PELINTEKO-OHJELMISTOJEN KEHITTYMINEN 2000-LUVULLA

Lappeenrannan–Lahden teknillinen yliopisto LUT

Tietotekniikan kandidaatintyö

2023

Matti Bragge

Tarkastaja(t): Tohtori Erno Vanhala

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUT Teknis-luonnontieteellinen

Tietotekniikka

Matti Bragge

Pelinteko-ohjelmistojen kehittyminen 2000-luvulla

Tietotekniikan kandidaatintyö

2023

42 sivua, 22 kuvaa ja 2 taulukkoa

Tarkastaja(t): Tohtori Erno Vanhala

Avainsanat: pelinkehitys, pelinteko-ohjelmisto, pelimoottori, Multimedia Fusion, Godot, käytettävyys, vertailu

Pelien muuttuessa monimutkaisemmiksi ohjelmiksi niiden kehittämiseen käytettävien työkalujen on samalla kehityttävä. Tässä työssä tutkitaan sitä, miten pelinteko-ohjelmistot ovat muuttuneet 2000-luvun aikana. Tutkimuksen näkökulmia ovat peliominaisuuksien toteuttamisen helppous, ohjelmien toimintojen määrä ja dokumentaatio.

Tutkimus toteutettiin empiirisenä tapaustutkimuksena, jossa vertaillaan kahta pelinteko-ohjelmistoa: vuonna 2001 julkaistua Multimedia Fusionia ja vuonna 2022 päivitettyä Godotia. Vertailu perustuu havaintoihin, jotka kerätään tekemällä sama pelidemo molemmilla ohjelmistoilla. Vertailussa arvioidaan ohjelmistoilla tehtyjä tehtäviä aikatehokkuuden ja tavoitevastaavuuden pohjalta.

Vertailusta saatiin selville, että merkittävää kehitystä on huomattavissa pelinteko-ohjelmistojen käyttömukavuuden, ohjelmavirheettömyyden, rajoittavien tekijöiden ja työskentelytehokkuuden osalta. Toisaalta ilmeni asioita, jotka onnistuivat yhtä helposti tai helpommin Multimedia Fusionilla sen yksinkertaisuuden ansiosta. Yksinkertaisuus toi kuitenkin usein rajoitteita, jotka monimutkaistivat tiettyjen asioiden saavuttamista. Nämä asiat eivät aiheuttaneet ongelmia Godotilla, joka tarjoaa käyttäjälle paljon enemmän vapautta monilla toiminnoillaan. Pelidemon tekemiseen meni Multimedia Fusionilla 3,5 tuntia ja Godotilla 1 tunti ja 40 minuuttia.

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

School of Engineering Science

Software Engineering

Matti Bragge

The evolution of game creation software in the 2000s

Bachelor's thesis

2023

42 pages, 22 figures and 2 tables

Examiners: Dr Erno Vanhala

Keywords: game development, game creation software, game engine, Multimedia Fusion, Godot, usability, comparison

As games become more complex programs, the tools that are used to develop them must also evolve. The aim of this thesis is to research how game creation software has changed during the 2000s. The research perspectives are the easiness of implementing game features, the number of functions in the software, and the documentation.

The research was carried out as an empirical case study that compares two pieces of software: Multimedia Fusion, which was released in 2001, and Godot, which was updated in 2022. The comparison is based on observations that were collected by creating the same game demo with both pieces of software. In the comparison, the tasks done with the software are evaluated based on time efficiency and goal correspondence.

The comparison revealed that significant developments can be observed in the comfort of use, the number of bugs, limiting factors, and efficiency of use of game creation software. On the other hand, some things were found to be as easy or easier with Multimedia Fusion due to its simplicity. However, the simplicity brought limitations often, which made accomplishing certain things more complicated. These things did not cause issues when using Godot, which offers the user much more freedom with its numerous features. Creating the game demo took 3,5 hours with Multimedia Fusion and 1 hour and 40 minutes with Godot.

LYHENNELUETTELO

Lyhenteet

ISO	International Organization for Standardization, kansainvälinen standardeja kehittävä organisaatio
PC	personal computer, henkilökohtainen tietokone
UPBGE	Uchronia Project Blender Game Engine, Blenderin päälle rakennettu pelimoottori
XNA	XNA’s not acronymed, Microsoftin tekemä kokoelma työkaluja peliohjelmointiin

Sisällysluettelo

1 Johdanto

Teknologian kehityksen myötä tietokonepelit ovat nähneet suuria muutoksia esimerkiksi koon, grafiikoiden ja ominaisuuksien osalta. Näiden muutoksien myötä myös pelien luomiseen käytetyt ohjelmistot ovat tarvinneet parannuksia, jotta yhä monimutkaisempia ominaisuuksia voidaan toteuttaa tehokkaammin ja yksinkertaisia asioita vaivattomammin. Aikaisemmin pelit saatiin toimimaan ohjelmilla, jotka oli räätälöity erityisesti pelin ajamiseen käytettävää laitteistoa varten (Ahmad, 2013). Nykypäivänä peliteollisuus on monen miljardin dollarin ala, ja pelimoottoreista on tullut uudelleenkäyttöön soveltuvia ohjelmistokehitystyökaluja (Ahmad, 2013). Vaikka suurin osa kaupallisista peleistä luodaan suljetun lähdekoodin ohjelmilla, hiljattain tunnetuksi on tullut Godot-projekti, jonka tavoitteena on rakentaa vankka ja tehokas avoimen lähdekoodin pelinteko-ohjelma (Politowski *et al.*, 2021). Godotin kaltaiset ilmaiset ohjelmat alentavat kynnystä pelien tekemiseen. Alan jatkuvan kasvun vuoksi on tärkeää tutkia yhtä siihen mahdollisesti vaikuttavaa tekijää, joka on pelien luomisen helpottuminen modernien pelinteko-ohjelmistojen ansiosta. Uusien, helposti saatavilla olevien työkalujen avulla myös yksittäiset henkilöt kykenevät tuottamaan korkealaatuisia pelikokemuksia kattavalla laitteistotuella (Ciesla, 2017). Pelien kehittäminen on mielekäs tapa opetella esimerkiksi ohjelmointia, joten on mielenkiintoista tutkia, kuinka paljon tämän kynnys on madaltunut. Tässä työssä tutkitaan, miten pelinteko-ohjelmat ovat 2000-luvun alusta kehittyneet ja helpottaneet pelien luomista erityisesti harrastajien näkökulmasta. Tarkasteltavia asioita ovat käytettävyys, dokumentaatio ja toiminnot.

Yksi pelinteko-ohjelmiin keskittyvistä aiemmista tutkimuksista on Faizi Noor Ahmadin artikkeli (2013), jossa käsitellään markkinoiden suosituimpien pelinteko-ohjelmien teknisiä tietoja. Kyseisen tutkimuksen tuloksena on vertailu näiden ohjelmien tuetuista ominaisuuksista, selvitystä niillä tehtyjen pelien käyttämistä teknologioista ja katsaus Call of Duty -pelisarjan käyttämän pelimoottorin, IW:n (Infinity Ward), muutoksista pelien välillä. Merkittyihin muutoksiin kuuluu muun muassa uusia visuaalisia tehosteita ja fysiikkaparannuksia. (Ahmad, 2013) Toisessa konferenssijulkaisussa tutkitaan pelinteko-ohjelmien eroja esimerkiksi dokumentaation tasossa, käyttöjärjestelmäyhteensopivuudessa ja aloittelijaystävällisyydessä. Tutkimuksessa tultiin tulokseen, että ei voida suositella yhtä ohjelmaa kaikkien muiden sijaan, koska niillä jokaisella on hyvät ja huonot puolensa. (Vohera *et al.*, 2021)

Kuten edellä mainituissa tutkimuksissa, muissakaan aiemmissa tutkimuksissa ei suoraan tutkita sitä, miten pelinteko-ohjelmat ovat kehittyneet helppokäyttöisyydessä, tai verrata todella vanhoja ohjelmia uusiin. Jotkin tutkimukset käsittelevät pelimoottoreita ja pelinteko-ohjelmia, jotka eivät ole julkisesti saatavilla, mikä eroaa tämän tutkimuksen harrastajanäkökulmasta. Tutkimukset ovat myös menetelmiltään usein pelkästään teoriapainotteisia, jolloin niistä puuttuvat empiiriset havainnot.

Tämän kandidaatintyön tärkeimpänä tavoitteena on saada selville, kuinka paljon pelinteko-ohjelmat ovat muuttuneet. Sen lisäksi, että aihe on mielenkiintoinen, tutkimuksen tuloksista saattaa ilmetä asioita, jotka eivät ole muuttuneet paljoa tai joihin uudempienkin ohjelmien kannattaa kiinnittää huomiota kehittyessään. Tavoitteena on tehdä konkreettisia havaintoja ja esittää vertailu siitä, mitä saman lopputuloksen saavuttaminen edellyttää vanhalla ja uudella ohjelmalla.

Päätutkimuskysymys on seuraava: Miten pelien tekeminen on muuttunut helpommaksi pelinteko-ohjelmistojen kehittyessä? Kuten edellisessä kappaleessa hieman johdateltiin, toisena tutkimuskysymyksenä on seuraava: Mitkä asiat ovat pysyneet lähes yhtä haastavina tai muuttuneet haastavammiksi toteuttaa pelinteko-ohjelmilla? Yksi työn rajauksista on se, että tutkimuksessa keskitytään pelinteko-ohjelmistojen kehityksen kannalta merkittävimpiin ohjelmistoihin. Toinen rajaus on se, että niitä ohjelmistoja, jotka eivät ole olleet julkisesti saatavilla, ei tutkita. Viimeisenä rajauksena on se, että pelinteko-ohjelmistojen kanssa käytettäviä erillisiä ohjelmia, kuten piirtosovelluksia, ei tutkita.

Työn rakenne sisältää johdantoluvun jälkeen luvun, jossa tarkastellaan muita tämän tutkimuksen kaltaisia töitä sekä niiden tuloksia. Kolmannessa luvussa kuvataan aineiston keräämistä, sen käsittelyä sekä tutkimusmenetelmää. Neljäs luku sisältää tutkimuksen tulokset. Viides luku, keskustelu, keskittyy tulosten kommentointiin. Kuudes ja viimeinen pääluku on yhteenveto tutkimuksesta.

2 Muu tutkimus

Tässä luvussa käsitellään muita tutkimuksia tai tieteellisistä töistä, joiden aiheet ovat samankaltaisia tämän kandidaatintyön kanssa. Tutkimuksista käydään läpi aihe, tulokset ja tutkijoiden mahdolliset johtopäätökset. Tutkimukset esitetään kronologisessa järjestyksessä omina kappaleinaan.

Yhdessä tämän tutkimuksen aihepiirin kuuluvassa artikkelissa käsitellään pelitekniikoiden käyttämistä kolmiulotteisten käyttöliittymien luomiseen digitaalisille kirjastoille (Dupire, Topol & Cubaud, 2005). Tutkimuksessa kuvataan tärkeitä toimintoja käyttöliittymille sekä 3D-toteutuksien ongelmia aiemmin ja tutkimuksen luomisen aikana. Ongelmia esiintyi ennen paljon laitteistojen suorituskyvyssä. Eri tekniikoita käyttävien ratkaisujen rajoitteita käyttöliittymän luomisessa on eritelty tarkemmin tutkimuksen toiseksi viimeisessä kappaleessa. Tutkimustuloksissa esitetään taulukko eri tekniikoiden hyvistä ja huonoista puolista. Vertailukohteina on esimerkiksi renderointinopeus ja -laatu, animaatioiden toteutus sekä tiedostojen lataaminen. (Dupire, Topol & Cubaud, 2005.)

Käyttäjän tunteiden tunnistamisesta peleissä potentiaalisilla pelinteko-ohjelmien toiminnallisuuksilla ja mitä se edellyttäisi ovat keskustelun aiheita toisessa tutkimuksessa (Hudlicka, 2009). Siinä päädytään tulokseen, että vaatimukset idean toteuttamiseksi reilusti ylittävät affektiivisen tietojenkäsittelyn ja mallintamisen nykyisen tason. Tuloksista johdetaan, että toteutusta varten affektiivisen tietojenkäsittelyn on edistyttävä paljon, mutta affektiivisten pelimoottoreiden kehittämistä tulisi harkita. Syyksi kerrotaan, että kyseiset edistysaskeleet ovat tarpeellisia, jos peleistä halutaan tehdä tehokkaita opetus- ja hoitovälineitä. (Hudlicka, 2009.)

Toisessa julkaisussa (Doss *et al.*, 2011) käsitellään pelinkehitysalustoja, joita käytetään opetusvälineinä. Tutkimalla alustojen ominaisuuksia kuten käyttöliittymää, virheenkorjausmekanismeja sekä ohjelmointikieltä, tutkimuksen tuloksena on tiivistelmä kuuden eri alustan ominaisuuksista, joita kannattaa ottaa huomioon, jos niitä aikoo käyttää opetuksessa. Tutkittuja alustoja ovat Greenfoot, Game Maker, XNA (XNA’s not acronymed), Scratch, Alice ja Pygame. XNA on Microsoftin tekemä kokoelma työkaluja peliohjelmointia varten.

Johtopäätös on, että kehitystyökalun vahvuuksien ja ohjelman sovittaminen yhteen voi vaikuttaa huomattavasti itse ohjelmaan sekä opiskelijoiden menestykseen. (Doss *et al.*, 2011)

Muussa tutkimuksessa tutkitaan yleisimmin käytettyjä pelinteko-ohjelmia hyötypelien kehittämiseen (Cowan & Kapralos, 2014). Yleisiä ohjelmissa esiintyviä työkaluja on listattu tutkimuksen taustaan keskittyvässä kappaleessa: kenttä-, skripti-, materiaali- ja äänieditori. Julkaisun tulososiossa esitetään taulukko, jossa vertaillaan ohjelmien tukemia ominaisuuksia. Lisäksi tuloksissa on pylväskaavio siitä, kuinka usein tietyt termit esiintyvät akateemisissa tietokannoissa jokaisen eri ohjelman yhteydessä. Kaaviosta nähdään esimerkiksi, että termillä ”educational game” eli ”opetuksellinen peli” on eniten esiintymiä Second Life -ohjelman yhteydessä. Viimeisessä kappaleessa esitetään johtopäätös, että hyötypelien tekijät käyttävät enimmäkseen pelinteko-ohjelmia, jotka ovat suunniteltu erityisesti viihdepelien luomista varten. Lopussa ehdotetaan, että hyötypelien tekemiseen tarkoitetuista ohjelmista puuttuu monia ominaisuuksia, joita kaupalliset ohjelmat tarjoavat. (Cowan & Kapralos, 2014.)

Yhdessä artikkelissa vertaillaan suosittuja pelinteko-ohjelmia, niillä tuotettuja moderneja pelejä ja eri julkaisualustojen suosiota (Mishra & Shrawankar, 2016). Lisäksi siinä testataan pelimoottorin vaikutusta pelien suoritustehokkuuteen neljällä eri näytönohjaimella. Tuloksissa merkitään muun muassa, että Cry Enginellä tehty Crysis 3 (Electronic Arts, 2011) hyödyntää prosessoria paremmin, kuin Vision Enginellä kehitetty Max Payne 3 (Rockstar Games, 2012). Artikkelissa on pelinteko-ohjelmien vertailun lisäksi lyhyet katsaukset seuraavista pilvipelaamisalustoista: GaiKai, StreamMyGame, OnLive ja GamingAnywhere. (Mishra & Shrawankar, 2016.)

Vanhanaikaisia pelinteko-ohjelmistoja on käsitelty eräässä kirjaluvussa (Ciesla, 2017). Luvussa kerrotaan tietoa ohjelmistoista kuten The Quill (Gilsoft, 1983), Pinball Construction Set (Electronic Arts, 1983), Garry Kitchen’s GameMaker (Activision, 1985) ja STOS BASIC (Mandarin Software, 1988). Esittelyyn otetut ohjelmistot on valittu sillä periaatteella, että ne sisältävät kaikki tarvittavat työkalut suhteellisen esityskelpoisen pelin luomista varten. Ohjelmistoista esitellään esimerkiksi niiden taustatiedot, rajoitteita sekä alustat, joille ohjelmat on suunniteltu. Ohjelmistojen huomattavimmat rajoitteet liittyvät enimmäkseen niillä tuotettujen pelien suorituskykyyn:

> *"Most often, games produced with these systems performed shoddily with a low frames per second (fps) rate. The key was either keeping your projects as elementary as possible or investing in additional Mandarin Software products. The STOS/AMOS compiler accelerated programs written with the base software considerably."* (Ciesla, 2017)

Yleisiä alustoja ohjelmistoille ovat Commodore 64, IBM PC (personal computer), Apple 2, Amiga ja Atari ST (Ciesla, 2017).

Jatkona edelliseen kappaleeseen, saman kirjan viidennessä luvussa kirjoittaja arvostelee ilmaisia pelinteko-ohjelmistoja kaupallisen potentiaalin, käytettävyyden, audiovisuaalisten ominaisuuksien ja tuen näkökulmasta sekä yleisesti. Arvosteltuja ohjelmia ovat Unity, Construct Classic/Construct 2, Ren'Py, Gamelooper, Stencyl ja Godot. (Ciesla, 2017.) Taulukossa 1 on yhteenveto luvussa annetuista arvosanoista, joissa suurempi numero tarkoittaa parempaa. Tutkimuksessa ei suoraan kerrota syytä sille, miksi Godotin yleisarvosana on viisi, vaikka sen yksittäisten osa-alueiden arvosanat ovat kaikki neljiä.

Taulukko 1: Ilmaiset pelinteko-ohjelmat arvioituna asteikolla 1 – 5 (Ciesla, 2017)

Ohjelma	Unity	Construct	Ren'Py	Gamelooper	Stencyl	Godot
Kaupallinen potentiaali	5	4	4	3	4	4
Käytettävyys	4	4	4	3	4	4
Audiovisuaaliset ominaisuudet	5	4	4	3	4	4
Tuki	5	5	4	3	4	4
Yleisarvosana	5	4	4	3	4	5

Toisen kirjan kuudennessa luvussa keskustellaan Blender-mallintamisohjelman pelimoottorin tulevaisuudesta ja esitellään Blenderin päälle rakennettuja avoimen lähdekoodin pelimoottoreita, jotka voisivat toimia Blenderin omalle moottorille vaihtoehtoina sen poistuessa ohjelmiston uusista versioista (Guevarra, 2020). Kyseisen pelimoottorin yksi vahvuuksista on Guevarran mukaan siinä, että se on suunniteltu yksinkertaistamaan monimutkaisia moottoriominaisuuksia sellaisiksi, että niiden käyttämiseen ei tarvita ohjelmointitaitoja. Vaihtoehtoina esiteltyjä pelimoottoreita ovat UPBGE (Uchronia Project Blender Game Engine) ja Armory3D. Luvun lopussa kerrotaan, että Blenderin oma moottori on poistettu, joten nämä

vaihtoehdot tulevat toimimaan korvaavina moottoreina sille niin kauan kuin sitä ei oteta uudestaan kehitykseen. (Guevarra, 2020.)

Yhdessä artikkelissa käsitellään avoimen lähdekoodin pelimoottoreita kolmesta eri perspektiivistä: kirjallisuus, koodi ja ihminen (Politowski *et al.*, 2021). Kirjallisuusperspektiivi kattaa kirjat ja akateemiset julkaisut, koodiperspektiivi kattaa pelimoottoreiden staattiset, historialliset sekä yhteisölliset ominaisuudet ja ihmisperspektiivin kautta tarkastellaan pelimoottorikehittäjien kyselyvastauksia liittyen tutkimuskysymykseen. Tutkimuksessa vastataan siihen, onko avoimen lähdekoodin pelimoottoreilla samanlaisia ominaisuuksia kuin tavallisilla ohjelmistokehyksillä. Tuloksissa kerrotaan, että vaikka vastaus on kyllä, niin ohjelmistokehysprojekteilla on muun muassa pidemmät elinajat, useampia julkaisuja ja enemmän suosiota kuin pelimoottoriprojekteilla. Johtopäätöksiksi on merkitty, että pelimoottorien kehittäjien tulisi ottaa käyttöön samanlainen ohjelmistokoodin laadun työkalupakki ja keskittyä enemmän dokumentaatioon. (Politowski *et al.*, 2021.)

Pelinteko-ohjelmistoja ja niiden ominaisuuksia hyötypelien luomisen ja pelillistämisen näkökulmasta arvioidaan muussa tutkimuksessa (Sharif & Yousif Ameen, 2021). Tutkimuksen tuloksissa on lyhyiden pelinteko-ohjelmistoesittelyiden lisäksi kaksi taulukkoa, joissa näytetään ohjelmistojen yksityiskohtia sekä analyysiä niiden toiminnoista. Eri arvosteluperiaatteiden pohjalta on myös arvioitu parhaat vaihtoehdot ohjelmien välillä. Esimerkiksi fysiikoiden osalta suositellaan Unreal Engineä, Unityä tai Cry Engineä, ja skriptauksen puolesta parhaiksi arvioidaan Unreal Engine ja Godot niiden useiden skriptauskielien ansiosta. Kuten monissa muissakin pelinteko-ohjelmistoja vertailevissa tutkimuksissa, tässäkin tullaan samaan johtopäätökseen, että yhtä parasta ohjelmistoa ei voida määrittää yleisesti. (Sharif & Yousif Ameen, 2021.)

3 Tutkimusprosessi

Tässä luvussa kuvataan työn aineiston keräämistä ja sen käsittelyä. Lisäksi luvussa esitetään työn tutkimusmenetelmä ja se, miten sitä noudatetaan työssä.

Tutkimuksen kirjallisen aineiston keräämiseen käytettiin pääasiallisesti Google Scholar -hakukonetta, josta haettiin tieteellisiä tekstejä esimerkiksi hakusanoilla pelinteko-ohjelma ja pelimoottori, sekä niiden englanninkielisillä vastineilla game creation software ja game engine. Etsimiseen käytettiin myös Scopus-viitetietokantaa. Koska työn aiheesta on suhteellisen vähän suomeksi kirjoitettua tutkimusta, englanninkielisillä hakusanoilla oli enemmän tuloksia. Kirjallista aineistoa avataan edellisessä luvussa ja niihin viitataan viidennessä luvussa.

Työn teknisen osan aineiston kerääminen tapahtui tekemällä havaintoja kahdesta eri pelinteko-ohjelmistosta vertailua varten. Ohjelmistoja käytettiin mahdollisimman identtisten pelidemojen luomiseen, ja havainnot kohdistuivat ohjelmistojen helppokäyttöisyyteen, dokumentaatioon, ajankäyttöön ja toimintoihin. Käytettävät ohjelmistot etsittiin kirjallista aineistoa lukemalla sekä tutkimalla Wikipedian pelinteko-ohjelmistokategoriaa (2019). Ensimmäisen ohjelmiston, Multimedia Fusionin, valinta perustui siihen, että sen julkaisuvuosi sijoittui mahdollisimman lähelle 2000-luvun alkua ja sen kokeiluversion lataamiseen löydettiin toimiva arkistoitu sivu. Toisen ohjelmiston, Godotin, valintaperusteet olivat toisessa tutkimuksessa mainittu viimeaikainen suosio kehittäjien keskuudessa (Politowski *et al.*, 2021), laaja dokumentaatio ja se, että se on ilmainen avoimen lähdekoodin ohjelmisto.

Työn tutkimusmenetelmä on empiirinen tapaustutkimus, jossa pyritään löytämään eroja vanhan ja uuden pelinteko-ohjelmiston välillä vertailemalla niiden käyttämisestä tehtäviä havaintoja. Menetelmän soveltamisessa vertailuun keskeistä on se, että havaintojen arviointiin luodaan vankka perusta ennakkoasenteiden välttämiseksi (Conradi & Wang, 2003). Tämän vuoksi tässä työssä tehdyssä vertailussa arvioidaan ajankäyttöä ja saavutettujen tulosten vastaavuutta tavoitteisiin. Tarkoituksena on poistaa subjektiivisuus tuloksista mahdollisimman hyvin. Erik Frøkjær et al. (2000) esittävät ISO:n (International Organization for Standardization) luoman määritelmän käytettävyydestä kolmena aspektina: vaikuttavuus, tehokkuus ja käyttäjätyytyväisyys. Vaikuttavuudella tarkoitetaan tavoitteiden saavuttamisen tarkkuutta ja

täydellisyyttä, tehokkuus kuvaa vaikuttavuuden ja resurssien, kuten ajan, käytön suhdetta ja käyttäjätyytyväisyys käyttäjien mukavuutta sekä positiivisia asenteita ohjelmiston käyttöä kohtaan. Erityisesti vaikuttavuus ja tehokkuus ovat siis linjassa tämän tutkimuksen vertailu-perusteiden kanssa.

4 Pelidemon toteutus

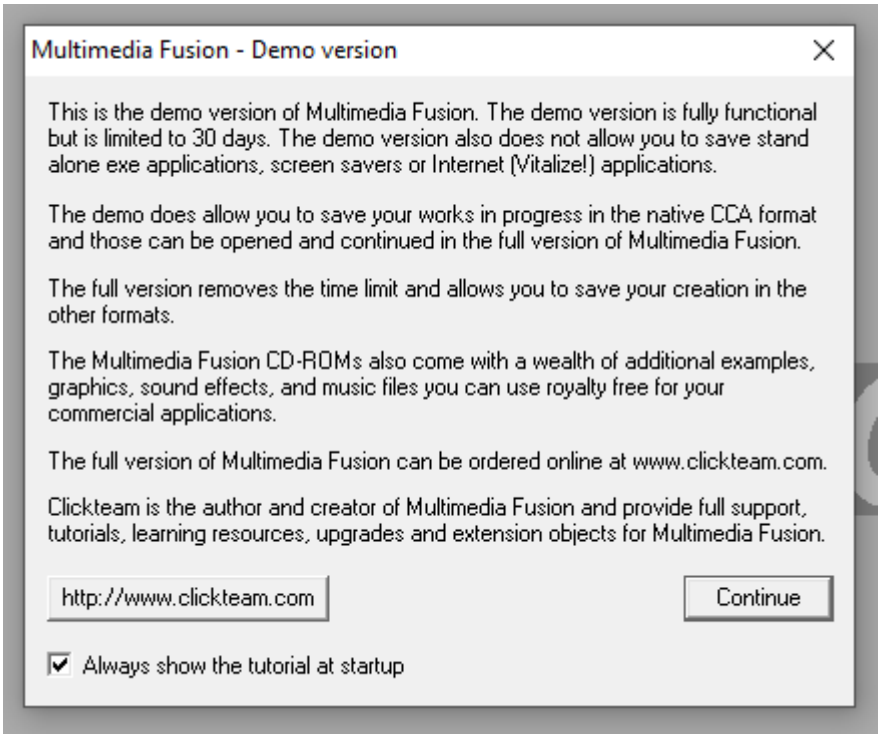
Tässä tutkimuksessa tehtiin pieni peli kaksi kertaa eri ohjelmistoilla, ja toteutusten aikana tehtyjä havaintoja esitellään tässä luvussa. Ensimmäinen ohjelmisto on vuonna 2001 julkaistu Multimedia Fusion 1.5, ja toinen on viime vuosina suosiota kerännyt Godot 3.5. Luvun lopussa esitetään yhteenveto havainnoista.

4.1 Tavoite

Tavoitteena on luoda kaksiulotteinen peli, jossa pelaaja voi liikkua ja hyppiä pelihahmolla. Hahmolla on animaatiot liikkumiselle ja hyppäämiselle. Hypätessä kuuluu äänitehoste. Pelialueen keskellä on pudotus, johon pudotessa peli alkaa alusta. Pudotuksen toisella puolella on maaliobjekti, jonka luokse pelaajan pitää päästä voittaakseen pelin. Maalia koskettaessa ruudulle ilmestyy tekstiä Roboto-fontilla. Pelin grafiikat ja äänitehoste tuodaan ohjelman ulkopuolisista tiedostoista. Pelin toimii ilman ohjelmavirheitä, ja sen kuvatarkkuus on 160 pikseliä kertaa 90 pikseliä venytettynä suuremmaksi.

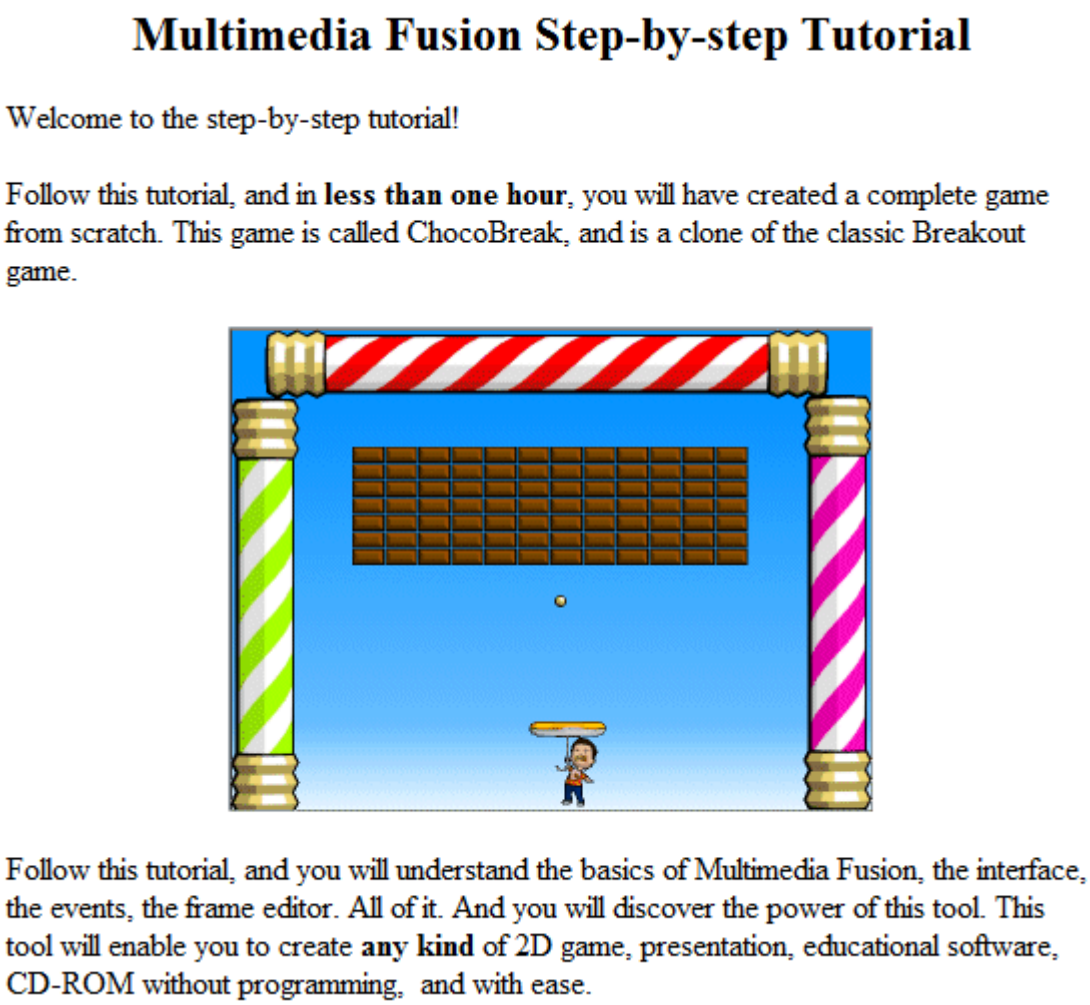
4.2 Multimedia Fusion

Heti ohjelman käynnistyessä ruudulle ilmestyy ilmoitus, joka esittää mahdollisuuden avata tekstimuotoinen opastuskurssi (engl. tutorial) käynnistymisen yhteydessä. Tämä ilmoitus näytetään kuvassa 1.



Kuva 1: Ohjelman käynnistyessä avautuva ilmoitus (Clickteam, 2001). Kirjoittajan ottama kuvakaappaus.

Tämä on hyvä keino auttaa käyttäjä alkuun ilman, että hänen tarvitsee etsiä itse mitään. Kurssi neuvoo, miten ohjelmalla ja sen asennuksen mukana tulevilla resursseilla luodaan valmis klooni klassisesta Breakout-pelistä. Kuvassa 2 on opastuskurssin ensimmäinen sivu.



Multimedia Fusion Step-by-step Tutorial

Welcome to the step-by-step tutorial!

Follow this tutorial, and in **less than one hour**, you will have created a complete game from scratch. This game is called ChocoBreak, and is a clone of the classic Breakout game.

Follow this tutorial, and you will understand the basics of Multimedia Fusion, the interface, the events, the frame editor. All of it. And you will discover the power of this tool. This tool will enable you to create **any kind** of 2D game, presentation, educational software, CD-ROM without programming, and with ease.

Kuva 2: Kuvakaappaus opastuskurssin ensimmäisestä sivusta (Clickteam, 2001). Sivulla on kuva pelistä ChocoBreak (Clickteam, n.d.). Kirjoittajan ottama kuvakaappaus.

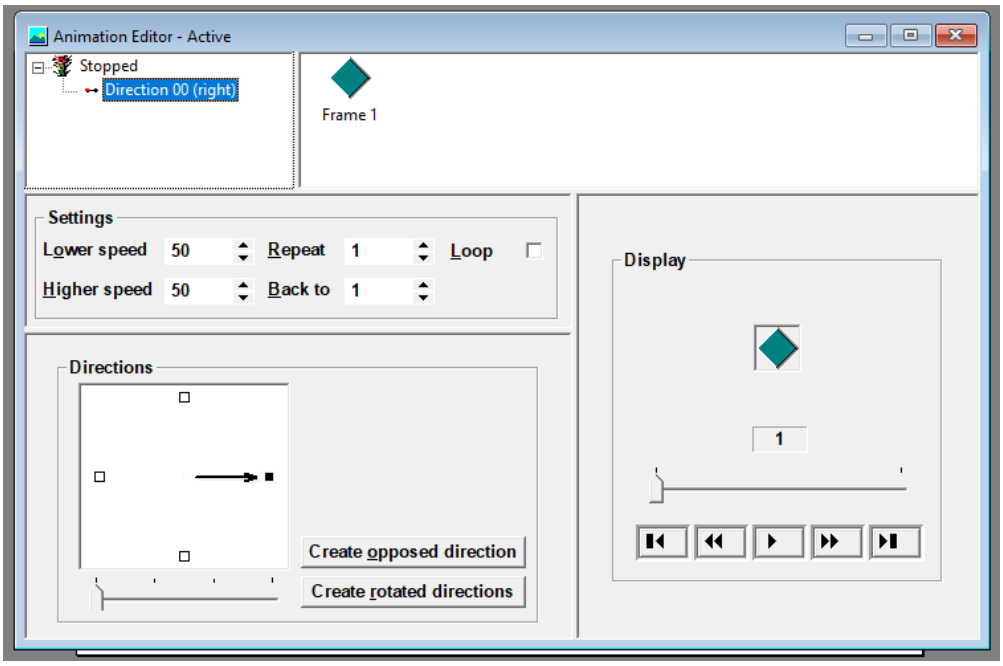
Vaikka kurssi tutustuttaa käyttäjän tehokkaasti ohjelman kanssa työskentelyyn, syvällisempi dokumentaatio olisi tarpeen selittämään niitä asioita, joita kurssi ei käy läpi. Esimerkiksi omien kuvatiedostojen käyttämistä peliobjektien animaatioissa ei neuvota kurssilla. Ohjelman ohjevalikko sisältää lisää vaihtoehtoja, mutta tämän tutkimuksen toteutusympäristössä nämä eivät toimi. Esimerkiksi ohjelman help-työkalu mahdollistaa sen, että käyttäjä painaa mitä tahansa asiaa ohjelmassa avatakseen sopivan ohjesivun. Idea on käyttäjäystävällinen, mutta sivujen sisältöä ei tässä tutkimuksessa pystytä tutkimaan mainitusta syystä.

Ohjelman ylälaidassa sijaitsee työkalupalkki, joka sisältää pikakuvakkeita. Palkki esitetään kuvassa 3.

Kuva 3: Työkalupalkki (Clickteam, 2001). Kirjoittajan ottama kuvakaappaus.

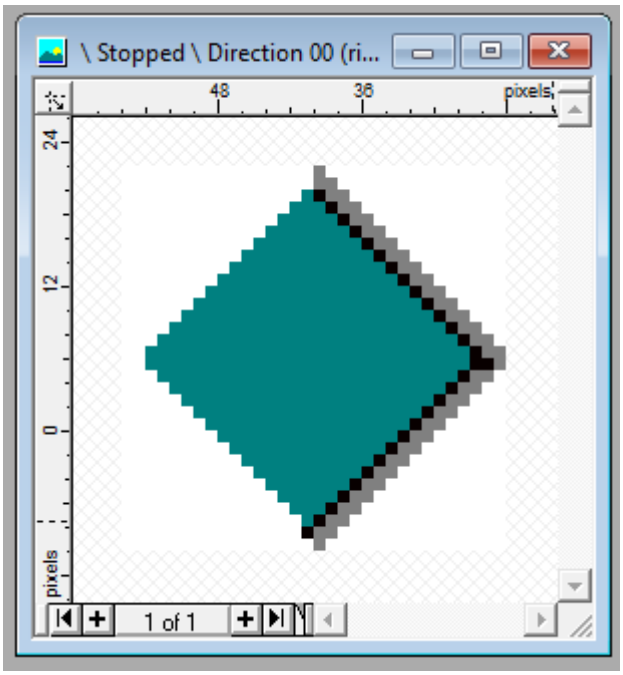
Nämä kuvakkeet osoittautuivat ulkonäöiltään osin hankalatulkintaisiksi, mikä hidasti työskentelyä. Hiiren osoittimen pitäminen kuvakkeen päällä hetken näyttää vastaavan toiminnon nimen, mikä auttaa lievittämään ongelmaa.

Jotta molemmilla tutkimuksessa käytetyillä ohjelmilla luodut pelit vastaisivat toisiaan mahdollisimman tarkasti, tavoitteena oli käyttää niissä tätä tutkimusta varten luotuja grafiikoita. Tässä onnistuminen vaatii Multimedia Fusionissa monta vaihetta. Peliobjektin luomisen jälkeen sen animaatioita pitää muokata painamalla hiiren oikeaa painiketta ja valitsemalla menun ensimmäisen vaihtoehdon. Muokkausikkuna näkyy kuvassa 4.



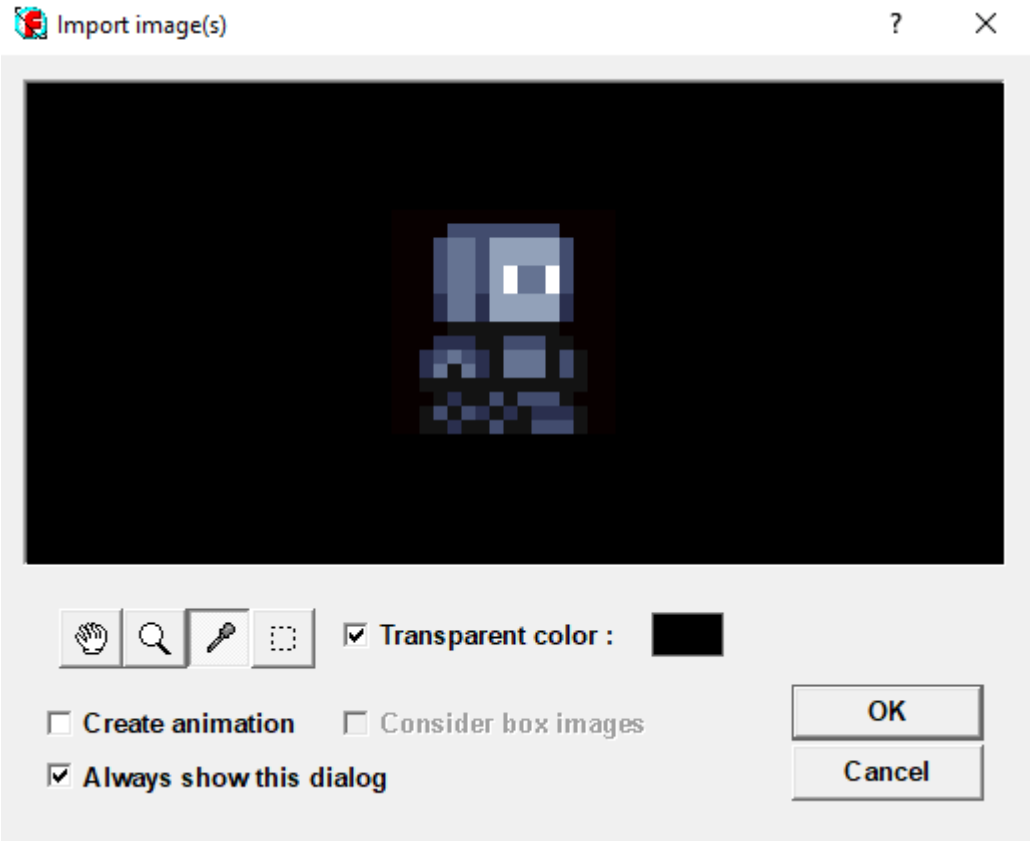
Kuva 4: Animaation muokkausikkuna (Clickteam, 2001). Kirjoittajan ottama kuvakaappaus.

Avautuvasta ikkunasta näkee objektin animaatiot ja niihin liittyvät asetukset sekä kuvaruudut (engl. frame). Ruutuja voi muokata painamalla niitä hiiren oikealla painikkeella ja valitsemalla menun ensimmäisen vaihtoehdon. Tätä varten käytettävä ikkuna on kuvassa 5.



Kuva 5: Animaation kuvaruutujen muokkausikkuna (Clickteam, 2001). Kirjoittajan ottama kuvakaappaus.

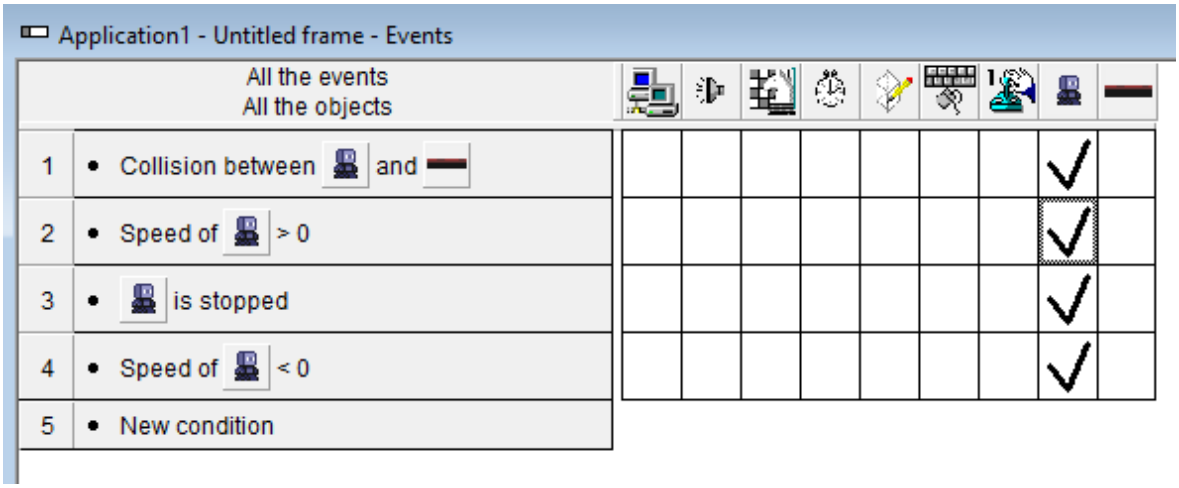
Avautuvan ikkunan alareunassa on plussalla merkitty painike, jonka avulla käyttäjä voi lisätä animaatioon oman kuvatiedostonsa. Läpinäkyviä kuvatiedostoja ohjelma ei tue, mutta tilalle syntyvän mustan taustan saa poistettua valitsemalla värin, jonka ohjelma muuttaa itse läpinäkyväksi. Kuvassa 6 esitetään ikkuna, joka avautuu kuvan tuomisen yhteydessä.



Kuva 6: Tuodun kuvatiedoston muokkausikkuna (Clickteam, 2001). Kirjoittajan ottama kuvakaappaus.

Yksi suurimmista hidasteista Multimedia Fusionin käyttämisessä on se, että animaatioruutujen lisääminen on tehtävä yksi ruutu kerrallaan. Animaatioiden muokkaamisen prosessin opetteleminen on myös aikaa vievää. Muussa tutkimuksessa ehdotetaan, että tehokkuuden tai opittavuuden mittarina voi käyttää vaadittujen askelten määrää tehtävien suorittamiseen (Kazman, Bass & Bosch, 2003). Kyseisellä mittarilla Multimedia Fusion jättää paljon parantamisen varaa tässä tehtävässä. Lisäksi usean ikkunan avautuminen tekee ohjelman työalueesta sekavan, eikä muiden ikkunoiden alle jääviä ikkunoita saa helposti tuotua esiin. Tämä hidasti työskentelyä huomattavasti.

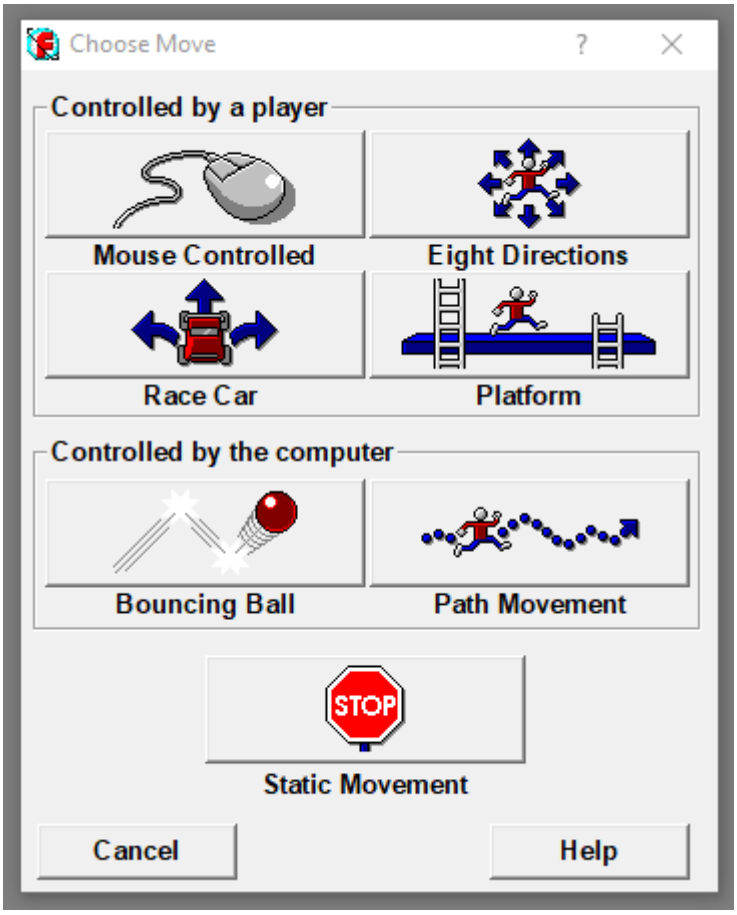
Pelilogiikan lisäämistä varten Multimedia Fusionissa ei tarvitse osata mitään ohjelmointikieltä, vaan logiikka koostuu tapahtumalauseista (engl. events), jotka sisältävät ehtoja ja toimintoja. Ehto on esimerkiksi näppäimen painaminen, ja toiminto on esimerkiksi peliobjektin animaation toistaminen. Näistä tapahtumalauseista näytetään esimerkkejä kuvassa 7.



Kuva 7: Tapahtumalauseikkuna (Clickteam, 2001). Kirjoittajan ottama kuvakaappaus.

Pelilogiikan lisääminen on yksinkertaistettua, ja se on selitetty ohjelman opastuskurssissa selkeästi. Tapahtumalauseiden muokkausvalikossa objektit näkyvät omina kuvinaan, mikä auttaa hahmottamaan lauseita silmäyksellä. Toisaalta samannäköiset objektit saattavat sekoittua helposti keskenään, koska kuvat ovat pieniä. Yhdelle toiminnolle voi asettaa useampia ehtoja, ja ehtoja voi kääntää päinvastaisiksi. Koodin kirjoittamisen puutteen vuoksi työskentely on nopeaa, mutta rajoitteita tulee vastaan usein mahdollisissa toiminnoissa.

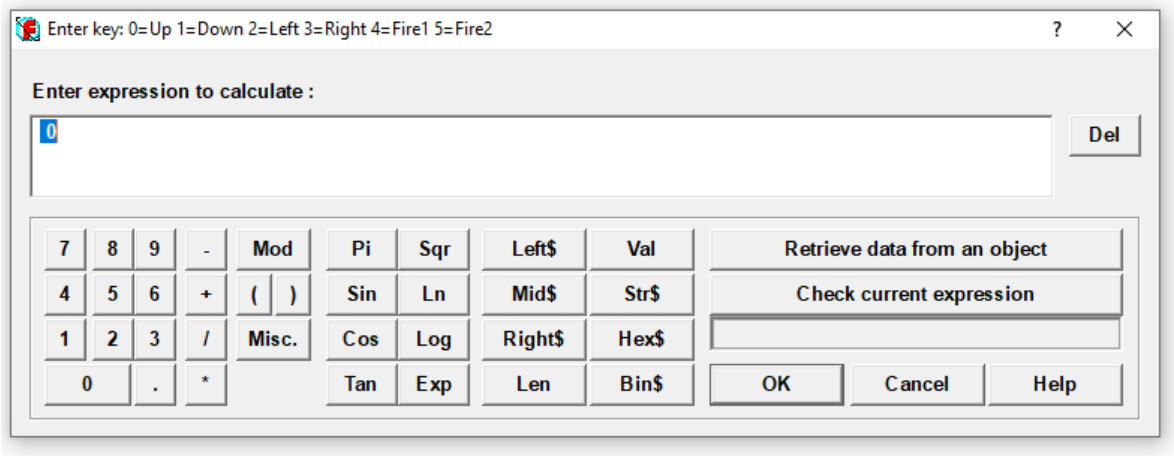
Ohjelma sisältää seitsemän erilaista valmista liikkumismuotoa peliobjekteille. Näiden lisääminen onnistuu nopeasti kahdella hiiren painalluksella. Eri liikkumismuodot ovat kuvassa 8.



Kuva 8: Liikkumismuotojen valintaikkuna (Clickteam, 2001). Kirjoittajan ottama kuvakaappaus.

Kuvan 8 valikko sisältää havainnollistavia kuvakkeita liikkumismuodoille. Platform-vaihtoehdon valitsemalla pelihahmon saa liikkumaan sivulta kuvatun tasohyppelypelin mukaisesti ilman, että painovoimaa tai hyppimistä toteuttaa itse. Toiminto on helppokäyttöinen, mutta ominaisuuksiltaan rajallinen. Ohjelma tarjoaa säädeltäviä arvoja liikkumisen muokkaamista varten, mutta esimerkiksi hypyn korkeutta ei voi muuttaa pelilogiikan tapahtumalauseissa. Tällainen toiminto vaatisi, että käyttäjä toteuttaisi objektin liikkumisen itse käyttämällä objektille lisättäviä muuttujia ja tapahtumalauseita, mikä on paljon monimutkaisempaa.

Ohjelmassa on mahdollista muuttaa pelaamiseen käytettäviä näppäimiä, mutta niitä on käytössä vain kuusi, ja ne on muutettava tapahtumalauseen sisällä. Kuvassa 9 näkyy ikkuna, jonka avulla näppäimen voi kytkeä johonkin toimintoon.

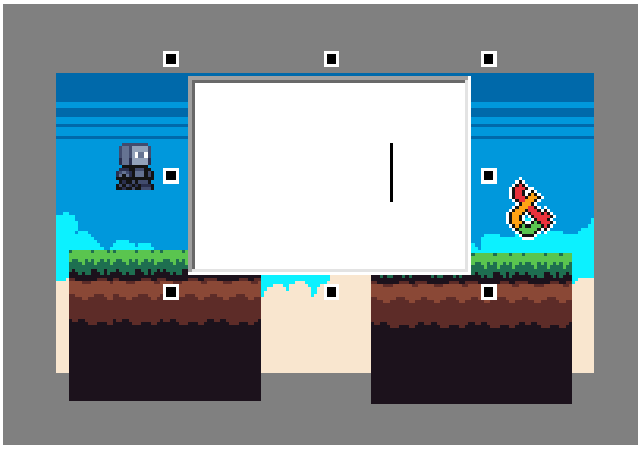


Kuva 9: Näppäimen kytkeminen toimintoon (Clickteam, 2001). Kirjoittajan ottama kuvakaappaus.

Eri toiminnot on listattu ikkunan otsikkopalkissa, mikä tekee niistä vaikeasti huomattavia ja hankalalukuisia. Matemaattisten toimintojen tarjoaminen tässä luo mahdollisuuksia, mutta voi myös näyttää epäselkeältä.

Peliobjektien siirtäminen tarkasti on Multimedia Fusionilla hankalaa, koska se ei tarjoa objektien kohdistamista esimerkiksi ruudukon tai muiden objektien mukaan. Ainoat tavat siirtää objekteja on hiirellä raahaaminen ja nuolinäppäimien painaminen. Pelialuetta muokatessa näkymää ei voi myöskään suurentaa, mikä vaikuttaa käytettävyyteen negatiivisesti. Monimutkaisissa ohjelmistoissa työskentelynäkymän suurentaminen on tärkeä elementti tehokasta käyttöä ajatellen (Albers, 2011). Lisäksi objekteja skaalatessa niiden palauttamiseen alkuperäiseen kokoonsa ei löydy intuitiivista keinoa, vaan skaalaus on peruttava heti sen tekemisen jälkeen tai objekti on luotava uudestaan.

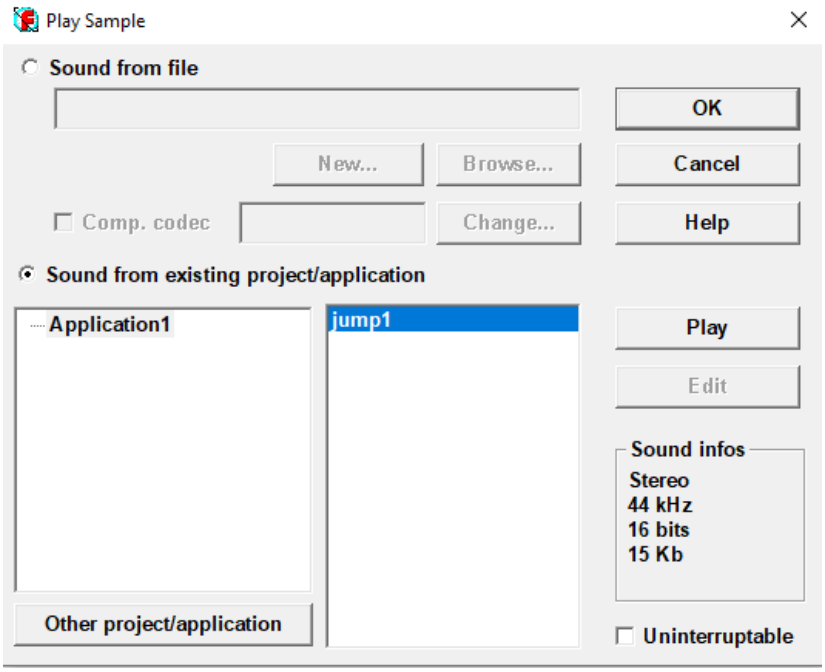
Taustakuvan saa helposti luotua peliruudulle, mutta oman grafiikan saa vaihdettua siihen pelkästään kopioimalla ja liittämällä sen leikepöydältä, mikä monimutkaistaa prosessia. Pelialueeseen voi lisätä tekstiobjektin ja kirjoittaa siihen suoraan. Kuva 10 on kuvakaappaus tekstilaatikosta, joka sisältää valkoista tekstiä.



Kuva 10: Valkoista tekstiä tekstilaatikossa (Clickteam, 2001). Kirjoittajan ottama kuvakaappaus.

Tekstiobjektia lisätessä ohjelma tarjoaa työkalupalkissa suhteellisen hyvät muokkausvaihtoehdot kuten tekstin lihavoimisen ja värin muuttamisen. Toisaalta tekstin ääriviivan väriä ei saa muutettua, ja valkoista tekstiä kirjoittaessa se ei erotu tekstilaatikon valkoisesta taustasta.

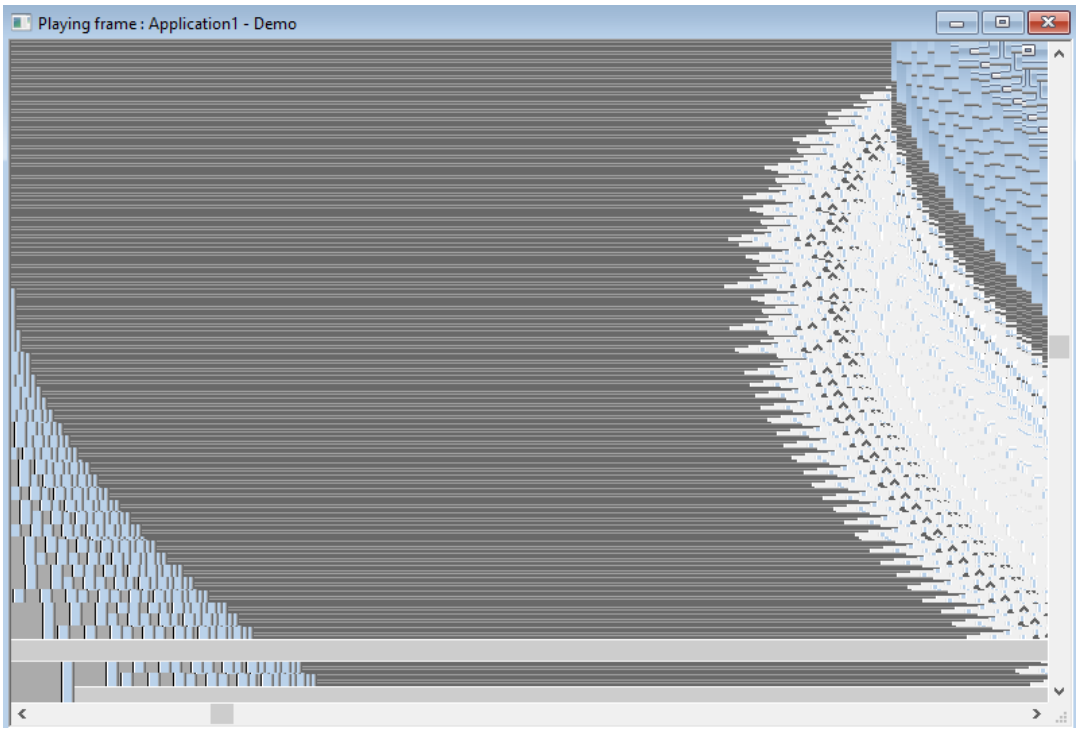
Pelin aloittaminen alusta pelaajan pudotessa ja äänitehosteen toistaminen olivat suoraviivaisia toteuttaa. Kuvassa 11 näkyy ikkuna, jossa voi säätää toistettavaa äänitehostetta.



Kuva 11: Äänitehosteikkuna (Clickteam, 2001). Kirjoittajan ottama kuvakaappaus.

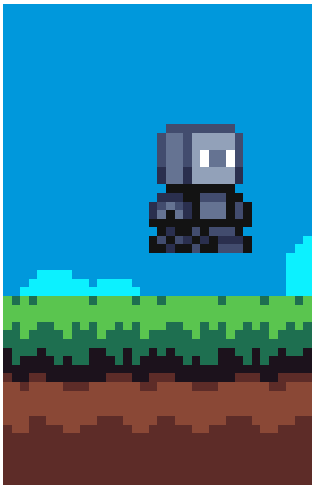
Kuitenkin ääniefektin yhdistäminen suoraan pelaajan hyppäämiseen ei onnistunut, vaan se toistuu aina näppäintä painaessa, mikä eroaa tavoitteesta.

Peli-ikkunan skaalaamista suuremmaksi ohjelman tarjoamalla toiminnolla ei saatu toimimaan tutkimuksen toteutusympäristössä, vaan se johti ohjelmavirheeseen. Ohjelmavirhe esitellään kuvassa 12.



Kuva 12: Ikkunan skaalaamisesta johtuva ohjelmavirhe

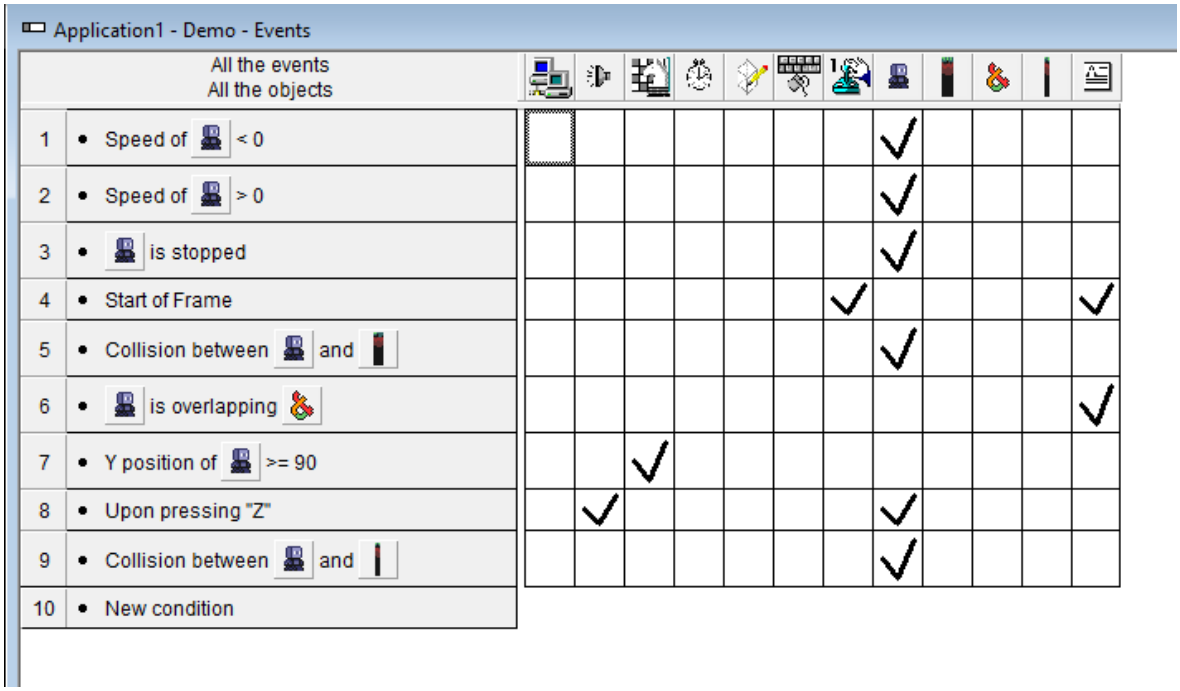
Virheen vuoksi tätä ei oteta huomioon tavoitteen saavuttamisen vertailussa. Pelissä ilmeni toinenkin ohjelmavirhe, joka näkyy kuvassa 13. Nämä virheet aiheuttivat sen, että lopullinen peli ei vastannut tavoitteita täydellisesti.



Kuva 13: Pelaajahahmo pysähtyy ilmassa ohjelmavirheen takia

Pelin tekemisessä kesti Multimedia Fusionilla noin 3,5 tuntia. Kuvissa 14 ja 15 näkyy kuvakaappaukset lopullisesta pelistä ja sen tapahtumalauseista.

Kuva 14: Kuvakaappaus valmiista pelistä



Kuva 15: Tapahtumalauseikkuna (Clickteam, 2001), jossa on valmiin pelin tapahtumalauseet. Kirjoittajan ottama kuvakaappaus.

Peli toimii lähes tavoitteiden mukaisesti, mutta kaikki niistä eivät täyttyneet seuraavien ongelmien vuoksi:

- Pelaajahahmo pysähtyy ilmaan laskeutuessaan hypyn jälkeen
- Pelaajahahmo jää kiinni seiniin hypätessä
- Animaatiot eivät aina käynnisty
- Äänitehosteen toistaminen ei ole kytketty hyppäämiseen vaan näppäimen painallukseen

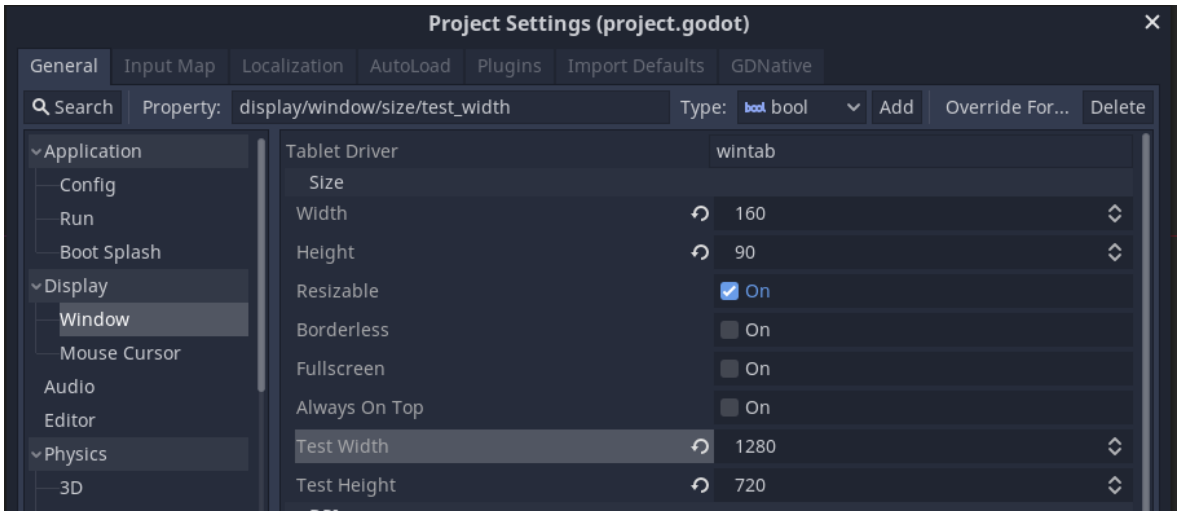
4.3 Godot

Pelin rakentaminen alkoi valitsemalla juurisolmuksi (engl. root node) kaksiulotteinen näyttämö (eng. 2D scene). Kuten Godotin dokumentaatiossa kerrotaan, näyttämö voi olla esimerkiksi hahmo, esine tai pelikenttä. Kaikki näyttämöt koostuvat juurisolmusta, jolla voi

olla loputon määrä lapsisolmuja. Solmuluokkia on erilaisia eri tarkoituksiin. Esimerkiksi Sprite-solmulla näyttämölle voidaan piirtää liikkumaton kuva. Solmut ovat objekteja, jotka perivät ominaisuuksia yliluokiltaan. (Godot Docs, 2022a.)

Godotin työkalupalkista löytyy apuvalikko, joka tarjoaa linkkejä dokumentaatioon, kysymyspalstalle sekä muihin yhteisösivuihin. Näiltä sivuilta löytyy paljon opastuskursseja erilaisten pelien tekemiseen, mitkä auttavat oppimaan järkeviä toteutustapoja peliominaisuuksille kuten pelihahmon liikkumismuodoille. Ylimpänä vaihtoehtona apuvalikossa on avunhakemistoiminto, josta voi hakea ja avata luokkien toimintoja ja ominaisuuksia selittäviä viitteitä luettavaksi ohjelman sisällä. Tämä toiminto nopeutti työskentelyä huomattavasti, koska se vähensi tarvetta vaihtaa ohjelman ja nettiselaimen välillä.

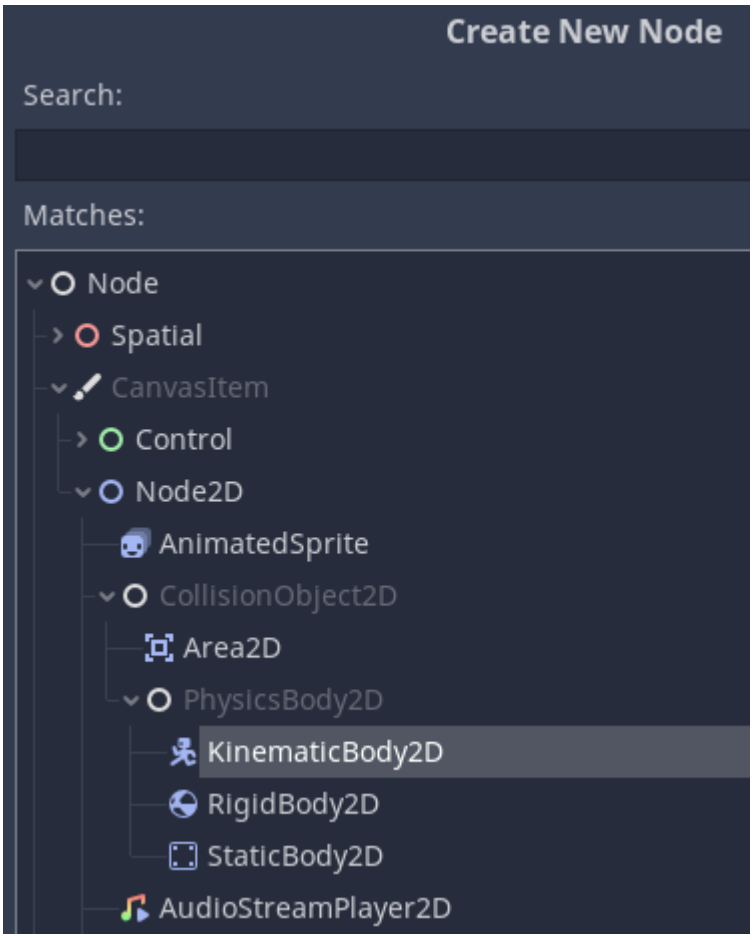
Pelin kuvatarkkuuden muuttaminen tavoitetta vastaavaksi onnistui helposti asetusikkunan hakutoiminnon ansiosta. Lisäksi peli-ikkunan skaalaaminen suuremmaksi toimi vaihtamalla arvoja testausleveydelle ja -korkeudelle. Kuvassa 16 näkyy molemmat asetukset.



Kuva 16: Peli-ikkunan asetukset (Linietsky & Manzur, 2022). Kirjoittajan ottama kuvakaappaus.

Kuvassa näkyvät myös useat muut asetukset ja välilehdet, joita ei tutkimuksessa käytetty. Jotta itse pelikuva saatiin venymään ikkunan kokoiseksi, pelin venytysasetuksia piti muuttaa. Tavoitellun lopputuloksen saaminen oli helppoa kokeilemalla eri vaihtoehtoja. Vaihtoehtojen eroja selitetään myös Godotin dokumentaatiossa.

Pelihahmon lisääminen tapahtui lisäämällä näyttämölle lapseksi KinematicBody2D-solmu, joka tarjoaa toimintoja pelifysiikoita noudattavan liikkuvan hahmon toteuttamiseen. Oikean solmutyypin löytäminen valikosta oli hidasta, koska se on sijoitettu usean aliluokan alle. Jos solmuluokan nimen tietää etukäteen, tarjottu hakutoiminto on hyödyllinen. Valmiilla solmuluokilla on myös uniikit kuvakkeet, jotka antavat käyttäjälle idean niiden suunnitelluista käyttötarkoituksista. Kuvassa 17 on valikko, josta luotava solmu valitaan.

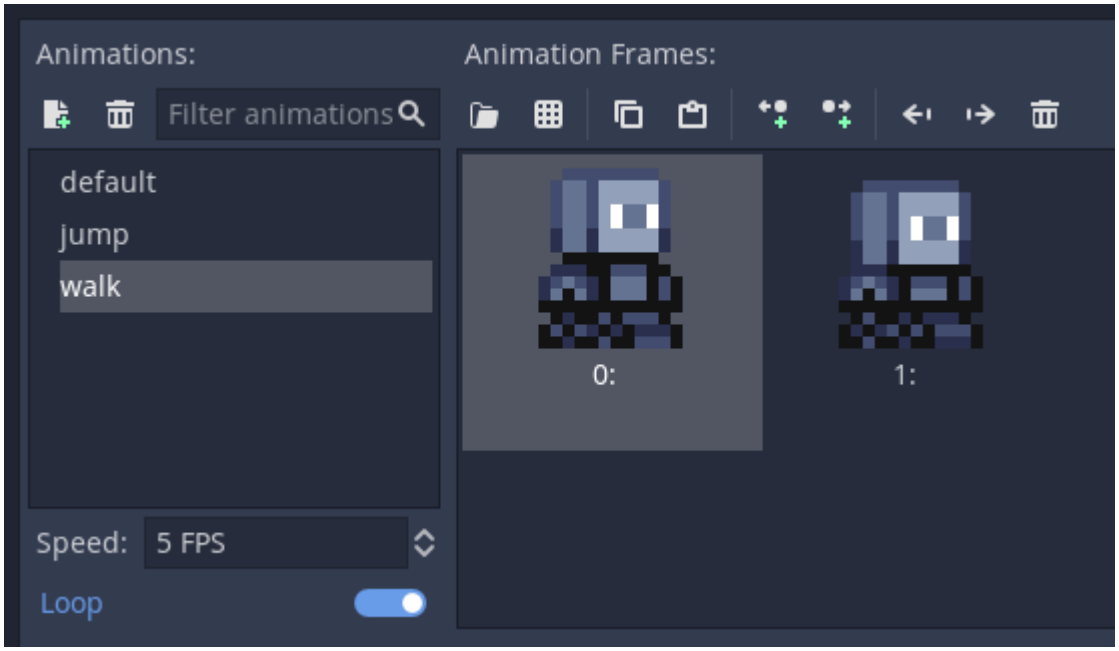


Kuva 17: Uuden solmun luomisikkuna (Linietsky & Manzur, 2022). Kirjoittajan ottama kuvakaappaus.

Kuvassa näkyy vain pieni osa solmuluokista. Godotin solmujärjestelmä ei ole aivan yhtä yksinkertainen kuin Multimedia Fusionin objektit, mutta se myös mahdollistaa paljon enemmän ja on looginen.

Peliin tuodut kuvat näkyivät aluksi sumeina, mikä johtui niiden tuontiasetuksista. Ongelma oli helposti korjattavissa vaihtamalla tuontiasetukset ohjelman vasemmassa reunassa olevasta valikosta 2D pixel -esiasetukseen (engl. preset), joka on tarkoitettu pienikokoisille kuville.

Animaation lisääminen oli merkittävästi intuitiivisempaa, vaikka siihen tarkoitettu työskentelyalue ei olekaan heti näkyvissä. AnimatedSprite- eli animaatiokuvasolmun ominaisuuksien tarkasteluvalikosta piti ensin luoda resurssi kuvaruuduille, jota painamalla ohjelman alareunaan avautuu animaatioiden muokkausvälilehti. Kuvia voi vaivattomasti raahata tiedostovalikosta suoraan animaatioon, liittää leikepöydältä tai yhdestä kuvatiedostosta voi leikata useamman ruudun. Lisäksi Godot tukee läpinäkyvien kuvien tuomista suoraan. Nämä vaihtoehdot tekevät työskentelystä joustavaa. Kuvassa 18 näkyy kuvakaappaus pelihahmon animaatioista.

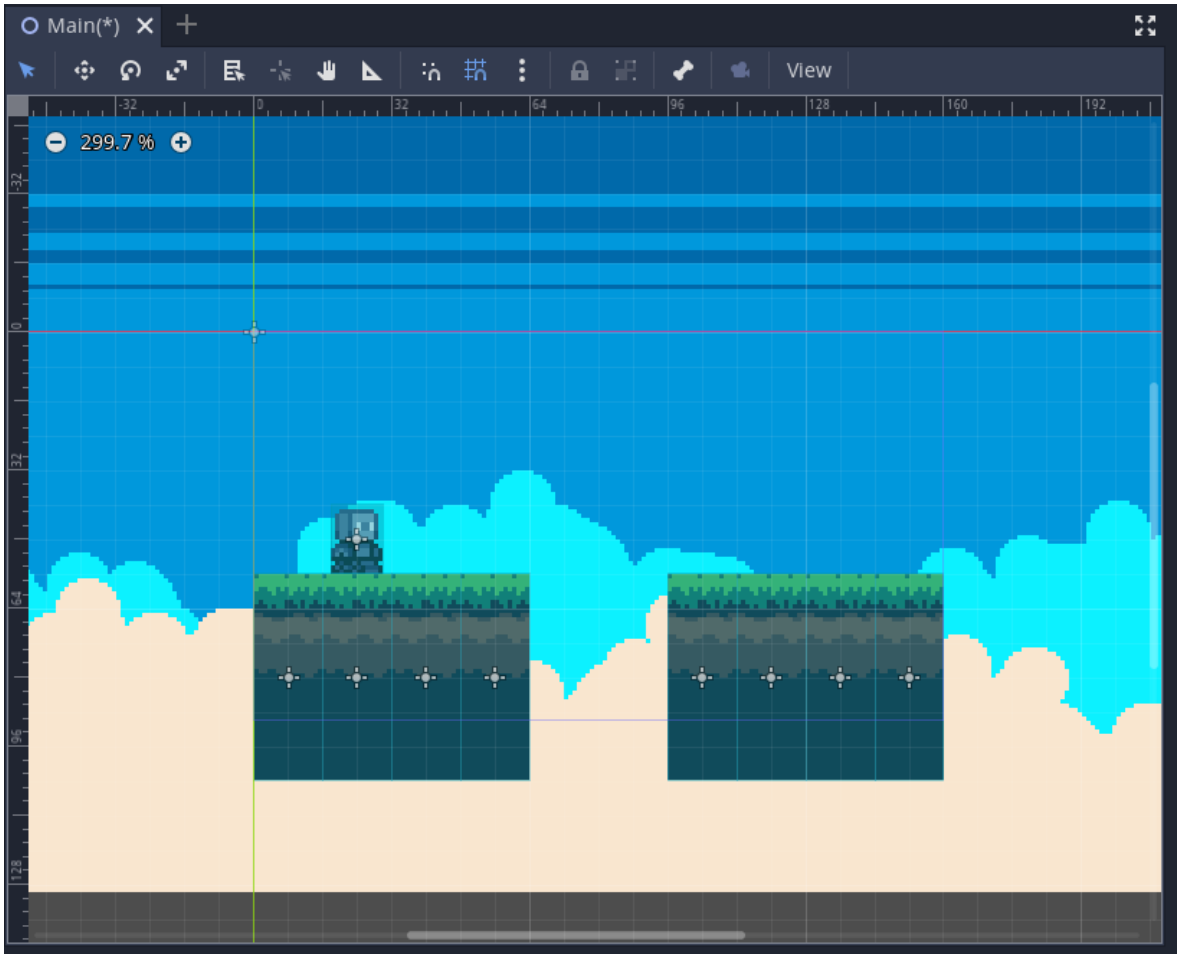


Kuva 18: Animaatioiden muokkausvälilehti (Linietsky & Manzur, 2022). Kirjoittajan ottama kuvakaappaus.

Kuvan vasemmassa laidassa näkyy lista kyseisen solmun animaatioista, jotka käyttäjä voi nimetä itse. Välilehti on rakenteeltaan samankaltainen kuin Multimedia Fusionin animaatioiden muokkausikkuna.

Jotta peliobjektit eivät liikkuisi toisistaan läpi, niille piti lisätä CollisionShape2D- eli kaksiulotteiset törmäysmuotosolmut. Godot näyttää niiden puuttuessa varoituskuvakkeen, mikä auttoi välttämään hämmennyksen. Muodon voi valita useista valmiista muodoista tai luoda oman. Tässä työssä käytettiin nelikulmioita, joiden säätäminen oikeisiin mittoihin onnistui helposti muuttamalla leveyttä ja korkeutta tarkasteluvalikosta.

Peliobjektien siirtäminen näyttämössä oli sujuvaa, ja siihen on useita aputyökaluja. Objekteja voi liikuttaa itse määritellyn ruudukon puitteissa, mikä tekee pelikentän tarkasta rakentamisesta helppoa. Lisäksi näkymää voi suurentaa ja pienentää. Kuvassa 19 on pelihahmo ja maasto aseteltuna peliympäristöön.



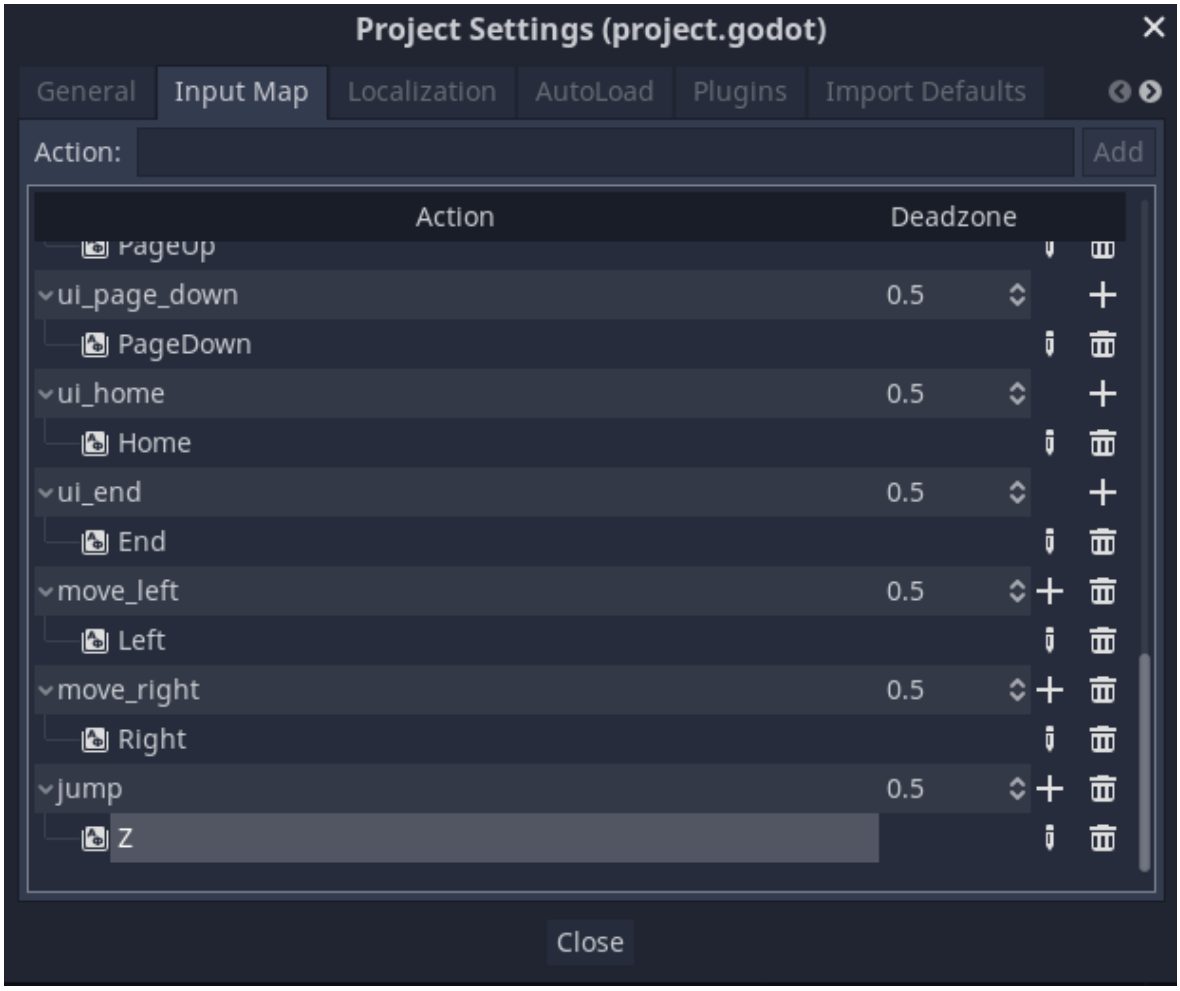
Kuva 19: Peliympäristön muokkausnäkymä (Linietsky & Manzur, 2022). Kirjoittajan ottama kuvakaappaus.

Kuvan näkymän ylälaidassa on kuvakkeita, joita painamalla saa esimerkiksi ruudukon näkyviin tai valitun solmun lukittua paikalleen. Myös törmäysmuodot näkyvät, mikä auttaa objektien rajojen hahmottamisessa.

Godot eroaa Multimedia Fusionista paljon pelilogiikan toteuttamisessa, koska siinä käytetään ohjelmointikieltä. Ohjelmointi tuo omat haasteensa, mutta sen avulla pelidemo saatiin helpommin tismalleen tavoitteiden mukaiseksi. Godotin suositeltuna ohjelmontikielenä on sen oma GDScript, joka on syntaksiltaan Python-kielen kaltainen korkean tason ohjelmointikieli (Godot Docs, 2022b). GDScriptin opettelemiseen löytyy kattavat ohjeet Godotin dokumentaatiosta. Sen lisäksi Godot tukee myös C#-kieltä, jota ei käytetä tässä työssä.

Godotissa jokaiseen solmuun voi liittää yhden skriptin eli komentosarjan. Näiden kirjoittamiseen ohjelmasta löytyy oma editori yläreunasta valittavana välilehtenä. Sisäänrakennettu editori toimi tutkimuksessa virheettömästi ja mahdollisti nopean vaihtamisen ohjelmoimisen ja pelinäyttämön muokkaamisen välillä.

Pelissä käytettävien syötteiden säätäminen ja lisääminen hoituu Godotissa projektiasetuksista, jossa voi lisätä äärettömän monta itse nimettyä abstraktia toimintoa ja yhdistää ne näppäimistön, hiiren tai peliohjaimen syötteisiin. Tämä kartoitus helpottaa pelilogiikan ohjelmointia merkittävästi, koska syötteisiin voi viitata koodissa niihin yhdistettyjen toimintojen nimellä. Esimerkki syötteiden kartoituksesta on kuvassa 20.



Kuva 20: Syötteiden kartoituksen asetusikkuna (Linietsky & Manzur, 2022). Kirjoittajan ottama kuvakaappaus.

Kuvassa näkyy jokaisen toiminnon alla lista syötteistä, jotka siihen on yhdistetty. Yhteen toimintoon voi siis yhdistää useita syötteitä, mistä on hyötyä, jos haluaa pelin tukevan useita syötelaitteita samanaikaisesti.

Pelin tasohyppelymekaniikat piti tehdä ilman samanlaista esirakennettua liikkumismuotoa kuin Multimedia Fusionissa, mikä teki kyseisestä askeleesta pidemmän Godotilla toteutettaessa. Toisaalta siihen löytyi helposti tarvittavat tiedot dokumentaatiosta, ja lopputulos on tavoitteiden mukainen. Äänitehosteen toistaminen onnistui käyttämällä siihen tarkoitettua solmua, jota kutsuttiin pelaajahahmon skriptissä. Kuvassa 21 on edellä mainittu skripti.

```gdscript
extends KinematicBody2D

const MOVE_SPEED      = 50.0    # Liikkumisvauhti
const JUMP_IMPULSE    = 100.0   # Hypyn korkeus
const FALL_ACCELERATION = 200.0 # Putoamiskiihtyvyys

var input_direction = Vector2.ZERO # Syötteen suunta
var velocity        = Vector2.ZERO # Pelaajan fyysinen nopeus

onready var audio = get_node("../AudioStreamPlayer")

func _physics_process(delta):
    # Syötteen päivittäminen
    input_direction.x = (Input.get_action_strength("move_right")
            - Input.get_action_strength("move_left"))

    if is_on_floor() and Input.is_action_just_pressed("jump"):
        velocity.y -= JUMP_IMPULSE
        $AnimatedSprite.play("jump")
        audio.play()

    # Pelaajan liikuttaminen
    velocity.x = input_direction.x * MOVE_SPEED
    velocity.y += FALL_ACCELERATION * delta
    velocity = move_and_slide(velocity, Vector2.UP)

    if not input_direction.x == 0:
        # Pelaajan kuvan kääntäminen syötteen suuntaan
        $AnimatedSprite.flip_h = (input_direction.x < 0)
    if is_on_floor():
        # Kävely- ja seisomisanimaatioiden toisto
        if not input_direction.x == 0:
            if not $AnimatedSprite.animation == "walk":
                $AnimatedSprite.play("walk")
        else:
            $AnimatedSprite.play("default")

    # Pelin aloittaminen uudelleen kentältä pudotessa
    if global_position.y > 90:
        get_tree().reload_current_scene()

```

Kuva 21: Pelaajahahmoon liitetty skripti

Kuvan skriptissä on kommenttirivit mukaan lukien 40 riviä, joten tässä työssä pärjättiin suhteellisen pienellä rivimäärällä.

Tekstin lisääminen peliin onnistui yhtä helposti kuin Multimedia Fusionillakin. Kuitenkin fontin vaihtaminen on monimutkaisempaa Godotilla, koska oletusfontin lisäksi se ei tarjoa

muita vaihtoehtoja valmiina. Sen sijaan omat fontit täytyy löytyä tiedostoina peliprojektin resurssikansiosta. Tämä hidastaa myös erilaisten fonttien kokeilua.

Jotta teksti tulisi näkyviin pelaajan osuessa maaliobjektiin, maaliobjektin oli tunnistettava törmäys. Tämä toteutettiin Godotin signaalijärjestelmällä, jonka avulla pelaajahahmon tuleminen maalin rajaamalle alueelle voitiin käsitellä skriptissä. Kyseinen toteutustapa ei ollut aivan yhtä suoraviivainen kuin Multimedia Fusionin tapahtumalauseen käyttäminen.

Godotilla pelin tekemiseen meni yhteensä noin 1 tunti ja 40 minuuttia. Yleisesti ottaen ohjelma sisältää paljon työskentelyä helpottavia ominaisuuksia. Lisäksi siinä on toimintoja lähes joka tarpeeseen. Dokumentaatio on selkeää, kattavaa, kätevästi saatavilla ja sisältää paljon esimerkkejä. Käyttöesimerkit dokumentaatiossa vähentävät virheitä, parantavat onnistumisprosenttia ja kehittäjien tyytyväisyyttä (Sohan *et al.*, 2017). Tuotettu peli on ulkonäöltään, toiminnallisuudeltaan ja ääniltään täysin tavoitteiden mukainen. Kuvassa 22 on kuvakaappaus valmiista pelistä.

Kuva 22: Lopullinen pelidemo Godotilla tehtynä

Kuvassa näkyvästä tekstistä voi havaita, että fontti skaalautuu venytetyn ikkunan kuvatarkkuuden mukaan. Tämän voisi välttää muuttamalla venytysasetuksia siten, että fontin pikselikoot noudattaisivat alkuperäistä kuvatarkkuutta.

4.4 Vertailu

Taulukossa 2 esitetään tiivistetty vertailu siitä, kuinka hyviksi eri asiat koettiin tutkimuksessa käytetyillä ohjelmistoilla. Arviot perustuvat aikatehokkuuteen ja saavutettuun lopputulokseen verrattuna tavoitteeseen. Arviot ovat asteikolla 1–5, jossa suurempi luku tarkoittaa parempaa. Viiva tarkoittaa sitä, että kyseistä asiaa ei kyetty arvioimaan kunnolla teknisistä syistä.

Taulukko 2: Ohjelmistoilla tehtyjen tehtävien tiivistetty vertailu

Ohjelma	Multimedia Fusion	Godot
Pelin kuvatarkkuuden asettaminen	5	5
Pelikuvan skaalaaminen	-	5
Ohjelman eri työskentelyalueiden välillä siirtyminen	2	4
Tarvittavien tietojen hakeminen dokumentaatiosta	-	5
Peliobjektien tarkka asettelu peliympäristöön	1	5
Kuvatiedostojen tuominen animaatioihin	2	5
Animaatioiden toistaminen	3	5
Pelaajahahmon liikkumisen toteuttaminen	3	4
Syötteiden kytkeminen toimintoihin	2	5
Äänitehosteen toistaminen	3	4
Tekstin lisääminen peliin	5	4
Tekstin näyttäminen pelaajan päästessä maaliin	5	5
Pelin aloittaminen alusta pelaajan pudotessa	5	5

Taulukon mukaisesti useimmissa asioissa Godot koettiin paremmaksi ajankäytön ja tavoite-vastaavuuden näkökulmasta. Muutamissa kohdissa Multimedia Fusion osoittautui yhtä hyväksi tai paremmaksi.

5 Keskustelu

Tässä luvussa kommentoidaan työn tuloksia. Kommentit keskittyvät tuloksien merkitykseen eri näkökulmista ja tutkimuksen riskien huomioimiseen.

Työn tuloksista voisi päätellä, että osa vanhemmasta ohjelmistosta tehdyistä havainnoista ovat olleet kehityskohteina 2000-luvun aikana julkaistuissa pelinteko-ohjelmistoissa. Se, että vanhempi Multimedia Fusion koettiin useilla tavoilla hankalakäyttöisemmäksi kuin uudempi Godot, voi tarkoittaa, että kehitys on selvästi havaittavissa tämän tutkimuksen asettamista näkökulmista. Tulokset voivat myös tarkoittaa, että pelinkehityksen harrastajan on huomattavasti helpompaa toteuttaa tavoitteidensa mukainen peli nykyaikaisten ohjelmistojen avulla.

Toimintojen määrä on tuloksissa tärkeässä asemassa ohjelmistoja vertaillessa, koska tavoitteita oli helpompaa saavuttaa Godotin tarjoamien toimintojen ansiosta. Tämä havainto vastaa toisen tutkimuksen tuloksia, joissa mainitaan viihdekäyttöön tarkoitettujen ohjelmien olevan suositumpia myös hyötypelien kehittämisessä ominaisuuksiensa vuoksi (Cowan & Kapralos, 2014). Lisäksi muun tutkimuksen johtopäätöksissä mainittiin, että pelimoottorien kehittäjien tulisi keskittyä dokumentaatioon (Politowski *et al.*, 2021). Tämä havainto näkyy myös tässä tutkimuksessa, koska Godotin laadukas dokumentaatio helpotti sillä työskentelyä paljon.

Työn ensimmäiseksi tutkimuskysymykseksi asetettiin seuraava: Miten pelien tekeminen on muuttunut helpommaksi pelinteko-ohjelmistojen kehittyessä? Tähän kysymykseen tulokset tarjoavat useita havaintoja, kuten parannukset animaatioiden toteutuksessa ja yleisessä työskentelymukavuudessa. Vanhemman ohjelmiston toimintojen rajoitteiden vuoksi tietyt tavoitteet eivät toteutuneet täysin sillä tehdyssä pelidemossa, kun taas uudemmalla ohjelmistolla yhtä huomattavia esteitä ei esiintynyt. Tämän voi huomata siitä, että kaikki asetetut tavoitteet toteutuivat lyhyemmässä ajassa Godotilla, vaikka aikaa kului koodin kirjoittamiseen toisin kuin Multimedia Fusionilla, jonka kanssa koodia ei tarvinnut kirjoittaa. Toiseksi tutkimuskysymykseksi valittiin seuraava: Mitkä asiat ovat pysyneet lähes yhtä haastavina tai muuttuneet haastavammiksi toteuttaa pelinteko-ohjelmilla? Tähänkin kysymykseen tulokset antavat esimerkkejä, kuten tekstin lisääminen ja pelin aloittaminen alusta. Pelihahmon

liikkumismuodon toteuttamisen voi ajatella olevan haastavampaa Godotilla, jos vastaavuutta tavoitteeseen ei ota huomioon.

Tuloksiin tuo epätarkkuutta se, että tutkimuksessa vertaillaan vain kahta ohjelmistoa keskenään. Tämä huomioitiin ohjelmistojen valinnassa siten, että niiden julkaisujen välinen aikaero olisi mahdollisimman suuri tapahtuneen kehityksen korostamiseksi. Tulokset ovat kuitenkin melko ohjelmakohtaisia, joten useammalla ohjelmalla tulokset olisivat tarkempia. Pelilogiikan toteuttamisen yksinkertaisuus eroaa ohjelmistoissa paljon, joten tutkimuksessa havainnoitiin sekä ajankäyttöä että tavoitteiden toteutumista, kun arvioitiin helppokäyttöisyyttä.

6 Yhteenveto

Tässä työssä tavoitteena oli tutkia sitä, miten pelinteko-ohjelmistot ovat kehittyneet ja muuttuneet helppokäyttöisemmiksi 2000-luvulla. Lisäksi tutkittiin niitä asioita, jotka ovat pysyneet yhtä haastavina tai muuttuneet haastavammiksi toteuttaa pelinteko-ohjelmistoilla. Aihetta tutkittiin tekemällä sama pelidemo vuonna 2001 julkaistulla Multimedia Fusionilla ja vuonna 2022 päivitetyllä Godotilla. Demojen toteutusprosessin aikana havainnoitiin asioita, jotka liittyivät tutkimuskysymyksiin.

Aihetta tutkittiin, koska peliteollisuuden merkitys kasvaa jatkuvasti, ja pelinteko-ohjelmistojen kehittyminen on mahdollisesti yksi siihen vaikuttava tekijä. Pelien kehittämisellä nuoret harrastajat voivat opetella ohjelmointia ja muita taitoja, joten sen helpottumisella voi olla positiivisia vaikutuksia. Tutkimuksen avulla voitiin myös löytää potentiaalisia kehityskohteita demon toteutuksessa käytetylle Godotille.

Tutkimuksen tuloksissa esitettiin demojen toteutuksen aikana tehdyt havainnot ja niiden pohjalta laadittu arviointitaulukko vanhemman ja uudemman ohjelman ominaisuuksista. Godotin helppokäyttöisyys arvioitiin paremmaksi seuraavissa asioissa: peliobjektien tarkka asettelu, kuvatiedostojen tuominen animaatioihin, animaatioiden toistaminen, pelaajahahmon liikkumisen toteuttaminen, syötteiden kytkeminen toimintoihin, äänitehosteen toistaminen ja ohjelman työskentelyalueiden välillä siirtyminen. Multimedia Fusionin helppokäyttöisyys puolestaan arvioitiin hieman paremmaksi tekstin lisäämisessä peliin. Dokumentaatiota ja pelikuvan skaalaamista ei pystytty ohjelmien välillä vertailemaan, koska Multimedia Fusionin kohdalla ne eivät toimineet tutkimuksen toteutusympäristössä. Muissa asioissa ohjelmat koettiin pitkälti yhtä hyviksi.

Tuloksista voisi tehdä johtopäätöksen, että tutkimuksessa havaittuihin vanhemman ohjelman hankaliin puoliin on löydetty ratkaisuja 2000-luvun uudemmissa pelinteko-ohjelmistoissa. Ensimmäiseen tutkimuskysymykseen tulokset vastaavat, että huomattavia parannuksia on tapahtunut pelinteko-ohjelmistojen käyttömukavuuden, ohjelmavirheettömyyden, rajoittavien tekijöiden ja työskentelytehokkuuden osalta. Toiseen tutkimuskysymykseen saatiin vastaus, että pelinteko-ohjelmistojen toimintojen suuremman määrän takia yksinkertaisuudesta on jouduttu karsimaan esimerkiksi tekstin lisäämisessä peliin.

Jatkossa tuloksia voisi hyödyntää siten, että niiden avulla esitettäisiin konkreettisia parannuksia pelinteko-ohjelmistoissa. Jos aiheesta tehtäisiin jatkotutkimus, siinä voisi verrata useampia ohjelmistoja. Koska Godot on avoimen lähdekoodin ohjelma, tuloksien pohjalta sitä voisi myös kehittää yhä helppokäyttöisemmäksi.

Lähteet

Ahmad, F.N. (2013) ‘An Overview Study of Game Engines’, 3(5), s. 1673–1693.

Albers, M.J. (2011) ‘Design and Usability: Beginner Interactions with Complex Software’, *Journal of Technical Writing and Communication*, 41(3), s. 271–287. Saatavissa: https://doi.org/10.2190/TW.41.3.d.

Clickteam (n.d.) 'ChocoBreak' [videopeli].

Ciesla, R. (2017) *Mostly Codeless Game Development: New School Game Engines*. 1st ed. 2017. Berkeley, CA: Apress : Imprint: Apress. Saatavissa: https://doi.org/10.1007/978-1-4842-2970-5.

Clickteam (2001) 'Multimedia Fusion 1.5' [ohjelmisto]. Haettu osoitteesta http://web.archive.org/web/20060626234953/http://www.clickteam.com/English/download_section.php?PID=8&SID=5&Sname=Main+Download.

Conradi, R. & Wang, A.I. (eds) (2003) *Empirical methods and studies in software engineering: experiences from ESERNET*. Berlin ; New York: Springer (Lecture notes in computer science, 2765).

Cowan, B. & Kapralos, B. (2014) ‘A Survey of Frameworks and Game Engines for Serious Game Development’, teoksessa *2014 IEEE 14th International Conference on Advanced Learning Technologies*. *2014 IEEE 14th International Conference on Advanced Learning Technologies (ICALT)*, Athens, Greece: IEEE, s. 662–664. Saatavissa: https://doi.org/10.1109/ICALT.2014.194.

Doss, K. *et al.* (2011) ‘Work in progress - A survey of popular game creation platforms used for computing education’, teoksessa *2011 Frontiers in Education Conference (FIE)*. *2011 Frontiers in Education Conference (FIE)*, Rapid City, SD, USA: IEEE, s. F1H-1-F1H-2. Saatavissa: https://doi.org/10.1109/FIE.2011.6143110.

Dupire, J., Topol, A. & Cubaud, P. (2005) ‘USING GAME ENGINES FOR NON 3D GAMING APPLICATIONS’, s. 1–4.

Godot Docs (2022a) ’Godot's design philosophy’ [verkkoaineisto]. Saatavissa: https://docs.godotengine.org/en/stable/getting_started/introduction/godot_design_philosophy.html (Viitattu 30.12.2022).

Godot Docs (2022b) ’GDSCript basics’ [verkkoaineisto]. Saatavissa: https://docs.godotengine.org/en/3.5/tutorials/scripting/gdscript/gdscript_basics.html (Viitattu 4.1.2023).

Frøkjær, E., Hertzum, M. & Hornbæk, K. (2000) ‘Measuring usability: are effectiveness, efficiency, and satisfaction really correlated?’, teoksessa *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. *CHI00: Human Factors in Computing Systems*, The Hague The Netherlands: ACM, s. 345–352. Saatavissa: https://doi.org/10.1145/332040.332455.

Guevarra, E.T.M. (2020) *Modeling and Animation Using Blender: Blender 2. 80: the Rise of Eevee*. Berkeley, CA: Apress L.P.

Hudlicka, E. (2009) ‘Affective game engines: motivation and requirements’, teoksessa *Proceedings of the 4th International Conference on Foundations of Digital Games - FDG ’09*. *the 4th International Conference*, Orlando, Florida: ACM Press, s. 299–305. Saatavissa: https://doi.org/10.1145/1536513.1536565.

Kazman, R., Bass, L. & Bosch, J. (2003) ‘Bridging the gaps between software engineering and human-computer interaction’, teoksessa *25th International Conference on Software Engineering, 2003. Proceedings. 25th International Conference on Software Engineering, 2003. Proceedings.*, Portland, OR, USA: IEEE, s. 777–778. Saatavissa: https://doi.org/10.1109/ICSE.2003.1201281.

Linietsky, J. & Manzur, A. (2022) 'Godot Engine 3.5.1' [ohjelmisto]. Saatavissa: https://godotengine.org/download.

Mishra, P. & Shrawankar, U. (2016) ‘Comparison between Famous Game Engines and Eminent Games’, *International Journal of Interactive Multimedia and Artificial Intelligence*, 4(1), s. 69–76. Available at: https://doi.org/10.9781/ijimai.2016.4113.

Politowski, C. *et al.* (2021) ‘Are game engines software frameworks? A three-perspective study’, *Journal of Systems and Software*, 171, s. 1–10. Saatavissa: https://doi.org/10.1016/j.jss.2020.110846.

Sharif, K.H. & Yousif Ameen, S. (2021) ‘Game Engines Evaluation for Serious Game Development in Education’, teoksessa *2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. *2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Split, Hvar, Croatia: IEEE, s. 1–6. Saatavissa: https://doi.org/10.23919/SoftCOM52868.2021.9559053.

Sohan, S.M. *et al.* (2017) ‘A study of the effectiveness of usage examples in REST API documentation’, teoksessa *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Raleigh, NC: IEEE, s. 53–61. Saatavissa: https://doi.org/10.1109/VLHCC.2017.8103450.

Vohera, C. *et al.* (2021) ‘Game Engine Architecture and Comparative Study of Different Game Engines’, teoksessa *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Kharagpur, India: IEEE, s. 1–6. Saatavissa: https://doi.org/10.1109/ICCCNT51525.2021.9579618.

Wikipedia (2019) ’Category:Video game development software’ [verkkoaineisto]. Saatavissa: https://en.wikipedia.org/wiki/Category:Video_game_development_software. (Viitattu 3.12.2022).