Opinnäytetyö (AMK)

Tietotekniikka

Mediatekniikka

2014

Sami Eklund

2D-ANIMAATIOTYÖKALUT

– kartoitus ja alustava suunnittelu HactEnginelle

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Mediatekniikka

2014 | 38

Ohjaaja yliopettaja, FT Mika Luimula

Sami Eklund

2D-ANIMAATIOTYÖKALUT– KARTOITUS JA ALUSTAVA SUUNNITTELU HACTENGINELLE

Tämä opinnäytetyö toteutettiin Indium Games -nimiselle, pelialalla toimivalle startup-yritykselle. Indium Games on kehittänyt omaa pelimoottoriaan jo jonkin aikaa ja pelimoottori on nimetty HactEngineksi. Opinnäytetyön tarkoituksena on kartoittaa ja tehdä alustava suunnitelma HactEngineen tuleville animaatiotyökaluille, työ rajataan 2D-animaatiotyökaluihin.

Opinnäytetyössä tutkittiin jo olemassa olevien ohjelmien ratkaisuja tutoriaalin omaisesti, jotta niistä sai tarkankuvan. Työssä käytiin läpi kolme ohjelmaa, jotka liittyvät animaatioiden tai pelien tekemiseen. Nämä ohjelmat olivat Adobe After Effects, Unity3D ja Godot, ohjelmat valittiin eri syistä. Unity3D on nykyään paljon käytetty pelikehittäjien keskuudessa. Godot-pelimoottori on taas tänä keväänä tullut avoimenlähdekoodinlisenssin alaiseksi. After effects ei ole pelimoottori vaan videoneditointiohjelma, toisin kuin Unity3D ja Godot, joten se antaa erilaisen näkökulman.

Eri ratkaisuja tutkittiin tekemällä yksinkertainen hahmon kävelyanimaatio. Animaatio toteutettiin käyttämällä samaa hahmoa, mutta se muokattiin aina kunkin ohjelman vaatimusten mukaan, jotta sitä voitiin käyttää animointiin.

Vertailuiden perusteella voitiin todeta, mitkä ovat tavanomaisimmat animaatiotyökalut ja miten ne toimivat. Sen pohjalta pystyttiin myös hyvin poimimaan ohjelmista niin hyviä ratkaisuja kuin ratkaisuja, joita voidaan käyttää HactEnginessä jatkokehittämisen jälkeen. Tuloksien avulla pystyttiin suunnittelemaan alustavasti, millaiset 2D-animaatiotyökalut voisivat HactEnginessä olla, ja miten niitä voitaisiin jatkossa kehittää paremmaksi.

.

ASIASANAT:

HactEngine, animaatio, pelimoottori, Unity3D, After Effects, Godot

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Media Technology

2014 | 35

Instructor Principal Lecturer, Phd Mika Luimula

Sami Eklund

2D-ANIMATION TOOLS – MAPPING AND A PRELIMINARY PLAN FOR HACTENGINE

This thesis was made for Indium Games, which is a startup company that works in the field of games. Indium Games has been developing its own game engine for a while, the engine is named HactEngine. The purpose of this thesis is to map and do a preliminary plan for the animation tools for HactEngine, the thesis was defined to 2D animation tools.

In the thesis, solutions from existing programs were compared. There were three different programs that were compared, which are used to do animation or games. These programs were Adobe After Effects, Unity3D and Godot, the programs were selected for different reasons. Unity3D is currently very popular among game developers. Godot was released under a open source license this spring. Unlike the other two After Effects is not a game engine, so it can give a different perspective.

The different solutions of these programs were compared by doing a simple walking cycle with them. The animation was done with the same character, but it was modified when necessary if needed, so that it could be used with the programs for animation.

Based on the comparison, it was possible to list the most common animation tools and how they work. It was also possible to find usable solutions from the programs and also solutions that could be useful to HactEngine after developing then further. With the results it was possible to make a preliminary plan for HactEngines 2D animation tools and how they might be developed further in the future

.

KEYWORDS:

HactEngine, animation, game, engine, Unity3D, After Effects, Godot

SISÄLTÖ

KUVAT

1 JOHDANTO

Tämä opinnäytetyö tehdään turkulaiselle pelialalla toimivalle startup-yritykselle nimeltään Indium Games. Indium Games on jo jonkin aikaa kehittänyt omaa pelimoottoria. Moottoria ei ole rakennettu minkään olemassa olevan pelimoottorin pohjalta, vaan se on alusta asti tehty itse. Pelimoottori on nimetty HactEngineksi, ja se tullaan lyhyesti esittelemään tässä opinnäytetyössä, sillä se asettaa pohjan suunnittelutyölle. Opinnäytetyön tarkoituksena on luoda suunnitelma animaatiotyökalusta HactEngineen, aihe rajataan koskemaan 2D-animaatiotyökaluja. Animaatiotyökaluilla olisi tarkoitus pystyä luomaan peleissä käytettävää animaatiota, esimerkiksi hahmoanimaatioita. Animaatiotyökalujen suunnittelussa täytyy huomioida HactEnginen asettamat vaatimukset, etenkin jos pohditaan valmiin ratkaisun integroimista pelimoottoriin. HactEnginessä on jo valmiina värimäärittelyt esimerkiksi käyttöliittymän kannalta.

Opinnäytetyö koostuu kahdesta eri osa-alueesta. Työn teoreettisessa osassa tutkitaan jo olemassa olevia ohjelmia, joissa voidaan toteuttaa animaatioita. Tavoitteena on kartoittaa tarvittavat työkalut ja niiden toiminta, jotta niiden käyttö olisi mielekästä. Koska kyseessä on pelimoottori, keskitytään animaatiotekniikoista vain pelinkehityksen kannalta oleellisiin tekniikoihin. Opinnäytetyöhön on valittu kolme ammattikäytössä olevaa ohjelmaa, joilla toteutetaan lyhyt animaatio eri tekniikoilla. Tällä tavalla pyritään kartoittamaan ohjelmista parhaimmat ratkaisut, joista voitaisiin ottaa mallia.

Työn toisessa osassa suunnitellaan löydösten perusteella animaatiotyökaluja sekä selvitetään, millainen käyttöliittymä animaatio-osiolla tulisi olla HactEnginessä. Suunnitelmassa eritellään tarvittavat työkalut ja niiden ominaisuuksia mahdollisimman tarkasti. Lisäksi pyritään selkeästi esittämään työkalujen väliset suhteet ja sen, miten ne rakentavat yhdessä toimivan kokonaisuuden. Tavoitteena on myös esittää ehdotus työkalujen ulkoasusta ottaen huomioon HactEnginen asettamat vaatimukset. Pelimoottorin toteutusta pohditaan yhteistyössä HactEnginen kehitystiimin kanssa.

2 ANIMAATIOTEKNIIKAT

2.1 Pala-animaatio

Cutout- eli pala-animaatiossa nimensä mukaisesti käytetään paloja animaation luomisessa. Pala-animaatiossa hahmot ja ympäristö koostuvat paloista, joita animaattori liikuttaa kameran edessä luoden illuusion liikkeestä. Tällöin jokaista eri asentoa ei tarvitse piirtää erikseen, vaan ne voidaan luoda asettelemalla palat uusiksi haluttuun asentoon. [1]

2.2 Stopmotion-animaatio

Stopmotion-animaatiossa on kyse liikkeen pysäyttämisestä. Tässä tekniikassa voidaan käyttää esimerkiksi kameraa ja nukkea. Kameralla otetaan kuva nukesta, minkä jälkeen nukkea liikutetaan vähän. Mitä enemmän kuvia saadaan liikeradasta, sitä sujuvammalta liikeanimaatio näyttää. [1]

2.3 Cel-animaatio

Cel-animaatio on animaatiotekniikka, jossa käytetään läpinäkyviä kalvoja. Animaattori maalaa animaatioon liittyviä elementtejä kalvoille tarpeen mukaan, jotka sitten valokuvataan. Paikallaan pysyvän taustalle riittää yksi kalvo, mutta hahmon liike vaatii niitä kuitenkin enemmän. Hahmo animaatiossa animaattori tekisi yhdestä liikeradasta useita kalvoja, joita sitten näytetään animaattorin määrittelemässä järjestyksessä, näin kalvo sarja luo liikeanimaation. Disneyn vanhoja animaatioita tehtiin tällä tekniikalla.[1]

2.4 Digitaalinen 2D-animaatio

Nykyään harvemmin käytetään perinteisiä animaatiotekniikoita animaatioiden tuottamiseen. Digitaaliset ohjelmat ovat kehittyneet, ja niiden avulla on animaa-

tioiden luonti helpompaa. Helppoudella tarkoitetaan tässä tuotannon suoraviivaistamista ja toistuvien työtehtävien pois jättämistä. Digitaalisessa animaatiossa tulevat usein vastaan pala- ja cel-animaatiotekniikoiden tapaisia tekniikoita..

Peleissä molemmat mainitut tekniikat ovat hyvin edustettuna. Disneyn käyttämä cel-animaatotekniikka, jossa hahmon animaatio piirretään kuva kuvalta, on hyvin samankaltainen sprite sheet -animaation kanssa. Sprite-sana viittaa pelialalla yleensä kuvaan, joka on piirretty läpinäkyvälle taustalle esimerkiksi png-tiedostomuotoon.

Sprite-animaatiossa hahmon liike piiretään kuva kuvalta. Tämän jälkeen pelille kerrotaan, missä järjestyksessä kuvia luetaan ja kuinka nopeasti. Näin pelaaja näkee kuvien sijaan esimerkiksi kävelyanimaation. Sprite-kuvat sisältävää kuvaa kutsutaan sprite sheet -kuvaksi. Yksi sprite sheet -kuva voi sisältää kymmeniä kuvia vain yhtä animaatiota varten. Mitä enemmän liikkeessä on kuvia, sitä sulavammalta animaatio näyttää. [2]

Pala-animaatio on myös paljon käytetty tekniikka. Sen jonka etuna on, että animaattori merkitsee pisteet ja tekee näissä kohdissa muutoksia hahmon asentoon, ohjelma laskee näiden pisteiden väliset kuvat. Näin animaattorin ei tarvitse tehdä kuin liikkeen aloitus- ja lopetustilat, koska ohjelma liikuttaa paloja ja luo tilojen väliset kuvat automaattisesti. Esimerkiksi suosittu South Park –sarja on luotu käyttämällä tätä tekniikkaa.[1] Oman kokemukseni mukaan pelimoottorit sallivat näiden molempien tapojen käytön tai jonkinlaisen oman sovellutuksen näistä tekniikoista. Kuten aikaisemmin mainittiin, eri tekniikat eivät ole toisiaan poissulkevia.

2.5 3D-animaatio

3D-animaation tekemiseen käytetään usein 3D-ohjelmaa esim. Blender-ohjelmistoa. Nykyään useat pelimoottorit tukevat 3D-objekteja ja niitä voidaan hyödyntää eri tavoin pelimoottorissa. Yleensä pohjustustyöt eli mallinnus ja riggaus tehdään erillisellä 3D-ohjelmalla, mutta animaatiot tehdään pelimoottorissa.

Riggauksella tarkoitetaan työvaihetta, jossa luodaan hahmolle luuranko. Tämä luuranko sisältää niveliä ja luita. Luurangon avulla saadaan siis hahmolle taipuvia osia esimerkiksi kädet tai jalat. 3D-grafiikan etuna on, että objekteja ja hahmoja ei tarvitse piirtää eri kulmista kuvakulman vaihtuessa, kuten 2D-grafiikan kohdalla. [3]

3 AMMATTIKÄYTÖSSÄ OLEVAT OHJELMAT

Tässä osassa kuvataan ammattikäytössä olevia ohjelmia, joita on käytetty pelien tai animaatioiden tekemiseen. Jotta jokaisesta ohjelmasta saataisiin samankaltainen kokemus, toteutetaan kaikilla ohjelmilla sama hahmoanimaatio. Työkuvataan tutoriaalin omaisena, jotta työkalujen toiminta olisi selkeä. Tavoitteena on kartoittaa ohjelmien hyvät ratkaisut, jotta niistä voitaisiin rakentaa toimiva kokonaisuus HactEngineen. Työkalujen tutkimisen ohella tarkastellaan ohjelmien käyttöliittymään liittyviä ratkaisuja ja poimitaan niistä käyttökelpoisimmat ideat.

3.1 Adobe After Effects

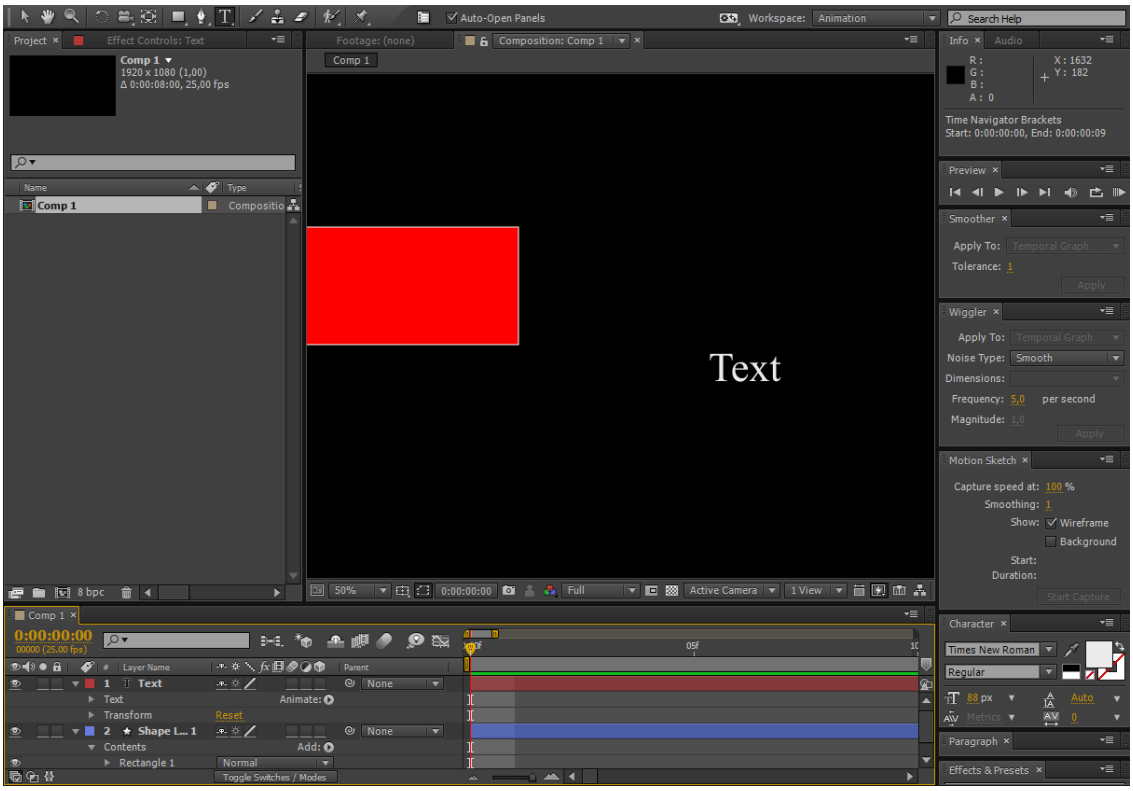
Adobe After effects on Adoben tuoteperheeseen kuuluva videoiden editointiohjelma, jota käytetään paljon myös erikoisefektien lisäämiseen ja animaatiovideoiden luomiseen [4]. Tässä työssä olen käytettiin After Effects -ohjelman CC eli Creative Cloud –versiota, ja DuIK -lisäosasta käytettiin versiota 10. Animaatiotyökaluja tutkittiin tekemällä ohjelmien työkaluilla hahmon kävelyanimaatio.

After Effects -ohjelmaa ei sinänsä suoranaisesti käytetä pelien animaatioiden tekoon, mutta se on hyvin käytetty animaatioiden tekemiseen muihin tarkoituksiin. Animaatioiden teossa käytetään kuitenkin paljon samoja tekniikoita. Ohjelma ei siis ole suoranaisesti pelin tekijöiden käytössä, mutta se voi sisältää silti hyviä ratkaisuja animaatiotyökalujen toteutukseen liittyen.

After Effects tukee myös monia tiedostotyyppejä kuten esimerkiksi muiden Adobe-tuotteiden ja monen 3D-ohjelman tiedostotyyppejä. Esimerkiksi Adobe Photoshop -tiedoston voi tuoda ohjelmaan yhtenä tiedostona tai suoraan sellaisenaan, jolloin siinä säilyy Photoshop-ohjelmassa määritellyt tasot. [4]

3.1.1 Käyttöliittymä

After Effectin käyttöliittymä on toimiva, koska se on selkeä ja sallii käyttäjän tehdä siihen muutoksia (Kuva 1.). Käyttöliittymä rakentuu työkalupalkista ja muutamasta pääruudusta, jotka ovat projekti-ikkuna, aikajana ja videoikkuna. [5]



Kuva 1. After Effects käyttöliittymä.

Projekti-ikkuna sisältää tietoa projektista, sen sisään listautuu kaikki videoon tuodut tiedostot ja siihen luodut kompositiot. [5]

Ohjelmassa yksittäistä ”kansiota” kutsutaan kompositioksi, ja se voi sisältää esimerkiksi toisia kompositioita. Käytetään esimerkkinä hahmoanimaatiota, joka on toteutettu pala-animaatiotekniikalla. Hahmon eri osat, kädet, jalat, torso ja pää voivat olla eri kompositioita. Esimerkiksi pääkompositio voi sisältää kompositiot silmille ja suulle, jotka sisältävät animaatiota. Liikuttaessa päätä liikkuvat

myös pääkomposition sisälle tehdyt osat sen mukana. Tällä tavoin voidaan luoda hyvinkin monimutkaista animaatiota. [6]

Aikajanalla näkyy valitun komposition tasot ja niihin asetetut avainkuvat eli keyframe-pisteet. Aikajanalla tehdään myös halutut transformaatiot. Janaa voidaan myös zoomata, jolloin avainkuvat voidaan asettaa hyvinkin tarkasti halutuun kohtaan. [5]

Kolmas oleellinen ikkuna on itse videoikkuna. Tässä ikkunassa näkyy, miltä video näyttää kulloisellakin hetkellä. On siis ymmärrettävää, että nämä kolme ikkunaa ovat kytköksissä toisiinsa ja siksi yleensä aina samanaikaisesti näkyvissä. [5]

After Effects sisältää myös paljon lisäikkunoita, joita käyttäjä voi tuoda näkyville niin halutessaan. Kaikilla näillä ikkunoilla on oma tarkoituksensa. Kuitenkin ohjelma tarjoaa käyttäjälle valmiita ikkuna-asetelmia esimerkiksi animaation tekoon. Tällaisen valmiin asetuksen valittuaan käyttöliittymä muokkaantuu sisältämään ennalta määrätyt ikkunat. Käyttäjä voi silti siirtää, poistaa tai lisätä ikkunoita haluamallaan tavalla. On myös mahdollista asettaa omia ikkuna-asetelmiaan ohjelman muistiin myöhempää käyttöä varten. [5]

After Effectsissä erittäin toimivaa on ikkunoiden käytön vapaus. Käyttäjä voi itse valita niiden paikan, sillä ne ankkuroituvat helposti toisiin ikkunoihin. Käyttäjä voi myös itse valita niiden koon, jolloin pudotusvalikot vaihtuvat listoiksi tai toisinpäin. [3]

Värimaailmaltaan käyttöliittymä on myös hyvä, se on selkeä ja värit ovat sovussa keskenään. Myöskään painikkeiden ja muiden osien animaatiot eivät ole häiritseviä mutta silti sulavia. Kaikkien painikkeiden toiminnot eivät heti selviä vain katsomalla tai edes välttämättä heti kokeilemalla joten ne vaativat perehtymistä.

3.1.2 Sprite sheet -animaatio

After Effectsin aikajanaa voidaan lähentää niin paljon, että videon erilliset kuvat eli framet erottuvat, joten sillä voidaan tehdä perinteistä ns. kuva kuvalta eli

frame by frame -animaatiota. Ohjelmassa ei voi itse piirtää eikä siinä ole laajaa valikoimaa työkaluja, joilla luoda monimutkaista grafiikkaa. Animaatiossa käytettävä grafiikka on siis tehtävä toisella ohjelmalla ja tuotava After Effects -ohjelmaan. Varsinaista tukea ei ohjelmassa ole sprite sheet -kuvan käyttöön. Hahmoanimaatiossa hahmon eri liikekuvat voidaan tehdä eri tasoille. Kuten esimerkiksi Photoshopissa ja tiedosto voidaan tuoda siten sellaisenaan After Effectsiin. Animaatioon vaaditut kuvat voidaan tuoda myös erillisinä sprite-kuvina ja järjestellä eri tasoille After Effectsissä. Tämän jälkeen kuvat voidaan järjestää aikajanalle, jossa käyttäjä voi asettaa kuinka, kauan kukin kuva on näkyvissä ja kuinka kauan animaatio kestää. [7]

3.1.3 Pala-animaatio

After Effects on hyvä ohjelma pala-animaation tuottamiseen. Toiminnaltaan se muistuttaa paljon Adobe Flash -ohjelmaa, koska molemmat kuuluvat samaan Adobe-tuoteperheeseen.

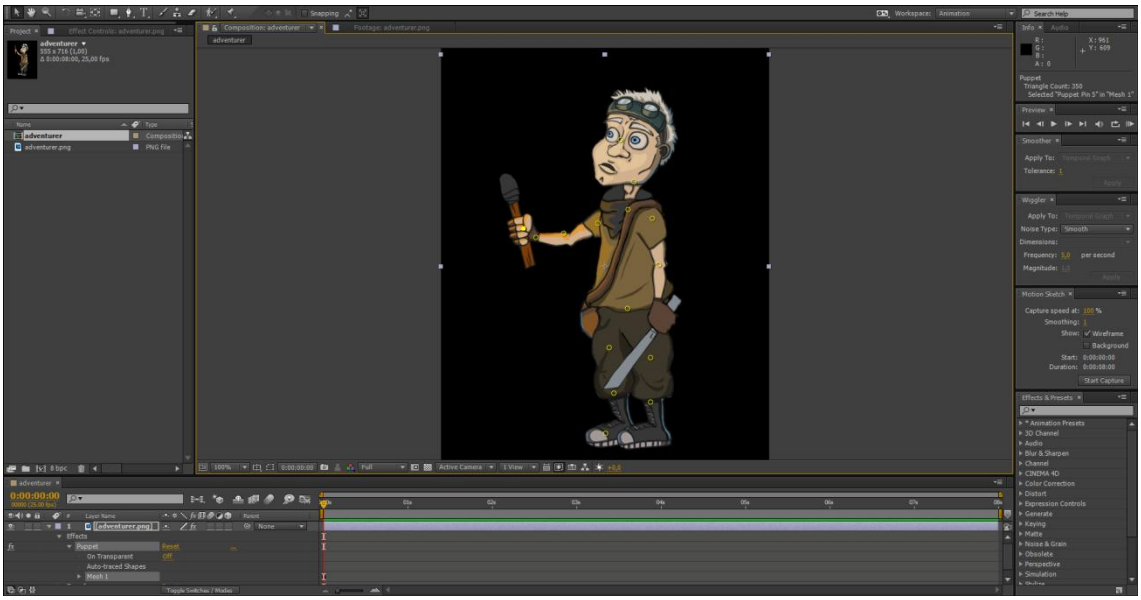
Kompositioon tuodut osat ovat kompositiossa omilla tasoillaan: ylin taso on aina lähempänä kameraa. Jokaisella eri tasolla eli osalla on oma rivinsä aikajanalla. Näitä osia voidaan liikuttaa erikseen, ja niille voidaan asettaa omat avainkuvat eli keyframe-pisteet. [8]

Avainkuva on tärkeä ja hyvin hyödyllinen työkalu. Avainkuvalla merkitään muutoksen kohta aikajanalla. Muutos voi olla esimerkiksi kuvan läpinäkyvyys, sijainti ruudulla tai kierto. Avainkuvan avulla merkitään lähtö- ja loppukohta, ohjelma tekee itse tilojen väliset kuvat ja luo näin katkeamattoman animaation. Monessa ohjelmassa, kuten Flashissa, tätä on kutsuttu nimillä ”tween” tai ”motion tween”. Avainkuvien sijainnilla voidaan vaikuttaa muutoksen nopeuteen esimerkiksi siirtämällä kahta avainkuvaa lähemmäs toisiaan, muutos tapahtuu nopeammin. Näin voidaan helposti kertoa ohjelmalle, milloin mitäkin pitäisi tapahtua ja mihin osaan se vaikuttaa. Jokaiselle muutokselle voidaan asettaa omat avainkuvansa, joten on mahdollista asettaa osalle monta muutosta tapahtumaan saman animaation aikana. Esimerkiksi voidaan määrätä laatikon liikkuvan ruudun halki

samalla kiertyen keskikohtansa ympäri ja samalla välkkyen. Pala-animaatiossa tärkeää on ryhmitellä osat oikein kompositioihin, jotta ne liikkuvat järkevästi kokonaisuuksien mukana. [8]

3.1.4 Puppet tool –työkalu

After Effects sisältää työkalun nimeltä Puppet tool Tällä työkalulla kuvasta voidaan luoda eräänlainen nukke, jolloin sen eri osia voidaan liikuttaa. Kun hahmon sprite-kuva on tuotu After effectsiin ja sille on luotu kompositio, voidaan sitä muokata Puppet tool -työkalun avulla. Työkalulla voidaan lisätä kuvassa 2. näkyviä keltaisia pisteitä hahmon taittokohtiin esimerkiksi kaulaan ja niveliin. Työkalu kartoittaa kuvan ja tekee siitä eräänlaisen ”meshin”, jolloin kuvan päälle rakentuu verkko. Hahmon raajoja voidaan nyt liikuttaa, ja ne taipuvat merkittyjen pisteiden kohdalta. Kuitenkin usein kyseessä on rasteroitua grafiikkaa, mikä tässä tapauksessa tarkoittaa sitä, että liikuttaessa raajoja, kuva usein venyy. Tuloksena voi olla suttuisuutta ja kuvan vääristymistä. [9]



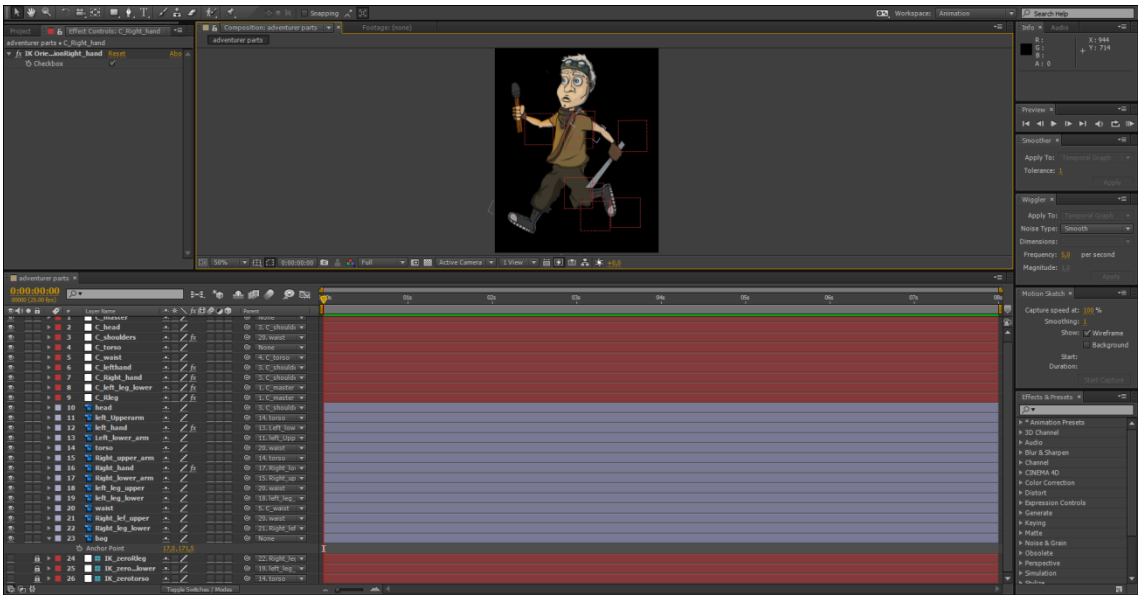
Kuva 2. Puppet tool -työkalu.

Puppet tool -työkalu on varmasti hyvä pienissä animaatioissa ja siinä pystyy liikuttamaan esimerkiksi raajoja kuvan päälle, vaikka kuvassa ei olisi ollut tasoja

alun perin. Omasta mielestäni se ei kuitenkaan vastaa sellaisenaan HactEnginen tarpeita, eikä se ole erityisen monikäyttöinen työkalu. Kuitenkin sitä voisi hyödyntää osana isompaa kokonaisuutta. Sen avulla voisi esimerkiksi korjata tapahtuneita vääristymiä.

3.1.5 DuIK-lisäosa

DuIK on ilmainen lisäosa After Effects -ohjelmaan, tai tarkemmin sanottuna se on JavaScript-tiedosto, joka lisää muutamia toimintoja ohjelmaan. Se sisältää muutamia ongelmia mutta kuitenkin se on saanut hyvää palautetta. Se tuo After Effects ohjelmaan uutena työkaluna 2D-riggauksen. (kuva 3.). [10]



Kuva 3. DuIK lisäosa.

DuIK toimii vain pala-animaation kanssa, sillä se vaatii eri osia joiden välille rakennetaan suhteita. DuIKissa voidaan luoda suhde 2–3 palan ja ohjaimen välille; kuitenkaan paloja ei voi olla tätä vähemmän tai enemmän. Esimerkiksi hahmon raajat ovat usein kolmessa osassa, jotta niitä voidaan liikuttaa mahdollisimman luontevasti. [10]

DuIK kutsuu tätä IK lyhenteellä, joka on lyhenne sanoista inverse kinematics, DuIK antaa myös mahdollisuuden päättää kääntösuunnan, jonka avulla raajat

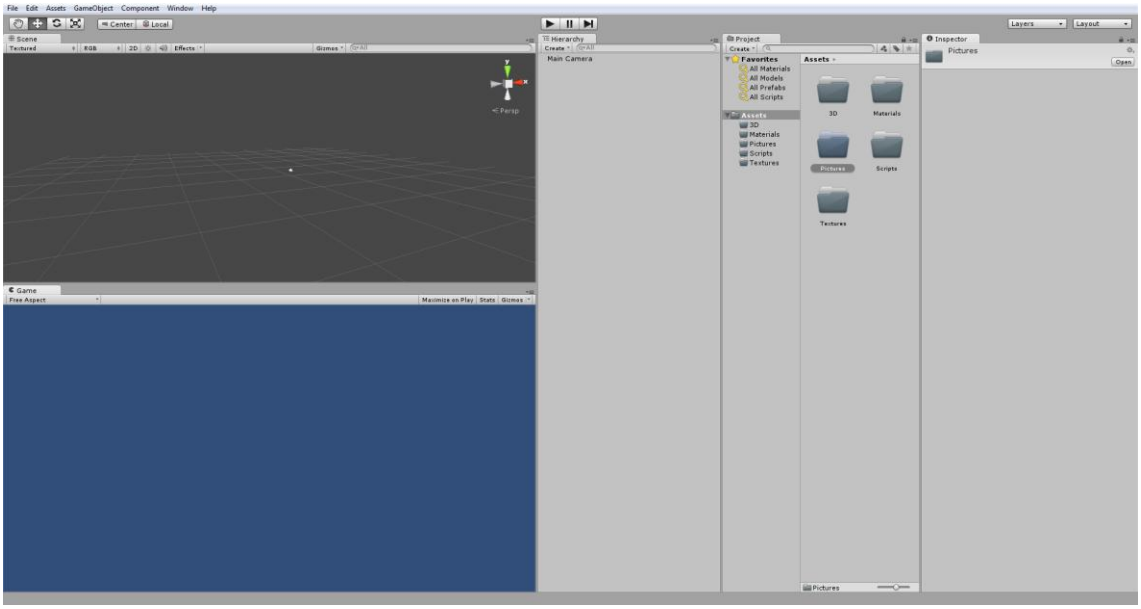
eivät käänny väärin. Kinematiikkaa käytetään 3D-riggauksessa, DuIKin toiminta muistuttaakin paljon riggaus-prosessia. 3D-riggauksessa on kaksi kinematiikka-vaihtoehtoa, joilla voi hallita liikettä. DuIKin käyttämä inversed Kinematics, jossa siis hahmon osia liikutetaan hierarkiassa ylöspäin liikkuen. Forward kinematics, joka on yleisempi tapa, jossa liikutaan hierarkiaa alaspäin. Forward kinematicsia käytettäessä esimerkiksi käden animaatiossa liikuttaisiin hartioista kämmeniin, kun taas inversed kinematicsia käytettäessä tehtäisiin juuri päinvastoin. Inversed kinematicsin käytössä on vaarana, että raajat taittuvat väärin. Tämä johtuu siitä että tietokone siirtää kämmentä suorinta reittiä. On huomattava, että DuIK sisältää mahdollisuuden estää väärä taipuminen. Inversed kinematicsin luuranko rakenteen avulla paloista kostuvan hahmon animointi muistuttaa nuken asettelua, mikä on ajatukseltaan paljon 3D- animaation kaltaista. [10]

3.2 Unity3D

Unity 3D on kaupalliseen käyttöön tarkoitettu pelimoottori, johon myydään erilaisia lisenssejä. Pelimoottori tukee niin 2D- kuin 3D-grafiikkaa. Työssä on käytetty Unity3D:n ei-kaupallista versiota 4.3.4, jossa on mukana uudet 2D-työkalut. Ohjelmalla on tarkoitus toteuttaa samat hahmoanimaatiot kuin aikaisemmilla ohjelmilla ja tutkia niin käyttöliittymää kuin työkaluja. Tarkoituksena on keskittyä 2D-grafiikan animointiin, mutta ei kuitenkaan suljeta 3D-puolta pois, sillä nämä kaksi kulkevat usein yhdessä. [11,12]

3.2.1 Käyttöliittymä

Unity3D:n käyttöliittymä on modulaarinen, joten käyttäjä voi itse muokata siitä omanlaisensa. Käyttäjä voi myös halutessaan käyttää valmiiksi asetettuja vaihtoehtoja. Kuvassa 4. näkyvä asetus on yksi näistä, sen nimi on 2 by 3. Tässä vaihtoehdossa näkyvissä ovat kaikki tarpeelliset ikkunat. [13]



Kuva 4. Unity3D käyttöliittymä.

Scene-ikkuna on ikkuna, jossa peli rakennetaan ja jonne peliin halutut objektit vedetään. Tätä näkymää voi käyttää 3D- ja 2D-näkymässä, näkymää voidaan vaihtaa milloin tahansa. Game-näkymä näyttää käyttäjälle miltä lopputulos näyttää kameran kautta. Hierarchy-ikkunasta käyttäjä näkee kaikki valitussa Scenessä olevat objektit. Project-ikkunassa on listattuna kaikki projektiin tuodut asiat ja sieltä ne saadaan tuotua helposti Hierarchy-ikkunaan. Project-ikkunassa voidaan luoda kansioita ja täten organisoida käytetty materiaali selkeästi. Ikkunan sisällä on toinen ikkuna, jossa näkyy projekti ikkunassa valitun kansion sisältö, kuvassa 4 valittuna on Assets-kansio. Viimeisenä ikkunana on Inspector. Tässä ikkunassa näkyy lisätietoa valitusta objektista ja siinä voidaan vaikuttaa sen asetuksiin. Kokenut käyttäjä voi lisätä asetuksia ja muokata niitä luomalla

objektille skripti-tiedoston. Skriptejä voidaan tehdä käyttämällä esimerkiksi mukana tulevaa monodeveloper ohjelmaa.[13]

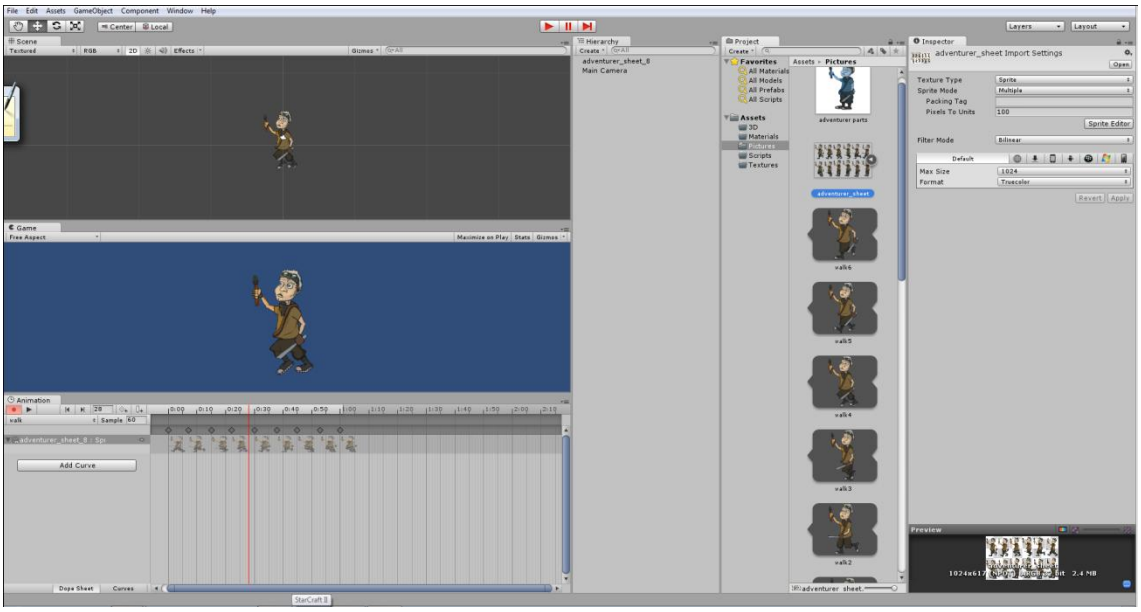
3.2.2 Sprite sheet -animaatio

Unity3D ohjelmaan voidaan tuoda sprite sheet –kuva png-tiedostona, joko vetämällä se ohjelmaan tai kopioimalla se suoraan projekti kansioon. Jotta kuvaa voidaan käyttää animoimiseen, pitää siitä erotella hahmon eri kuvat eli spritet. Unity3D ohjelmasta löytyy Sprite Editor -työkalu juuri tätä toimenpidettä varten,(kuva 5.).

Kuva 5. Sprite Editor -työkalu.

Sprite Editor -työkalun avulla käyttäjä voi leikata yksittäiset kuvat erilleen, jolloin niitä voidaan käyttää animaation tekemiseen. Slice-valikosta ne voidaan leikata automaattisesti. Unity3D käyttää kuvan tyhjää tilaa tunnistamaan erilliset kuvat. Valikko sisältää muutamia kohtia, joilla automaatti valintaa voidaan säätää. Minimum size -kohdasta voidaan säätää yhden kuvan valinnan kokoa. Jos koko asetetaan liian suureksi voi, kaksi erillistä kuvaa päätyä samaan kuvaan. Pivot-kohdasta voidaan asettaa kuvan pivot pisteen sijainti. Kuva kiertyy pivot pisteen kohdasta. Animaation kannalta on tärkeää asettaa pivot pisteet jokaisen kuvan kohdalla samaan paikkaan, jotta animaatio pysyy sulavana. [14]

Käyttäjä voi myös valita kuvat itse vetämällä neliön kuvien ympärille, (kuva 5). Valinta neliöt ovat aina suorakulmaisia, joten sprite sheet -kuvan tekovaiheessa on tärkeää pitää kuvat tarpeeksi kaukana toisistaan. Valitsemalla kuvan ilmestyy sprite-ikkuna, josta voidaan muokata jokaisen kuvan asetuksia erikseen. Kuvat voidaan esimerkiksi nimetä uudestaan, yksilöllisillä nimillä. Sprite Editor -työkalu sisältää myös Trim-toiminnon, joka tiivistää valinnan mahdollisimman lähelle kuvaa. [14]



Kuva 6. Unity3D Sprite sheet -animaatio.

Sprite-kuvien erottelun jälkeen voidaan toteuttaa animaatiot, joiden sprite-kuvat ohjelmaan tuotu sprite sheet -kuva sisältää. Animaatioiden toteuttamiseen tarvitaan Animation- ja Animator-ikkunoita. Animation-ikkuna sisältää aikajanan ja sen avulla voidaan kuvista tehdä animaatio. Sprite sheet -animaation tekemiseksi valitaan Animation-ikkunasta Add curve -painike, jonka jälkeen animaatiolle annetaan nimi ja se tallennetaan. Tämän jälkeen valitaan uudelleen Add curve -painike ja avautuvasta listasta valitaan Sprite. Tämän jälkeen animaatioon tarvittavat kuvat voidaan vetää aikajanalle. Vedetyt kuvat näkyvät pikku kuvina aikajanalla ja niiden kohta janalla on merkitty avainkuvalla, eli Keyframe-pisteellä. Avainkuvia siirtämällä voidaan päättää kuinka kauan kestää ennen kuin kuva vaihtuu, huomattavaa on että liian suuret välit voivat saada animaati-

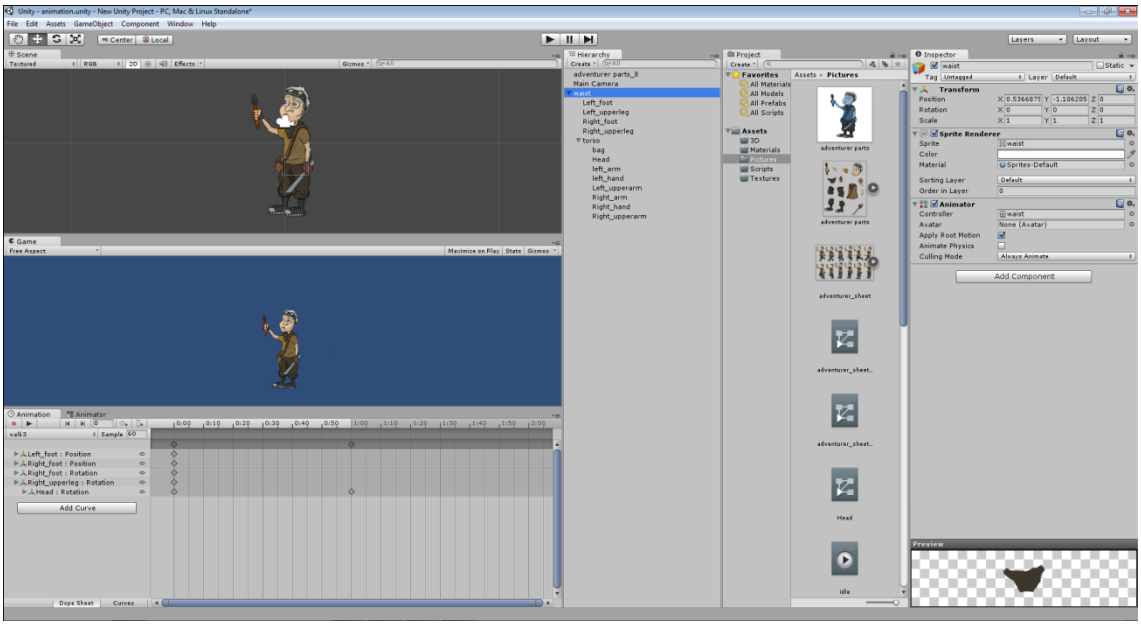
on näyttämään katkonaiselta. Liian pienet välit vastaavasti nopeuttavat liikettä. Avainkuvien väliä voidaan myös muokata vaihtamalla Animation-ikkunasta löytyvän sample-ruudun arvoa.

Yksi ”Sprite sheet” kuva voi sisältää kuvat useammalle animaatiolle. Valitsemalla ”Animation” ikkunasta animaation nimi, voidaan valitulle objektille luoda uusi animaatio. Yhdelle objektille voidaan luoda täten monta eri animaatiota, esimerkiksi hahmolle voidaan luoda kaikki siihen tarvittavat animaatiot yhteen käyttämällä vain yhtä kuvaa. [15]

3.2.3 Pala-animaatio

Unity3D-pelimoottoriin voidaan tuoda sprite sheet -kuva, joka sisältää hahmon animoitavat osat. Kuvan tulee sisältää kaikki mahdolliset osat, joita hahmon animaatiossa tarvitaan. Osat voidaan eritellä kuvasta käyttämällä sprite Editor -työkalua, (ks. Luku 3.2.2).

Kun hahmon eri osat on eroteltu, voidaan hahmolle tehdä animaatioita. Animaatio tekeminen tapahtuu hyvin samankaltaisesti verrattuna sprite sheet –animaatioon(kuva 7).



Kuva 7. Unity3D Pala-animaatio.

Erotellut osat vedetään Scene-ikkunaan ja järjestetään haluttuun järjestykseen. Inspector-ikkunassa voidaan vaihtaa osien kerrosta, jotta ne näkyvät oikeassa järjestyksessä. Jos osille luo tässä vaiheessa animaatioita ne luodaan erikseen ja ne linkittyvät eri osiin eivätkä itse hahmoon. Tästä syystä on järkevää luoda uusi objekti Hierarchy-ikkunaan ja asettaa hahmon osat sen alle eli sen lapsiobjekteiksi. Tämän jälkeen on hyvä ryhmitellä osia esimerkiksi asettaa yläruumin osat torso osan alle, tällöin muut osat ovat sen lapsia ja liikkuvat sen mukana. Valitaan luotu objekti, jonka alla osat ovat ja luodaan animaatio Animation-ikkunassa painamalla Add curve -painiketta ja annetaan animaatiolle haluttu nimi. Tämän jälkeen voidaan tehdä sen eri osille animaatiot painamalla Add curve -painiketta. Valitaan listasta osa, jolle halutaan tehdä animaatio ja mitä sille halutaan tehdä, esim. Transform tai Rotation. Tämä toimenpide toistetaan jokaiselle halutulle osalle, kuten on tehty kuvassa 7.[16]

Osille tulevat omat rivinsä ja avainkuvat eli keyframe-pisteet. Näiden avulla voidaan toteuttaa jokaiselle osalle haluttu animaatio. Valtaosaan riittää pelkkä kiertoanimaatio, mutta koska osat eivät ole yhteydessä toisiinsa voi, niitä joutua siirtämään, jolloin niihin lisätään position-animaatio. Tätäkin voidaan osittain välttää osien ryhmittelyllä. Esimerkiksi käden animaatiossa voidaan kiertää koko kättä, minkä jälkeen voidaan kiertää kyynärpäätä ja rannetta. Joten osien lapsi/vanhempi suhteisiin kannattaa siis panostaa, täten animaation tekeminen helpottuu.[16]

3.2.4 Puppet2D

Puppet 2D on maksullinen lisäosa Unity3D:hen, ohjelman tarkoituksena on antaa paremmat välineet 2D hahmojen animointiin.[17]

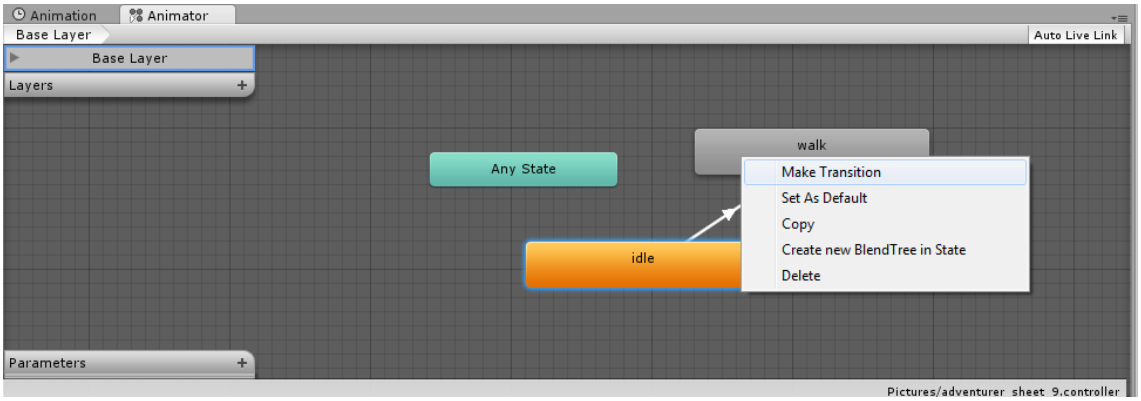
Ohjelma toimii melkein kuten pala-animaatio, mutta se tuo ns. IK-riggauksen 2D-ympäristöön. Hahmon osat ovat erikseen kuten pala-animaatiossa, mutta niille luodaan luita ja niiden välille luodaan suhteita. Tässä suhteessa se muistuttaa 3D-animaatiota. Hahmo-animaatiossa hierarkian ylimpänä on yleensä vyötärö, johon muut osat liittyvät. Animaatio ja sen valmistelu muistuttaa siis

hyvin paljon pala-animaatiota, sama hierarkiarakenne toimii myös normaalissa pala-animaatiossa. Luiden määrittämisen ja osien ryhmittelyn jälkeen voidaan hahmoa liikuttaa kuin nukkea, kuten 3D-animaatioissa. Hyötynä on, että hahmoa ei tarvitse enää liikutella osa kerrallaan kyynärpäästä kämmeneen, vaan voidaan suoraan liikuttaa kämmentä ja muut käden osat liikkuvat perässä.[17]

Tämän lisäksi ohjelmassa on After Effectsin Puppet tool -työkalun tyyppinen lisätyökalu, (ks. Luku 3.2.4). Sen toiminta on hyvin samankaltainen, vaikka se eroaakin hivenen After Effectsin työkalusta. Puppet2D-ohjelma luo kuvista ”meshin”, eli siis 2D-tason, joka koostuu polygoneista. ”Meshin” verteksien eli digitaalisessa avaruudessa sijaitsevien koordinaattipisteiden avulla voidaan korjata mahdollisia vääristymiä tai muokata 2D-kuvaa halutusti ja hallitusti.[17]

3.2.5 Animaatioiden hallinta

Animaatiot eivät kuitenkaan vaihdu itsestään, vaan niiden vaihtumiselle on asetettava ehdot. Kuvassa 8. näkyvät viivat kertovat pelille, mihin animaatioon siirrytään tietyn ehdon täyttyessä. Näitä ehtoja voidaan asettaa skripti-tiedostoissa, jotka sitten liitetään itse objektiin. Skripteissä ehtojen muutujat voidaan myös asettaa siten, että ne näkyvät objektin Inspector–ikkunassa, tällöin niiden arvoja on helppo vaihtaa ilman skripti-tiedoston muuttamista. Animator-ikkunassa voidaan luoda yhteyksiä useiden animaatioiden välillä, näin voidaan saada hahmolle hyvinkin laaja ja käytännöllinen animaatioreaktiopuu (kuva 8.).[18]



Kuva 8. Animator-ikkuna.

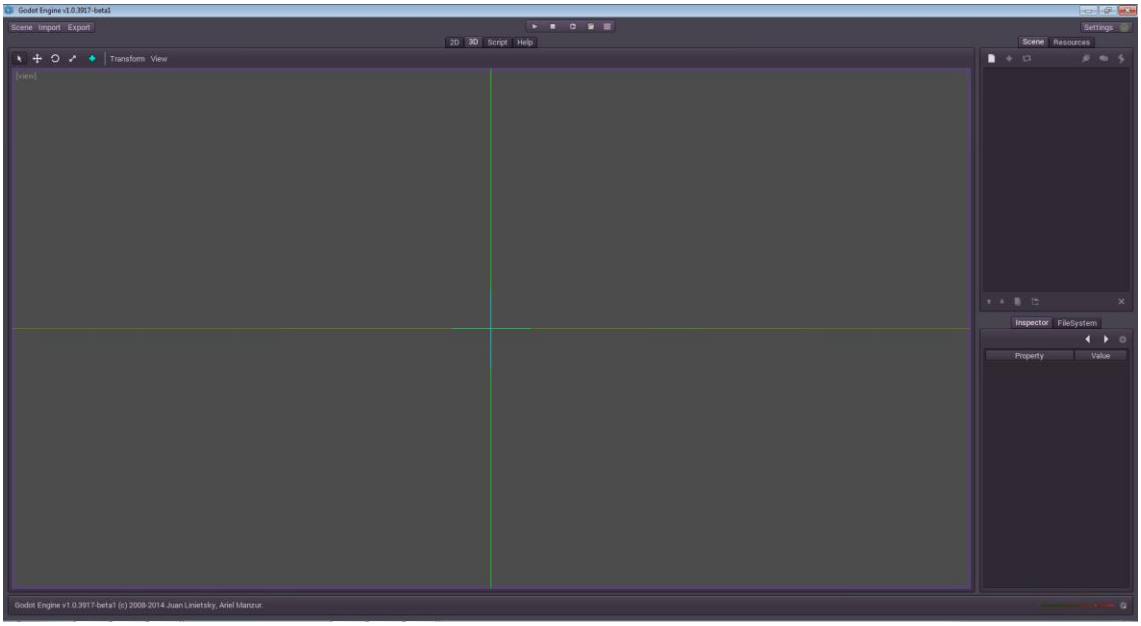
Tässä yhteydessä joudutaan kuitenkin tekemään aina omat animaatiot, esimerkiksi juoksua ja ampumisanimaatiota ei voida sekoittaa keskenään. Käyttäjä joutuu siis tekemään eri hahmon tila-animaatiot erikseen eli ampumisen seisten, juosten ja kävellessä. [18]

3.3 Godot

Godot on avoimen lähdekoodin pelimoottori. Sitä on tehty jo kauan ja sillä on myös toteutettu useita pelejä. Kuitenkin avoimeksi pelimoottori tuli vasta tänä keväänä, tästä syystä etenkin sivuston materiaali ja dokumentointi on joiltain osilta hyvin puutteellista. Godot-pelimoottorista on ulkona vasta beta-versio, joten se voi vielä kokea muutoksia.[19]

3.3.1 Käyttöliittymä

Godot pelimoottorin käyttöliittymä koostuu muutamasta eri ikkunasta ja rakenne muistuttaa paljon Unity 3D:n käyttöliittymää. Käyttöliittymä ei ole kovinkaan muokattava. Ikkunoiden kokoa voidaan muuttaa, mutta niiden sijaintia ei. Ikkunoita ei ole useita, ja osa niistä on samassa ikkunassa välilehtenä (kuva 9.).

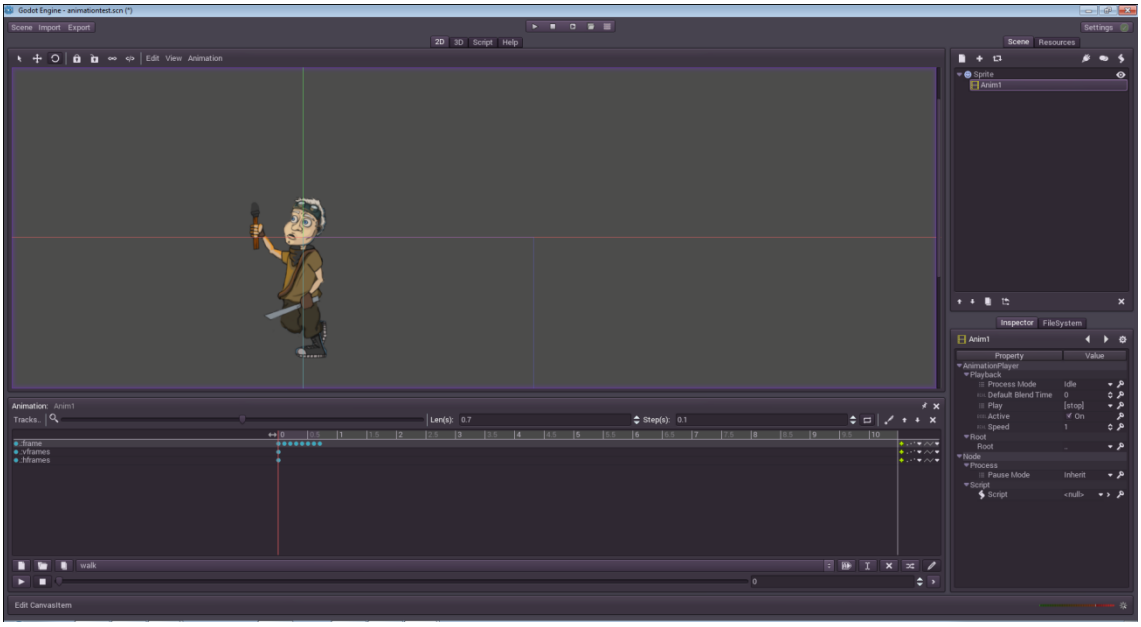


Kuva 9. Godot-käyttöliittymä.

Editor-ikkunassa rakennetaan itse peli, ja siihen tuodaan objektit Scene-ikkunasta. Scene-ikkunan välilehtenä on Resources-ikkuna, jossa nähdään projektiin tuodut tiedostot. Editor-ikkunan alapuolelle voi tuoda esiin Animation-ikkunan. Tämän ikkunan voi pitää näkyvillä tai piilottaa. Sen alapuolella on output-ikkuna, josta käyttäjä näkee eri ilmoituksia, esimerkiksi virheilmoitukset. [20]

3.3.2 Sprite sheet -animaatio

Godotin sprite sheet -kuva tuki on hyvin perustasoa. sprite sheet -kuva pitää valmistella tietyllä tavalla, jotta sitä voidaan käyttää ilman ongelmia. Valmisteluiden jälkeen voidaan sprite sheet –kuvat tuoda pelimoottooriin joka leikkaa ne erilleen automaattisesti. Tämän jälkeen voidaan luoda sprite sheet -animaatio(kuva 10.).



Kuva 10. Godot Sprite sheet -animaatio.

Ensin luodaan Sprite node -komponentti Scene-ikkunassa, tämän jälkeen valitaansen tiedostoksi Inspector-ikkunassa sprite sheet -kuva. Sprite sheet -kuva ilmestyy kokonaisuudessaan Editor-ikkunaan. Inspector-ikkuna sisältää paljon eri asetuksia. ”Hframe” ja ”Vframe” kohdilla kerrotaan Godotille, kuinka monta riviä ja saraketta sprite sheet -kuva sisältää. Samalla nähdään kun Editor-ikkunassa oleva sprite sheet -kuva rajaantuu yhteen spriteen, mikäli sprite sheet -kuva on tehty oikein[21,22]. Godot-ohjelmassa sprite-kuvat pitää sijoittaa ruudukkoon ja varmistaa, että ne on sijoitettu yhtä etäälle toisistaan.

Sprite-kuvien erottelu sprite sheet -kuvasta on hyvin suoraviivaista eikä ohjelma anna paljonkaan valtaa käyttäjälle vaikuttaa siihen, kuinka ne erotellaan.

Erottelun jälkeen voidaan tehdä animaatiota. Animaatio-ikkuna on kuitenkin tässä vaiheessa vielä tyhjä. Jotta animaatio voidaan luoda, täytyy prite node -komponentin alle luoda Animationplayer-komponentti. Tämän jälkeen Animaatio-ikkunaan ilmestyy aikajana ja muita tarvittavia painikkeita. [21,22]

Valitaan Scene-ikkunasta sprite node -komponentti, jotta saadaan Inspector-ikkunaan näkyviin sen tiedot. Ikkunasta voidaan luoda avainkuvia eli keyframe-pisteitä. Muuttujia, joista avainkuvia voidaan luoda Animaation-ikkunan aika-

janalle, on merkitty avain kuvalla. Sprite sheet -animaatiossa luodaan avainkuvat Inspector-ikkunasta löytyvästä frame-muuttujasta. Muuttujan viereisestä valintalaatikosta valitaan haluttu aloituskuva. Tämän jälkeen painetaan sen vieressä sijaitsevaa avainkuvaketta. Animaation-ikkunan aikajanalle ilmestyy ympyrä, joka merkitsee, että siihen on luotu avainkuva. Jokaiselle halutulle kuvalle luodaan oma avainkuva aikajanalle samalla tavalla. Käyttäjä voi siirtää avainkuvia aikajanalla, jos hän haluaa muuttaa niiden vaihtumisnopeutta.[21,22]

4 HACTENGINE

HactEngine on Indium Game Storiesin kehityksen alla oleva pelimoottori, tällä hetkellä sitä kehittää aktiivisesti kolmen henkilön tiimi. HactEngine on suunniteltu julkaistavan avoimen lähdekoodin lisenssin alaisena, koska sen ympärille halutaan luoda pelimoottorin kehityksestä kiinnostunut yhteisö. Yhteisö mahdollistaisi moottorin nopeamman kehityksen ja tietoisuus sen olemassa olosta leviäisi helpommin. HactEngine on suunniteltu modulaariseksi, jotta siihen olisi helppo tehdä erilisiä työkaluja. Modulaarisuus mahdollistaa uusien työkalujen testaamisen ja kehittämisen vaarantamatta valmista ohjelmistoa. Modulaarisuuden avulla ohjelmiston toiminta ei ole riippuvainen lisäosista, joten käyttäjä voi valita tarvitsemansa lisäosat, mikä tekee pelimoottorista muokattavamman.

4.1 Qt-kirjasto

HactEngine käyttää alustanaan suomalaisen ohjelmistoyhtiön Digian kehittämää ja avoimella lisenssillä levitettävää Qt-ohjelmistokirjastoa. Qt on ohjelmistojen ja graafisten käyttöliittymien kehitysympäristö, se on myös alusta riippumaton. Nokialle kehitti Qt:lla Symbian- ja MeeGo-käyttöjärjestelmiä mutta myi omistuksen Digialle, siirtyessään Windows-puhelimiin.[23]

Qt ei sinänsä ole peleille suunnattu ohjelmistokirjasto, mutta se tarjoaa HactEnginelle erittäin hyvän alustan. Qt:n ansiosta sama lähdekoodi saadaan pyöri mään niin PC- kuin mobiililaitteissa, jolloin pelinlähdekoodia ei tarvitse erikseen kääntää.[23]

4.2 Ohjelmointikielet

HactEnginessä käytetään useaa eri kieltä ja niillä on omat tarkoituksensa. Seuraavaksi käydään läpi käytetyt kielet ja niiden käyttötarkoitukset.

4.2.1 C++

C++ on erittäin nopea ja tehokas olio-ohjelmointikieli. Se on hyvin suosittu ja käytetty kieli ohjelmistokehityksessä. Tämän takia se on valittu myös HactEnginen kehitykseen. C++ on käännettäväkieli, joten muutoksien jälkeen se on aina käännettävä, jotta tehdyt muutokset tulevat voimaan. Pelikoodi pyritään pitämään skriptikielenä, jotta C++ koodia ei tarvitsisi muuttaa, tällöin peliä ei tarvitse käynnistää uudelleen nähdäkseen muutokset. [23]

4.2.2 Lua

Lua on hyvin käytetty ja suosittu skriptikieli, sitä on käytetty useissa peleissä. Luan syntaksi on erilainen verrattuna moneen muuhun kieleen, vaikka se onkin saanut vaikutteita C-ohjelmointikielistä. Se on myös hyvin yksinkertainen kieli ja sitä käytetään HactEnginessä pääosin pelien logiikan skriptaamiseen, mutta myös editorien toiminnallisuuksien toteuttamiseen. [23]

4.2.3 GLSL

GLSL(OpenGl Shader Language) on OpenGl Shadereissa käytettävä kieli. GLSL ei ole skriptikieli, mutta kuten Lua se voidaan kääntää ajonaikaisesti. Shadereita voidaan siis muuttaa pelin ollessa käynnissä ja niiden vaikutus nähdään heti. Shaderit ladataan erillisistä lähdekooditiedostoista ja niillä voidaan vaikuttaa esimerkiksi valaistukseen. [23]

4.2.4 XML

XML (Extensible Markup Language) on kieli, jota käytetään järjestelmien väliseen tiedon välitykseen ja formaattina tiedon tallentamiseen. HactEnginessä XML:ää käytetään mm. asetustiedostoissa ja pelimaailmaa kuvaavissa tiedos-

toissa. yhtenä syynä XML kielen valintaan oli tavoite käyttää selkokielisiä tiedostoja. [23]

4.2.5 QML

QML (Qt Meta Language) on kieli joka perustuu JavaScript-ohjelmointikieleen. Se on osa QtQuick-kirjastoa, jolla tehdään käyttöliittymäsuunnittelua ja muistuttaa osittain CSS-kieltä. Sillä on helppo suunnitella ja toteuttaa käyttöliittymiä. QML-kieltä käytetään pelimoottorin editorien toteutuksessa, sitä käytetään ydessä Luan kanssa. QML-kielen avulla toteutetaan painikkeiden ulkoasu ja perustoiminnallisuuksia. [23]

4.3 Käyttöliittymä

HactEnginessä on jo tehty eräitä työkaluja, joten myös käyttöliittymän osalta on tehty jo muutamia linjauksia. Käyttöliittymässä olevia värejä on jo mietitty, mutta lopullisia värejä ei ole vielä toistaiseksi sovittu. Käyttöliittymän pohja väreinä toimii kaksi eri harmaan sävyä, korosteväreinä toimivat sininen ja oranssi. [20]

Käyttöliittymää on rakennettu siten, että työkalut voidaan toteuttaa modulaarisena. Tarkoituksena on, että käyttäjä voi itse muuttaa ikkunoiden järjestystä ja valita näkyville vain tarvitsemansa työkalut. [23]

5 HACTENGINE 2D-ANIMAATIOTYÖKALUT

Tässä luvussa käydään läpi löydökset ja pohditaan niiden perusteella mahdollisia toteutustapoja HactEnginen animaatiotyökaluille. Tehdään myös ehdotuksia mahdollisista sivuavista työkaluista sekä käydään lyhyesti läpi ehdotuksia käyttöliittymän osalta, koska se sivuaa myös oleellisesti työkaluja. Tätä osaa kirjoitettaessa on myös konsultoitu osin HactEnginen kehitystiimiä ja heidän ajatuksiaan niin käyttöliittymän kuin muiden yleisten asioiden ja tietenkin animaatiotyökalujen kehityksestä.

5.1 Käyttöliittymä

Tutkittujen ohjelmien käyttöliittymät olivat hyvin samankaltaisia. Ne antoivat käyttäjälle mahdollisuuden muokata niitä hyvinkin paljon. Käyttäjä pystyy siirtämään ruutuja ja vaihtamaan niiden kokoa. Käyttäjälle annetaan muutama valmis valikkoasetus, joista tämä voi valita.

Kuva 11. HactEngine käyttöliittymä.

Käyttöliittymää on jo toteutettu jonkin verran, ja yllämainittuja seikkoja on otettu sen tekemisessä huomioon (kuva 11.). Käyttöliittymä on saanut jonkin verran vaikutteita esimerkiksi Unity3D:n käyttöliittymä rakenteesta. Käyttöliittymää on kehitetty ennen ja samanaikaisesti opinnäytetyön kanssa, kehitys jatkuu vielä sen jälkeenkin. Käyttöliittymän toteutuksessa on ollut mukana koko Indium Gamesin kehittäjätiimi, mutta suurimman kehitystyön on tehnyt Antti Tujula, joka toimii ohjelmoijan tehtävissä. Itse olen vaikuttanut käyttöliittymän visuaaliseen puoleen ja toteuttanut siinä käytetyt kuvakkeet. Käyttöliittymän rakenne ja toiminta perustuu myös osittain opinnäytetyön vertailun tuloksiin. Sen tarkoituksena on siis olla hyvin muokattavissa. Käyttäjän tulee voida tallentaa tekemiään ikkunajärjestyksiä. Näin käyttäjän on helppo aina käyttää ohjelmaa hyväksi toteamallaan tavalla, eikä muutoksia tarvitse tehdä joka kerta uudestaan. Kuitenkin tarjotaan valmiiksi asetettuja kokonaisuuksia, jotka kehitystiimi on kokenut itse toimiviksi kokonaisuuksiksi. Käyttäjälle tulee olla myös helppoa tuoda takaisin poistettu ikkuna. Tämän takia ikkunat voisivat olla kaikki esimerkiksi listattuna saman valikon alle. Valittu näkymä olisi kirjoitettuna ruudun reunalla, ja sitä klikkaamalla saisi valikon auki, josta sen ruudun näkymää voisi vaihtaa. Ikkunoita voisi myös välilehdittää, jotta tarpeelliset ruudut olisivat helposti saatavilla. Kun niitä ei juuri tarvita, ne voitaisiin piilottaa. Yhdessä ikkunassa voisi siis olla asetettuna useampi näkymä, joista vain yksi kerrallaan olisi näkyvissä.

Animaatio työkalujen osalta tärkeitä ikkunoita ovat, editori, hierarkia-, resurssi manageri ja animaatio -ikkunat. HactEngineen tulee oma ikkuna scriptien ja koodin tuottamiseen, mutta se todennäköisesti on mahdollista irrottaa liittymästä. Näin se voidaan siirtää esimerkiksi toiselle näytölle.

5.2 Sprite sheet -animaatio

Sprite sheet -animaatio on hyvin käytetty tapa tehdä animaatioita peleihin ja siksi tuki sille tarvitaan myös HactEngineen. Se on myös työkaluista ehkä yksinkertaisin toteuttaa. Yksinkertaistettuna spirte sheet -kuvan toiminta on seuraavan kaltainen; ohjelmalle kerrotaan jokaisen kuvan sijainti ”sprite sheetissä”,

animaatio ikkunassa kerrotaan, milloin mitäkin kuvaa näytetään ja kuinka kauan.

Kaikkienläpikäytyjen ohjelmien animaatiotyökalut olivat suhteellisen samankaltaisia. Unity3Dn Sprite Editor -työkalu oli mielestäni hyvin käyttökelpoinen, samanlaista työkalua voitaisiin miettiä myös HactEngineen. Työkalulla oli helppo ”leikata” kuvat erilleen, jolloin niitä pystyttiin käyttämään animaation teossa. Ainut huomautuksen aihe työkalussa oli, että siinä sprite-kuvien valinta sprite sheet -kuvasta tehtiin suorakulmaisessa muodossa. Minkä seurauksena sprite-kuvat täytyy sprite sheet -kuvan tekovaiheessa sijoittaa tarpeeksi erilleen. Tämän johdosta kuvatiedostoon jää paljon tyhjää tilaa. Tyhjä tila vie silti muistia, joten mitä vähemmän sitä on, sitä parempi.

HactEnginen ”Sprite Editor” voisi olla muuten samankaltainen kuin Unity3D ohjelmassa (ks. Luku 3.2.2). Sillä erolla että erilliset kuvat rajattaisiin pisteillä, jotka yhdistyvät suorilla viivoilla, kuten kuvassa 12. Pisteitä voisi lisätä siis enemmän kuin neljä, jolloin kuvan reunoja pystytään seuraamaan suurpiirteisesti.

Kuva 12 Sprite Editor -työkalu ehdotus.

Näin sprite sheet -kuvaan voidaan kuvat asetella lomittain, jolloin kuvaan mahtuu enemmän tietoa ja jää vähemmän tyhjää tilaa. Rajatut kuvat voitaisiin nimetä uusiksi kuten Unityn sprite editorissa. Minkä jälkeen niitä voitaisiin käyttää

kuten yksittäisiä kuvia, vaikka ne tosiasiassa ladataankin edelleen sprite sheet -kuvasta.

Tämän jälkeen yksittäisistä kuvista voidaan luoda animaatioita. Animaatiot luotaisiin animaatio ikkunassa, jossa olisi animaatioiden tuottamiseen tärkeät työkalut. Tämän ikkunan toiminto voitaisiin rakentaa samantyyppiseksi kuin testatuissa ohjelmissa. Ohjelmien animaatioikkunat ja niiden toiminta oli hyvin samankaltaista. Ensin ikkunassa luotaisiin uusi animaatio, minkä jälkeen animaatioon liittyvät sprite kuvat tuotaisiin aikajanalle. Jokainen kuva olisi oma avainkuvansa, jolloin käyttäjä voi vaihdella niiden vaihtumisnopeutta siirtämällä niitä aikajanalla. Käyttäjä voisi myös toistaa animaation, jotta hän näkee miltä se näyttää. Tällöin hän myös pystyy korjaamaan mahdolliset viat tai vaihtamaan avainkuvien paikkaa aikajanalla. Jokaisen tuodun kuvalla pitää olla mahdollisuuksia vaihtaa tiettyjä asioita. Näitä ovat sijainti, kierto, leveys ja korkeus. Itse animaatio näkyisi eri ikkunassa, esimerkiksi Editor-ikkunassa. On myös mahdollista, että animaation luomiseen tueksi annetaan oma Preview-ikkuna.

5.3 Pala-animaatio

Kaikissa läpikäydyissä ohjelmissa oli pala-animaatio mahdollisuus, mutta se ei ollut läheskään täydellinen, kuitenkin lisäohjelmilla siitä saatiin paljon käytettävämpi. Lisäosat toivat pala-animaatiota myös lähemmäs 3D-animaatiota. Tästä syystä Hactenginen pala-animaatiossa kannattaa lähteä mieluummin lisäohjelmien suuntaan, joissa mukaan on tuotu 3D-animaatiosta tuttu riggaus. Perus pala-animaatio tuki, jollainen esimerkiksi Unity3D ohjelmassa on ilman lisäosia, riittää pieniin tausta-animaatioihin. Hahmoanimaatiossa Puppet2D tyylinen työkalu on hyvin paljon käytännöllisempi. Tavoitteena kuitenkin on luoda työkalu, jolla on helppo ja nopeaa luoda animaatiota.

Kehittäjätiimin kanssa käydyssä keskustelussa kannatettiin myös pala-animaatiota ja korostettiin sen monia hyötyjä, verrattuna sprite sheet -kuvien käyttöön. Pala-animaatiossa voidaan yksittäisiä paloja vaihtaa kesken pelin, rikkomatta animaatiota. Tämä on hyvin käytännöllistä peleissä, joissa pelaaja

vaihtaa esimerkiksi varusteita. Pala-animaation tekeminen on suhteellisen helppoa, vaikka valmistelu vie oman aikansa. Kuitenkaan käyttäjän ei tarvitse piirtää jokaista kuvaa liike radasta erikseen, vaan hän voi käyttää hyödyksi avainkuvia eli Keyframe-pisteitä. Käyttäjän ei tarvitse piirtää virheellisiä kuvia uudestaan vaan muokata virhe kohdan palojen sijaintia tai kiertoa. Tämä säästää paljon aikaa. Kuitenkin HactEnginen kehittäjätiimin kanssa käydyssä keskustelussa todettiin, että 2D-riggauksen lisääminen pala-animaation kasvattaisi huomattavasti sen käytännöllisyysarvoa. Myös hahmojen animointi olisi hivenen miellyttävämpää, vaikka luiden ja lapsi-vanhempisuhteiden oikea luominen vie oman aikansa oppia.

6 YHTEENVETO JA JOHTOPÄÄTÖKSET

Opinnäytetyössä käytiin läpi perinteisiä animaatiotekniikoita, joista saatiin selville muutama pelialalle käytännöllinen animaatiotekniikka. Työssä vertailtiin kolmea ohjelmaa toteuttamalla niissä saman hahmon kävelyanimaatio, jotta saatiin käsitys jo olemassa olevista ratkaisusta. Huomattiin että ne olivat perusteiltaan hyvin samankaltaisia verrattuna perinteisten animaatiotekniikoiden kanssa.

Ohjelmilla toteutettiin sprite sheet –animaatio, pala-animaatio ja pala-animaatio jossa käytettiin myös 2D-riggausta. Näitä työkaluja pitäisi siis löytyä myös tulevaisuudessa HactEnginestä. Godot-pelimoottorilla ei pystytty toteuttamaan pala-animaatiota, eikä siitä johtuen päästy tutkimaan käytännön kautta myöskään sen 2D-riggaus ratkaisuja. Ohjelmia vertaillessa opittiin, että nämä kolme animaatiotekniikkaa ovat tällä hetkellä vallitsevia. Pala-animaatio ja 2D-riggaus ovat suhteellisen uusia ja siksi osassa ohjelmista ne ovat vielä karkeita käyttää. Kuitenkin oli selkeää että ne ovat pelimoottorissa oleellisia työkaluja ja niiden toteuttamiseen pitää käyttää aikaa.

Saatiin myös hyvä kuva ohjelmien käyttöliittymistä, vaikka se ei ollut opinnäytetyön päätutkimuksen kohde. On se läheisesti siihen liittyvä seikka, kun pohditaan animaatiotyökaluja ja siihen oleellisia ikkunoita. Kuten myös painikkeet ja animaatiota sivuavat lisätyökalut, esimerkiksi animaationhallintaan liittyen.

Vertailuun olisi voitu ottaa enemmän pelimoottoreita, mutta se olisi pidentänyt opinnäytetyön pituutta huomattavasti. Enkä usko että se olisi tuonut huomattavaa lisäarvoa vertailulle. Pohjimmiltaan pelimoottorit ovat hyvin samankaltaisia. Työssä käytettiin Unity3D-pelimoottorista ilmaista versiota, joka ei sisällä kaikkia ominaisuuksia, joita maksullinen pro versio sisältää. Tästä syystä siitä olisi voinut olla hyvä käyttää maksullista versiota, jotta sen koko potentiaali olisi tullut esille. Kuitenkin tässä tapauksessa ei maksullisen pro versiota ollut mahdollista käyttää.

HactEnginen alpha versio on tarkoitus julkaista vuoden 2014 loppuun mennessä. Alpha versiolla on tarkoitus testata pelimoottoria aikaisessa vaiheessa. Ani-

maatiotyökaluista alpha versio sisältää sprite sheet –animaatioon liittyviä työkaluja. Tästä syystä opinnäytetyön olisi voinut rajata pelkästään sprite sheet –animaatioon, jolloin sen eri osista olisi saanut kattavamman ja tarkemman tutkimuksen. Kuitenkin toteutettu tutkimus antaa hyvän yleiskuvan tarvittavista 2D-animaatio työkaluista ja sen pohjalta on helppo rajata pienempiä osa-alueita jatkotutkimuksia varten.

Tulevaisuudessa HactEnginen animaatiotyökaluja tullaan suunnittelemaan ja toteuttamaan vaiheittain. Alpha versioon tehdään sprite sheet –animaatiotyökalut ja tämän jälkeen suunnitellaan tarkemmin pala-animaatioon liittyviä työkaluja. Pala-animaation ohessa voidaan suunnitella myös 2D-riggausta, se sivuaa myös tekniikkana 3D-riggausta, joten se voidaan toteuttaa myös sen rinnalla. On huomattavaa että kaikki eri animaatiotavat tulevat käyttämään myös samoja perustyökaluja.

Vaativampien animaatiotyökalujen teknillisessä toteutuksessa voidaan ottaa esimerkkiä Godot-pelimoottorin ratkaisuista. Godot on avoin ja MIT-lisenssin ansiosta sen koodia voidaan käyttää vapaasti. Lisenssi antaa myös mahdollisuuden käyttää tehtyjä ratkaisuja suljetuissa ja maksullisissa ohjelmissa, vaikka alkuperäinen koodi onkin avointalähdekoodia.

Opinnäytetyö pysyi yleisellä tasolla eikä esimerkiksi päästy syvälle työkalujen teknilliseen puoleen, kuitenkin siitä saa hyvin selville 2D-animaatiotyökalujen nykytilan. Vastaavanlaisia tutkimuksia on hyvä tehdä aika-ajoin, sillä ohjelmat kehittyvät ja niihin tehdään parannuksia. On siis hyvä seurata millaisia ratkaisuja muut yritykset tekevät ja pohtia niiden käyttökelpoisuutta. Näin voidaan löytää hyviä kehitysratkaisuja HactEngineen.

LÄHTEET

[1] Wyatt. A. The Complete Digital Animation Course. USA:Thames & Hudson.2010

[2] Krull.M, Tutorial: Walking animation for low-res sprites, [www-dokumentti]. Saatavilla: http://www.manningkrull.com/pixel-art/walking.php (Luettu:4.6.2014)

[3] Squidoo, Learn about 3d Character Rigging in Blender 3D, [www-dokumentti]. Saatavilla: http://www.squidoo.com/BlenderCharacterRigging (Luettu:4.6.2014)

[4] Wikipedia, After Effects, [www-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/Adobe_After_Effects. (Luettu 16.3.2014)

[5] Lynda.com, After Effects CC Tutorial: Exploring the interface and elements of After Effects, [www.dokumentti]. Saatavilla: http://www.youtube.com/watch?v=8v19kr2LHcE.. (Luettu 16.3.2014)

[6]Adobe, Composition Basics, [www-dokumentti]. Saatavilla: http://helpx.adobe.com/after-effects/using/composition-basics.html. (Luettu 4.6.2014)

[7] George Maestri, Rigging replacement animation in After Effects, [www-dokumentti]. Saatavilla: http://www.lynda.com/3D-Animation-Animation-tutorials/Rigging-replacement-animation-After-Effects/734/58891-4.html. (Luettu 17.3.2014)

[8] Lynda.com, After Effects CC tutorial: Understanding animation, [www-dokumentti]. Saatavilla: https://www.youtube.com/watch?v=h9o6gDaLwVM. (Luettu19.3.2014)

[9] George Maestri, Rigging with the Puppet tool in After Effects, [www-dokumentti]. Saatavilla: http://www.lynda.com/3D-Animation-Animation-tutorials/Rigging-Puppet-tool-After-Effects/734/58892-4.html. (Luettu19.3.2014)

[10] Famos, DuIK Tools - Inverse Kinematics for After Effects, [www-dokumentti]. Saatavilla http://vimeo.com/7872163. (Luettu 22.3.2014)

[11] Unity3d.com, Unity, [www-dokumentti]. Saatavilla: http://unity3d.com/unity. (Luettu 28.3.2014)

[12] Unity3d.com, Showcase, [www-dokumentti]. Saatavilla: http://unity3d.com/showcase. (Luettu 28.3.2014)

[13] Unity3d.com, Learning the interface, [www-dokumentti]. Saatavilla: http://docs.unity3d.com/Documentation/Manual/LearningtheInterface.html. (Luettu 29.3.2014)

[14] Unity3d.com, The Sprite Editor, [www-dokumentti]. Saatavilla: http://unity3d.com/learn/tutorials/modules/beginner/2d/sprite-editor. (Luettu 5.4.2014)

[15] Hunter Ehrismann, Unity 2d animator and animation, [www-dokumentti]. Saatavilla https://www.youtube.com/watch?v=FNp_y_Ickhs. (Luettu 6.4.2014)

[16] Red Hoodie, Tutorial - Using Unity 2D's Dope Sheet for Animation, [www-dokumentti]. Saatavilla: http://totallysweetredhoodie.blogspot.fi/2013/11/tutorial-using-unity-2ds-new-dope-sheet.html. (Luettu 6.4.2014)

[17] Jaimie Nimann, Puppet2D a 2D skeletal animation tool for Unity, [www-dokumentti]. Saatavilla: http://www.puppet2d.com/#!documentation/c1p9k. (Luettu 12.4.2014)

[18] Unity3d.com, Animator, [www-dokumentti]. Saatavilla: https://docs.unity3d.com/Documentation/Manual/Animator.html. (Luettu 13.4.2014)

[19] Godotengine.or, Godot Game engine, [www-dokumentti]. Saatavilla: http://www.godotengine.org/wp/. (Luettu) 19.4.2014)

[20] Godotengine.org, Scenes and nodes, [www-dokumentti]. Saatavilla: https://github.com/okamstudio/godot/wiki/tutorial_scene. (Luettu 19.4.2014)

[21] Godotengine.org, Animation tutorial, [www-dokumentti]. Saatavilla: https://github.com/okamstudio/godot/wiki/tutorial_animation. (Luettu 20.4.2014)

[22] El Canal de Shackra, Godot Engine - setting the animation for a sprite, [www-dokumentti]. Saatavilla: https://www.youtube.com/watch?v=DUKH9qr9mJQ. (Luettu 20.4.2014)

[23] Tujula.A. HactEnginen kehittäjä. Keskustelu 23.5.2014.