

2.3 Mobiilipelit

Mobiilipelillä tarkoitetaan peliä, jota voidaan pelata mobiililaitteella. Mobiililaitteeksi kutsutaan matkapuhelimia ja niin sanottuja tablet-laitteita. Suomalaisen Nokian ollessa matkapuhelinvalmistajien johtava toimija 1990-luvulla, oli heillä jo ajatuksia siitä, että puhelinta voisi käyttää muuhunkin kuin puhumiseen ja tekstiviestien lähettämiseen. Nokian legendaarinen matopeli syntyi, kun Nokian markkinointiosasto pyysi suunnittelijoitaan tekemään pelejä, joita voisi pelata kännykällä. (Riikonen 2016.) Matopeli ei ollut kuitenkaan varsinaisesti Nokian insinöörien keksintö, vaan kyseinen pelityyppi oli saanut alkunsa jo vuonna 1976, jolloin Gremlin-niminen pelistudio julkaisin pelin nimeltä Blockade (1976). Saman pelistudion seuraavana vuonna julkaisema Hustle (1977) oli vielä lähempänä Nokian matopeliä. (Sega Retro 2019.) Nokia jatkoi pelien tuomista matkapuhelimiin ja julkaisi jopa käsikonsolimaisen N-Gagen vuonna 2003. Se oli tavallaan pelikonsolin ja puhelimen sekoitus. Vaikka N-Gage ei lopulta saavuttanut suurta suosiota, sitä myytiin kuitenkin noin kolme miljoonaa kappaletta. (Keane 2015.)

Nykyään lähes kaikki julkaistavat uudet puhelimet ovat ns. älypuhelimia, joihin voidaan asentaa erilaisia sovelluksia, joita siis myös mobiilipelit ovat. Mobiilipelimarkkinat mullistuivat varsinaisesti sen jälkeen, kun Apple esitteli iPhone-puhelimensa vuonna 2007. Aikaisemmin pelejä tehtiin suuremmassa mittakaavassa lähinnä konsoleille, kotitietokoneille ja kannettaville pelikonsoleille, kuten Sony Playstation Portablelle. (Bajarin 2017.)

Pelialan hurja kasvu on näkynyt voimakkaasti myös Suomessa, sillä vuonna 2009 Suomen pelialan kokonaisliikevaihto oli 87 miljoonaa euroa ja vain neljä vuotta myöhemmin se oli ylittänyt jo 900 miljoonan euron rajan (Karjalainen, Lehtonen & Niipola 2014, 15-16). Vuosi 2017 oli kolmas peräkkäinen vuosi, jolloin alan kokonaisliikevaihto ylitti Suomessa kahden miljardin euron rajan. Vuonna 2017 ala työllisti Suomessa lähes 3000 ihmistä ja uusia kotimaisia pelejä julkaistiin kaikkiaan noin 150 kappaletta, eli uusi peli lähes joka toinen päivä. (Neogames 2018.) Kautta aikain suomalaisittain suurimpia menestystarinoita ovat olleet vuonna 2009 ensimmäisen osansa saanut Rovion

Angry Birds -pelisarja ja Supercellin pelit Clash of Clansin (2012) johdolla. Kyseiset yritykset ovat tuoneet pelialalle paljon näkyvyyttä ja uusia pelialan yrityksiä on syntynyt Suomeen niiden vanavedessä hurjaa vauhtia. (Karjalainen ym. 2014, 15-16.) Pelialan viime vuosien nopeasta kasvusta kertoo myös se, että vuonna 2015 alalla oli aktiivisia yrityksiä 260 kappaletta, joista peräti 69% oli ollut toiminnassa alle viisi vuotta (Lappalainen 2015, 8).

Kolikolla on kuitenkin kääntöpuolensa. Vaikka peliala voi hyvin ja toinen toistaan upeampia menestystarinoita on syntynyt Suomeen useita, ei takaiskuiltakaan ole vältytty. Tästä esimerkkinä käy vaikkapa se, että Rovio palkkasi Angry Birdsin nousukiidon aikana vuonna 2013 yli 300 työntekijää ja vain kaksi vuotta myöhemmin yhtiö ilmoitti irtisanovansa Suomesta lähes 200 työntekijää. (Yle 2015.)

2.4 Pelisuunnittelun roolijako

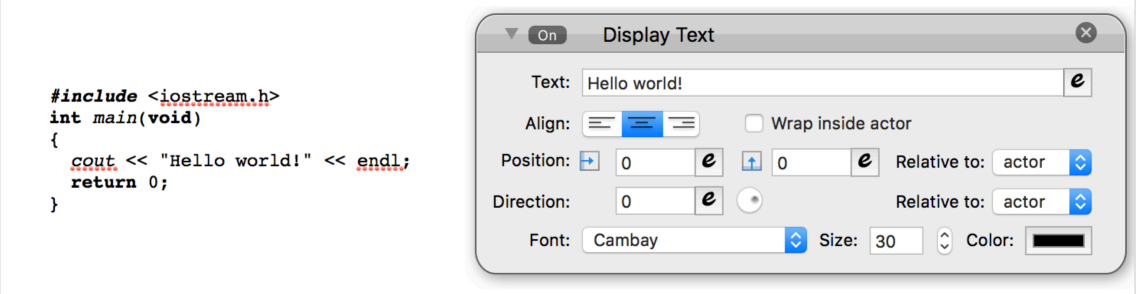
Pelien suunnitteluun osallistuvien henkilöiden roolijako muodostuu suunniteltavan pelin vaatimusten mukaan ja samaa työtehtävää voi tietenkin hoitaa useampi ihminen. Toisaalta tilanne voi olla myös se, että sama henkilö hoitaa useampaa eri työtehtävää. Kokonaisen pelin voi suunnitella myös yksin, mikäli suunnittelijan taidot ja käytettävissä olevat resurssit riittävät. Erilaisia pelinkehityksen rooleja tai tehtävänimikkeitä ovat perinteisesti olleet esimerkiksi konseptisuunnittelija, pelisuunnittelija, käsikirjoittaja, tuottaja, graafikko, ohjelmoija, testaaja ja markkinoija (Vuorela 2007, 15).

Erilaisia titteleitä voidaan tietenkin kehitellä aina tarpeen mukaan lisää. Hyvänä esimerkkinä tästä on Rovion entisen markkinointijohtajan Peter Vesterbackan humoristinen Mighty Eagle-titteli, joka nimi juonsi juurensa maailmanmaineeseen nousseesta Angry Birds -pelisarjasta (Graydon 2013, 87).

3 Visuaalinen ohjelmointi ja pelisuunnittelu

3.1 Mitä visuaalinen ohjelmointi tarkoittaa?

Visuaalisessa ohjelmoinnissa ohjelmointi tapahtuu graafisen käyttöliittymän kautta, kokoamalla ohjelmiston kirjastossa olevista elementeistä toimiva ohjelma. Elementit sisältävät erilaisia ominaisuuksia ja niiden avulla voidaan esimerkiksi lukea ja kirjoittaa dataa. Visuaalista ohjelmointia verrataan usein Lego-palikoiden kokoamiseen. Ero perinteiseen ohjelmointikoodin kirjoittamiseen on siis selvä, vaikka molempia kutsutaankin ohjelmoinniksi (kuva 2). Visuaaliseen ohjelmointiin tarkoitetut ohjelmistot ovat yleensä helposti omaksuttavia ja niissä pystyy nopeasti näkemään ohjelmoinnin tulokset. (Dehouck 2015.)



Kuva 2. Hello world! -esimerkki C++ -ohjelmointikielellä ja GameSaladilla tehtynä. Käytetty GameSalad Inc:n luvalla 27.5.2019.

Visuaalisella ohjelmoinnilla on myös heikot kohtansa. Ohjelmistot ovat usein suljettuja järjestelmiä, eivätkä niiden ominaisuudet välttämättä riitä kaikkien tarpeisiin. Verrattaessa perinteiseen ohjelmointiin niiden tuottama koodi on yleensä kankeampaa ja hankalasti optimoitavissa. (Simões 2015.)

3.2 Visuaalisen ohjelmoinnin historiaa

1980-luvun alussa kotitietokoneet alkoivat yleistymään ja monen suomalaiskodinkin pöydältä löytyi esimerkiksi Commodore 64 (1981). Muita

kotitietokoneita olivat esimerkiksi ZX Spectrum (1982) ja BBC Micro (1981), joka oli tosin tunnettu lähinnä Englannissa. Tuolloin pelejä voitiin ohjelmoida käyttäen BASIC-ohjelmointikieltä. Se ei kuitenkaan tukenut monien pelien vaatimaa grafiikkaa, vaan sopi paremmin esimerkiksi pelkkää tekstiä sisältäviin peleihin, kuten vaikkapa tekstiseikkailuihin. Tilanteen helpottamiseksi kehitettiin ohjelmistoja, joiden avulla muutkin kuin kehittyneimmät ohjelmoijat pystyisivät tekemään kunnollista grafiikkaa sisältäviä pelejä. Tällaisia ohjelmistoja olivat esimerkiksi The Quill (1983), Games Designer (1983) ja Graphic Adventure Creator (1985). Vaikka niiden tuottama ohjelmointikoodi ei ollutkaan saman tasoista kuin kehittyneiden ohjelmoijien tuotokset, pystyi niillä kuitenkin helposti toteuttamaan ja testaamaan peli-ideoita. Ne toimivat tavallaan esikuvina sellaisille ohjelmistoille kuin Applen HyperCard (1987), Macromedia Director (1993) ja Adobe Flash (1996), joiden avulla pystyttiin suunnittelemaan myös pelejä, vaikka se ei välttämättä niiden alkuperäinen käyttötarkoitus ollutkaan. (Zagalo & Branco 2015, 9–10.)

Erilaisia ohjelmointia helpottavia ohjelmistoja on kehitetty jatkuvasti ja esimerkiksi mobiilipelien suunnittelu ilman ohjelmointikoodin näkemistä on nykyään mahdollista visuaalisen ohjelmoinnin avulla.

3.3 2D-mobiilipelin suunnittelu visuaalisen ohjelmoinnin avulla

Visuaalisen ohjelmoinnin mahdollistavia pelinsuunnitteluohjelmistoja on nykyään tarjolla melko paljon. Valitsin alla olevat ohjelmistot esimerkeiksi sen perusteella, että ne ovat 2D-ohjelmistoja, eivätkä vaadi välttämättä koodausta ja niillä pystyy tekemään julkaisukelpoisia pelejä Android- ja iOS-pohjaisille laitteille. Näistä ohjelmistoista minulla on eniten kokemusta GameSaladista. Olen tutustunut kahden muunkin ohjelmiston käyttämiseen, mutta varsinaisia peliprojekteja en ole niillä tehnyt. Eri ohjelmistot eroavat toisistaan melko paljon, joten niiden suora vertailu on haasteellista.

Stencyl (2011) on Stencyl-nimisen yrityksen tuote, joka muistuttaa GameSaladia melko paljon ja kuuluu sen kovimpiin kilpailijoihin. Stencyl on

hinnoiteltu siten, että sen käyttö on ilmaista, mutta mikäli käyttäjä haluaa julkaista pelin tietokoneelle tai mobiililaitteisiin, tulee hänen maksaa lisenssimaksu. Lisenssityyppejä on tarjolla GameSaladin tapaan kaksi, Basic-lisenssin vuosiveloitus on 99 dollaria ja Pro-lisenssin 199 dollaria. (Stencyl 2019.)

Construct 3 (2018) on Construct-ohjelmistosarjan kolmas osa. Se on selainpohjainen ohjelmisto, jossa voi halutessaan ohjelmoida myös koodin avulla, mutta pakollista se ei ole. Lisenssityyppejä on saatavilla useita erilaisia. Tavallinen yksityishenkilöille tarkoitettu lisenssi maksaa 99 dollaria vuodessa, opiskelijalisenssin hinta on puolet tästä. Näiden lisäksi on saatavilla yrityslisenssejä, joita löytyy kolme erilaista. (Construct 2019.)

GameSalad on Yhdysvaltain Texasissa sijaitsevan GameSalad-nimisen yrityksen tuote. Yritys on perustettu vuonna 2007, jolloin sen nimenä oli Gendai Games. Myöhemmin nimeksi vaihtui GameSalad ja sen samanniminen ohjelmisto julkaistiin siis vuonna 2010. GameSaladista on tullut vuosien mittaan varsin suosittu, sillä on ollut yli miljoona käyttäjää. Yrityksen tavoitteena on ollut tehdä pelialaa mullistava ohjelmisto, jonka avulla koodaustaidottomatkin saisivat tehtyä pelejä ilman ensimmäisenkään koodirivin kirjoittamista. (GameSalad 2018a.) Alunperin GameSalad-ohjelmisto julkaistiin pelkästään Applen Mac-käyttöjärjestelmälle, mutta vuodesta 2012 alkaen se on ollut saatavilla myös Microsoft Windows -käyttöjärjestelmälle. Nykyään GameSaladilla suunniteltuja pelejä voidaan julkaista esimerkiksi Applen iOS, Googlen Android ja HTML5-alustoille. Tarkempia tilastoja tai luetteloita GameSaladilla tehdyistä peleistä ja niiden menestyksestä ei ole julkistettu, mutta GameSalad kertoo nettisivuillaan, että heidän ohjelmistoaan käyttämällä on tehty yli 75 peliä, jotka ovat yltäneet Applen App Storessa sadan ladatuimman pelin joukkoon, mukaan lukien muutama latauslistan kärkeen noussut peli. (GameSalad 2018a.)

Valitsin GameSaladin opinnäytetyön ohjelmistoksi sen vuoksi, että se on mielestäni erittäin helppokäyttöinen ja tarjoaa tarpeeksi ominaisuuksia pelinkehitystä aloittelevalle. Pääsin sen käyttöön nopeasti kiinni ja sen

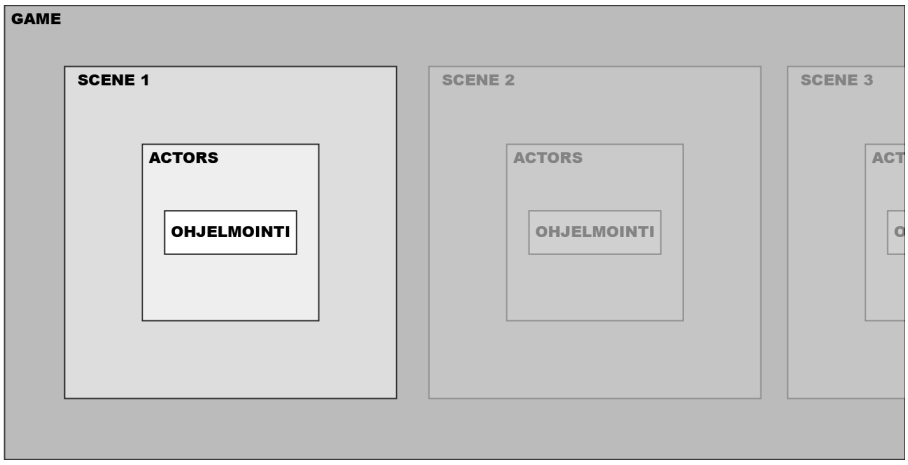
käyttäminen tuntuu hyvin luontevalta. Sen avulla voi nopeasti suunnitella ja testata pelejä, joten se sopii erinomaisesti myös erilaisten prototyyppien toteuttamiseen. GameSalad on maksullinen ohjelmisto ja tarjolla on kaksi erilaista lisenssiä. Basic ja Pro. Pro-lisenssin hinta on 299 dollaria ja Basic-lisenssin 199 dollaria. Opiskelijat voivat hankkia lisenssit puoleen hintaan. (GameSalad 2019.)

4 GameSaladin käyttäminen

4.1 Toimintaperiaate, keskeiset ominaisuudet ja toiminnot

Ohjelmiston toiminta perustuu niin sanottuun drag-and-drop, eli raahaa ja pudota -tyyppiseen visuaaliseen ohjelmointiin, jossa käyttäjä näkee pelkästään graafisen käyttöliittymän, eikä ollenkaan varsinaista koodia. Kun peli on saatu julkaisua tai testaamista varten valmiiksi, ladataan pelitiedosto GameSaladin serverille, jossa siitä muodostetaan varsinainen pelitiedosto. Tämän jälkeen pelitiedosto voidaan tallentaa serveriltä käyttäjän tietokoneelle, josta se voidaan siirtää edelleen johonkin sovelluskauppaan, kuten Applen App Storeen. (GameSalad 2018b.) GameSaladilla ei siis pysty muodostamaan julkaisuvalmista pelitiedostoa suoraan ohjelmasta tietokoneelle, vaan se täytyy aina tehdä internetin välityksellä GameSaladin servereiden kautta.

Ennen GameSaladin käytön aloittamista on hyvä tutustua sen keskeisiin ominaisuuksiin ja toimintoihin. GameSaladin sisäinen hierarkia jakautuu periaatteessa kolmeen osaan, joita ovat Game, Scene ja Actor. Game tarkoittaa tässä koko peliä, eli pelin muut elementit ovat sen sisällä (kuva 3).



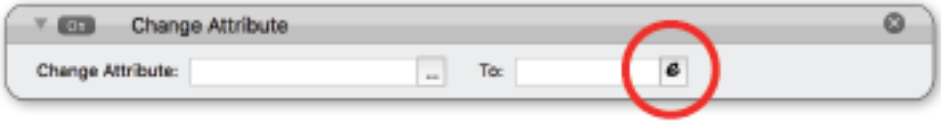
Kuva 3. GameSaladin sisäinen hierarkia.

Actor on GameSaladissa elementti, jonka sisällä itse ohjelmointi suoritetaan. Niitä voisi kutsua hahmotyypeiksi. Erilaisia hahmotyyppejä voivat olla esimerkiksi pelissä ohjattavat hahmot tai vaikkapa vihollishahmot, pelin napit ja pelin taustakuvat. Hahmotyyppien valmiista attribuuteista löytyy paljon erilaisia asetuksia, joilla voidaan säätää esimerkiksi niiden fysiikka- ja grafiikkaominaisuuksia. Hahmotyyppejä on olemassa periaatteessa kahdenlaisia, alkumuotoisia ja Scene-malliaia hahmotyyppejä. Palaan näiden eroihin esimerkkipelin suunnittelussa luvussa viisi. (GameSalad 2018b.)

Sceneä voisi kuvailla kenttäpohjaksi, jonka sisällä kaikki tapahtuu ja johon pelissä käytettävät hahmotyypit sijoitetaan. Peli voidaan jakaa useampaan eri Sceneen, eli esimerkiksi pelin päävalikolle ja eri kentille omansa. On kuitenkin mahdollista suunnitella kokonainen peli käyttämällä vain yhtä Sceneä. Pelityypin mukaan suurten kenttämäärien hallinta on usein kuitenkin helpompaa, mikäli kentät on jaettu erilleen. Tämä on kuitenkin siis pelikohtaista ja riippuu täysin pelintekijän valinnoista. (GameSalad 2018b.)

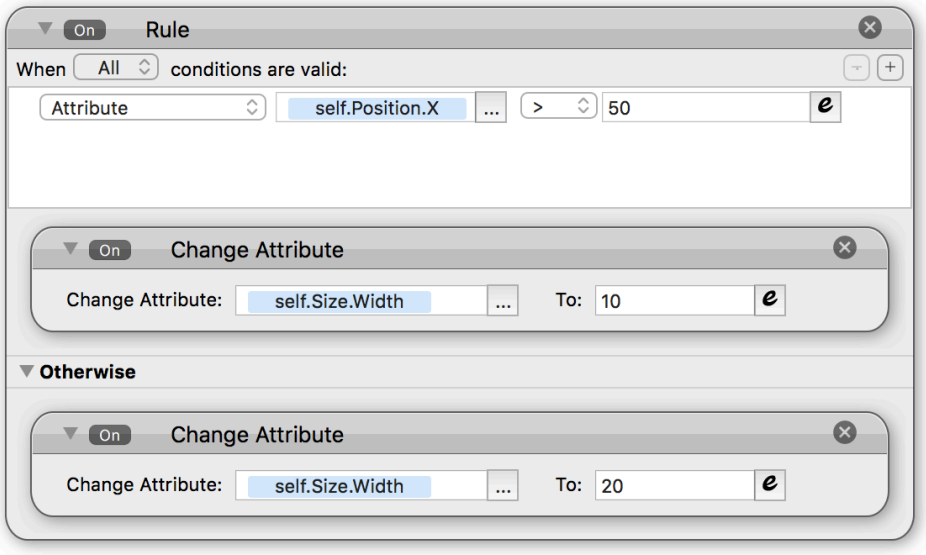
Pelin ohjelmointi tapahtuu hahmotyyppien Behaviors-toimintojen avulla. Niiden avulla hahmotyypit saadaan käyttäytymään halutulla tavalla. Mikäli halutaan esimerkiksi vaihtaa hahmotyypin käyttämää kuvaa, onnistuu se valitsemalla Behaviors-valikosta Change Image -toiminto. Erilaisia Behaviors-toimintoja löytyy GameSaladista yli 60 kappaletta. Niiden käyttäminen on melko suoraviivaista, sillä ne on mielestäni nimetty hyvin kuvaavasti. Lisäksi valikosta löytyy selkeät ohjeet jokaisen toiminnon käyttämiseen. Monimutkaisempien

matemaattisten kaavojen muodostamiseen on saatavilla Expression Editor, joka muistuttaa ominaisuuksiltaan funktiolaskinta (kuva 4). (GameSalad 2018b.)



Kuva 4. GameSaladin Expression Editorin avaaminen. Käytetty GameSalad Inc:n luvalla 27.5.2019.

Tärkeä osa Behaviors-toimintojen käyttöä ovat Rules-säännöt, joiden avulla hahmotyypit voidaan ohjelmoida toimimaan eri tavalla eri tilanteissa. Kuvassa 5 on yksinkertainen esimerkki tilanteesta, jossa muutetaan hahmotyypin leveys kymmenen pikselin suuruiseksi, mikäli sen keskikohta on hiekkalaatikon x-akselilla suurempi kuin 50 pikseliä. Muussa tapauksessa vaihdetaan hahmotyypin leveydeksi 20 pikseliä eli kyseessä on perinteinen ehtolause.



Kuva 5. Yksinkertaisen ehtolauseen muodostaminen GameSaladissa. Käytetty GameSalad Inc:n luvalla 27.5.2019.

Attribuutit (Attributes) ovat nimettyjä arvoja ja ominaisuuksia, joiden avulla hahmotyypit voivat esimerkiksi käsitellä tietoa, liikkua ja vaikkapa vaihtaa ulkomuotoaan. Jokaisella hahmotyypillä on omia attribuutteja, joiden lisäksi on olemassa myös Game- ja Scene -tason attribuutteja. Game-tason attribuutit ovat käytössä koko pelissä ja Scene-attribuutit vain kyseisessä Scenessä.

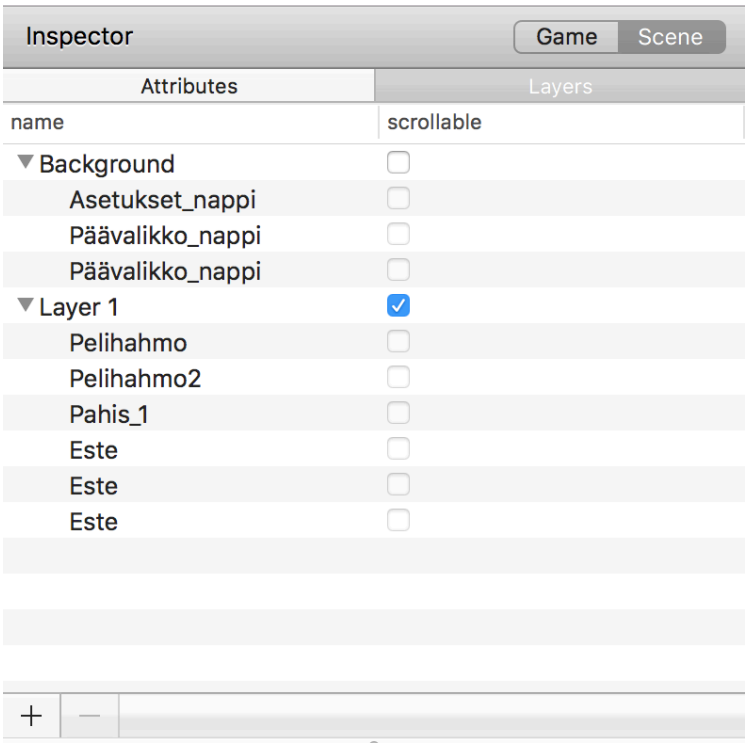
Esimerkki hahmotyyppien omista attribuuteista ovat koordinaatit, jotka kertovat hahmotyypin sijainnin Scenen x- ja y-akseleilla. Attribuutit voivat sisältää numeerista tai tekstimuotoista dataa, joskus molempiakin, riippuen valitun attribuutin tyypistä. Eri attribuuttityypit löytyvät listattuna taulukosta 1. (GameSalad 2018b.)

Taulukko 1. GameSaladin attribuuttityypit.

Attribuutti	**Datan tallennusmuoto**	**Huomioitavaa**
Boolean	TRUE ja FALSE	Myös 1 ja 0
Integer	Kokonaisluku	Pyöristyy kokonaislukuun
Text	Teksti- ja numeerinen data	
Real	Desimaaliluku	
Angle	Kulmaluku	
Index	Kokonaisluku	Vain positiiviset luvut

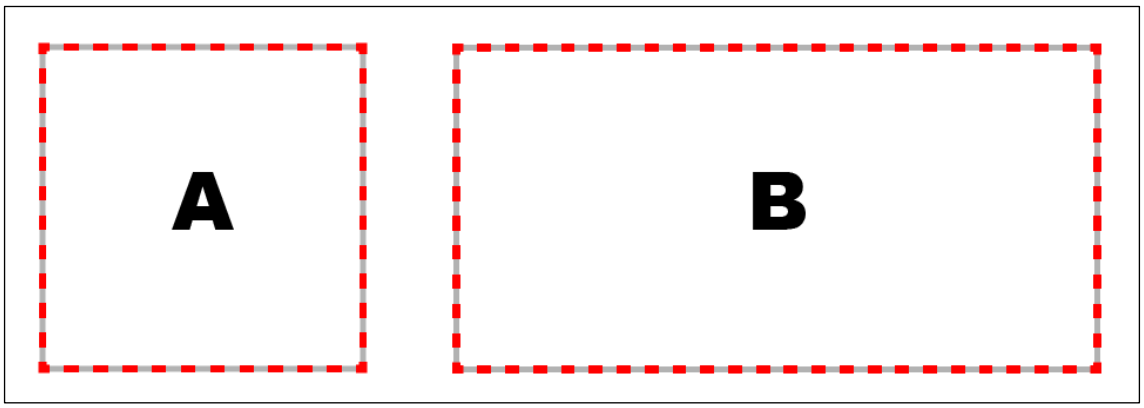
Peliä pelattaessa laitteen näytölle ilmestyvä kuva muodostetaan Scenen kameran avulla. Kameran attribuutteihin kuuluvat esimerkiksi sen koko ja seuranta-alue, jonka avulla sitä voidaan liikuttaa valitun hahmotyypin liikkeiden mukaisesti. Kameran saa seuraamaan hahmotyypin sijaintia käyttämällä hahmotyypin Control Camera -toimintoa.

GameSaladissa voidaan käyttää avuksi tasoja, jotka ovat monille tuttuja esimerkiksi kuvankäsittelyohjelmista (kuva 6). Niiden avulla voidaan säätää hahmotyyppien keskinäistä järjestystä ja tasoja voidaan myös asettaa lukituiksi, jolloin ne eivät reagoi kameran liikkeeseen. Tämä on hyödyllistä esimerkiksi pelin ohjausnappien kohdalla, eli ne pysyvät tällöin kiinteästi ruudulla.



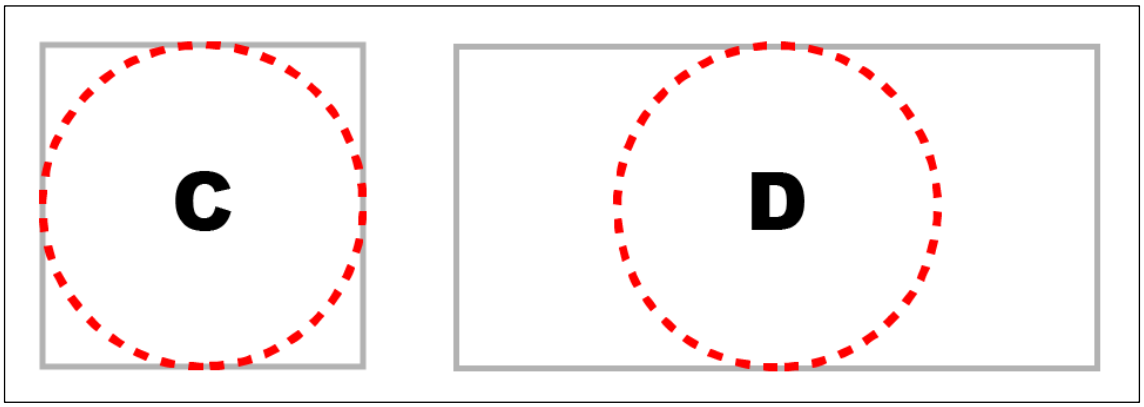
Kuva 6. Tasot GameSaladissa. Käytetty GameSalad Inc:n luvalla 27.5.2019.

Kun tehdään peli, jossa hahmotyypit voivat osua toisiinsa, on niille tärkeää ottaa käyttöön sopiva törmäysmuoto. Törmäysmuotovaihtoehdot löytyvät hahmotyyppien attribuuteista kohdasta Collision Shape. Valittavana on suorakaide, pyöreä ja käyttäjän itsensä tekemä muoto, jolloin törmäysmuoto voi olla minkä muotoinen tahansa. Käyttäjän tekemät törmäysmuodot tehdään kolmannen osapuolen ohjelmistoilla, joista tallennettu tiedosto ladataan GameSaladiin. Linkki ajan tasalla oleviin ohjeisiin ja käytettävissä oleviin ohjelmistoihin kerrotaan GameSaladissa kyseisen törmäysmuodon valinnan yhteydessä. Hahmotyypille valittu törmäysmuoto vaikuttaa myös esimerkiksi pelin käyttöliittymän nappien toimintaan, sillä kosketusalue muodostuu törmäysmuodon mukaan. Mikäli esimerkiksi napin grafiikka on ulkomuodoltaan pyöreä, halutaan luultavasti kosketusalueenkin olevan pyöreä. Kuten alapuolella oleva grafiikka esittää, on suorakaiteen muotoinen törmäysmuoto aina hahmotyypin ulkomuotojen mukainen, koska hahmotyypitkin ovat aina suorakaiteen muotoisia (kuva 7).



Kuva 7. Suorakaiteen muotoinen törmäysmuoto punaisella katkoviivalla.

Sen sijaan pyöreää törmäysmuotoa käytettäessä on tilanne se, että se on halkaisijaltaan hahmotyypin lyhyemmän sivun kokoinen. Tällöin hahmotyypistä voi jäädä joissakin tapauksissa iso osa törmäys- ja kosketusalueen ulkopuolelle (kuva 8).



Kuva 8. Pyöreä törmäysmuoto eri kokoisissa hahmotyypeissä.

4.2 GameSaladin käyttäminen

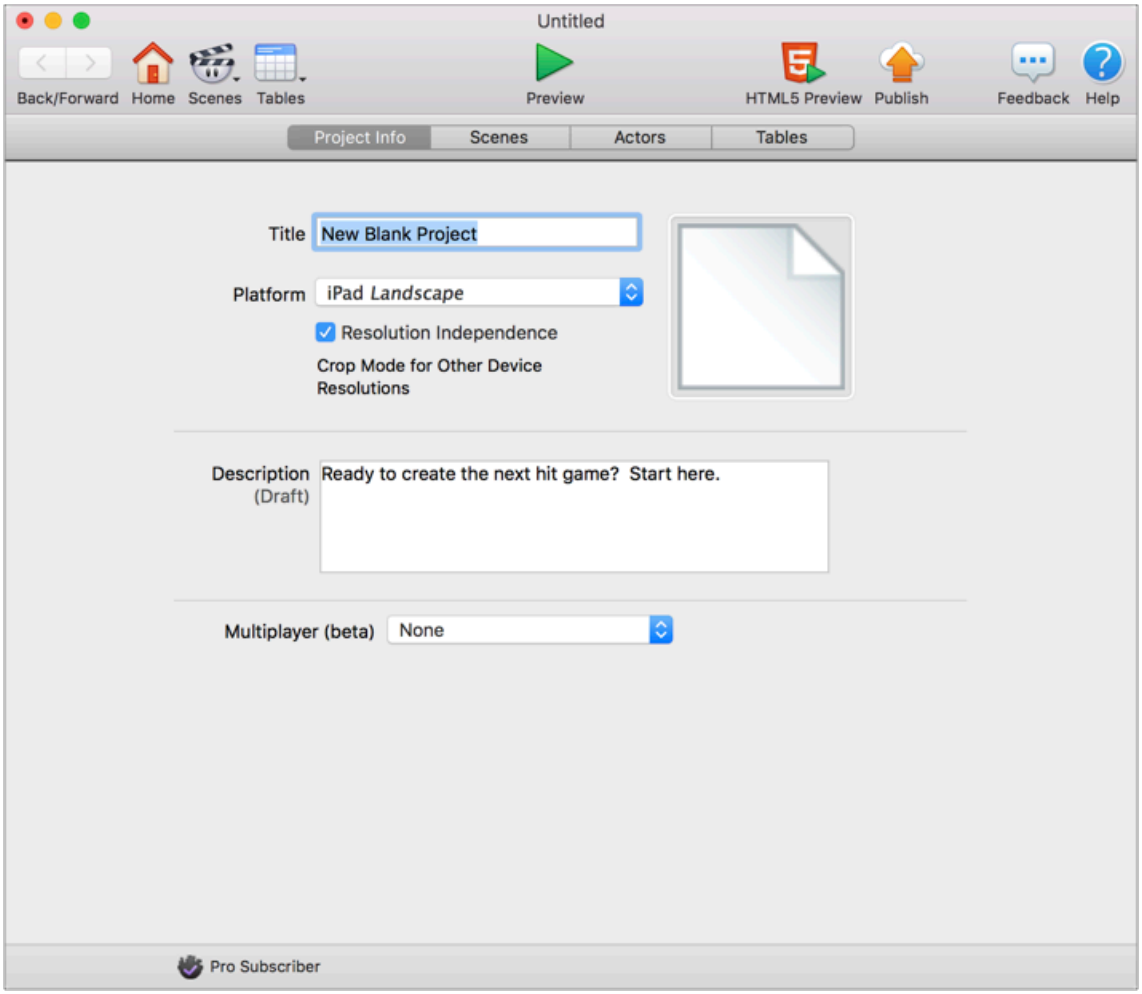
Kun GameSalad käynnistetään, avautuu ensimmäiseksi aloitusvalikko, jossa voi aloittaa uuden tai avata aiemmin tallennettuja peliprojekteja. Valikosta löytyy myös esimerkiksi linkkejä opiskelumateriaaleihin GameSaladin nettisivuille ja YouTubeen GameSaladin omalle kanavalle (kuva 9).

Kuva 9. GameSaladin aloitusvalikko. Käytetty GameSalad Inc:n luvalla 27.5.2019.

Mikäli aloitetaan uusi peliprojekti, tai jatketaan jo aikaisemmin aloitettua, aukeaa se uuteen ikkunaan. Avautuneen ikkunan Project Info -välilehdelle voidaan syöttää pelin tietoja, kuten projektin nimi ja kuvaus (kuva 10). Projektin alkuvaiheessa oleellisin on Platform-valikko, jossa valitaan se laitteisto, jolle peli tullaan suunnittelemaan.

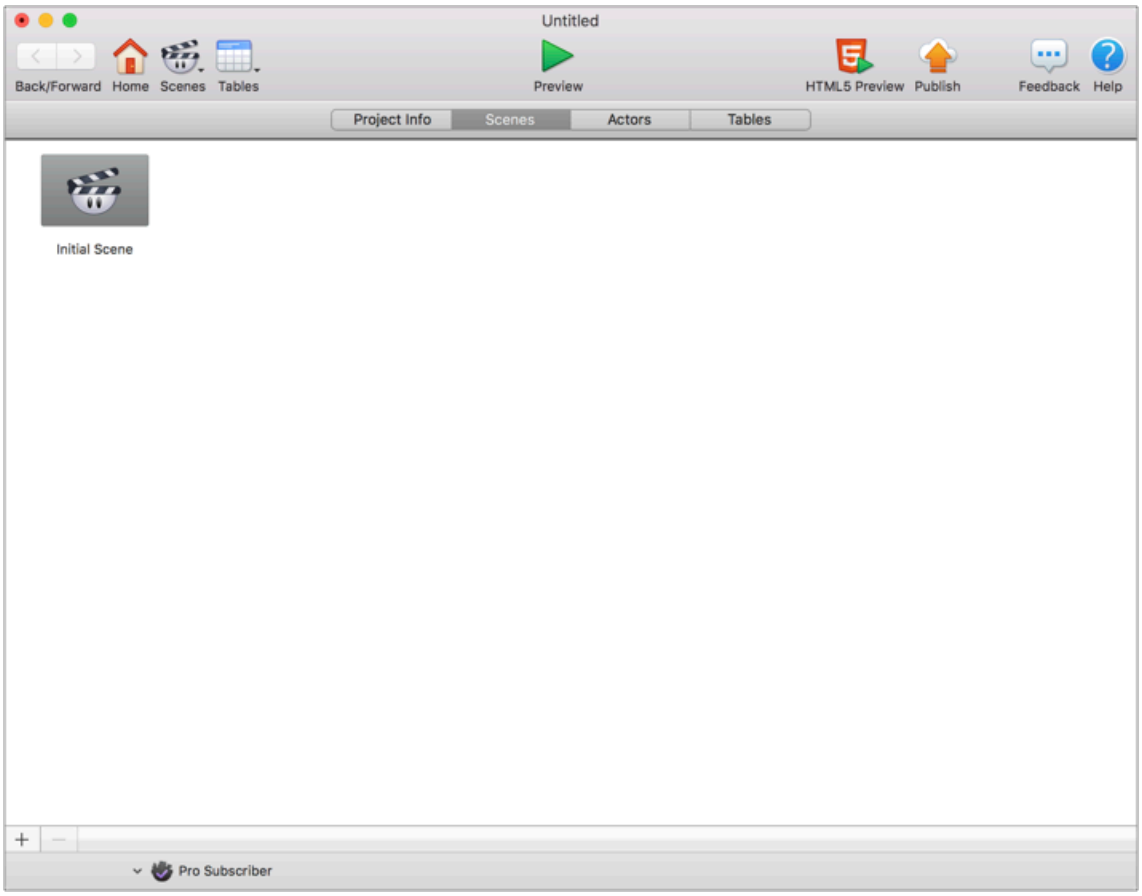
Mikäli peli aiotaan julkaista esimerkiksi Applen App Storeen siten, että se olisi yhteensopiva kaikkien Applen mobiililaitteiden kanssa, kannattaa valikosta valita iPad ja pitää myös Resolution Independence -asetus valittuna. Näin toimitaan sen vuoksi, että vanhemman sukupolven iPad-laitteissa näytön resoluutio oli heikompi (1024*768 pikseliä) kuin uudemmissa Retina-näytön omaavissa laitteissa (2048*1536 pikseliä). Asia on ratkaistu GameSaladissa siten, että pelin suunnittelu tehdään edelleen vanhemman iPadin mittojen mukaan, mutta grafiikka luodaan Retina-näytön mittojen mukaisesti. Resolution

Independence -asetuksen ansiosta GameSalad muodostaa kuvista automaattisesti sopivan kokoiset versiot heikomman resoluution omaaville laitteille, jolloin niiden ei tarvitse turhaan käyttää laitteen suorituskykyä vieviä isoja kuvatiedostoja.



Kuva 10. GameSaladin käyttöliittymän Project Info -välilehti. Käytetty GameSalad Inc:n luvalla 27.5.2019.

Scenes-välilehdellä voidaan hallita niiden järjestystä, nimiä ja lukumäärää (kuva 11). Siirtyminen valittuun Sceneen tapahtuu tuplaklikkaamalla kyseessä olevan Scenen kuvaketta. Siirtyminen Scenestä toiseen onnistuu myös yläpalkissa olevan Scenes-napin avulla, jota klikkaamalla avautuu alasvetovalikko.

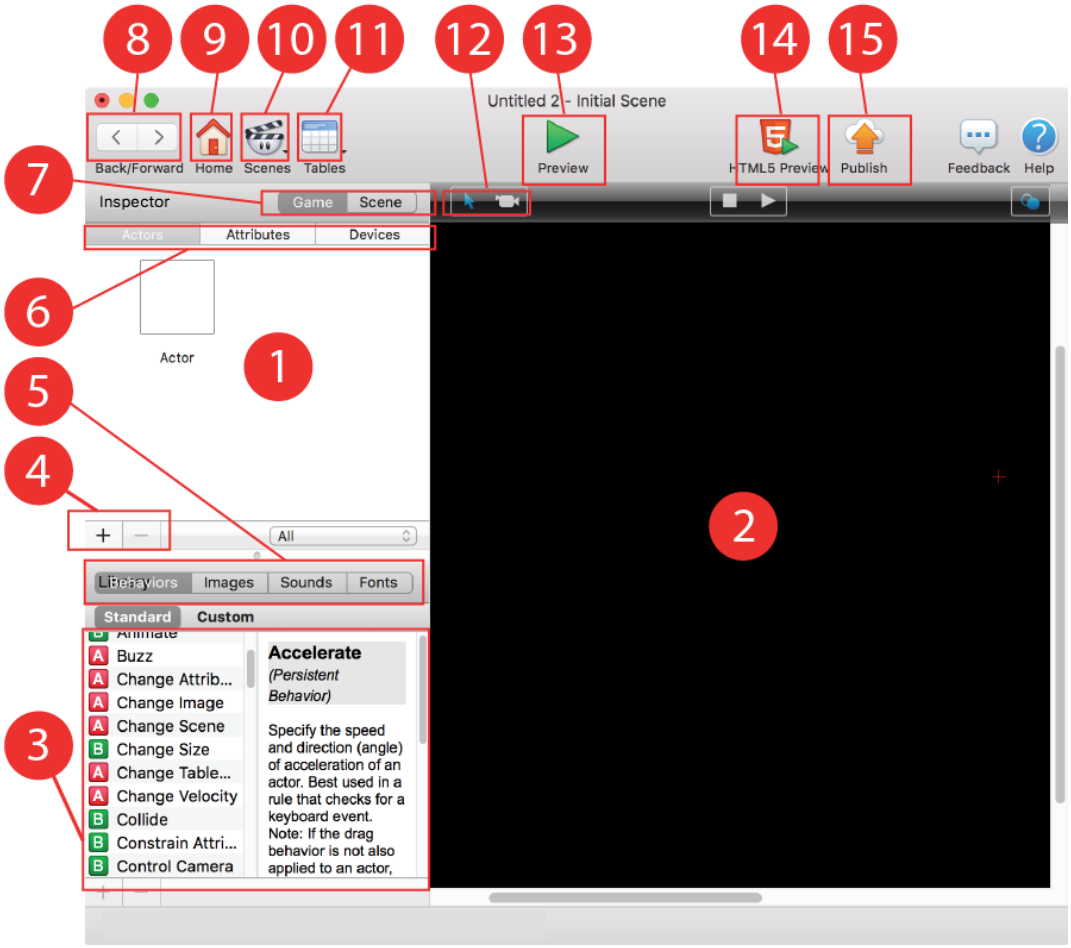


Kuva 11. GameSaladin käyttöliittymän Scenes-välilehti. Käytetty GameSalad Inc:n luvalla 27.5.2019.

Actors-välilehdeltä löytyvät peliin luodut hahmotyypit, joita voidaan hallita samaan tapaan kuin Scenejä edellisessä välilehdessä. Vasemmassa reunassa on lisäksi ryhmittelyikkuna, johon voidaan muodostaa ryhmiä tagien avulla. Tämä on hyödyllistä lähinnä silloin, kun halutaan pitää pelin eri hahmotyypit järjestyksessä. Ryhmiä voidaan käyttää toki hyväksi myös ohjelmoinnissa, sillä joitain pelin toimintoja voidaan laittaa koskemaan yhden hahmotyypin sijaan tiettyä ryhmää. Tämä on järkevää esimerkiksi peleissä, joissa on erilaisia esteitä, joihin pelihahmo voi törmätä. Tällöin jokaista estettä ei tarvitse ohjelmoida erikseen, vaan voidaan käyttää luotua ryhmää hyväksi.

Tables-välilehden avulla hallitaan niin sanottuja tauluja (Tables), joiden avulla voidaan muodostaa peliin tietokantoja. Tämä on hyödyllistä varsinkin projekteissa, joissa käsitellään isoja datamääriä. Tauluihin tallennetaan ja niistä luetaan dataa eri muodoissa, valittavana on Text, Integer, Real, Boolean ja Angle. Data tallennetaan ja luetaan tiettyjen Behaviors-toimintojen avulla.

Scene-näkymä sisältää melko paljon erilaisia toimintoja (kuva 12). Jokaisen napin ja valikon vierestä löytyy kuitenkin englanninkielinen selite, joten niiden käyttämisen oppii nopeasti. Taulukko 2 sisältää Scene-näkymän toimintojen kuvaukset.



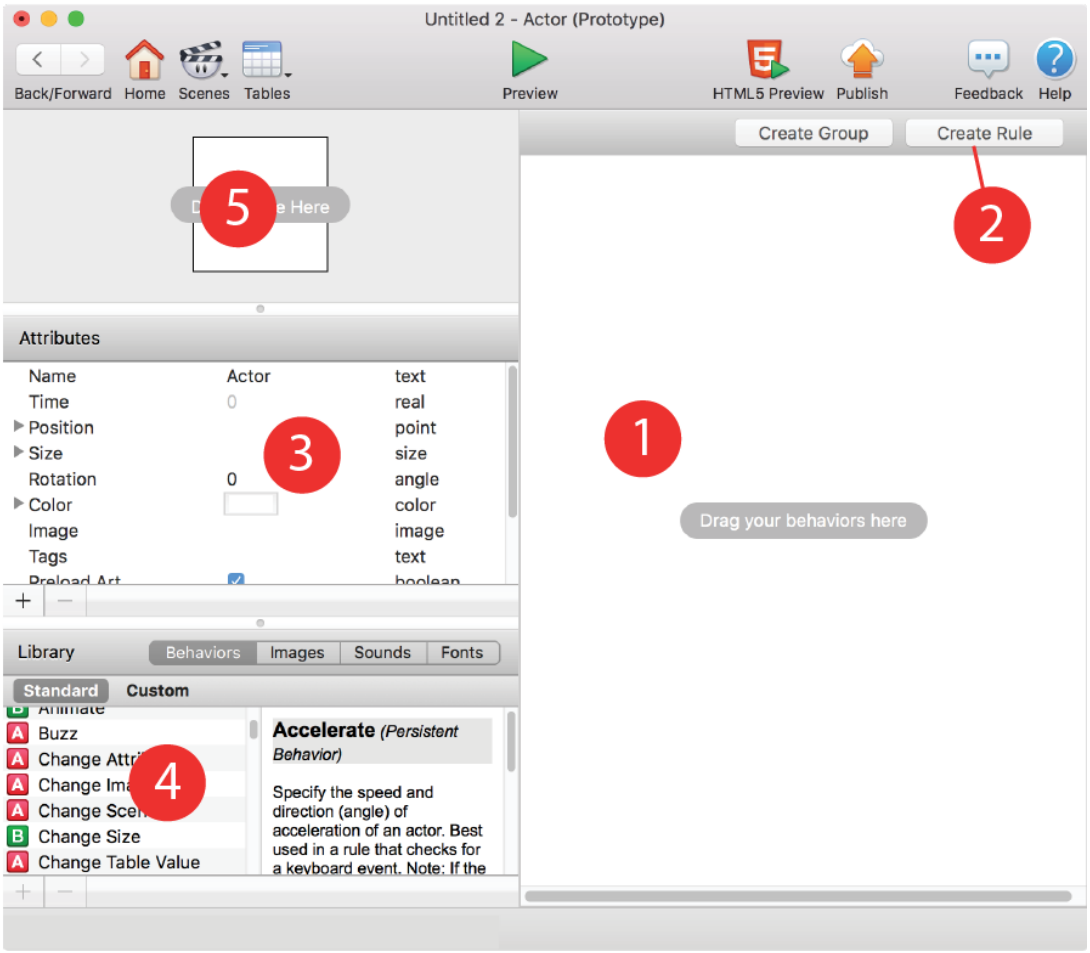
Kuva 12. GameSaladin Scene-näkymä. Käytetty GameSalad Inc:n luvalla 27.5.2019.

Taulukko 2. GameSaladin Scene-näkymän toiminnot.

1	Hahmotyyppien ”varasto”, jossa niitä voidaan lisätä, poistaa ja muokata.
2	Scene-ikkuna, pelin kenttäpohja
3	Behaviors-toiminnot
4	Hahmotyyppien lisäys ja poisto
5	Kirjasto kuville, musiikille, ääniefekteille ja fonteille.
6	Välilehdet, joista voi valita hahmotyypit, attribuutit tai laitteen asetukset.
7	Game- ja Scene -attribuuttien hallinta, eli lisäys, poisto ja nimeäminen.
8	Siirtyminen edelliseen tai seuraavaan näkymään

9	Koti-painike
10	Scenes -alasvetovalikko
11	Tables -hallinta
12	Scenen kameran seuranta-alueen hallinta
13	Pelin testaus sisäisesti GameSaladissa
14	Pelin HTML5 -testaus tietokoneen oletusselaimessa
15	Julkaisu -painike

Actor-näkymään (kuva 13) pääsee tuplaklikkaamalla hahmotyyppiä joko Scenes-näkymässä, tai sitten GameSaladin päävalikon Actors-välilehdellä. Taulukko 3 sisältää Actor-näkymän toimintojen kuvaukset. Mikäli hahmotyyppiä tuplaklikataan Scene-ikkunassa, voidaan siitä tehdä Scene-tyyppinen hahmotyyppi, jolloin se on käytössä vain kyseisessä Scenessä. Tällöin sillä pääsee käsiksi kyseessä olevan Scenen attribuutteihin.



Kuva 13. GameSaladin Actor-näkymä. Käytetty GameSalad Inc:n luvalla 27.5.2019.

Taulukko 3. GameSaladin Actor-näkymän toiminnot.

1	Ohjelmointialue, jonne raahataan Behaviors-toiminnot.
2	Rules-painike.
3	Hahmotyypin omat attribuutit.
4	Kirjasto Behaviors-toiminnoille, kuville, musiikille, ääniefekteille ja fonteille.
5	Hahmotyypin kuva, joka voidaan vaihtaa raahaamalla kuva kirjastosta.

4.3 GameSaladin vahvuudet ja heikkoudet

GameSaladin vahvuuksia ovat mielestäni ennen kaikkea pelien tekemisen nopeus ja helppous. Ohjelmiston peruskäytön oppii nopeasti ja sen avulla voi myös nopeasti testata uusia peli-ideoita. Virallinen ohjemateriaali on hyvin dokumentoitu, saatavilla on runsaasti erilaisia tutoriaaleja ja näiden lisäksi täytyy mainita GameSaladin aktiivinen keskustelufoorumi, jota kannattaa hyödyntää ongelmatilanteiden yhteydessä. Myös Youtube on erinomainen paikka etsiä apua GameSaladin käyttämiseen.

Ohjelmiston kenties suurimpana heikkoutena pidän sitä, että sillä ei voi suoraan luoda pelitiedostoa, vaan se täytyy aina tehdä internetin kautta. Tämän vuoksi käyttäjä on ikään kuin GameSaladin armoilla, sillä mikäli verkkoyhteys GameSaladin servereihin ei toimi, on pelien julkaisu mahdotonta. Mikäli GameSalad esimerkiksi menisi konkurssiin tai he päättäisivät lopettaa ohjelmiston ylläpidon, ei pelien julkaiseminen olisi enää mahdollista. Tämän vuoksi isojen ja pitkäkestoisten peliprojektien tekeminen GameSaladin avulla on hieman riskialtista.

GameSalad on suljettu ohjelmisto, joten siihen ei pysty asentamaan ns. kolmannen osapuolen tekemiä lisäosia. On kuitenkin olemassa joitakin ohjelmia, joilla pystyy muokkaamaan GameSaladilla tallennettuja tiedostoja. Mainitsemisen arvoinen ohjelma on LevelEditor (2016), jonka avulla voidaan nopeuttaa esimerkiksi tasohyppelytyyppisen pelin kenttien luomista. GameSalad ei sisällä ns. snap-to-grid -ominaisuutta, jonka avulla elementtejä

voisi sijoittaa kenttiin tarkasti ja nopeasti halutun suuruisen ruudukon avulla. Hahmotyyppien tarkka sijoittaminen olisi helpompaa, mikäli ruudukon saisi määritettyä esimerkiksi 10 pikselin kokoiseksi. LevelEditor sisältää tämän ominaisuuden lisäksi myös zoomauksen ja useiden hahmotyyppien, samanaikaisen siirtämisen. Ohjelma on ilmainen, ja sen saa ladattua tekijänsä, eli suomalaisen MantuApps Oy:n kotisivuilta. (MantuApps 2019.)

GameSaladin heikkoutena voidaan mainita myös se, että se on täysin 2D-pohjainen ohjelmisto. Lisäksi siitä puuttuu tuki vektorigrafiikalle, joten peleissä käytettävät kuvatiedostot tuodaan siihen rasterigrafiikkana, mieluiten PNG-tiedostoina (Portable Network Graphics).

5 Esimerkkipelin suunnittelu

5.1 Peliprojektin hallinta

Pelisuunnittelussa aikataulussa pysymisen kannalta on erittäin tärkeää, että alkuperäiseen suunnitelmaan ei tehdä suuria lisäyksiä peliprojektin kuluessa. Vaarana on myös se, että keskitytään liikaa asioihin, jotka ovat tuttuja ja helposti havaittavia. Esimerkiksi grafiikkaa tai pelivalikon tyyliä saatetaan muokata jatkuvasti, jolloin pelin alkuperäiset ideat ja tavoitteet voivat unohtua. (Manninen 2007, 32.)

Olen huomannut myös omissa aiemmissa peliprojekteissani, että mikäli projektiin tehdään muutoksia ja lisäyksiä jatkuvasti, saattaa pelistä tulla lopulta ikuisuusprojekti, jonka valmiiksi saaminen voi olla lopulta erittäin haastavaa. Kannattaa pitää mielessä, että jokainen muutoksista johtuva ylimääräinen päivä lykkää pelin valmistumista yhdellä päivällä. Peliprojektin aikana grafiikan suunnittelu, ohjelmointi ja testaus tapahtuvat usein rinnakkain, sillä esimerkiksi muutokset grafiikassa saattavat aiheuttaa muutoksia ohjelmointipuolella.

5.2 Pelin suunnittelussa käytetyt ohjelmistot

Pelin ohjelmoinnin suoritin siis GameSalad-ohjelmistolla. Olen perehtynyt opiskeluaikana kyseisen ohjelmiston käyttöön mielestäni erittäin hyvin ja tunnen sen ominaisuudet. Grafiikan tekemiseen käytin Adobe Illustrator CC -ohjelmistoa, jonka avulla voidaan luoda vektoripohjaista grafiikkaa. Koska GameSalad ei ainakaan toistaiseksi tue vektorigrafiikkaa, tuli kuvatiedostot tallentaa Illustratorista esimerkiksi PNG-muodossa. Päädyin Illustratorin käyttöön sen vuoksi, että minulla oli jo valmiiksi hankittuna Adoben CC -ohjelmistoille lisenssi. Tähän tarkoitukseen voisi tietenkin käyttää mitä tahansa muutakin graafiseen suunnitteluun soveltuvaa ohjelmistoa, jolla pystyy tallentamaan kuvat rasterimuodossa.

Pelin ääniefektit nauhoitin itse ja äänenkäsittelyn suoritin Audacityllä (2000), joka on ilmainen avoimen lähdekoodin äänenkäsittelyohjelmisto. Pelin taustamusiikiksi valitsin incompetech.com -sivustolta yhdysvaltalaisen Kevin MacLeodin säveltämän kappaleen ”Two Finger Johnny”. Hänen tekemäänsä musiikkia saa käyttää vapaasti, kunhan muistaa merkitä peliin tarvittavat tiedot Creative Commons -käyttöluvan mukaisesti. (Creative Commons 2019; incompetech.com 2019.)

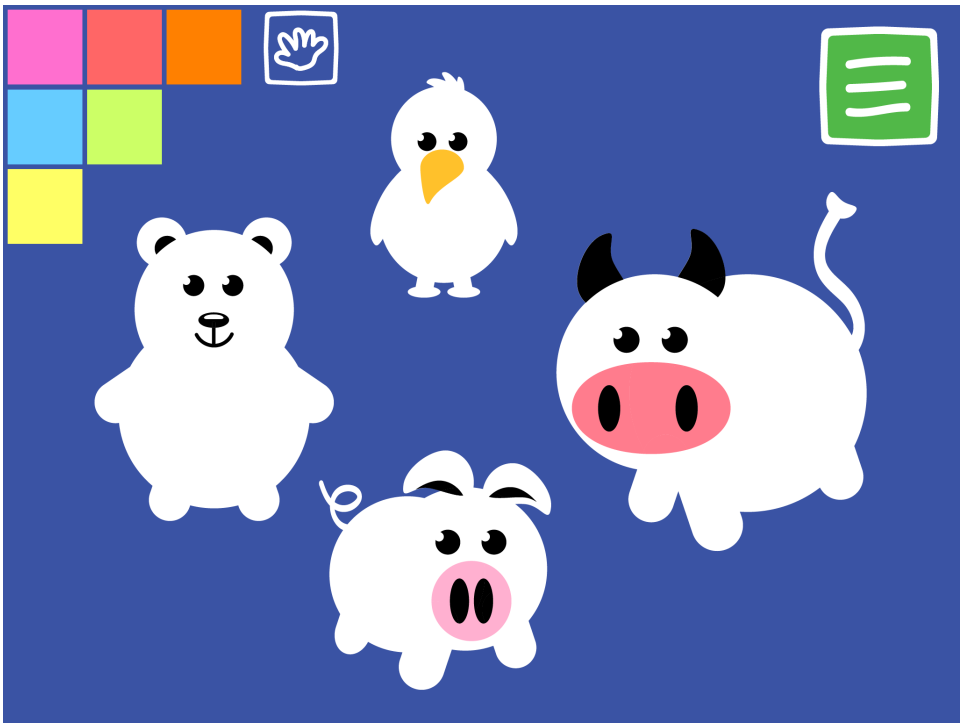
5.3 Pelin ideointi

Peliprojektin ideoinnissa mietin sitä, että millainen peli olisi oman kehittymiseni kannalta paras ja kokonsa puolesta sopiva opinnäytetyön esimerkkipeliksi. Useita erilaisia peli-ideoita mietittyäni päädyin lopulta siihen, että haluan tehdä pelin, jota voisivat pelata parivuotiaat lapsetkin. Eli kyseessä olisi mahdollisesti joidenkin lasten ensimmäinen askel mobiilipelien ja -laitteiden maailmaan.

Koska pelin kohderyhmänä on lapset, oli asiaa hyvä pohtia myös eettisestä näkökulmasta. Millainen pienille lapsille suunnattu peli saa olla? Onko heille ylipäänsä järkevää suunnitella pelejä? Pitäisi olla itsestään selvää, että lapsia tulee suojella haitallisena pidetyltä sisällöltä, joten esimerkiksi seksi, päihteet ja

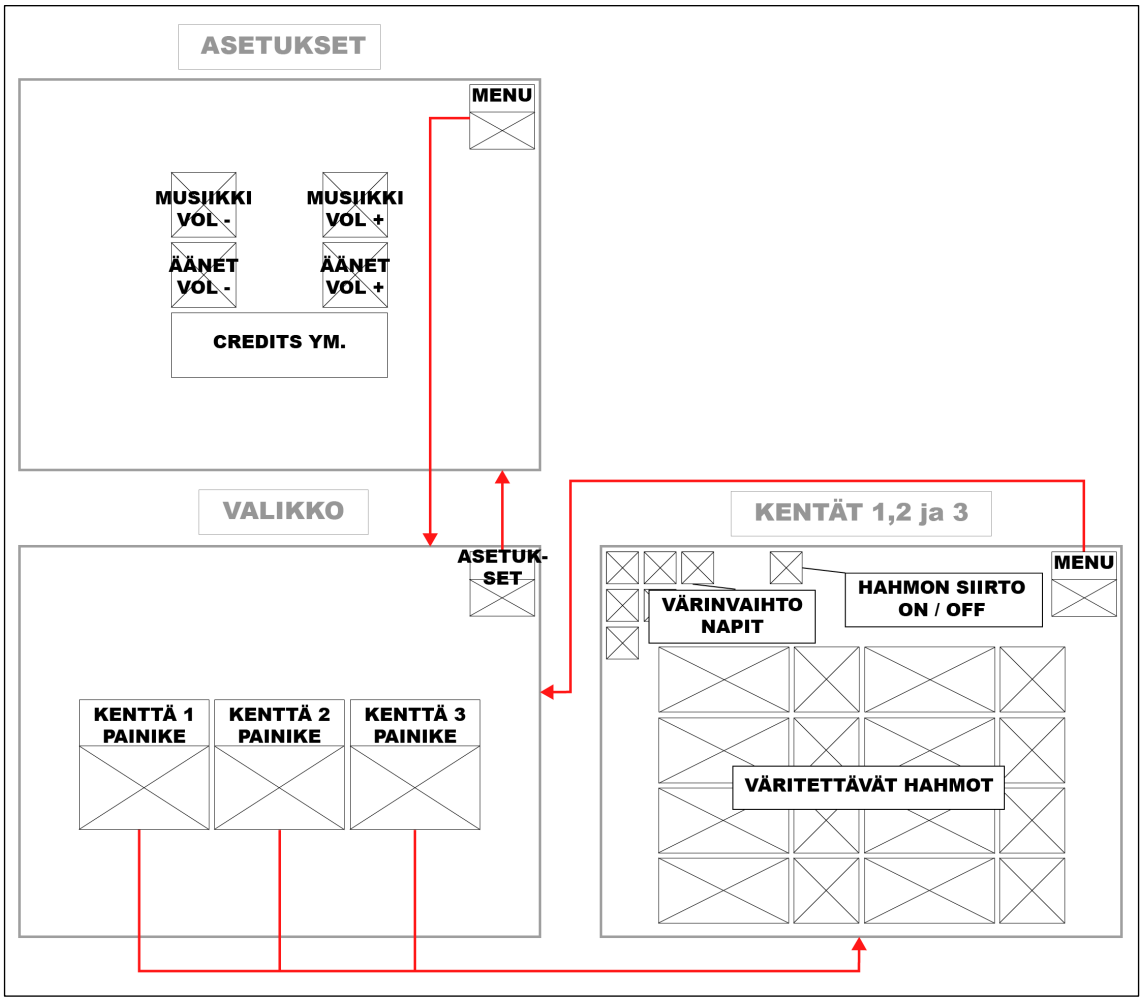
väkivalta eivät kuulu lasten peleihin. Pienet lapset saattavat pelätä myös erikoiselta näyttäviä hahmoja, kuten mörköä tai noitaa. (Kansallinen audiovisuaalinen instituutti 2019.) Maailman terveysjärjestö WHO:n huhtikuussa 2019 julkaistujen suositusten mukaan alle vuoden ikäisten lasten ei tulisi saada ollenkaan staattista ruutuaikaa ja 2–4-vuotiaidenkin vain korkeintaan yhden tunnin verran vuorokaudessa (WHO 2019, 9). Pienille lapsille suunnattuihin mobiilipeleihin voisikin mielestäni lisätä vanhemmille muistutuksen suositellusta ruutuajasta ja siitä, että pelien pelaaminen ei korvaa lasten leikkimistä ja sosiaalista kanssakäymistä, kuten yhteisiä lukuhetkiä.

Suunniteltavaa peliä miettiessäni syntyi idea yksinkertaisesta värityskirjatyyppisestä, erityisesti Applen iPadille tehdystä sovelluksesta, jossa leikkimielisten hahmojen värittäminen onnistuisi niitä sormella koskettamalla. Peli koostuu kolmesta kentästä, joista jokaisella on oma teemansa: ajoneuvot, eläimet ja kasvit. Pelin hahmot ovat aluksi kaikki samanvärisiä, eli valkoisia. Halutun värin voi valita ruudun vasemmasta yläkulmasta ja tämän jälkeen kyseinen väri voidaan siirtää hahmoihin niitä sormella koskettamalla. Pelin hahmoja voidaan myös siirtää, siirtotilan saa päälle kun sormella napautetaan ihmisen kättä kuvaavaa nappia värinvalintanappien vieressä (kuva 14).



Kuva 14. Esimerkkipelin 2. kenttä.

Pyörittelin mielessäni erilaisia ideoita siitä, miltä suunniteltavan pelin tulisi näyttää ja mitä sisältöä siihen kannattaisi laittaa. Tein myös muutamia erilaisia prototyyppejä GameSaladilla, joista valitsin mielestäni parhaiten toimivan. Suunnittelu lähti tämän jälkeen käyntiin perinteisesti kynän ja paperin avulla, joilla loin myös pelistä alustavan rautalankamallin (kuva 15). Toteutin tämän jälkeen rautalankamallin tietokoneella Adobe Illustrator CC:n avulla. Idean rautalankamallin hyödyntämiseen pelin suunnittelussa sain opintojeni aikana suorittamastani Peli- ja mobiiliympäristöt -opintojaksolta, jossa luotiin rautalankamalli kuvitteelliselle mobiilisovellukselle.



Kuva 15. Suunniteltavan pelin rautalankamalli.

Ennen suunnittelun aloittamista kannattaa päättää se, että tuleeko peli olemaan pelattavissa laitteen ollessa pysty- vai vaaka-asennossa, vai toimiiko se kenties kummankin vaihtoehdon kanssa. Tämän pelin suunnittelin toimimaan vain laitteen ollessa vaaka-asennossa.

Pelissä on siis rautalankamallin mukaisesti Päävalikko, Asetukset-näkymä ja kolme varsinaista pelikenttää. Minulla ei ollut aiempaa kokemusta rautalankamallin käyttämisestä peleissä, mutta siitä tuntui olevan paljon apua. Käytin rautalankamallia suunnittelussa pelin eri elementtien tilanvaraajina ja siitä selviää helposti myös se, miten pelin sisällä siirrytään näkymästä toiseen. Tämä yhdessä vuokaavioiden kanssa auttaa varmasti muistamaan tehdyt ratkaisut esimerkiksi silloin, kun projektin pariin palataan pidemmän tauon jälkeen. Koska pelin kohderyhmänä on lapset, en halunnut lisätä tähän peliin mitään mainoksia tai ostomahdollisuuksia esimerkiksi lisäominaisuuksille. Lasten peleihin kohdistuneet lisämaksut ovat aiemmin aiheuttaneet kritiikkiä pelialan yrityksiä kohtaan. Mikäli tämä peli joskus julkaistaan, tulee se olemaan niin sanotusti kertamaksullinen, eli hankinnan jälkeen siitä ei voi koitua käyttäjälle lisäkustannuksia.

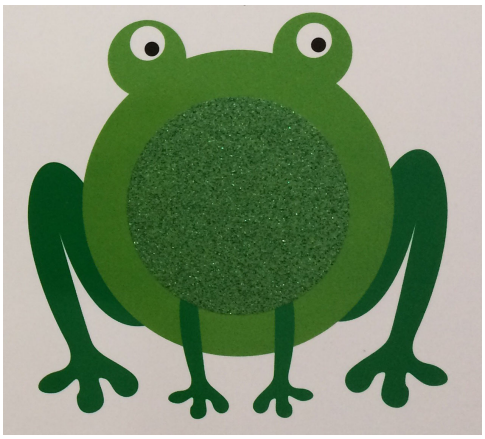
Erilaisia lapsille suunnattuja värityskirja-tyyppisiä sovelluksia on tarjolla melko paljon. Esimerkiksi Toonia Colorbook (2013) on ollut suosittu, sillä sitä on ladattu yli kaksi miljoonaa kertaa Applen App Storesta. Yhteinen piirre kokeilemissani sovelluksissa oli se, että niissä hahmojen väritys tapahtuu samaan tapaan kuin värikynällä oikeassakin elämässä, eikä hahmoa napauttamalla, kuten tämän opinnäytetyön esimerkkipelissä. Kokeilemissani sovelluksissa grafiikka oli tehty melko paksua ääriviivaa käyttäen ja värityksessä ideana oli pysyä viivojen sisäpuolella. Varsinkin ilmaiseksi ladattavissa peleissä oli tarjolla runsaasti maksullisia lisäominaisuuksia, joita ilman sovellusten käyttäminen tuntui puutteelliselta. Lisäksi näytölle ilmestyneet mainokset maksullisista lisäominaisuuksista heikensivät käyttökokemusta.

5.4 Grafiikka

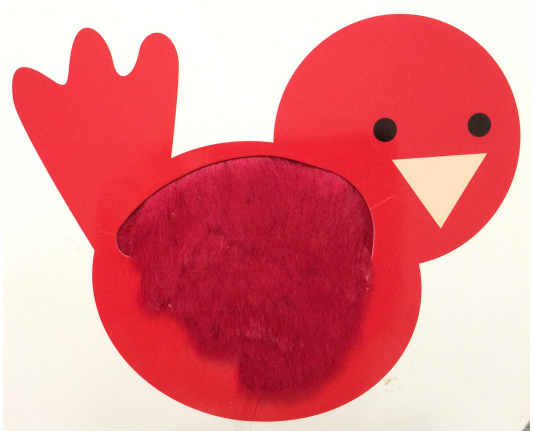
5.4.1 Kuvitusprosessi

Etsin esimerkkipelin kuvitustyyliin ideoita lasten kuvakirjoista ja myös internetistä Googlen kuvahaun avulla. Pyrin miettimään kuvakirjoissa tehtyjä ratkaisuja ja syitä niiden taustalla. Lähes kaikkien selailemieni taaperoikäisille

tehtyjen kuvakirjojen kuvitustyyli oli hyvin selkeä ja yksinkertainen. Kuvissa 16 ja 17 on kuvitusesimerkkejä lasten kuvakirjasta ”Värit: katso ja tunnustele”. Hahmojen värit erottuivat taustoistaan selvästi. Apulaisprofessori Maria Laukan mukaan selkeät rajat ja kompaktit väripinnat omaavat hahmot auttavat pieniä lapsia ymmärtämään kuvien sisältöä paremmin (Laukka 2001, 38).



Kuva 16. Valokuva kirjasta ”Värit: katso ja tunnustele” (Land 2014, 4). Käytetty Egmont Kustannus Oy Ab:n luvalla 4.6.2019.



Kuva 17. Valokuva kirjasta ”Värit: katso ja tunnustele” (Land 2014, 8). Käytetty Egmont Kustannus Oy Ab:n luvalla 4.6.2019.

Flat design -tyyli on minimalistista suunnittelua, jossa pyritään käyttämään ainoastaan yksinkertaisia, kaksiulotteisia ”litteitä” elementtejä. Siitä pyritään jättämään pois kaikki turhat koristelut, joilla perinteisessä suunnittelussa pyritään saamaan aikaan realistista vaikutelmaa. Tämän tyylin pohjimmainen ajatus on se, että vähemmän on enemmän. Tällöin saadaan aikaan parempi käyttäjäkokemus ja puhtaamman näköinen, selkeämpi lopputulos. Tyyliin

kuuluu olennaisena osana paksut, päätteettömät ja helppolukuiset kirjasintyypit. Flat design -tyylissä on perinteisesti käytetty kirkkaita värejä, joilla on suuri kontrasti taustavärin kanssa. Alkuperäiseen flat design -tyyliin ei siis kuulu mitkään kolmiulotteiset elementit, kuten varjot. Tämä on aiheuttanut ongelmia esimerkiksi sovellusten ja nettisivujen käyttöliittymien suunnitelussa, kun sovellusten käyttämiseen tarvittavia interaktiivisia elementtejä, kuten nappeja, ei ole voitu erottaa muusta grafiikasta esimerkiksi juuri varjojen avulla. Tämän vuoksi tästä tyylistä on kehitetty uudempi versio, jota suunnittelijat kutsuvat nimellä flat design 2.0. Siihen kuuluvat esimerkiksi syvyysvaikutelmaa antavat varjot, yksityiskohdat ja kuvan eri tasot, eli se ei ole välttämättä edeltäjänsä tavoin täysin litteä. (Esposito 2019.) Otin vaikutteita flat design 2.0 -tyylistä esimerkkipelin grafiikan luomisessa, eli pyrin pitämään grafiikan melko minimalistisena, mutta lisäsin kuitenkin pieniä yksityiskohtia.

Esimerkkipelissä on jonkin verran tekstiä, että esimerkiksi kentän vaihtamiseen käytettävissä napeissa ja tietenkin pelin nimessä, joka löytyy päävalikosta. Kirjasintyyppinä oli käytössä Patrick Wagesreiterin suunnittelema Patrick Hand SC. Se on saatavana ilmaiseksi OFL-lisensoituna Google Fonts -sivustolta. (Google Fonts 2019.) Kyseinen kirjasintyyppi sopii mielestäni hauskan ja selkeän ulkomuotonsa ansiosta hyvin lapsille suunniteltuun sovellukseen, kuten myös flat design -tyyppiseen suunnitteluun. Koska pienille lapsille suunnattujen pelien käyttäminen pitäisi olla ongelmatonta myös ilman lukutaitoa, tulisi myös pelin nappien grafiikasta selvitä niiden käyttötarkoitus. Tämän vuoksi esimerkiksi kentän vaihtamiseen tarkoitettujen nappien grafiikasta löytyy kyseisten kenttien sisältämiä hahmoja.

Olin miettinyt jo rautalankamallia tehdessäni minkä kokoisia monet peliin tulevat hahmotyypit olisivat mitoiltaan pikseleissä. Tämän vuoksi pystyin luomaan Illustratorissa sopivan kokoiset piirtoalustat jokaiselle yksittäiselle kuvalle. Tein lisäksi oman piirtoalustan pelin eri näkymille, eli valikolle, asetuksille ja jokaiselle kentälle. Näiden avulla pystyin hahmottamaan pelin graafista kokonaisuutta paremmin.

Suunnittelin peliin kolme erilaista kenttää, joilla kaikilla oli omanlainen teemansa: kulkuneuvot, eläimet ja kasvit. Nämä eroavat toisistaan myös graafisen ulkonäkönsä puolesta. Ajoneuvot ovat teknisiä laitteita, joten käytin niiden piirtämisessä teräviä kulmia ja suoria linjoja. Eläimissä käytin enemmän pyöreitä muotoja. Kasveista pyrin tekemään selkeitä ja yksinkertaistettuja (kuva 18).

Kuva 18. Pelin hahmoja.

5.4.2 Kuvatiedostoilta vaaditut tekniset ominaisuudet

Kun kuvat halutaan ottaa pelissä käyttöön, tulee ne tietenkin saada oikeassa muodossa ulos Illustratorista. Tämä onnistuu helpoiten Export As.. -toiminnon avulla, jolloin voidaan määritellä myös se, että ajetaanko ulos yksittäinen kuva, vai esimerkiksi kaikki kuvat kerralla. Piirtoalustat kannattaa nimetä järkevästi, sillä jokainen syntyvä kuvatiedosto saa nimensä piirtoalustansa nimen mukaan. Suositeltava muoto GameSaladin kuvatiedostoille on PNG. Se tosin tukee myös monia muita tiedostomuotoja. Mikäli grafiikassa on kohtia, joiden on tarkoitus olla pelissä läpinäkyviä, tulee Illustratorissa valita Transparent background -asetus päälle eli kuvien tausta läpinäkyväksi. GameSaladiin tuotavien kuvatiedostojen tarkkuuden tulee ehdottomasti olla tasan 72 dpi eli 72 pistettä

tuumaa kohden. Mikäli näin ei ole, tulee kuvista pelissä todennäköisesti erittäin suttuisia. Kuvan mitat ja tarkkuuden voi tarkistaa esimerkiksi Applen Mac -tietokoneissa kuvien esikatselun avulla. (Novak 2014, 126.)

Tärkeä asia kuvatiedostoja luotaessa on myös se, että kuvin tulisi mielellään noudattaa niin kutsuttua Power of two -sääntöä, eli kuvan pikselimittojen tulee olla jaollisia kahdella. Suositeltavat kuvien mitat ovat 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 tai 2048 pikseliä. Mikäli kuvien mitat poikkeavat näistä, on se laitteiden muistin käytön suhteen epäedullista. sillä mikäli kuva on esimerkiksi 514 pikseliä leveä, käyttää laite siihen 1024 pikseliä leveää kuvaa vastaavasti muistia. Eli kuvan mitat pyöristetään ylöspäin seuraavaan ”Power of two”-lukuun. GameSaladin hyväksymä suurin kuvakoko on 2048*2048 pikseliä. (Novak 2014, 126.)

Kuvatiedostot kannattaa mielestäni nimetä mahdollisimman kuvaavasti, tämä helpottaa etenkin kuvien päivitystä. Kun kuvatiedostoja halutaan päivittää, se onnistuu helposti siten, että uudet kuvatiedostot ladataan GameSaladiin, jolloin päivitys tapahtuu automaattisesti, kunhan päivitettävien kuvien nimet vastaavat alkuperäisiä. Tein työtäni helpottamaan taulukkolaskentaohjelmalla listan, johon kokosin kaikkien kuvatiedostojen nimet, mitat ja muita lisätietoja, kuten esimerkiksi päivitystä vaativat kuvatiedostot (taulukko 4).

Taulukko 4. Kuvakaappaus suunnittelua helpottamaan tehdystä kuvalistasta.

Tiedoston nimi	Leveys*korkeus	Käyttökohde	Sijainti	Tila
kenttä_1.png	512*512	Kentänvaihtonappi_1	Päävalikko	ok
kenttä_2.png	512*512	Kentänvaihtonappi_2	Päävalikko	ok
kenttä_3.png	512*512	Kentänvaihtonappi_3	Päävalikko	ok
asetukset.png	256*256	Menu/asetukset-nappi	Päävalikko	ok
musiikki_up.png	256*256	Musiikin voimakkuus ylös-nappi	Asetukset	ok
musiikki_down.png	256*256	Musiikin voimakkuus alas-nappi	Asetukset	ok
ääniefektit_up.png	256*256	ääniefektit_up-nappi	Asetukset	ok
ääniefektit_down.png	256*256	ääniefektit_down-nappi	Asetukset	ok
päävalikko.png	256*256	Menu/asetukset-nappi	Päävalikko	ok
värinvaihto.png	256*256	Värinvaihtonappi	Asetukset	ok
hahmonsiirto.png	256*256	Hahmonsiirtonappi	Kentät 1,2,3	ok
hahmo1.png		hahmo1	Kenttä 1	ok
hahmo2.png		hahmo2	Kenttä 1	ok
hahmo3.png		hahmo3	Kenttä 1	ok
hahmo4.png		hahmo4	Kenttä 1	ok
hahmo5.png		hahmo5	Kenttä 1	ok
hahmo6.png		hahmo6	Kenttä 2	ok
hahmo7.png		hahmo7	Kenttä 2	ok
hahmo8.png		hahmo8	Kenttä 2	ok
hahmo9.png		hahmo9	Kenttä 2	ok

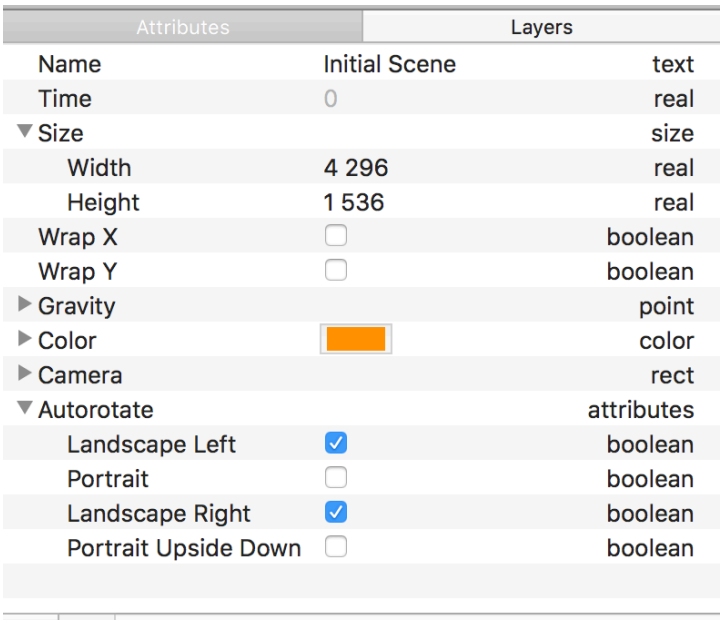
5.5 Ohjelmointi

5.5.1 Esivalmistelut ennen ohjelmointia

Ennen varsinaista ohjelmointia mietin rautalankamallin pohjalta sitä, että minkälaisia hahmotyyppejä peliin täytyy luoda. Kokosin näistä listan ja hahmottelin niiden toimintoja myös paperille vuokaavioiden muodossa. Vuokaavioiden avulla sain mietittyä pelissä tarvittavat attribuutit ja toiminnot jo etukäteen. Peliin suunnitellut vuokaaviot löytyvät opinnäytetyön liitteistä. Tämän jälkeen tein Illustratorilla pelin pohjapiirustuksen (liite 2), josta selviää hiekkalaatikon mitat, hahmotyyppien koot ja sijainnit x- ja y-akseleilla. Käytin pohjapiirustusta tehdessäni hyväkseni aiemmin tekemääni rautalankamallia, jossa pelin Scene ja suurin osa hahmotyypeistä olivat oikeankokoisia pelin väritettäviä hahmoja lukuun ottamatta. Tämän pohjapiirustuksen tarkoituksena oli se, että pystyin sen avulla sijoittamaan pelin hahmotyyppejä oikeisiin paikkoihin nopeammin. Periaatteessa rautalankamallin ja tämän pohjapiirustuksen voisi yhdistää, mutta se saattaisi olla hieman sekava ratkaisu. Kuten pohjapiirustuksesta selviää, pelin kaikki kentät löytyvät samasta Scenestä, jolloin siirtyminen kenttien välillä tapahtuu kameran koordinaatteja muuttamalla.

5.5.2 GameSaladin asetukset

Ensimmäinen muutos GameSaladin asetuksiin oli Scenen mittojen muuttaminen oikeiksi. Tämä tapahtuu menemällä Scenen asetuksiin, eli kohtaan Scene Attributes, joista löytyy kohta Size. Korkeudeksi muutin tehdyn pohjapiirustuksen mukaisesti 1 536 pikseliä ja leveydeksi 4 296 pikseliä (kuva 19).



Kuva 19. GameSaladin Scene-attribuutit. Käytetty GameSalad Inc:n luvalla 27.5.2019.

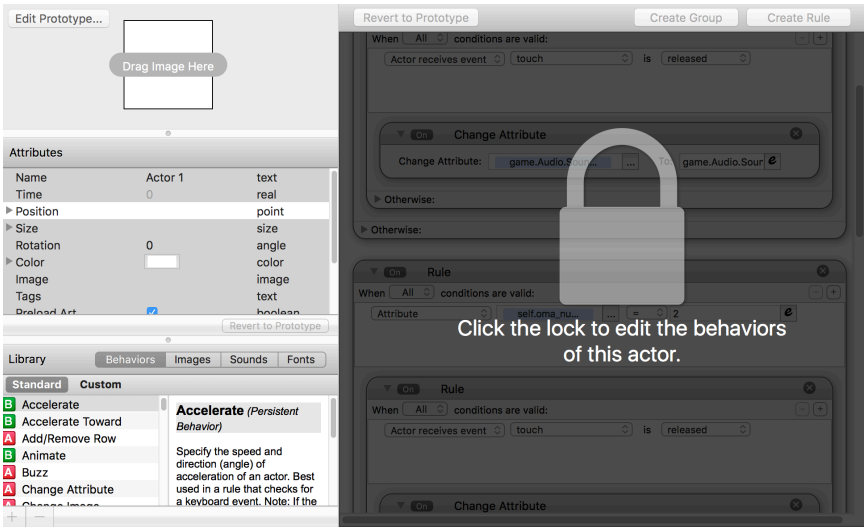
Koska peli suunnitellaan pelattavaksi siten, että, että iPad on vaakatasossa, tulee GameSaladin asetuksista valita pystyasennot pois käytöstä. Tämä onnistuu menemällä Scene Attributes-valikon Autorotate-kohtaan, josta valitaan päälle Landscape Left ja Landscape Right.

5.5.3 Päävalikon ja Asetukset -näkymän nappien suunnittelu

Ohjelmoin kaikkiin valikon nappeihin hieman animointia, että käyttäjä saisi palautetta niiden koskettamisesta. Nappien leveys ja korkeus kasvaa kahdeksan pikseliä ja ne kääntyvät tämän lisäksi satunnaisen asteluvun verran (5-10 astetta). Kun sormi nostetaan tai liu'utetaan pois niiden päältä, palautuu nappien koko ja kulma ennalleen.

Hahmotyypeistä ensimmäiseksi päävalikon kentänvaihtonapit, joiden avulla siirrytään päävalikosta haluttuun kenttään. Aloitin suunnittelun siten, että tein nappien toimintaa kuvaavan vuokaavion, josta selviää miten napit toimivat koskettaessa. Toteutin nämä napit siten, että sormi tulee nostaa kyseisen napin päältä, että sille ohjelmoidut tehtävät suoritettaisiin. Näin ollen pelkkä sormen liikuttelu näytön päällä ei aiheuta kameran siirtymistä. Näillä

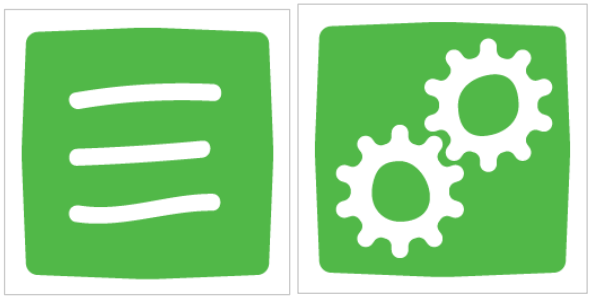
kentänvaihtonapeilla on siis periaatteessa vain yksi tehtävä, joka on liikuttaa pelin kameraa x-akselilla haluttuun kohtaan. Tein tässä vaiheessa myös aktiivinen_kenttä-nimisen Game-attribuutin, jonka arvo vaihdetaan aina kentänvaihdon yhteydessä kyseisen kentän mukaiseksi. Tätä attribuuttia tullaan käyttämään myöhemmin hahmojen läpinäkyvyyden poistamiseksi. On hyvä tietää, että esimerkiksi kameraan liittyvät attribuutit ovat Scenen sisäisiä attribuutteja, joihin ei pääse käsiksi normaalilla alkumuotoisella hahmotyypillä, vaan siitä täytyy tehdä Scene-mallin hahmotyyppi. Se onnistuu tuplaklikkaamalla sitä Scene-ikkunassa ja avautuneessa Actor-näkymässä klikataan lukkoa esittävää kuvaa (kuva 20). Scene-mallisten hahmotyyppien ohjelmointi tapahtuu avaamalla ne Scene-ikkunassa, eikä siis perinteisesti hahmotyyppien varaston kautta.



Kuva 20. Hahmotyypin muuttaminen Scene-malliseksi. Käytetty GameSalad Inc:n luvalla 27.5.2019.

Samaan tapaan toimii nappi, jolla siirrytään muista näkymistä Päävalikkoon ja Päävalikosta Asetukset-näkymään. Eli sille ohjelmoidaan kaksi varsinaista toimintoa. Päävalikkonäkymässä tämän napin käyttämä grafiikka vaihdetaan asetukset.png -kuvatiedosto ja kameran ollessa muissa näkymissä on käytössä päävalikkoon.png -kuvatiedosto (kuva 21). Testauksen aikana huomattiin, että kun tätä nappia käytettiin kameran ollessa jonkin kentän kohdalla, siirtyi kamera suoraan Päävalikon yläpuolella olevaan Asetukset-näkymään, vaikka tarkoitus oli siirtyä Päävalikkoon. Tämä ongelma ratkaistiin luomalla painettu_x_0-

niminen attribuutti, jonka avulla määriteltiin, että Asetukset-näkymään pääsee vain painamalla tätä nappia kameran ollessa Päävalikon kohdalla.



Kuva 21. Päävalikko- ja Asetukset -näkymiin siirtymiseen käytettävän napin grafiikat.

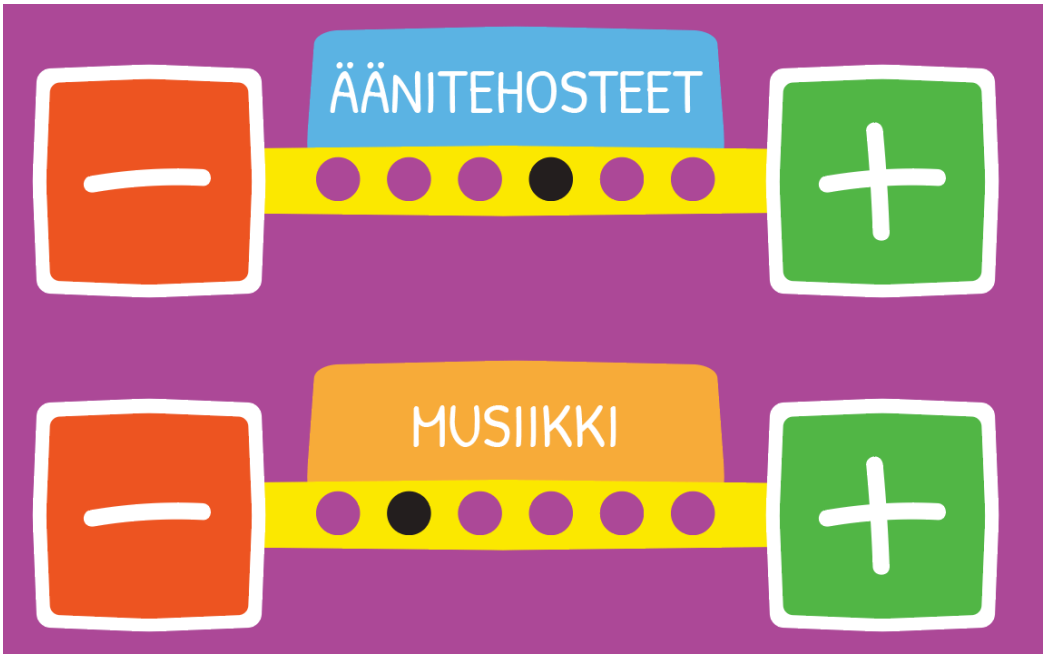
Pelin asetuksia varten tarvittiin neljä nappia eli äänitehosteiden ja musiikin voimakkuuden lisääminen ja vähentäminen. Ohjelmoin kaikkien neljän napin toiminnot yhden hahmotyypin sisälle siten, että tein tälle hahmotyypille ”oma_numero”-nimisen Integer-attribuutin, jonka avulla pystyin erottamaan eri napeille suunnitellut toiminnot toisistaan (taulukko 5). Raahasin näitä hahmotyyppejä Scene-ikkunaan tarvittavat neljä kappaletta. Tämän jälkeen asetin niiden ”oma_numero”-attribuutille arvoksi käyttötarkoituksen mukaan 1, 2, 3 tai 4.

Taulukko 5. Äänenvoimakkuusnappien ominaisuudet.

Napin nimi	Käyttötarkoitus	”oma_numero”-attribuutti
äänitehoste-	Äänitehosteet hiljemmalle	1
äänitehoste+	Äänitehosteet kovemmalle	2
musiikki-	Musiikki hiljemmalle	3
musiikki+	Musiikki kovemmalle	4

GameSaladissa äänentason asteikko on nollasta yhteen, yhden ollessa suurin voimakkuus. Näiden nappien ohjelmoinnin toteutin niin, että äänentaso muuttuu

0,2 yksikköä, kun sormi on nostetaan niiden päältä. Oli tarpeellista tehdä myöskin hahmotyyppi, josta ilmenisi esimerkiksi musiikin ja äänitehosteiden äänenvoimakkuus. Toteutin tämän niin, että tein osoittimen, joka liikkuu x-akselilla äänenvoimakkuuden tason mukaisesti. Käytin tämän hahmotyypin ohjelmointiin Constrain Attribute -attribuuttia, joka päivittää valitun attribuutin tilaa jatkuvasti. iPadille suunniteltaessa näytön x-akselin keskikohta GameSaladissa on 512 pikselin kohdalla. Ohjelmoin osoittimen liikkumaan 100 pikseliä keskikohdan molemmin puolin, jolloin osoitin on 412 pikselin kohdassa äänenvoimakkuuden ollessa nollassa ja 612 pikselin kohdassa äänenvoimakkuuden ollessa suurimmalla tasolla. Oli myös tarpeen tehdä hahmotyyppi, jonka tehtävänä oli näyttää äänenvoimakkuuden asteikko. Liitin näiden grafiikkaan myös selitetekstit ”ÄÄNITEHOSTEET” ja ”MUSIIKKI” (kuva 22). Valittu musiikki ja äänitehosteet tuodaan GameSaladin kirjastoon, jonka jälkeen ne ovat käytettävissä Play Sound- ja Play Music -toiminnoilla. Ei ole väliä mihin hahmotyyppiin Play Music -toiminto sijoitetaan, kunhan kyseinen hahmotyyppi on viety Scene-ikkunaan.

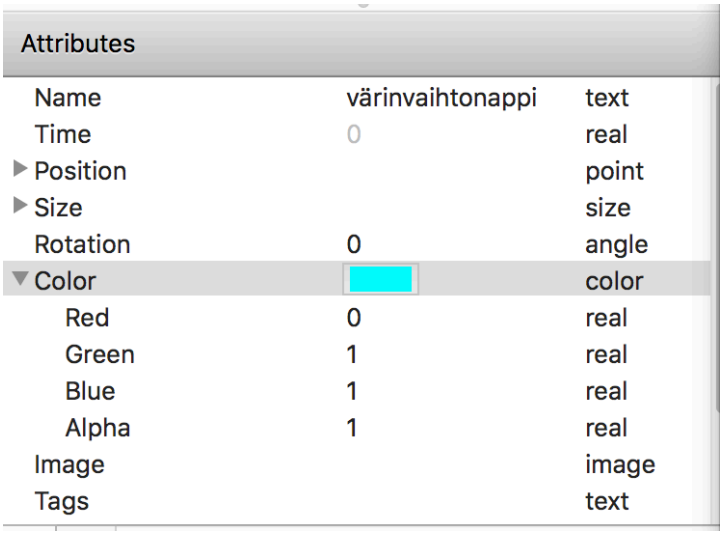


Kuva 22. Äänitehosteiden ja musiikin säätönapit.

5.5.4 Kenttien hahmotyyppien suunnittelu

Itse pelin idea on siis se, että pelaaja valitsee haluamansa värin, jolla sitten voidaan värittää pelissä esiintyviä hahmoja. Näitä hahmoja voidaan myös liikutella haluttuun paikkaan asettamalla hahmonsiirtotila päälle.

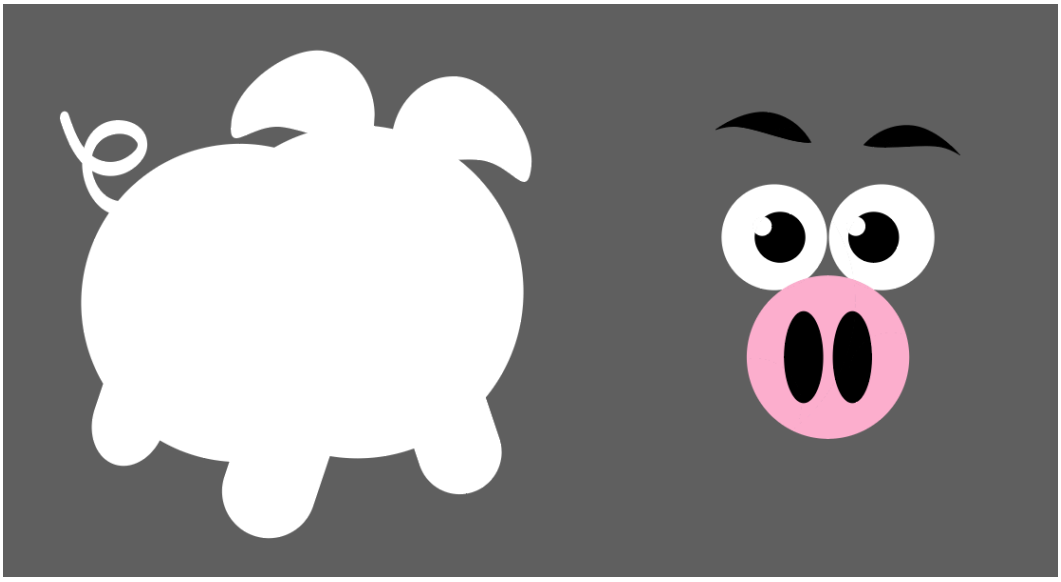
Aloitin värinvalintanappien ohjelmoinnin siten, että tein värejä varten kolme Game-attribuuttia, punaiselle, vihreälle ja siniselle omansa. Nappien toiminnan ohjelmoin siten, että kun värinvalintanappia kosketetaan, muuttaa se väreille luotujen Game-attribuuttien arvot omien väriarvojensa mukaisiksi (kuva 23).



Kuva 23. Hahmotyypin väriarvojen muokkaaminen. Käytetty GameSalad Inc:n luvalla 27.5.2019.

Seuraavaksi ohjelmoin pelin väritettävät hahmot. Näiden toimintatapa on käänteinen verrattuna värinvalintanappeihin, eli kun hahmoa kosketetaan, muuttuu sen väriarvot aiemmin luotujen Game-attribuuttien mukaisiksi. Suunnittelin näiden hahmotyyppien grafiikan puhtaan valkoiseksi, että värit toistuisivat niissä halutusti. Ulkonäöllisistä syistä johtuen halusin näihin hahmoihin joitain yksityiskohtia, kuten eläimille esimerkiksi silmät, joiden väri ei muuttuisi. Tämän toteutin tekemällä jokaiselle pelin hahmolle kaksi päällekkäin asetettavaa hahmotyyppiä, joista taaimmainen on väritettävä ja edessä oleva sisältää edellä mainittuja graafisia yksityiskohtia (kuva 24). Näiden hahmojen siirtelyn toteutin Constrain Attribute -toiminnolla, jonka avulla molemmat

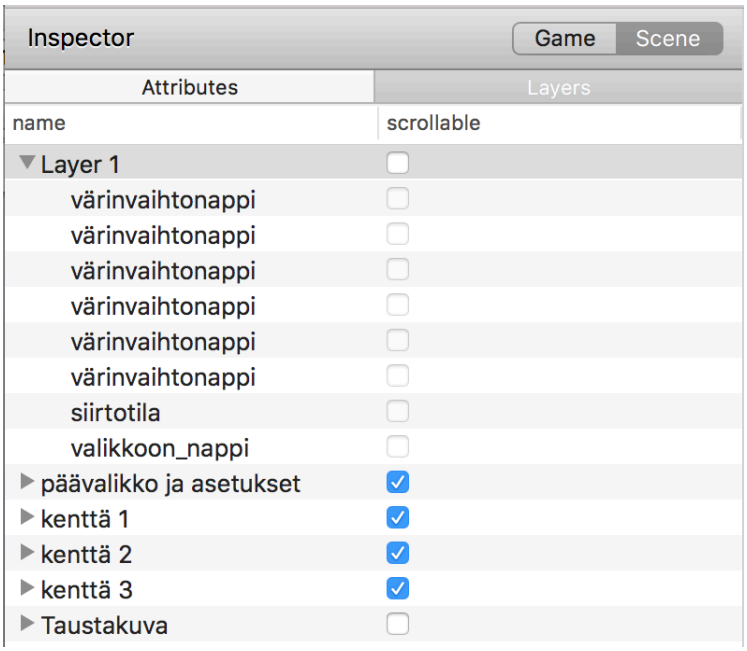
hahmotyypit seuraavat kosketusta laitteen näytöllä. Koska halusin, että vain yhtä hahmoa voisi liikutella kerrallaan, tein näille hahmoille oma_numero-nimisen Self-attribuutin ja lisäksi siirto_varattu_numerolle-nimisen Game-attribuutin. Kun hahmoa kosketetaan siirtotilan ollessa päällä, varataan siirtotila vain kyseisen hahmon käyttöön. Koska pelin hahmoja voi siirtää näytöllä vapaasti, oli vaarana se, että hahmo siirrettäisiin osittain viereisen kentän päälle. Eli esimerkiksi kulkuneuvoja sisältävän kentän reunaan voisi tulla näkyviin eläinhahmoja. Keksin tähän kaksi vaihtoehtoista ratkaisua: Hahmojen läpinäkyvyyden asettaminen pois päältä, kun pelaaja pelaisi toista kenttää. Toinen vaihtoehto olisi ollut siirtää kenttiä x-akselilla kauemmaksi toisistaan, jolloin hahmot eivät ulottuisi missään tapauksessa viereisten kenttien päälle. Käytin näissä hahmoissa ensimmäistä vaihtoehtoa. Toteutin sen siten, että tein näille hahmoille oma_kenttä-nimisen Integer-attribuutin, jonka arvon asetin hahmojen kenttää vastaavaksi, eli esimerkiksi kulkuneuvojen kohdalla arvoksi tuli 1. Nyt pystyin ohjelmoimaan hahmojen läpinäkyvyyden pois päältä, jos niiden oma_kenttä-attribuutti ei ollut arvoltaan sama kuin aiemmin luodun aktiivinen_kenttä-attribuutin. Seuraavaksi tein napin, jolla voi asettaa hahmojen siirtämisen mahdollistama tila päälle muuttamalla siirtotila-atribuutin arvoa.



Kuva 24. Väritettävä hahmotyyppi vasemmalla, yksityiskohtia sisältävä hahmotyyppi oikealla.

Pelissä on näiden lisäksi kolme hahmotyyppiä, joiden ainoa tehtävä on kuvien esittäminen. Näitä olivat Asetukset-näkymässä oleva hahmotyyppi, jossa kerron esimerkiksi pelin taustamusiikin tekijän vaatimat tiedot. Tämän lisäksi Päävalikon ylälaidassa on pelin nimeä esittävät hahmotyypit. Ohjelmoin näihin myös hieman liikettä, eli ne kääntyilevät edestakaisin näytöllä.

Kun kaikki hahmotyypit on saatu siirrettyä oikeille paikoilleen Scene-ikkunassa, kannattaa vielä tarkistaa, että ne ovat keskenään oikeassa järjestyksessä ja suunnitelmien mukaisilla tasoilla Layers-valikossa (kuva 25).



Kuva 25. Esimerkkipelin tasot (Layers). Käytetty GameSalad Inc:n luvalla 27.5.2019.

5.6 Pelin testaaminen ja valmistelu sovelluskauppoja varten

Testaaminen on hyvin tärkeä osa pelien suunnittelua. Sen avulla pyritään löytämään pelistä niin pienet kuin isommatkin suunnittelussa tapahtuneet virheet, eli niin sanotut bugit. Testausta voidaan tehdä heti, kun pelistä on saatu tehtyä ensimmäinen pelattava versio. Pelin kehittämisen kannalta on erittäin

tärkeää, että suunnittelijat voivat luottaa pelistä saatuun palautteeseen. Testauksen aikana suunnittelijan tulisi malttaa olla neuvomatta testaajaa ja tyytyä tarkkailemaan testaajan pelaamista. Koska suunnittelija saattaa tulla sokeaksi pelin virheille, on sille hyvä saada ulkopuolista testausapua. (Rouse 2004, 483–492.) GameSaladissa pelin testaus voidaan toteuttaa nopeasti ohjelman sisäistä esikatseluominaisuutta hyödyntäen. Mikäli peliä halutaan testata suoraan mobiililaitteessa, onnistuu se sovelluskaupoista ladattavan GameSalad Viewer -sovelluksen avulla, jolloin pelin lataaminen siihen onnistuu langattoman verkon kautta. Kun peli on saatu lähelle valmista, voidaan peliä testata tarkemmin AdHoc-tyyppisesti suoraan mobiililaitteeseen ladattuna, ilman erillisiä sovelluksia. (GameSalad 2018b.)

Kun peli on saatu valmiiksi, halutaan se yleensä saattaa ihmisten saataville. GameSaladissa julkaisuprosessi alkaa niin, että klikataan Publish-nappia GameSaladin käyttöliittymän oikeassa yläkulmassa, jonka tehdyn pelin tiedosto latautuu GameSaladin serverille. Julkaisuprosessi jatkuu nyt aukeavassa nettiselaimessa, jossa valitaan esimerkiksi kohdelaite ja syötetään pelin nimi, versionumero ja muita oleellisia tietoa. Kun kaikki pelin tiedot on saatu syötettyä, voidaan luoda varsinainen pelitiedosto ja ladata se sitten koneelle. Julkaisuprosessin ajan tasalla oleva ohjeistus kannattaa aina tarkistaa GameSaladin kotisivuilta. On hyvä muistaa, että Applen App Storeen julkaisua varten täytyy hankkia erikseen Applen Developer-lisenssi, joka maksaa vuodessa 99 dollaria (Apple 2019).

Pelien jakelu tapahtuu järkevimmin käyttämällä hyväksi sovelluskauppoja, joista merkittävimmät ovat Applen App Store ja Googlen Play. Applen App Store on tarkoitettu Applen iOS -käyttöjärjestelmää käyttäville laitteille eli iPhonelle ja iPadille. Google Play puolestaan on olemassa Android-käyttöjärjestelmällä varustettuja laitteita varten. Suunnittelin tämän pelin erityisesti Applen iPadille, mutta pienin muutoksin sen voisi julkaista myös muille laitteille. Esimerkkipelin pohjapiirustus, vuokaaviot, ohjelmointi ja kuvakaappauksia löytyvät tämän opinnäytetyön liitteistä.

6 Pohdinta

Tein tätä opinnäytetyötä periaatteessa yli vuoden ajan, tosin melko epäsäännöllisesti. Olen tyytyväinen siihen, että pelistä tuli toimiva ja sain toteutettua kaikki siihen suunnitellut asiat. Myös peliprojektin dokumentointi onnistui mielestäni hyvin. Koska en ole taustaltani ohjelmoija, en voi olla varma kaikkien ohjelmointiratkaisujeni järkevyydestä ja varmasti löytyisikin vaihtoehtoisia ratkaisumalleja moneen tilanteeseen.

Onnistuin mielestäni kehittämään omia taitojani melko paljon, varsinkin projektin hallinnan ja erityisesti ajankäytön optimoinnin suhteen. Huolelliset etukäteen tehdyt suunnitelmat, kuten rautalankamalli, vuokaaviot ja pelin pohjapiirustus tehostivat ohjelmoinnin ja grafiikan tekemistä huomattavasti. Varsinkin GameSaladissa hahmotyyppien sijoittelu onnistui hyvin nopeasti, kun niiden tulevat sijainnit olivat tiedossa etukäteen. Niiden ansiosta pelin pariin oli helppo palata pidemmänkin tauon jälkeen. Koska pidin alkuperäisestä suunnitelmasta kiinni, enkä lisännyt peliin uusia toimintoja, pystyin toteuttamaan pelin ilman ylimääräisiä viivästyksiä. Myös esimerkiksi kuvatiedostoista tekemäni listat otan varmasti käyttöön myös tulevissa projekteissa, sillä ne helpottavat ja nopeuttavat päivitysprosessia. Etukäteen tehtyjen rajausten ja tarkkojen suunnitelmien avulla projektin keston pystyy ennustamaan melko tarkasti, jolloin realistisen aikataulun laatiminen on helpompaa.

Opinnäytetyön tutkimuskysymyksenä oli se, että miten mobiilipeliprojekti on mahdollista toteuttaa ilman koodin näkemistä, millaisia tietoja ja taitoja tekijällä tulisi olla ja millaisia ohjelmistoja pelin tekemiseen tarvitaan. Mielestäni opinnäytetyöni vastasi kysymykseen hyvin, sillä siinä kerrottiin vaihe vaiheelta pelin suunnittelun eri osa-alueet. Peliprojekti kannattaa aloittaa huolellisella suunnittelulla ja peliin tulevan sisällön rajaamisella. Alkuperäisessä suunnitelmassa kannattaa pysyä, mikäli projekti halutaan saada valmiiksi aikataulun mukaisesti. Ohjelmointi, grafiikan luominen ja äänten nauhoittaminen onnistuu aloittelijaltakin, kunhan käytetään ensin hieman aikaa ohjelmistojen käytön opetteluun.

Tämän opinnäytetyön yhteydessä syntyneen pelin jatkokehitys olisi mielestäni melko helppoa, sillä projektin aikana syntyi paljon uusia ideoita. Pelin hahmoja voisi animoida ja nauhoittaa niille lisää erilaisia ääniefektejä. Hahmot voisi myös jakaa pienempiin osiin, jolloin ne voisivat sisältää useampaa eri väriä. Tämän lisäksi niille voisi ohjelmoida partikkeliefektejä, eli hahmon värjäytyessä siitä lentäisi vaikkapa tähtiä ilmaan. Myös esimerkiksi muokattavat väripaletit voisi olla toimiva ratkaisu, eli pelaaja saisi itse valita käytössä olevat värit. Myös tallennusmahdollisuus voisi olla hyvä ominaisuus tämän tyyppiseen peliin, eli käyttäjä voisi tallentaa keskeneräiset väritykset ja jatkaa niiden parissa myöhemmin.

Mikäli pelin graafiseen suunnitteluun olisi ollut enemmän aikaa, olisin luultavasti lisännyt peliin enemmän hahmoja ja tehnyt myös väritettäviä taustakuvia jokaiseen kenttään. Tämä olisi toki vaikuttanut myös pelin ohjelmointiin, joten työtuntien määrä olisi kasvanut myös sitä kautta. Olisi ollut myös mielenkiintoista testata erilaisia graafisia ratkaisuja isomman testiryhmän avulla.

Ideoita mahdollisille jatkotutkimuskysymyksille voisivat olla esimerkiksi huomaako pelaaja eroa visuaalisen ohjelmoinnin ja perinteisen koodauksen avulla tehdyissä peleissä ja onko visuaalisen ohjelmoinnin kehittyminen pelialalle uhka vai mahdollisuus.

Lähteet

Apple. 2019. Choosing a Membership. https://developer.apple.com/support/compare-memberships/. 24.5.2019.

Bajarin, T. 2017. How Apple's iPhone Changed These 5 Industries. Time. http://time.com/4832599/iphone-anniversary-industry-change/. 20.6.2018.

Barton, M. & Loguidice, B. 2008. The History of Spacewar!: The Best Waste of Time in the History of the Universe. Gamasutra. https://www.gamasutra.com/view/feature/132438/the_history_of_spacewar_the_best_.php. 18.5.2019.

Barton, M. & Loguidice, B. 2009. The History Of Pong: Avoid Missing Game to Start Industry. Gamasutra. https://www.gamasutra.com/view/feature/132293/the_history_of_pong_avoid_missing_.php?page=2. 18.5.2019.

Creative Commons. 2019. https://creativecommons.fi/lisenssien-kayttoohje/cc-lisenssin-merkitsemisohjeet/. 25.5.2019.

Crecente, B. 2015. Time killers: The strange history of wrist gaming. Polygon. https://www.polygon.com/a/smartwatch-history-guide-evolution. 4.6.2019.

Construct. 2019. https://www.construct.net. 25.5.2019.

Dehouck, R. 2015. The Maturity of visual programming. Craft ai. https://www.craft.ai/blog/the-maturity-of-visual-programming/. 27.5.2019.

Esposito, E. 2018. Your guide to using flat design. https://www.invisionapp.com/inside-design/your-guide-to-using-flat-design/ 20.5.2019.

Gameberry. 2019. Game & Watch. http://www.gameberry.net/laitteet/gameandwatch.php. 5.6.2019.

GameSalad. 2018a. About GameSalad. https://gamesalad.com/about/. 20.6.2018.

GameSalad. 2018b. GS Mac Cookbook. http://help.gamesalad.com/gamesalad-cookbook/. 6.9.2018.

GameSalad. 2019. Buy Now. https://get.gamesalad.com/upgrade. 25.5.2019.

Google Fonts. 2019. Patrick Hand SC. https://fonts.google.com/specimen/Patrick+Hand+SC. 25.5.2019.

Graydon, D. 2013. Angry Birds maailmaa rakentamassa: Suuri Angry Birds -kirja. Helsinki: Readme.fi Oy.

Herman, L. 2008a. Early Home Video Game Systems. Teoksessa Wolf, M. (toim.) The Video Game Explosion: A History from PONG to PlayStation® and Beyond. Englanti: Greenwood Press, 54–58.

Herman, L. 2008b. A New Generation of Home Video Game Systems. Teoksessa Wolf, M. (toim.) The Video Game Explosion: A History from PONG to PlayStation® and Beyond. Englanti: Greenwood Press, 115–118.

Herman, L. 2008c. Handheld Video Game Systems. Teoksessa Wolf, M. (toim.) The Video Game Explosion: A History from PONG to PlayStation® and Beyond. Englanti: Greenwood Press, 143–149.

Huizinga, J. 1967. Leikkivä ihminen. Helsinki: Werner Söderström Osakeyhtiö.

Incompetech.com. 2019. Music FAQ. https://incompetech.com/music/royalty-free/faq.html. 19.5.2019.

Kansallinen audiovisuaalinen instituutti. 2019. Ahdistus. https://kavi.fi/fi/meku/ikarajat/sisaltosymbolit/ahdistus. 2.6.2019.
Karjalainen, T., Lehtonen, M. & Niipola, J. 2014. The Playing Finn: Stories on successful game development and music export. Helsinki: Talentum.
Keane, J. 2015. Before Mobile Gaming Exploded, There Was the Nokia N-Gage. Paste. https://www.pastemagazine.com/articles/2015/12/before-the-mobile-gaming-explosion-there-was-the-n.html. 18.5.2019.
Land, F. 2014. Värit: katso ja tunnustele. Helsinki: Egmont Kustannus Oy Ab.
Lappalainen, E. 2015. Pelien valtakunta. Jyväskylä: Atena Kustannus Oy.
Laukka, M. Kuvakirjan vuosikymmenet. Teoksessa Rättyä, K. & Raussi, R. (toim.) Tutkiva katse kuvakirjaan. Helsinki: BTJ Kirjastopalvelu Oy, 27–50.
Manninen, T. 2007. Pelisuunnittelijan käsikirja – ideasta eteenpäin. Oulu: Kustannus Oy Rajalla.
Neogames. 2018. Suomen pelialan raportti 2017. https://www.neogames.fi/2017-finnish-games-industry-in-2017/. 12.9.2018.
Nirvi, N. 2009. Videopeli–peli. Pelit-lehti. https://www.pelit.fi/artikkelit/videopeli-peli/. 23.5.2019.
Novak, J. 2014. The Official GameSalad Guide to Game Development. USA: Cengage Learning.
Riikonen, P. 2016. Matopeli luikerteli kansan suosikiksi 1990-luvun lopussa. Yle. https://yle.fi/aihe/artikkeli/2016/08/11/matopeli-luikerteli-kansan-suosikiksi-1990-luvun-lopussa. 20.5.2019.
Rouse, R. 2004. Game Design: Theory and Practice (2nd Edition). USA: Wordware Publishing, Inc.
Räty, V. 1999. Pelien leikki. Espoo: Taideteollinen korkeakoulu.
Santaharju, T. 2015. Rovio päätti yt-neuvottelut – potkut 198:lle. Yle. https://yle.fi/uutiset/3-8397775. 10.5.2019.
Sega Retro. 2019. Blockade. https://segaretro.org/Blockade. 17.5.2019.
Simões, T. 2015. Visual Programming Is Unbelievable... Here’s Why We Don't Believe In It. Outsystems. https://www.outsystems.com/blog/visual-programming-is-unbelievable.html. 27.5.2019.
Vuorela, V. 2007. Pelintekijän käsikirja. Helsinki: BTJ Finland Oy.
WHO. 2019. Guidelines on physical activity, sedentary behaviour and sleep for children under 5 years of age. https://apps.who.int/iris/bitstream/handle/10665/311664/9789241550536-eng.pdf. 2.6.2019.
Zagalo, N. & Branco, P. 2015. The Creative Revolution That Is Changing the World. Teoksessa Zagalo, N. & Branco, P. (toim.) Creativity in the Digital Age. Englanti: Springler-Verlag London Ltd., 3-16.

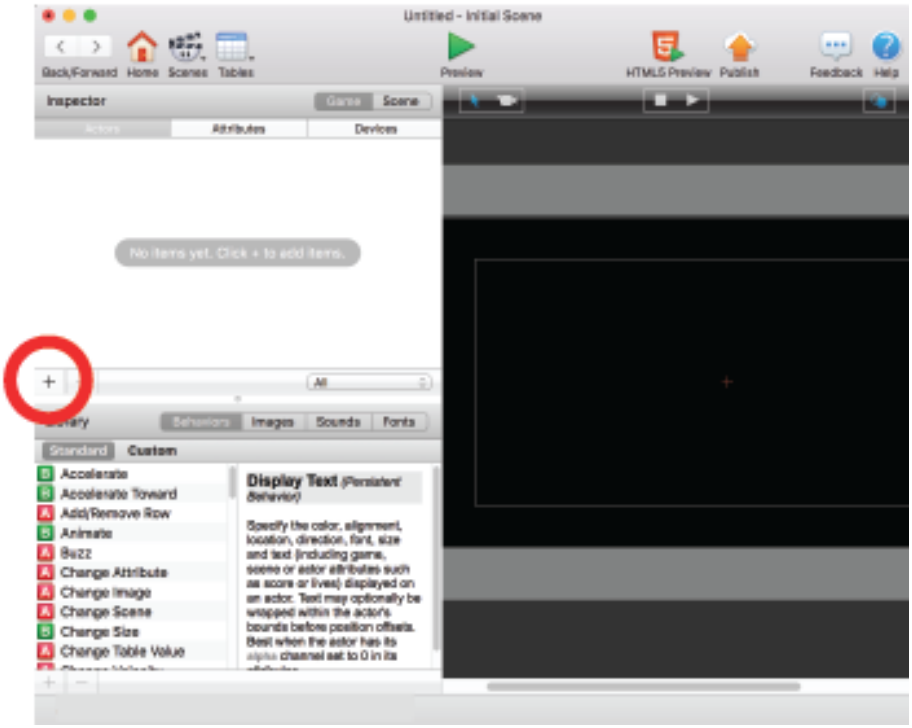
Ohjelmointiesimerkki ”Hello world”

GameSaladin käyttäminen kannattaa mielestäni aloittaa niin, että lähdetään liikkeelle yksinkertaisten harjoitusten ja kokeilujen kautta. Kokemuksen ja ymmärryksen lisääntyessä voidaan siirtyä monimutkaisempaan suunnitteluun. Perinteinen ”Hello world!” -tekstin näytölle tulostava sovellus on hyvä keino päästä sisälle GameSaladin ohjelmointilogiikkaan:

1. Käynnistetään GameSalad ja luodaan uusi projekti.

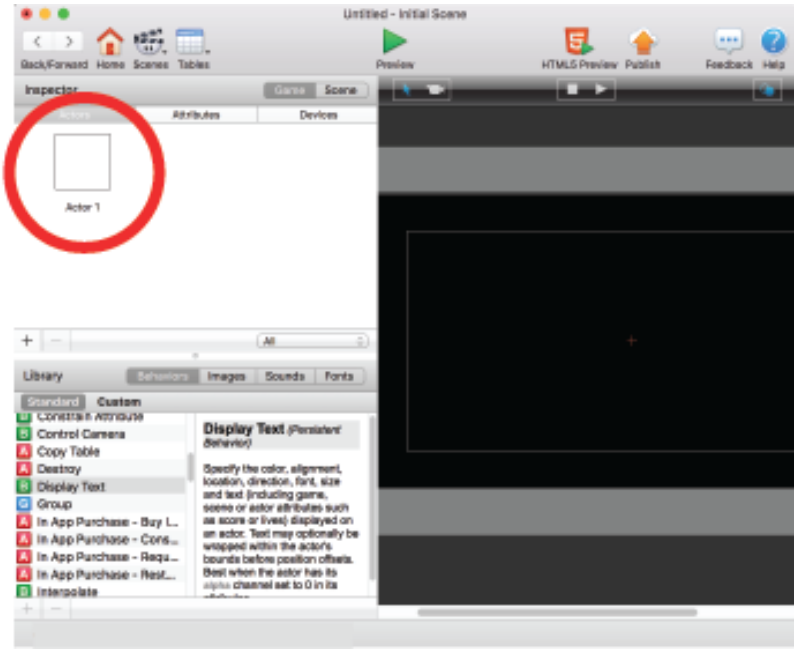
2. Siirrytään Scenes –välilehdelle, josta jatketaan Scene-näkymään tuplaklikkaamalla Initial Scene -kuvaketta.

3. Luodaan uusi hahmotyyppi (kuva 1).



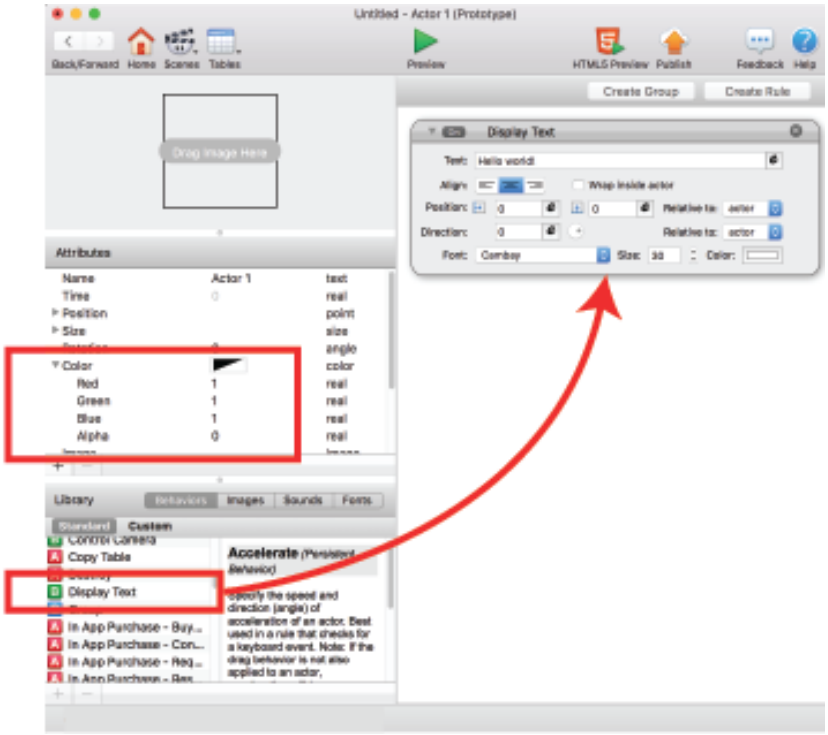
Kuva 1. Uuden hahmotyypin luominen. Käytetty GameSalad Inc:n luvalla 27.5.2019.

4. Avataan luotu hahmotyyppi tuplaklikkaamalla sitä.



Kuva 2. Hahmotyypin avaaminen. Käytetty GameSalad Inc:n luvalla 27.5.2019.

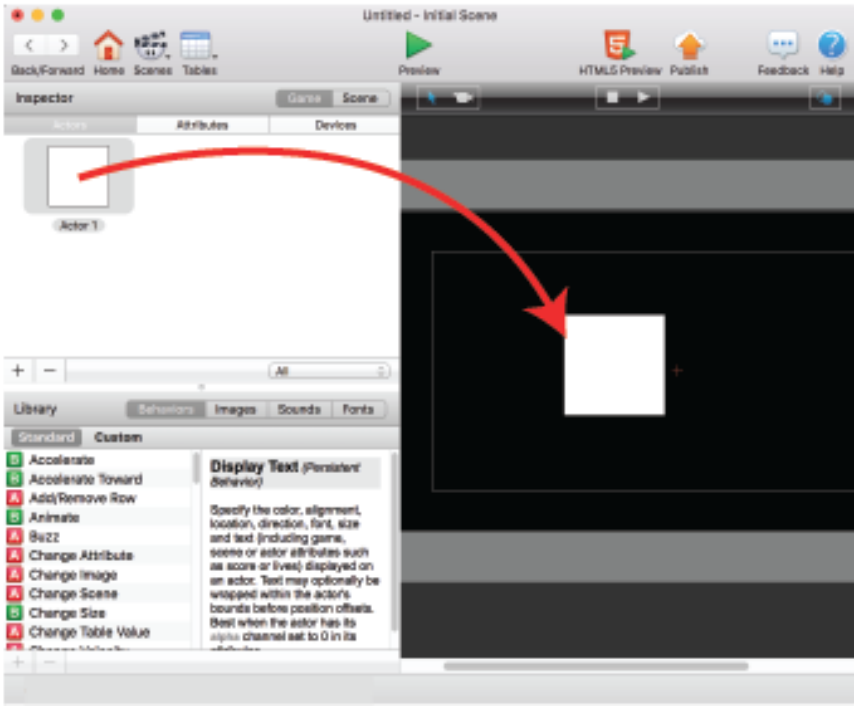
5. Etsitään auenneen hahmotyypin Behaviors-valikosta Display Text -toiminto ja raahataan se oikealla olevaan valkoiseen ohjelmointitilaan (kuva 3). Lisäksi muutetaan hahmotyypin läpinäkyvyys nollaksi.



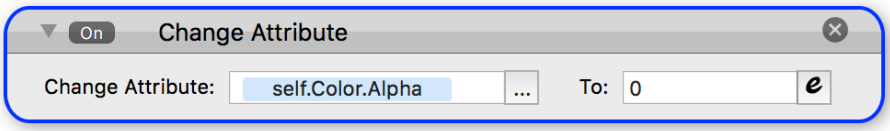
Kuva 3. Ohjelmointi ja attribuutin arvon muokkaaminen. Käytetty GameSalad Inc:n luvalla 27.5.2019.

6. Siirrytään takaisin scene-näkymään vasemmassa yläkulmassa olevan Back-nuolinäppäimen avulla.

7. Tämän jälkeen voidaan raahata ohjelmoitu hahmotyyppi Scene-ikkunaan (kuva 4). Huomaa, että hahmotyypin läpinäkyvyys on nyt nolla, eli sitä ei näy myöskään nyt Scene-näkymässä. Toinen keino läpinäkyvyyden poistamiseen olisi lisätä hahmotyyppiin Change Attribute -toiminto (kuva 5), jolloin se muuttuisi näkymättömäksi vasta pelin käynnistyessä.

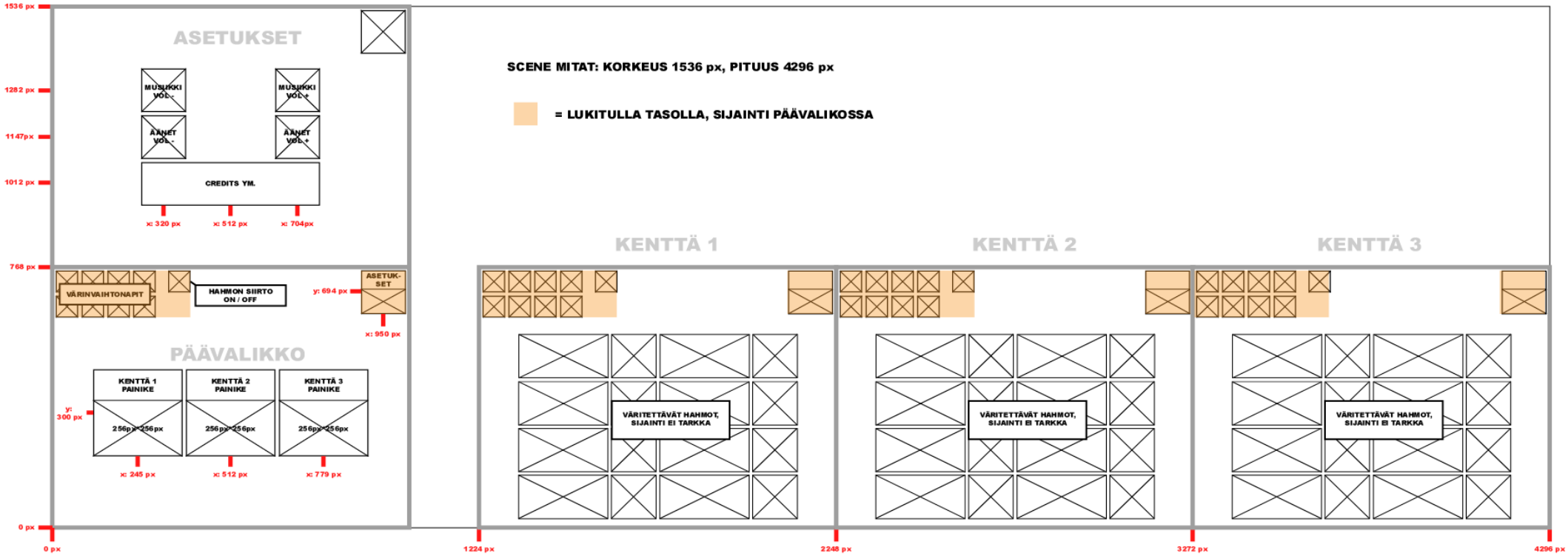


Kuva 4. Hahmotyypin raahaaminen Sceneen. Käytetty GameSalad Inc:n luvalla 27.5.2019.

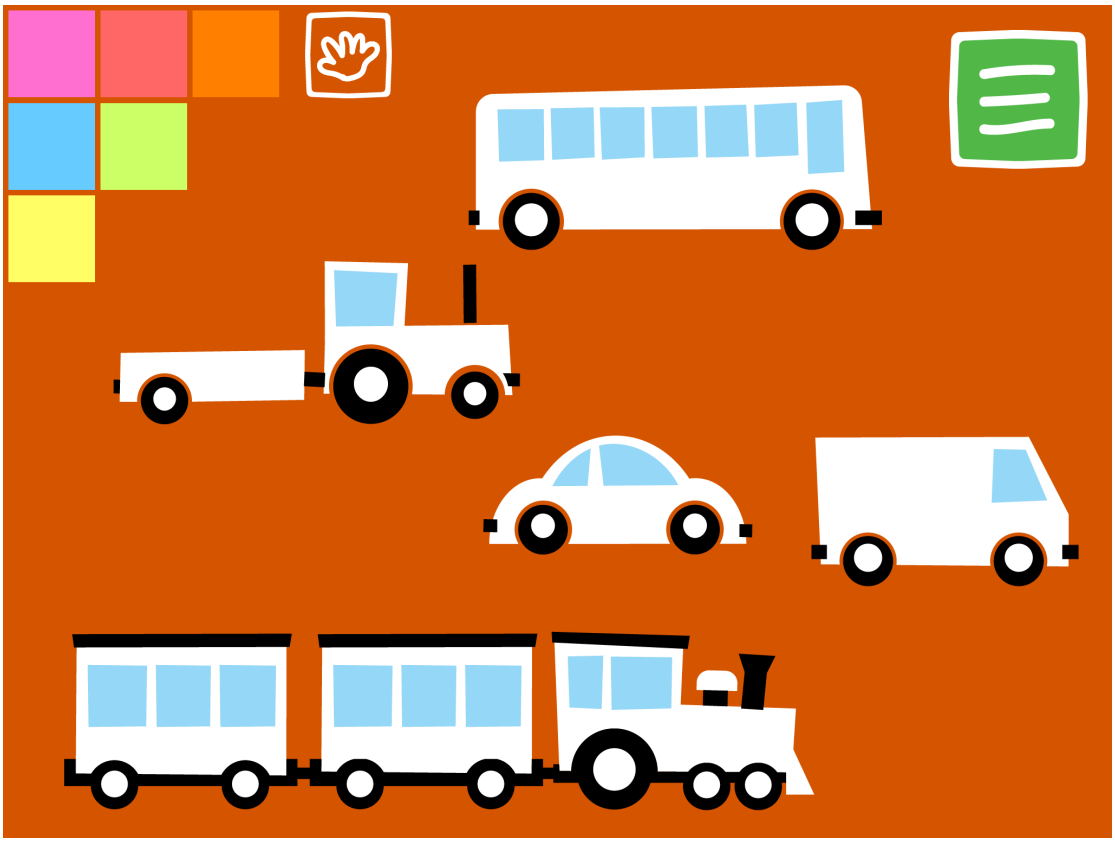


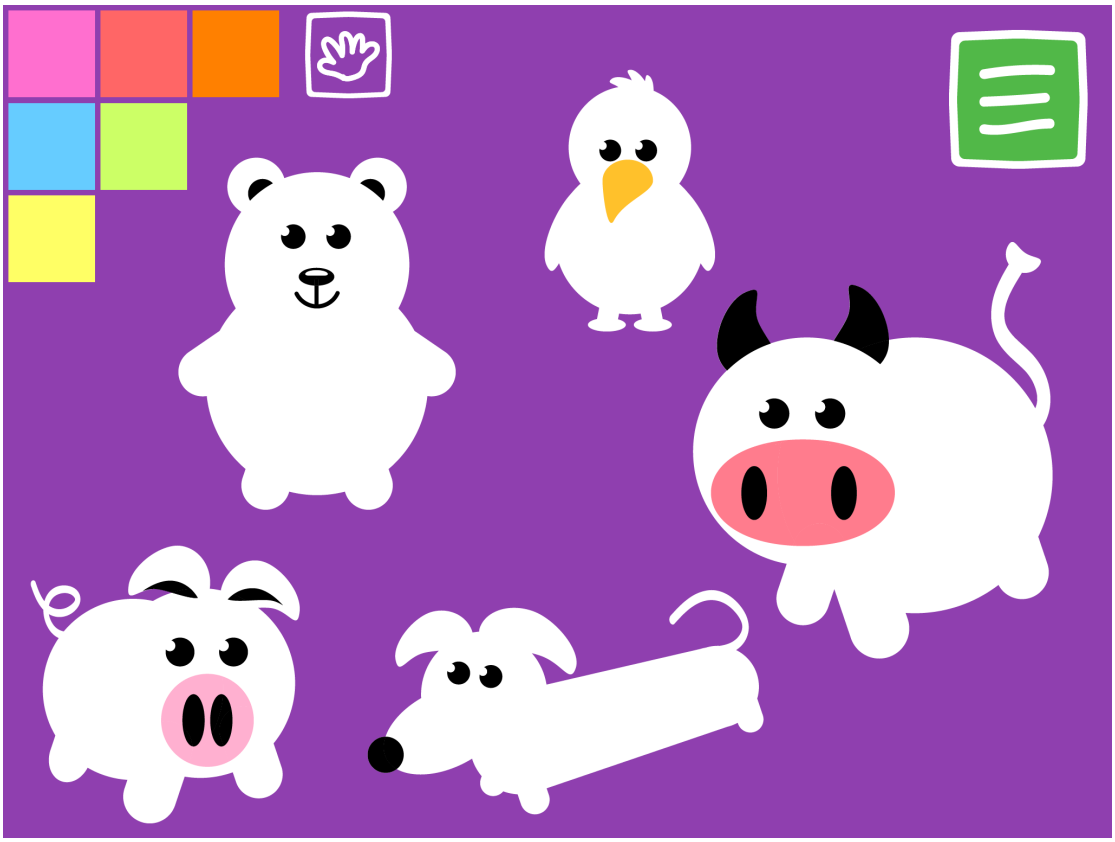
Kuva 5. Hahmotyypin läpinäkyvyyden muuttaminen Change Attribute -toiminnon avulla. Käytetty GameSalad Inc:n luvalla 27.5.2019.

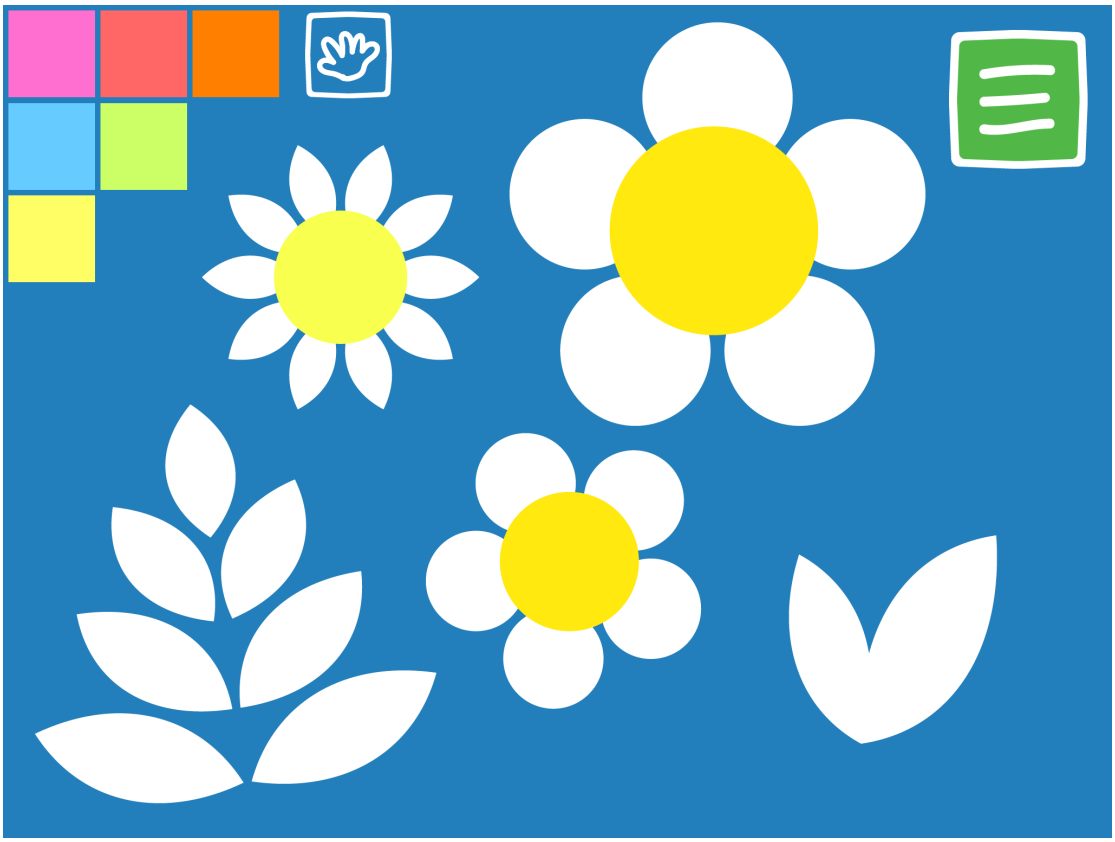
8. Preview-nappia painamalla ruudulle pitäisi ilmestyä pelin esikatselunäkymä, jossa lukee mustalla taustalla valkoisella tekstillä ”Hello world!”.


1536 px
1282 px
1147px
1012 px
768 px
0 px
0 px
1224 px
2248 px
3272 px
4296 px
ASETUKSET
MUSIIKKI VOL -
MUSIIKKI VOL +
ÄÄNET VOL -
ÄÄNET VOL +
CREDITS YM.
x: 320 px
x: 512 px
x: 704px
SCENE MITAT: KORKEUS 1536 px, PITUUS 4296 px
= LUKITULLA TASOLLA, SIJAINTI PÄÄVALIKOSSA
KENTTÄ 1
KENTTÄ 2
KENTTÄ 3
VÄRINVAIHTONAPIT
HAHMON SIIRTO ON / OFF
y: 694 px
ASETUK-SET
x: 950 px
PÄÄVALIKKO
y: 300 px
KENTTÄ 1 PAINIKE
KENTTÄ 2 PAINIKE
KENTTÄ 3 PAINIKE
256px 256px
x: 245 px
x: 512 px
x: 779 px
VÄRITETTÄVÄT HAHMOT, SIJAINTI EI TARKKA


TAAPERON
VÄRITYSKIRJA
KULKUPELIT
ELÄIMET
KASVIT



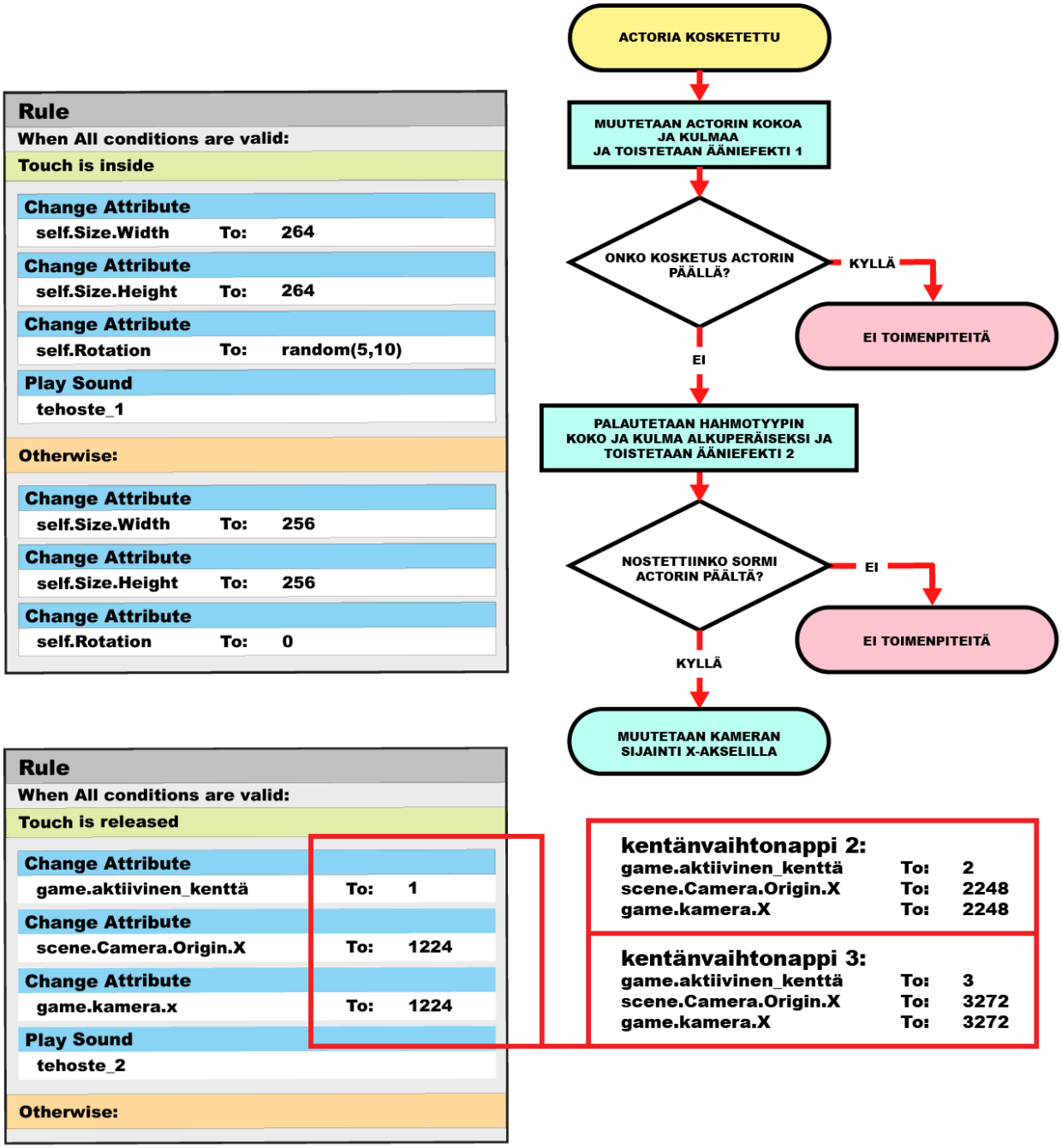




LUODUT GAME-ATTRIBUUTIT:

siirtotila	boolean
siirto_varattu_numerolle	boolean
punainen	real
vihreä	real
sininen	real
kamera.x	0
aktiivinen_kenttä	0

KENTÄNVAIHTONAPIT 1, 2 ja 3



Rule
When All conditions are valid:
Attribute game.kamera.X > 0

Rule
When All conditions are valid:
Touch is inside

Change Attribute
game.punainen To: self.color.red

Change Attribute
game.vihreä To: self.color.green

Change Attribute
game.sininen To: self.color.blue

Change Attribute
self.Size.Width To: 100

Change Attribute
self.Size.Height To: 100

Otherwise:

Rule
When All conditions are valid:
Attribute self.Color.Red = game.punainen
Attribute self.Color.Green = game.vihreä
Attribute self.Color.Blue = game.sininen

Change Attribute
self.Size.Width To: 100

Change Attribute
self.Size.Height To: 100

Otherwise:

Change Attribute
self.Size.Width To: 80

Change Attribute
self.Size.Height To: 80

Rule
When All conditions are valid:
Attribute game.siirtotila = true

Change Attribute
self.Color.Alpha To: 0

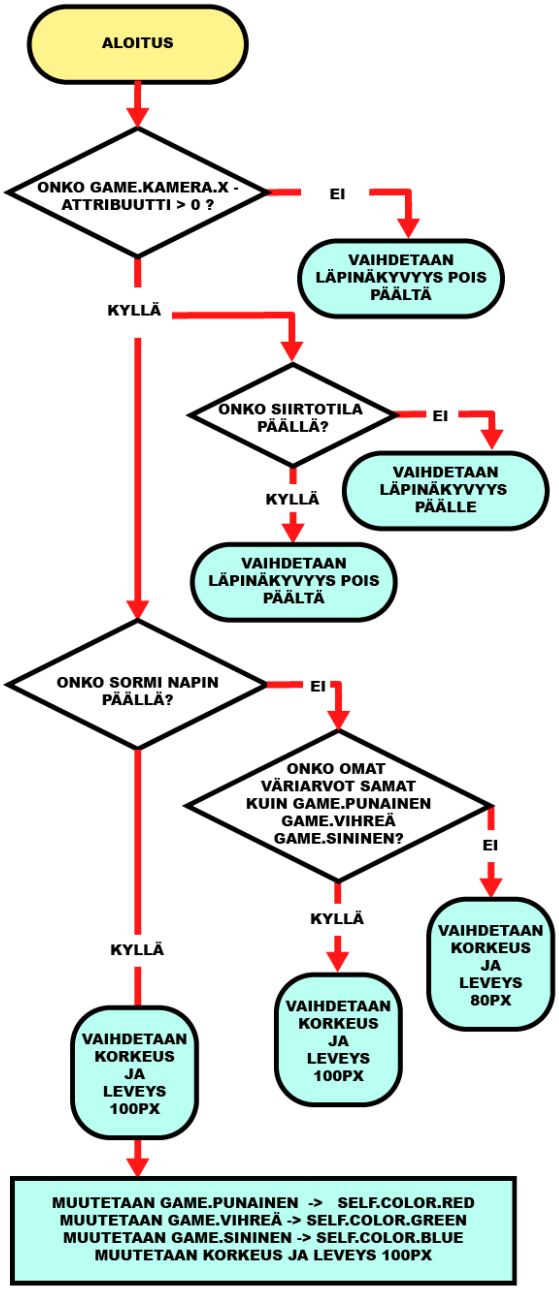
Otherwise:

Change Attribute
self.Color.Alpha To: 1

Otherwise:

Change Attribute
self.Color.Alpha To: 0

VÄRINVALINTANAPIT



PELIN NIMI/LOGO KÄÄNTYILEMÄÄN EDESTAKAISIN PÄÄVALIKKOON

Rule
When All conditions are valid:
Attribute self.Rotation = 0

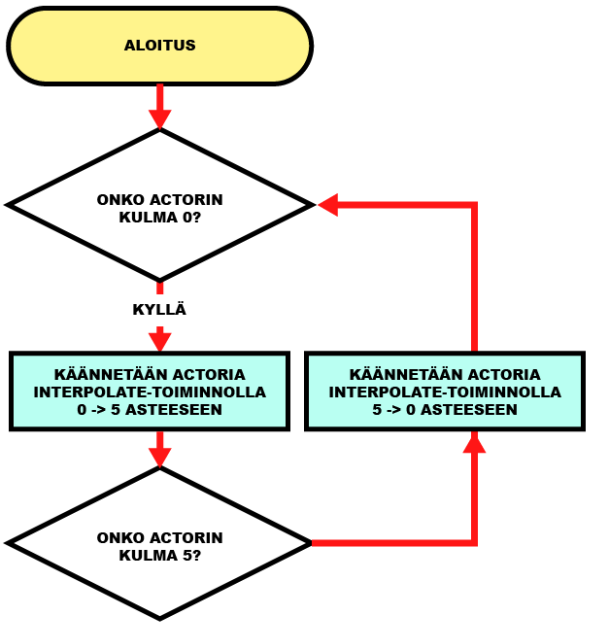
Interpolate			ease in/out
self.Rotation	To: 5	Duration:	3s

Otherwise:

Rule
When All conditions are valid:
Attribute self.Rotation = 5

Interpolate			ease in/out
self.Rotation	To: 0	Duration:	3s

Otherwise:



Rule
When All conditions are valid:
Attribute game.kamera.X > 0

Rule
When All conditions are valid:
Touch is inside

Change Attribute
self.Size.Width To: 136

Change Attribute
self.Size.Height To: 136

Change Attribute
self.Rotation To: random(5,10)

Otherwise:

Change Attribute
self.Size.Width To: 128

Change Attribute
self.Size.Height To: 128

Change Attribute
self.Rotation To: 0

Rule
When All conditions are valid:
Touch is released

Rule
When All conditions are valid:
Attribute game.siirtotila is false

Change Attribute
game.siirtotila To: 1

Otherwise:

Change Attribute
game.siirtotila To: 0

Otherwise:

Rule
When All conditions are valid:
Mouse button is up

Change Attribute
Attribute: game.siirto_varattu_numerolle To: 0

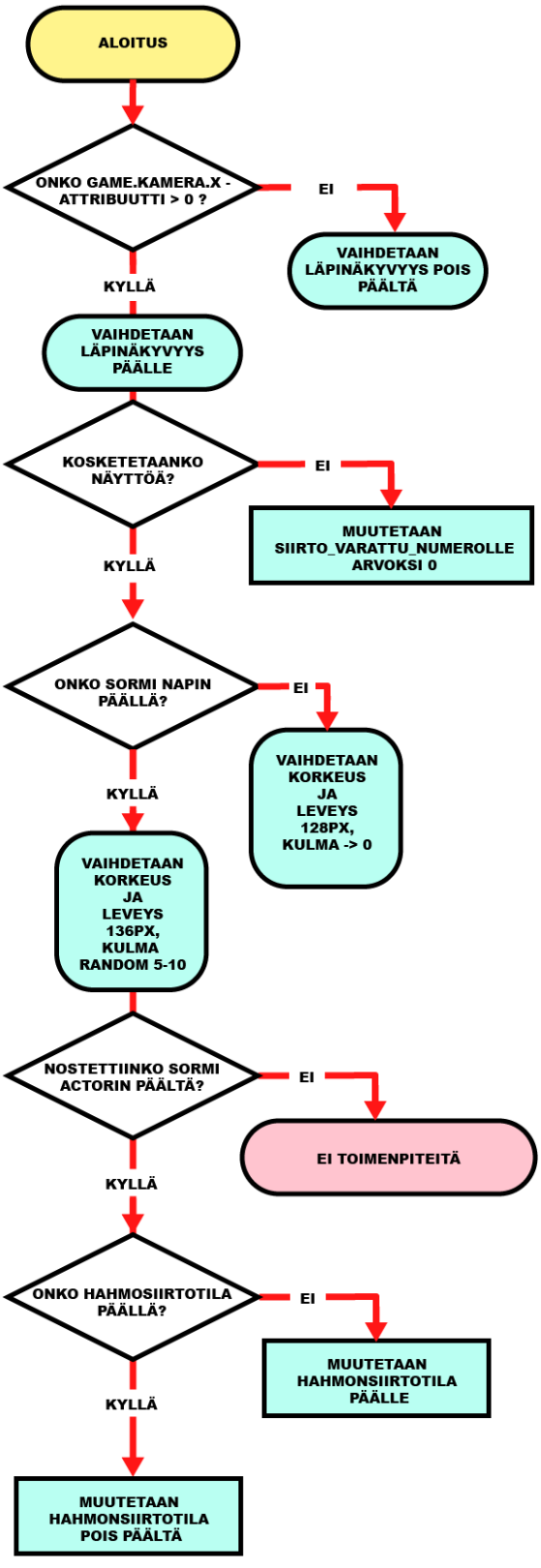
Otherwise:

Change Attribute
self.Color.Alpha To: 1

Otherwise:

Change Attribute
self.Color.Alpha To: 0

SIIRTOTILANAPPI



Rule
When All conditions are valid:
Attribute game.aktiivinen_kenttä = self.oma_kenttä

Rule
When All conditions are valid:
Attribute siirtotila is true

Rule
When Any conditions are valid:
Attribute game.siirto_varattu_numerolle = 0
Attribute game.siirto_varattu_numerolle = self.oma_numero

Rule
When All conditions are valid:
Touch is inside

Change Attribute
game.siirto_varattu_numerolle To: self.oma_numero

Constrain Attribute
self.Position.X To: Game.Mouse.Position.X+game.kamera.x

Constrain Attribute
self.Position.Y To: Game.Mouse.Position.Y

Otherwise:

Otherwise:

Otherwise:

Rule
When Any conditions are valid:
Attribute game.siirto_varattu_numerolle = 0
Attribute game.siirto_varattu_numerolle = self.oma_numero

Change Attribute
self.Color.Red game.punainen

Change Attribute
self.Color.Green game.vihreä

Change Attribute
self.Color.Blue game.sininen

Otherwise:

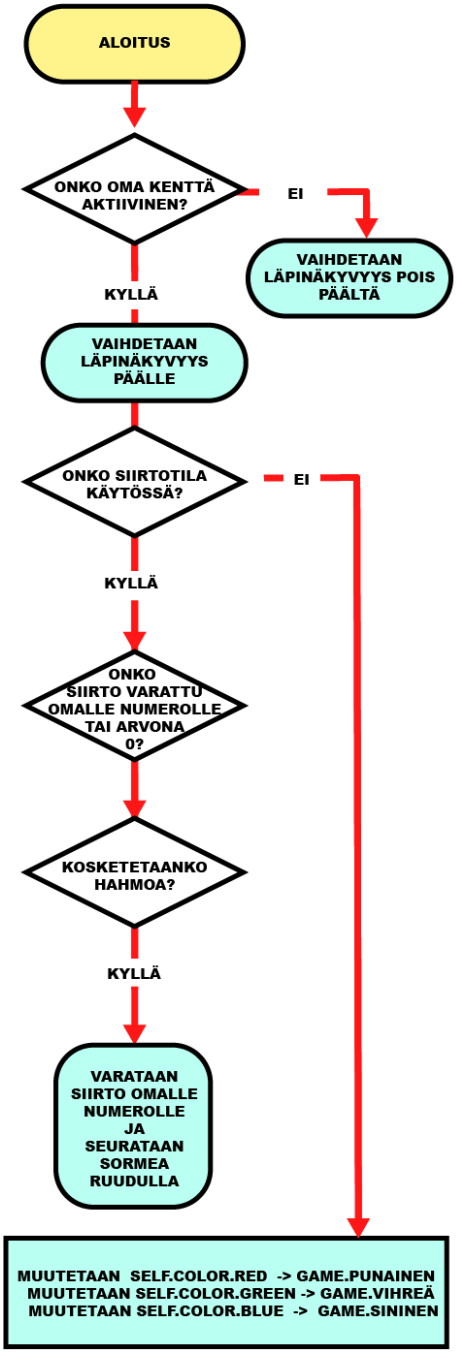
Change Attribute
self.Color.Alpha To: 1

Otherwise:

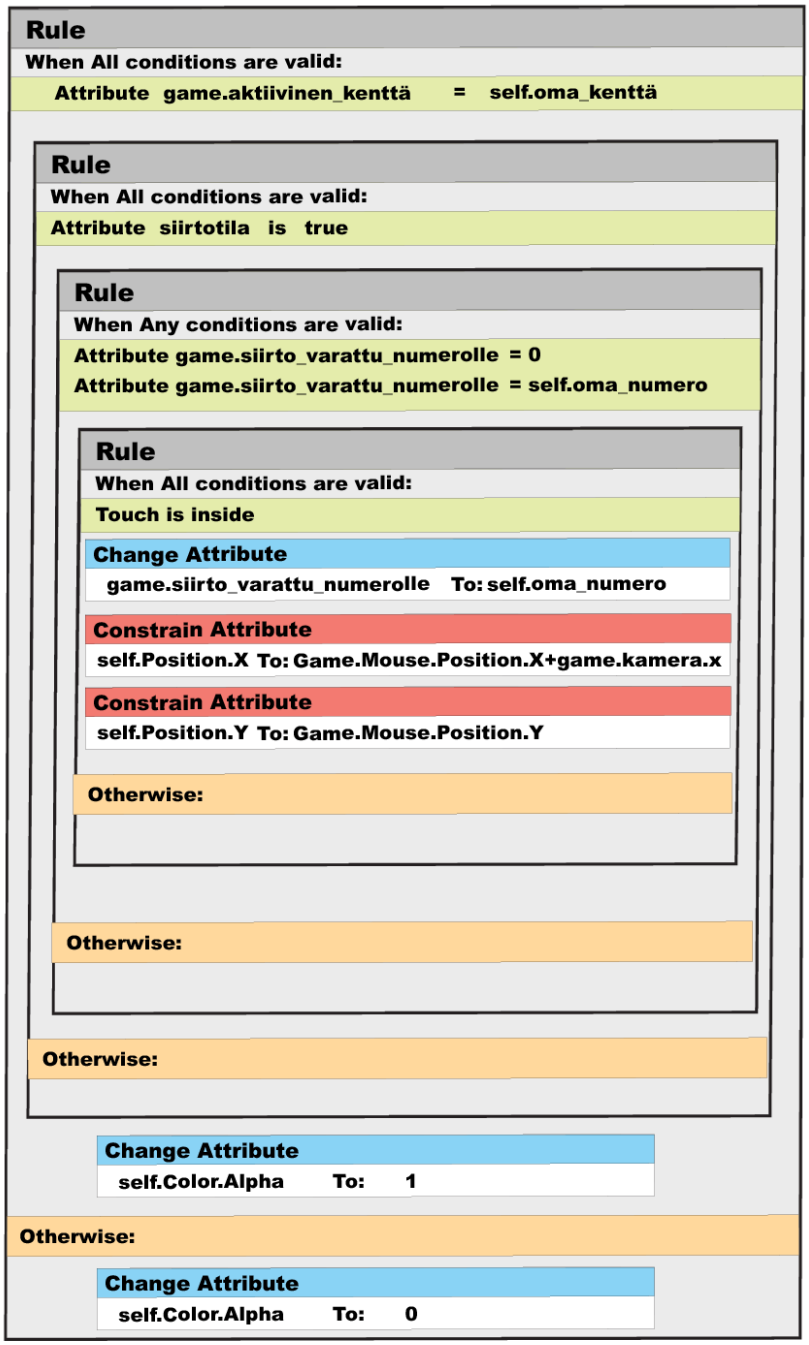
Change Attribute
self.Color.Alpha To: 0

VÄRITETTÄVÄ HAHMO

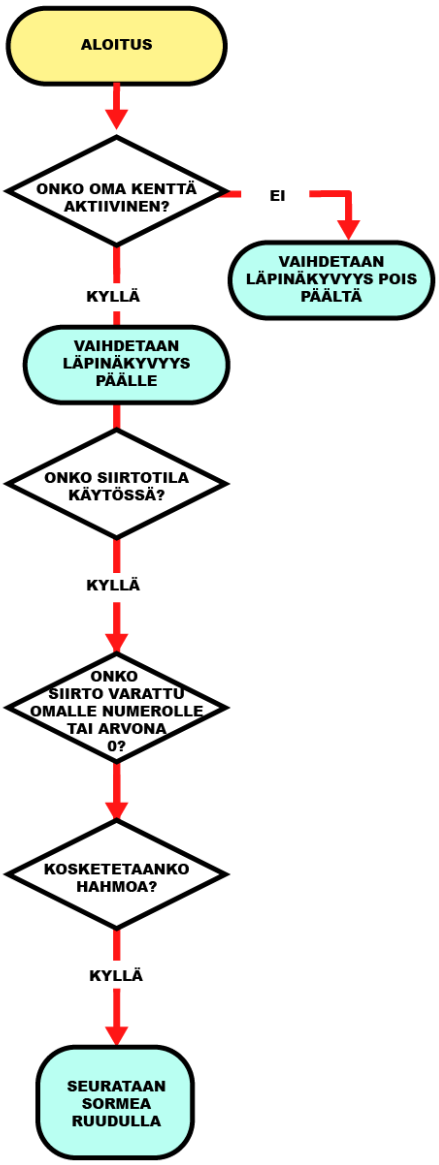
Lisätyt attribuutit:
self.oma_numero
self.oma_kenttä



HAHMOJEN YKSITYISKOHTIA SISÄLTÄVÄ HAHMOTYYPPI



Lisätyt attribuutit:
self.oma_numero
self.oma_kenttä



Rule
When All conditions are valid:
Attribute self.oma.numero = 1

Rule
When All conditions are valid:
Touch is released

Change Attribute
game.Audio.Sound.Volume To: game.Audio.Sound.Volume -0.2

Otherwise:

Otherwise:

Rule
When All conditions are valid:
Attribute self.oma.numero = 2

Rule
When All conditions are valid:
Touch is released

Change Attribute
game.Audio.Sound.Volume To: game.Audio.Sound.Volume +0.2

Otherwise:

Otherwise:

Rule
When All conditions are valid:
Attribute self.oma.numero = 3

Rule
When All conditions are valid:
Touch is released

Change Attribute
game.Audio.Music.Volume To: game.Audio.Music.Volume -0.2

Otherwise:

Otherwise:

Rule
When All conditions are valid:
Attribute self.oma.numero = 4

Rule
When All conditions are valid:
Touch is released

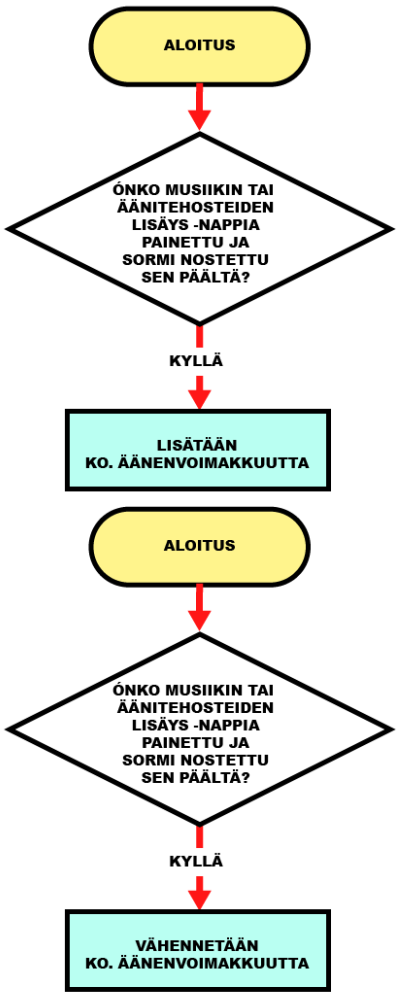
Change Attribute
game.Audio.Music.Volume To: game.Audio.Music.Volume +0.2

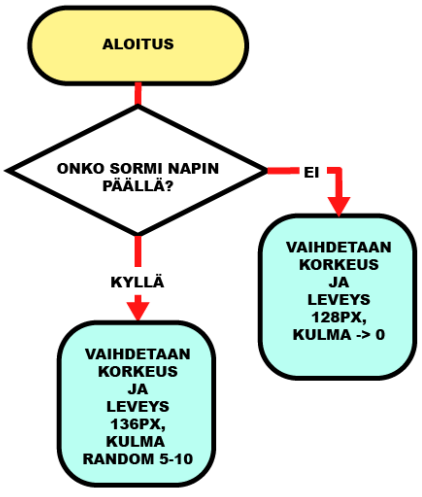
Otherwise:

Otherwise:

ÄÄNENVOIMAKKUUDEN SÄÄTÖNAPIT

Lisätyt attribuutit:
self.oma_numero integer





Rule
When All conditions are valid:
Attribute scene.Camera.X = 0
Attribute scene.Camera.Y = 0

Change Attribute
game.siirtotila To: 0

Change Attribute
game.kamera.x To:0

Change Image
Set Image to: asetuksiin_nappi

Otherwise:

Change Image
Set Image to: valikkoon_nappi

Rule
When All conditions are valid:
Touch is released
Attribute self.painettu_x_0 = false

Change Attribute
scene.Camera.Origin.X To: 0

Change Attribute
game.aktiivinen_kenttä To:0

Change Attribute
self.painettu_x_0 true

Otherwise:

Rule
When All conditions are valid:
Attribute scene.Camera.Origin.Y = 0
Attribute scene.Camera.Origin.X = 0
Attribute self.painettu_x_0 = false

Rule
When All conditions are valid:
Touch is released

Change Attribute
self.painettu_x_0 To: 1

Change Attribute
Scene.Camera.Origin.Y To: 768

Otherwise:

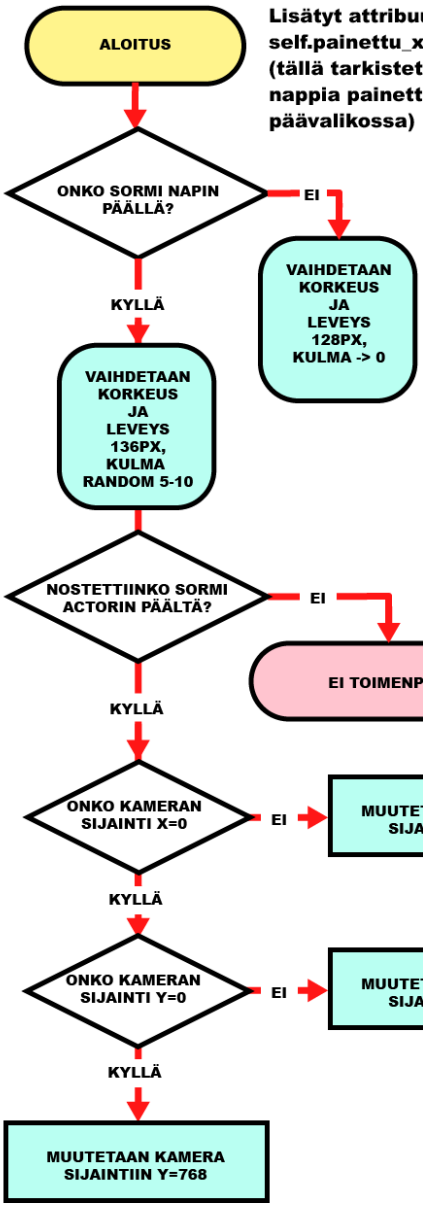
Otherwise:

When All conditions are valid:
Touch is pressed

Change Attribute
self.painettu_x_0 To: 0

Otherwise:

VALIKKO- JA ASETUKSET-NAPPI



Rule
When All conditions are valid:
Touch is released
Attribute scene.Camera.Origin.Y = 768
Attribute self.painettu_x_0 = false

Change Attribute
self.painettu_x_0 To: 1

Change Attribute
Scene.Camera.Origin.Y To: 0

Otherwise:

NÄIDEN LISÄKSI ESIMERKIKSI KOON MUUTTAMINEN KOSKETTAESSA SAMALLA LAILLA KUIN ESIMERKIKSI SIIRTOTILANAPIN KANSSA