Tampereen ammattikorkeakoulu

Animaatiotyökalun luominen Unity-pelimoottoriin

Case: Tweener

Severi Jokiperä

OPINNÄYTETYÖ
Lokakuu 2020

Tietojenkäsittely
Pelituotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Pelituotanto

JOKIPERÄ, SEVERI:
Animaatiotyökalun luominen Unity-pelimoottoriin
Case: Tweener

Opinnäytetyö 34 sivua, joista liitteitä 1 sivu
Lokakuu 2020

Tämän opinnäytetyön toimeksiantaja on Mohavi Creative Company oy, joka luo interaktiivisia ratkaisuja, kuten nettisivuja, sovelluksia ja pelejä yritysasiakkaille. Mohavi Creative on laajentamassa markkinointiaan yksityisasiakkaille luomalla omia pelejä. Opinnäytetyön tavoite oli suunnitella ja toteuttaa työkalu Unity-pelimoottoriin, jonka avulla voi helposti ja nopeasti toteuttaa yksinkertaisia animaatioita. Työkalun tulisi olla helppo käyttää, jotta projektitiimin tekijät, joilla ei ole paljoa kokemusta Unityn käytöstä, voivat käyttää sitä vaivatta. Koodin tulisi olla myös monikäyttöinen ja toimia eri tilanteissa, esim. objektien liikuttamiseen ja pyörittämiseen 3D-, 2D- sekä UI-ympäristöissä. Työkalua pitäisi voida käyttää ilman Unityn omia animaatio-työkaluja.

Opinnäytetyö luotiin osana peliprojektia, jossa sen tehtävänä oli helpottaa UI:n animaatioiden luomista niin projektin ohjelmoijille kuin graafikoille. Graafikot ja ohjelmoijat testasivat tuotosta, ja heidän tuotoksiaan puolestaan testattiin pelien prototyypissä. Testausvaiheessa ihmiset, joiden osaaminen Unity-pelimoottorin käytöstä vaihtelee, testasivat työkalun käytettävyyttä, ja tutkittiin kuinka intuitiivinen työkalu on. Kun työkalu täytti sen testausvaatimukset ja käyttöryhmän toiveet ja tarpeet, se lisätään osaksi yrityksen koodikirjastoa, jota yritys käyttää jokaisessa projektissaan. Työkalu saatetaan julkaista Unity-pelimoottoria varten luodussa kaupassa, Unity Store:ssa. Kaupassa kenen tahansa on mahdollista laittaa myyntiin luomiaan työkaluja muille kehittäjille.

Opinnäytetyön tuloksena syntyi lopullinen työkalu, jota on jo käytetty ja tullaan käyttämään toimeksiantajan tulevissa projekteissa. Tuotosta tulisi jatkuvasti jatkokehittää vastaamaan asiakkaiden uusia tarpeita sekä huolehtia, että ohjelma toimii myös Unity-pelimoottorin uusissa versioissa. Jatkokehityksessä on otettava huomioon Unity-pelimoottorin uudet ominaisuudet, jotta ohjelmaa voitaisiin hyödyntää myös niiden käytössä. Jatkokehityksessä tulisi myös harkita ohjelman käyttöliittymän päivittämistä yksinkertaisempaan muotoon, että sen pääasiallinen tehtävä nopeana ja helppona työkaluna pysyisi totena.

Asiasanat: pelituotanto, Unity, työkalu, käyttöliittymä

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Game Development

JOKIPERÄ, SEVERI:
Creating an Animation Tool for Unity Game Engine
Case: Tweener

Bachelor's thesis 34 pages, appendices 1 pages
October 2020

The commissioner of this thesis was the Finnish Company Mohavi Creative Company oy, which creates interactive solutions such as websites, software and games to business customers. Mohavi Creative is also widening its market to consumers by creating its own games. The objective of this thesis was to design and create a working tool for Unity game engine, which could easily and quickly create simple animations. The tool should be easy to use, so that members of the team who do not have much of experience with Unity could use it without effort. The tool should also be multi-functional and work in different situations, for example in moving and rotating objects in 3D, 2D and UI-space. The tool should be usable without Unity's own animation-tools.

This thesis was carried out as a part of a game project, where the main purpose was to help making UI-animations easier to the team’s programmers and artists. The tool was tested in the workflow of programmers and artists, whose output in turn was tested in the games prototypes. The main goal of testing was to see how intuitive the tool was with people who do not have much experience with Unity game engine. When the tool passes its design requirements and the needs and wishes of the user group, it will be added as part of the company's tool library the company uses in all of its projects. The tool will be published in Unity's Asset store, which is a store created solely for user-made tools that can be used in Unity.

As the result of this thesis, a tool was created, that has already been used and will be used in the company's future products. Even though the tool is complete for now, it still needs some refinement to fix possible bugs, and to make sure it works in the future versions of Unity. The tool should also be developed further to match the new needs of the customer, and to implement new features Unity's new versions might include. When developed further, the main goals of the tool to be fast and easy to use tool should be kept in mind.

Key words: game production, Unity, tool, user interface

SISÄLLYS

1 JOHDANTO....7
2 PELIMOOTTORI....9
2.1 Mikä on pelimoottori?....9
2.2 Unity-pelimoottori....11
3 KÄYTTÖLIITTYMÄ....13
3.1 Mikä on kättöliittymä?....13
3.2 Unityn käyttöliittymä....15
4 ANIMAATIOT....18
4.1 Mikä on animaatio?....18
4.2 Animaatiot Unityssä....19
5 CASE-TWEENER....20
5.1 Toimeksianto....20
5.2 Toteutus....20
5.3 Testaus....27
6 POHDINTA....31
LÄHTEET....32
LIITTEET....33
Liite 1. Kuva työkalusta ilman editor-skriptiä muokkaamassa ulkonäköä....33

LYHENTEET JA TERMIT

Unity	Yksi suosituimmista pelimoottoreista
2D	Two-Dimensional, kaksiulotteinen
3D	Three-Dimensional, kolmiulotteinen
UI	User Interface, käyttöliittymä
Modaus	Pelin muokkaaminen käyttäjien toimesta
Indie	Independent, itsenäinen, käytetään yleensä puhuttaessa pienistä peliyrityksistä.
Asset Store	Unityn luoma kauppa joka myy kehittäjien tekemää sisältöä.
Asset	Mikä tahansa peleihin käytettävä sisältö, kuten kuvat ja ääni efektit.
GUI	Graphical User Interface, graafinen käyttöliittymä
UX	User Experience, käyttäjäkokemus
C#	Ohjelmointikieli jota pelinkehittäjät käyttävät Unityssa
Komponentti	Komponentti tarkoittaa yhtä pientä pelin osaa joka luo tietyn ominaisuuden peliin
Frame	Yksi kuva animaatiossa
Keyframe	Yksi kuva animaatiossa joka sisältää arvoja animaation pyörittämiseen, kuten positio, rotaatio ja skaala.
FPS	Frames Per Second, kuinka monta kuvaa animaatio näyttää sekunnin aikana
Kurvi	2D-presentaatio siitä, miten animaatio kulkee alkupisteestä loppupisteeseen.
Looppi	Jonkin asian toistaminen monta kertaa
Skripti	Ohjelmointi tiedosto joka luo ominaisuuden pelissä tai ohjelmassa.
Tutoriaali	Opas, joka selittää tarkkaan miten jotain tehdään tai käytetään.
Metodi	Pieni osa skriptiä jolla on oma käyttötarkoituksensa skriptissä.
Muuttuja	Koodissa käytettävä arvo, joka voi olla sanoja, numeroita yms.

Public	Julkinen, käytetään muuttujissa kertomaan että kaikki koodit voivat päästä muuttujaan käsiksi.
Bugi	Jokin virhe ohjelman toiminnassa.

1 JOHDANTO

Unity on yksi tämän hetken suosituimmista pelimoottoreista. Unityn suuri suosio ei silti tarkoita, etteikö siinä olisi aina jotain parantamisen varaa. Jotkin sen ominaisuuksista eivät toimi oikein tai ne ovat erittäin vaikeita ja monimutkaisia käyttää. Tästä syystä Unity mahdollistaa omien työkalujen luomisen osaksi Unityn käyttöjärjestelmää helpottamaan pelin luomisprosessia halutulla tavalla.

Työskennellessäni Unityn kanssa projekteissa olen huomannut yksinkertaisten animaatioiden luomisen olevan monimutkainen prosessi. Yhden animaation luominen vaatii melkein 10 työvaihetta, vaikka animaatio olisi niinkin yksinkertainen kuin objektin siirtäminen paikasta A paikkaan B. Tästä syystä lähdin ajattelemaan, miten yksinkertaista olisi lisätä objektiin ominaisuus, jossa sille voisi vain antaa aloitus ja lopetus pisteen, ja se toimisi halutulla tavalla, leikaten suurimman osan pakollisista työvaiheista pois. Näin syntyi idea työkalulle "Tweener".

Opinnäytetyön toimeksiantaja, Mohavi Creative Company oy, on vuoden 2020 alussa luotu yritys, joka tuottaa interaktiivisia ratkaisuja yritysasiakkailleen. Nämä ratkaisut sisältävät nettisivuja, sovelluksia, ja pelejä. Mohavi Creative tähtää myös luomaan omia työkaluja ja pelejä kuluttaja-asiakkaille. Olen yksi kyseisen yrityksen perustajista ja pääasiallisessa vastuussa yrityksen ohjelmointi tarpeista sovellus- ja pelituotantoon liittyen.

Opinnäytetyön tavpotteena on ensisijaisesti luoda toimeksiantajalle työkalu yksinkertaisten animaatioiden luomiseen yrityksen tulevissa projekteissa. Työkalulle asetettiin tavoitteeksi olla helppokäyttöinen ja käytettävä ilman ylimääräistä opastusta. Työkalun tulisi myös toimia 2D-, 3D- ja UI-ympäristöissä. Opinnäytetyö toteutetaan osaksi yrityksen tämänhetkisiä pelinkehitysprojekteja, jolloin sen tulisi edistää projektin käyttöliittymän suunnittelua ja toteutusta. Työkalun ollessa valmis toimeksiantaja suunnittelee työkalun laittamista myyntiin Unityn omaan Asset Storeen.

Opinnäytetyön tarkoitus on tutkia pelimoottoreita ja niiden kehittämistä, käyttöliittymää ja sääntöjä hyvän käyttäjäkokemuksen luomiseen, sekä animaatioiden perusteita. Koska luon työkalun Unity-pelimoottorille, tutkin jokaista aihetta myös Unityn näkökulmasta ymmärtääkseni miten nämä aiheet on toteutettu käyttämässäni kehitysympäristössä, ja miten voisin hyödyntää niitä toteutuksessa.

2 PELIMOOTTORI

2.1 Mikä on pelimoottori?

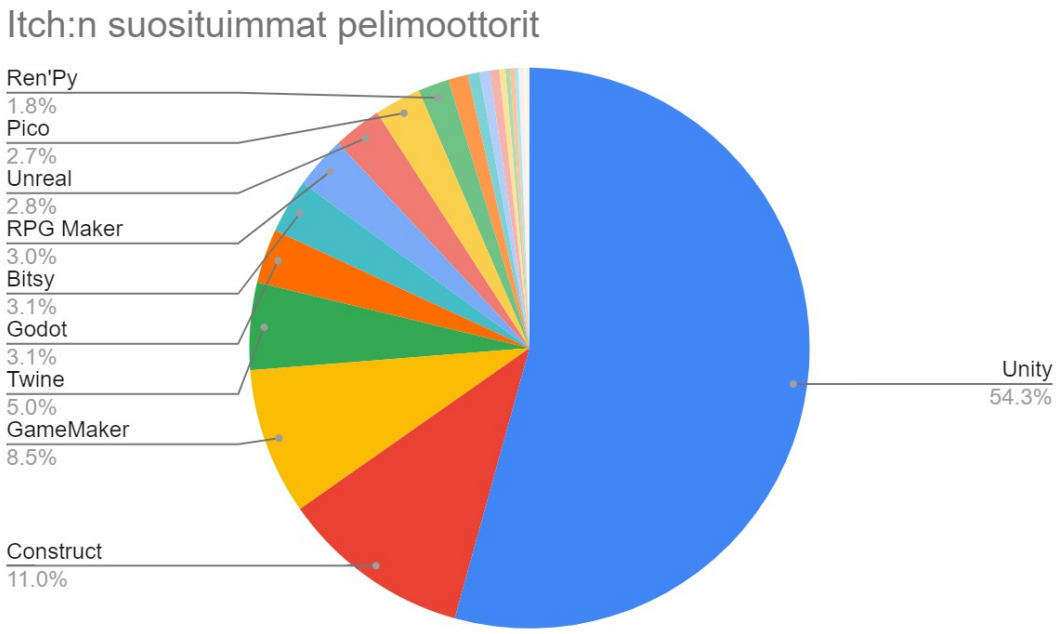
Sanan pelimoottorin määritelmä on hyvin epämääräinen, eikä ole vieläkään tarkkaa määritelmää, mitä pelimoottori saa ja ei saa sisältää. Pelimoottorin ero itse peliin on sen uudelleenkäytettävyys. Peli on kokonaisuus joka luodaan pelimoottorilla, mutta itse pelimoottori muodostuu pelin taustalla olevista pienistä osista, eli komponenteista, joita voidaan käyttää uudelleen muuhun tarkoitukseen. Sana pelimoottori onkin alun perin lähtöisin 90-luvun modauskulttuurista, kun pelaajat alkoivat muokata alkuperäisistä peleistä jotain täysin erilaista muokkaamalla ja käyttämällä uudelleen pelin ominaisuuksia muihin tarkoituksiin. (Gregory 2009)

Tänä päivänä pelimoottorin määritelmä on kuitenkin hieman selvempi. Pelimoottorilla tarkoitetaan erilaisia työkaluja sisältävää ohjelmaa, jota käytetään pelien tekemiseen. Pelimoottorin tarkoitus on vapauttaa pelinkehittäjä rutiinityötehtävistä ja varmistaa, että pelimoottori on käytettävissä luovilla suunnittelijoilla, joilla on erilaiset taustat ja jotka kokevat sovelluskehityksen haasteelliseksi. Pelimoottorin tarkoitus on helpottaa pelien tekemistä tekijän taustasta huolimatta. (Sung, Pavleas, Arnez, Pace, 2015)

Pelimoottoreita on nykyään useita, ja kaikilla on oma käyttötarkoituksensa. Jotkin suosivat täysin graafista käyttöliittymää, joissa ohjelmointi tapahtuu vetämällä palikoita toisiinsa. Toiset taas antavat käyttäjälle täysin vapaat kädet ohjelmoida peliin mitä tahansa. Jotkin ovat hyviä tekemään 2D-pelejä, toiset taas keskittyvät 3D-peleihin. Joissakin pelimoottoreissa on ohjelmointi pääosassa, kun taas jotkin pelimoottorit käyttävät kokonaan visuaalista ohjelmointiympäristöä. Pelimoottoreissa jokaiselle löytyy jotakin, mutta jos keskitytään suosituimpiin pelimoottoreihin, tulevat tietyt nimet aina vastaan.

Itch.io on erittäin suosittu pelien julkaisu- ja myyntisivu indie- (independent, itsenäinen) pelinkehittäjien keskuudessa. Kuka tahansa voi luoda sinne tunnukset ja julkaista pelinsä joko ilmaiseksi vai maksullisena. Itch.io pitää

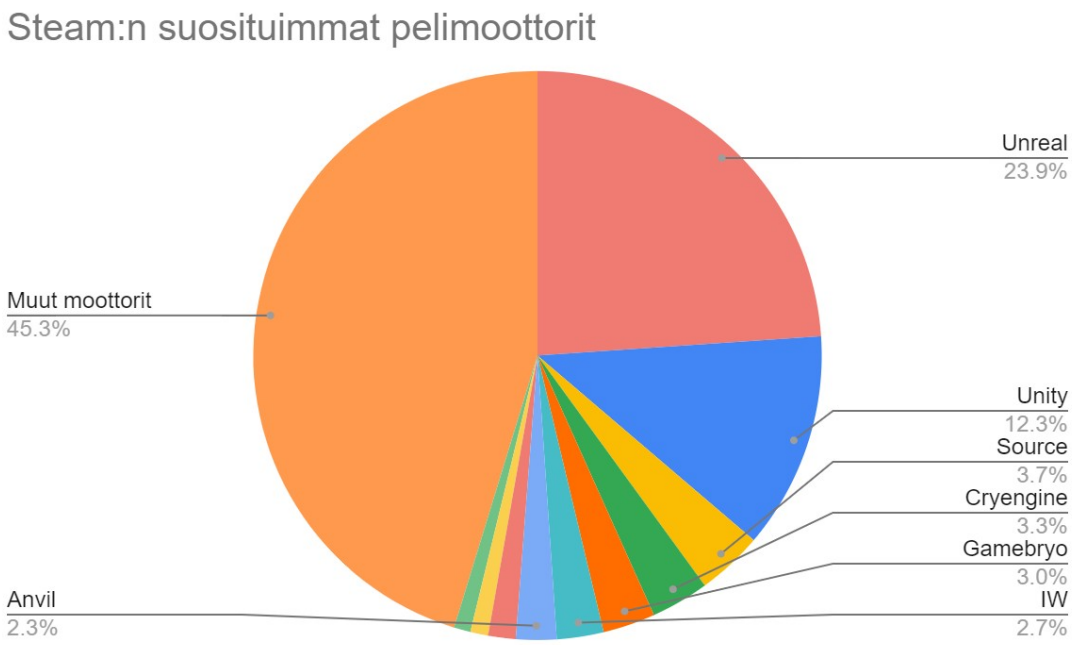
tarkkaa kirjaa siitä, millä pelimoottorilla kukin siellä julkaistu peli on tehty. Kuviosta 1 nähdään, että yli puolet sivuston peleistä on tehty Unity-pelimoottorilla. Toisena ja kolmantena olevia pelimoottoreita, Constructia ja GameMakeria, käytetään vain vajaaseen viidesosaan kaikista peleistä. Unity on selvästi ylivoimaisin pelimoottori sivustolla.



KUVIO 1. Ympyräkaavio itch.io-pelikaupan suosituimmista pelimoottoreista (itch.io, 2020)

Kuviosta 2 taas nähdään tämän hetken suosituimman tietokonepelikaupan, Steamin, tilaston peleistä. Epic Gamesin Unreal-pelimoottori on käytetyin pelimoottori, mutta Unity tulee heti toisena. Kaavio ei kuitenkaan ole virallinen, sillä Steam ei pidä kirjaa siitä, millä pelimoottorilla pelit on tehty. Kuvioon käytetty tilasto on käyttäjien itse keräämää dataa, joka ei sisällä läheskään kaikkia pelejä, joita Steamissä on myynnissä. Yksi Gamasutra-nettisivun blogaajista, Marcus Toftedahl (2019), loi ohjelman, joka ensin keräsi dataa Steamissä julkaistuista peleistä, etsi näiden pelien Wikipedia-sivun ja päivitti tilastoon pelimoottorin, jos se oli merkitty. Koska Wikipedia ei ole kaikkein luotettavin tiedon lähde, vain suosituimmilla peleillä on oma Wikipedia-sivu, ja vielä harvemmalla on merkitty, millä pelimoottorilla peli on tehty. Tilasto on erittäin suppea eikä kovin luotettava. Tästä huolimatta kaavio suuntaa antava mitä pelimoottoreita käytetään kaupallisissa julkaisuissa. Jos tilasto olisi

täydellinen, uskoisin Unityn olevan lähempänä Unreal-pelimoottoria, ellei jopa ylittänyt sitä tilastossa, mutta tämä on vain spekulaatiota.



KUVIO 2. Ympyräkaavio Steam-pelikaupan suosituimmista pelimoottoreista (Toftedahl, M, 2019)

2.2 Unity-pelimoottori

Unity3D (lyhyemmin vain Unity) on Unity Technologiesin alun perin vuonna 2005 kehittämä pelimoottori, jonka päätarkoitus oli toimia pelien kehitysalustana juuri 3D-peleille. Sen suosio alkoi, kun 2008 Apple Inc:n julkaisi oman Apple Store-sovelluskauppansa mobiililaitteille. Unity loi tuen Applen mobiililaitteille, ja siitä tuli nopeasti suosittu Apple-pelikehittäjien keskuudessa. Siitä lähtien Unityn suosio on noussut tasaisesti Unityn luodessa työkaluja pelien kehittämiseen muillekin alustoille, kuten pelikonsoleille ja nettisivuille. (Haas, 2014)

On monta asiaa, mitkä tekevät Unitystä niin suositun. Unity on ilmainen niin kauan kun sillä luotu peli ei ole tuottanut yli 100 000 dollaria viimeisen 12 kuukauden aikana. Juuri tästä syystä se on ollut erittäin suosittu juuri indie-pelinkehittäjien keskuudessa, joilla ei välttämättä ole rahoitusta, eikä siksi varaa hankkia maksullista pelimoottoria. (Haas, 2014) Jotkin Unityn ominaisuuksista

ovat kuitenkin ilmaisessa versiossa käytössä rajoitetusti, kuten Unityn verkko- ja moninpeli ominaisuudet. Unity tukee monia eri pelialustoja, ja pyrkii jatkuvasti lisäämään niitä. Nämä sisältävät esimerkiksi uusimmat pelikonsolit, kuten Nintendo Switch ja PS5, kuin myös jokapäiväiset laitteet, kuten tietokoneet, mobiililaitteet ja verkkoselaimet. (Unity Technologies)

Ehkä suurin syy Unityn menestykseen on kuitenkin käyttäjien luoma yhteisö, joka on syntynyt Unityn ympärille. Unityn nettisivut sisältävät vapaan keskustelupalstan, jossa kaikentasoiset käyttäjät voivat keskustella projekteistaan, Unityn käytöstä, sekä pyytää apua mahdollisiin ongelmiin, joita he kohtaavat projekteissa. Tässä ei kuitenkaan ole kaikki, sillä Unitylla on oma kauppa, Asset Store, jossa myydään assetteja, työkaluja pelin tekemiseen. Assetit tarkoittavat pelin kaikenlaista sisältöä, kuten kuvia, 3D-mallinnuksia ja ääniefektejä. Työkalut taas lisäävät Unityn ominaisuuksia, jotka helpottavat sen käyttöä. Jotkin työkalut voidaan myös lisätä peliin, jolloin ne luovat peliin tiettyjä ominaisuuksia, jotta pelin tekijän ei itse tarvitse nähdä aikaa ja vaivaa näiden ominaisuuksien ja työkalujen tekemiseen. Assettien tekemiseen tarvitaan yleensä ulkoisia ohjelmia, riippuen siitä minkälaisia assetteja käyttäjä haluaa tehdä, mutta työkaluja voi kuka tahansa tehdä Unityyn käyttämällä C#-ohjelmointikieltä.

3 KÄYTTÖLIITTYMÄ

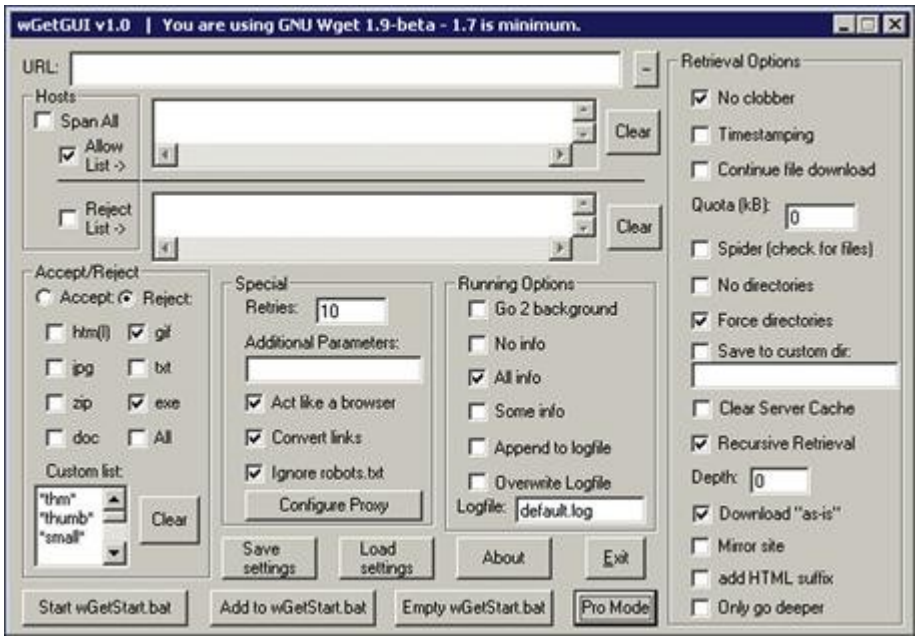
3.1 Mikä on kättöliittymä?

Käyttöliittymä, lyhyemmin UI (User Interface), on joukko välineitä, jotka antavat pelaajan olla vuorovaikutuksessa pelin kanssa. UI on siis mitä tahansa interaktiota käyttäjän ja pelin välillä. Kun puhutaan pelkästään näkyvästä UI:sta, käytetään termiä graafinen käyttöliittymä, lyhyemmin GUI (Graphical User Interface). GUI tarkoittaa siis vain käyttöliittymän graafista puolta, eli asioita, jotka ovat käyttäjälle näkyvissä, kuten nappeja ja valikoita, kun taas UI tarkoittaa koko käyttöliittymää kokonaisuutena, joka saattaa sisältää asioita jotka eivät välttämättä ole käyttäjälle graafisesti näkyvissä, kuten pikanäppäimiä. Nämä kaksi kuitenkin sekoitetaan usein yhdeksi, ja kun puhutaan ohjelman ja pelin graafisesta ulkoasusta, puhutaan usein pelkästään sen UI:sta. (Goldbold, 2018)

Hyvä UX (User Experience, käyttäjäkokemus) on kuin keskustelu ohjelman kanssa. Huono UI on luonnotonta ja keskittyy liikaa ohjelman yksityiskohtiin ja ominaisuuksiin koodin sisällä, joista käyttäjä ei välitä tai ymmärrä. Hyvä UI osaa näyttää asiat ystävällisesti mutta ammattimaisesti luoden helposti ymmärrettävän ja luonnollisen kokemuksen ohjelman kanssa. Hyvä UI ei vaadi käyttäjää muistamaan asioita, ja näyttää käyttäjälle vain kaiken oleellisen käyttäjä tarvitsee tehtävänsä suorittamiseen. (McKay, 2013)

Yleisin virhe minkä yritykset tekevät UI:n luomisessa on se, että he laittavat ohjelman ohjelmoijat joilla ei ole UX kokemusta luomaan ohjelmalle UI:n. Tämä vaikuttaa viisaalta valinnalta koska ohjelmoijat osaavat opastaa kenet tahansa käyttämään luomaansa ohjelmaa. Mutta kun itse käyttäjä yrittää käyttää ohjelmaa, siellä ei ole ketään opastamassa häntä. Tällöin UI:n on oltava se, joka opastaa käyttäjää ohjelman eri ominaisuuksiin. Ohjelmoijat usein luovat UI:n samalla tavalla kuin koodinkin, antamalla käyttäjälle kaikki työkalun ominaisuudet käyttöön yhdellä kertaa ilman järkevää jaottelua. Tämä luo monimutkaisen UI:n joka on täynnä ammattikieltä jota käyttäjä ei ymmärrä, ja

joka on täynnä valintoja ilman intuitiivista asettelua. Tästä näemme esimerkin kuvassa 1. (McKay, 2013)



KUVA 1. Esimerkki huonosta käyttöliittymästä (McKay, 2013)

Hyvän ohjelman UI:n tärkein tehtävä on olla intuitiivinen. Intuitiivisuus on kuitenkin erittäin laaja käsite jota ei käsitellä kovinkaan laajasti missään. Tästä syystä UX-suunnittelu ja konsultointi yrityksen UX Design Edgen johtaja, Everett McKay (2013) täsmentää että ”UI on intuitiivinen kun kohderyhmä ymmärtää sen käyttäytymisen ja vaikutuksen ilman päättelyä, muistamista, kokeilua, apua tai koulutusta” . Eli UI on hyvä silloin, kun käyttäjä pystyy käyttämään ohjelmaa ajattelematta asiaa juuri ollenkaan. Jokaisen ohjelman tulisi tähdätä siis pyytämään käyttäjää tekemään asioita kohta kerrallaan ilman että hänen tarvitsee ajatella asiaa liikaa.

McKay (2013) tiivistää hyvän UI:n 5 sääntöön, joiden pääajatus on tehdä UI siten, että se vaikuttaa hyvältä keskustelulta. Ohjelman tulee kommunikoida käyttäjän kanssa, olla selkeä, kohtelias, ystävällinen ja ammattimainen. Jos UI-elementti kertoo käyttäjälle jotain millä ei ole käyttäjälle väliä, se on kerrottu jo aiemmin jossain muualla, tai se on liian monimutkainen ymmärtää, tulisi UI-elementtiä joko muuttaa tai poistaa ohjelmasta kokonaan. UI tulisi luoda samalla tavalla, kun ohjelman käyttöä opettaisi jollekin toiselle henkilölle.

3.2 Unityn käyttöliittymä

Jotta voidaan ymmärtää Unityn käyttöliittymää, on ymmärrettävä, miten Unityssä luodaan pelejä, ja miten ne toimivat.

Unityssä koko peli luodaan lisäämällä peliin objekteja. Objekteja voi olla niin monta kuin halutaan, ja niille voidaan asettaa tarvittaessa positio, rotaatio, ja skaala. Objektit voivat olla pelissä sellaisenaan, tai ne voidaan asettaa jonkin toisen objektin alle, jolloin alempi objekti perii ylemmän objektin arvoja. Tällöin ylemmästä objektista käytetään nimitystä vanhempi, ja alemmasta, joka perii vanhempi-objektin arvoja, kutsutaan lapseksi.

Jokainen ominaisuus joka lisätään peliin pitää olla jossakin objektissa, joka on pelissä. Objektit voivat pitää sisällään käytännössä mitä tahansa: 2D-kuvia, 3D-mallinnuksia, tai ääniefektejä. Näitä ominaisuuksia objekteissa kutsutaan komponenteiksi. Jokaiselle ominaisuudelle on oma komponenttinsa, ja niitä voi lisätä objekteihin niin monta kuin käyttäjä haluaa. Unity sisältää laajan valikoiman komponentteja, mutta ne eivät silti riitä pelin tekemiseen. Tästä syystä käyttäjän on myös luotava omia skriptejä, eli koodinpätkiä, jotta hän saa pelin toimimaan halutulla tavalla. Jokainen skripti jonka käyttäjä luo voidaan automaattisesti lisätä objekteihin komponenttien tavoin luomaan ominaisuuksia peliin.

Unityn käyttöliittymä perustuu eri ikkunoihin, joita ohjelmassa on auki. Jokaisella ikkunalla on oma käyttötarkoituksensa pelin luomiseen. Ikkunoita voi olla auki niin monta kun käyttäjä haluaa, ja käyttäjä voi siirrellä ja luoda Unitystä haluamansa näköisen. 5 tärkeintä ikkunaa on auki heti ensimmäisellä kerralla kun avaat Unityn, ja nämä on numeroitu kuvassa 2.

Unity tärkeimmät ikkunat ovat:

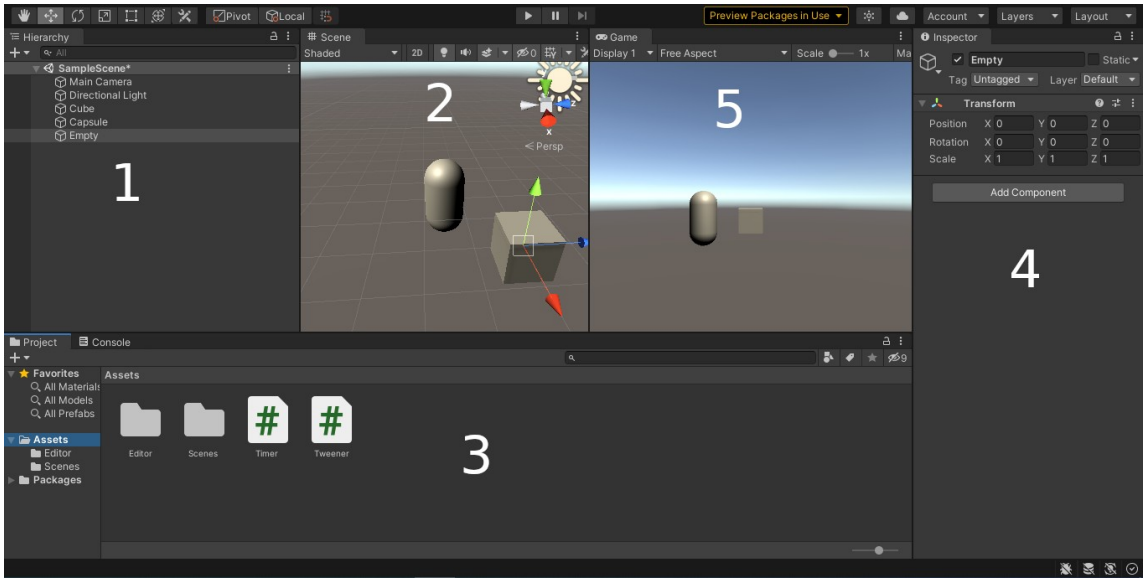
1) Hierarchy-ikkuna pitää sisällään kaikki objektit joita pelissä on.

2) Scene-ikkuna näyttää kaiken visuaalisen mitä pelissä on. Jos valitsemme objektin Hierarchy-ikkunassa, näemme Scene-ikkunassa missä kyseinen objekti on, ja miltä se näyttää. Voimme myös halutessamme liikuttaa, pyörittää ja skaalata objekteja Scene-ikkunassa.

3) Project-ikkuna pitää sisällään kaikki käyttäjän luomat tiedostot joita hän tarvitsee projektissa. Käyttäjä voi järjestellä tiedostot haluamallaan tavalla, luoda kansioita muutamalla klikkauksella, ja lisätä uusia tiedostoja projektiin vetämällä ne ikkunaan. Näitä tiedostoja voidaan myös lisätä peliin objekteiksi vetämällä ne joko Hierarchy-ikkunaan tai Scene-ikkunaan.

4) Inspector-ikkuna näyttää kaikki valitun asian tiedot. Nämä valittavat asiat voivat olla esim. tiedostoja Project-ikkunassa tai objekteja Hierarchy- tai Scene-ikkunassa. Inspector-ikkuna on ehkä tärkein ikkuna koko Unityn käyttöliittymässä koska se näyttää tarvittavaa informaatiota kaikista muista ikkunoista.

5) Game-ikkuna, joka näyttää miltä peli tällä hetkellä näyttää. Game-ikkuna näyttää pelin kameran perspektiivistä, ja kun käyttäjä testaa omaa peliään, käyttää hän Game-ikkunaa pelin testaamiseen.



KUVA 2. Kuvakaappaus Unity-pelimoottorin version 2020.1.5f1 käyttöliittymästä.

Unity sisältää monia muitakin ikkunoita, ja niillä jokaisella on oma käyttötarkoituksensa pelimoottorin sisällä. Käyttäjien on myös mahdollista luoda lisää ikkunoita itse tekemilleen työkaluille, ja moni Asset Storesta ostettava työkalu pitää sisällään uusia ikkunoita, jotka helpottavat pelin luomista.

Unityn UI saattaa vaikuttaa sekavalta, mutta ottaen huomioon kuinka monikäyttöinen pelimoottori on ja kuinka monia ominaisuuksia se pitää sisällään, on Unity tehnyt joitain hyviä valintoja. Koska Unity on jakanut kaikki

tärkeät ominaisuudet omiin ikkunoihinsa, joita käyttäjä voi avata ja sulkea haluamallaan tavalla, saa käyttäjä esille vain kaiken tarvitsemansa silloin kun käyttäjä haluaa.

Unity myös tiedostaa ettei jokainen ominaisuus ole niin intuitiivinen kuin mahdollista, minkä takia heillä on kattava oppimismateriaali Unityn käyttöön. Tämä oppimismateriaali voidaan asentaa tietokoneelle samalla kun asennetaan Unity-pelimoottori. Kaikki viralliset tutoriaalit ovat myös löydettävissä Unity Technologiesin omilla nettisivuilla sekä youtube-kanavalla. Käyttäjät ovat myös luoneet paljon tutoriaaleja pelimoottorin käyttöön, joten vaihtoehtoja on paljon mikäli haluaa pelimoottoria käyttää.

4 ANIMAATIOT

4.1 Mikä on animaatio?

Animaatio on audio-visuaalinen media, jossa yhdistetään liikkuva kuva ja ääni luomaan uusia kokemuksia. 1900-luvulla animaatio toteutettiin luomalla yksi frame, eli kuva, kerrallaan, ja kun niitä näytetään nopeasti yksi toisensa jälkeen, luodaan illuusio liikkuvasta kuvasta, eli animaatio. Tämä prosessi oli kuitenkin erittäin työlästä, minkä takia animaatioiden luominen oli hankalaa ja vaati paljon aikaa. Nykypäivän tietokone-teknologia on kuitenkin nopeuttanut animaatioiden luomista hurjasti, ja tämä pätee myös peleihin. (Sito, 2013)

Tavallinen animaatio, kuten piirrossarja, on staattinen, sillä se on samanlainen joka kerta kun katsot sen. Peleissä kuitenkin käytetään dynaamisia animaatioita. Tämä tarkoittaa, ettei animaatio pyöri aina samalla tavalla, koska animaation pitää reagoida käyttäjän liikkeisiin ja ohjaukseen (Nabors). Tästä syystä pelinkehittäjien pitää luoda monia erillisiä animaation pätkiä, jotka yhdistetään toisiinsa pelin sisällä riippuen siitä, mitä pelaaja tekee. Esimerkiksi kun pelaaja liikkuu, pyöritetään kävely animaatio, mutta kun pelaaja hyppää pitää sen vaihtaa sulavasti hyppy animaatioon ilman että se vaikuttaa keinotekoiselta. Animaatio ei voi odottaa että kävelyanimaatio loppuu, ja vasta sitten hypätä, koska sen pitää tuntua interaktiiviselta. (Cooper, 2019)

Animaation sulavuus riippuu myös siitä, kuinka suuri FPS (frames per second, kuvaa sekunnissa) animaatiossa on. Mitä enemmän kuvia animaatio näyttää sekunnin aikana, sitä paremmalta ja sulavammalta animaatio vaikuttaa. Peleissä FPS on 30-60 välillä, mikä tarkoittaa että se näyttää 30-60 kuvaa sekunnissa luomaan sulavan animaation pelille. Tämä vaatisi erittäin paljon työtä ilman nykyaikaisia tietokoneita ja ohjelmia animaation luomiseen. (Cooper, 2019)

Tietokoneet ja animaatio-ohjelmat ovat muuttaneet koko animaation luomisprosessin luomalla konseptin, keyframe. Keyframe on erityinen frame joka pitää sisällään tietoa kuvasta kuten sen paikan, rotaation ja koon. Kun

luodaan kaksi Keyframea, joiden välissä on useita, tavallisia frameja, alkaa ohjelma muuttamaan kuvan arvoja ensimmäisestä Keyframesta frame kerrallaan kunnes arvot ovat samat kuin jälkimmäisessä keyframessa. Toisin sanoen, ohjelma luo helposti usean framen pituisen animaation asettamalla sille vain alku- ja loppu keyframen. Näitä keyframeja pystytään luomaan niin monta kuin halutaan monelle eri kuvalle, jolloin saadaan luotua monimutkaisia animaatioita. (Sito, 2013)

4.2 Animaatiot Unityssä

Unityn animaatiot toimivat aiemmassa kappaleessa mainittujen komponenttien ja ikkunoiden avulla. Jotta animaatiot saadaan toimimaan Unityssä oikein, vaatii se monta eri työvaihetta. Käyttäjän on ensiksi luotava ainakin kaksi tiedostoa Unityn sisällä: Animation Clip, ja Animator Controller. Animation Clip-tiedosto on itsestään selvä. Se sisältää kaiken informaation yhdestä animaatiosta kuten sen keyframet ja kuvat, ja näitä tiedostoja tulisikin tehdä niin monta kuin haluaa erilaisia animaatioita. Animator Controller tulisi sisältää kaikki yhden objektin ja sen kaikkien lasten animaatiot. Animator Controller ohjaa, miten näiden animaatioiden välillä liikutaan käyttäjän haluamalla tavalla. Kun nämä tiedostot on luotu, käyttäjän on avattava Animator-ikkuna, jonka tarkoitus on muokata Animator Controller-tiedostoa. Käyttäjä lisää juuri luomansa Animation Clip-tiedoston sen sisään, ja tekee siihen tarvittavat muutokset, että animaatio pyörii vasta silloin kuin käyttäjä haluaa. Tämän jälkeen käyttäjä lisää Animator-komponentin haluamaansa objektiin, ja yhdistää Animator Controller-tiedoston siihen. Nyt käyttäjän on avattava toinen ikkuna nimeltään Animation, jotta hän voi vihdoin luoda haluamansa animaation objektille. Viimeiseksi käyttäjän on luotava koodi, joka laukaisee animaation Animator-komponentissa. (Unity Technologies Docs)

Tämä prosessi on ymmärrettävää, mikäli käyttäjä haluaa tehdä monimutkaisia animaatioita jotka kontrolloivat useita objekteja samanaikaisesti. Mutta mikäli käyttäjä haluaa vain luoda yksinkertaisen animaation paikasta A paikkaan B, prosessi on aivan liian pitkä ja monimutkainen. Animator-komponentti myös lisää muita ongelmia. Animator-komponentti lukitsee objektit paikoilleen, minkä

takia niitä ei pysty liikuttamaan edes silloin kun animaatio ei pyöri. Tämä ei ole käytännöllistä, ja aiheuttaa useita ongelmia kun pelistä yrittää tehdä mahdollisimman interaktiivisen. Tämä on korjattavissa, mutta aiheuttaa ylimääräistä vaivaa ja työvaiheita käyttäjälle.

5 CASE-TWEENER

5.1 Toimeksianto

Opinnäytetyön tavoitteena oli suunnitella toimeksiantajalle, Mohavi Creative Company oy:lle, monikäyttöinen työkalu yksinkertaisten animaatioiden luomiseen. Alunperin työkalu kehitettiin osaksi yrityksen luomaa asiakasprojektia. Työkalun oli tarkoitus vain helpottaa kyseisen projektin suunnittelua ja UI:n luomista, mutta Mohavi Creative koki työkalun niin hyödylliseksi, että päätti jatkokehittää työkalun omaksi tuotteeksensa. Työkalua aiotaan käyttää yrityksen tulevissa projekteissa, sekä mahdollisesti myydä muille kehittäjille.

Työkalusta haluttiin kehittää mahdollisimman intuitiivinen, että myös tiimin jäsenet, joilla ei ole kokemusta Unityn käytöstä osaisivat käyttää sitä pelin kehityksessä. Työkalun tulisi myös olla monikäyttöinen että sen myyminen laajemmalle asiakaskunnalle olisi helpompaa. Työkalun alkuperäinen tarkoitus oli pelkästään helpottaa UI:n luomista peliin, mutta opinnäytetyön aikana se tulisi jatkokehittää toimimaan myös 2D- ja 3D-ympäristöissä. Työkaluun lisätään myös muita tarvittavia ominaisuuksia, kuten toimivuus pelimoottorin fysiikkamoottorin kanssa, sekä pelimoottorin nopeusarvojen muutoksiin reagointi.

5.2 Toteutus

Projektin nimi, Tweener, tulee animaatioissa ja peleissä käytettävästä ”Tween” sanasta, joka puolestaan tulee englannin kielen sanoista inbetween, eli välissä. Tween tarkoittaa juurikin kahden Keyframen välissä olevaa animaatiota, jonka tietokone luo puolestasi. Sanaa käytetään myös verbinä, tweening tai inbetweening, joka tarkoittaa tällaisen animaation tekemistä. Tweener siis tarkoittaa tällaisen animaation tekijää, mikä työkalu periaatteessa on. Nimi on kuitenkin vain väliaikainen nimi projektille, sillä Tweenerin kaltaisia työkaluja on muitakin, ja työkalulle tulisi keksiä erottuvampi nimi kun sitä aletaan myymään.

Tweenerin suunnitteluprosessi aloitettiin suunnittelemalla, minkälainen työkalusta haluttiin. Unityssä voidaan luoda työkalu kahdella eri tavalla. Työkalu voi olla joko kokonaan oma ikkunansa Unityssä, tai se voidaan vain liittää komponenttina objekteihin. Ikkunan luominen työkalulle on järkevää tilanteissa, kun käyttäjälle halutaan välittää paljon visuaalista informaatiota, jota hän tarvitsee työkalun käyttämiseen. Koska työkalun päätarkoituksellinen tehtävä on toimia nopeana ja intuitiivisena kokemuksen, päätimme tehdä työkalusta komponentin. Komponenttien käyttö on ensimmäinen asia, minkä jokainen käyttäjä oppii Unitystä, joten aloittelijat voivat helposti oppia työkalun käytön. On myös nopeampaa lisätä komponentti objektiin ja muokata komponenttia suoraa, sen sijaan että avattaisiin kokonaan uusi ikkuna vain tätä tarkoitusta varten.

Seuraavaksi aloimme suunnitella, mitä kaikkia ominaisuuksia työkalulle tarvitaan. Työkalun tarvitsee aloittaa ja lopettaa ajanlasku halutulla nopeudella silloin kuin halutaan. Tätä varten luotiin oma ajastin-skripti joka pitää silmällä jokaisen tweener-skriptin aloitus-, nopeus- ja kestoaikaa. Ajastin tarkistaa, pitääkö työkalun toistaa itseään useasti, vai pyöriikö se vain kerran. Ajastin myös pitää huolen pyöriikö Tweener eteen- vai taaksepäin, vai molempiin suuntiin vaihtelevasti. Koska ajastin-skripti on yksinkertainen ja helppo tehdä, siitä pyrittiin tekemään tarpeeksi intuitiivinen että käyttäjät voivat halutessaan käyttää sitä projekteissaan sellaisenaan.

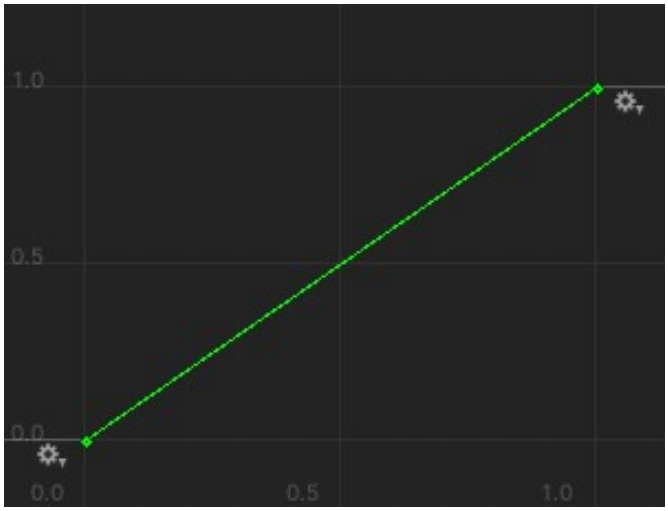
Ajastimen jälkeen lähdettiin luomaan itse animaatiota. Käytämme työkalussa samaa ideaa kuin tietokone-animaatiotkin, mutta pienemmässä skaalassa. Periaatteessa luomme työkalulla aloitus- ja lopetus-keyframet, ja kuljemme animaatiossa näiden kahden välillä. Tätä varten tarvitsemme käyttäjältä sekä aloitus-, että lopetusarvot, jotta voimme luoda animaation. Ensimmäiseen versioon joka luotiin vain UI käyttöön muuttamaan objektien positiota, luotiin työkalulle aloitus ja lopetus kentät joihin käyttäjä pystyy lisäämään haluamansa aloitus ja lopetus position. Koska työkalu haluttiin monikäyttöisemmäksi, lisättiin aloitus ja lopetus kentät myös rotaatiolle ja skaalalle. Tämä kuitenkin täyttää nopeasti työkalun editor-ikkunan turhalla informaatiolla jota ei tarvita. Tästä syystä työkalulle päätettiin luoda eri moodeja jotka sisältävät vain tarvittavat kentät juuri kyseiselle moodille. Nämä olivat: Position, Rotation, Scale, sekä

Transform. Position sisälsi arvon vain objektin positiolle, Rotation rotaatiolle ja Scale skaalalle, mutta viimeinen, Transform, sisälsi kentät aloitus ja lopetus objekteille. Transform- moodin ansiosta, sen sijaan että käyttäjän tarvitsee kirjoittaa työkaluun aloitus ja lopetus arvot positiolle, rotaatiolle ja skaalalle, käyttäjä voi vain joko luoda tyhjät objektit tai käyttää jo pelissä muuten olevia objekteja, asettaa niihin halutun position, rotaation ja skaalaan, ja asettaa ne työkaluun. Tällöin työkalu saa kaikki 3 arvoa ilman, että työkalun UI täyttyy kaikella tällä informaatiolla. Tämä myös mahdollistaa animaation helpon muokkaamisen pelin aikana muutamalla vain objektien arvoja.

Projektin edetessä luotiin vielä kaksi moodia lisää: Color, ja Random Shake. Color-moodiin voidaan laittaa objektin aloitus ja lopetus väri, ja animaatio muuttaisi näiden kahden välillä, kun taas Random Shake- moodiin laitetaan nopeus ja etäisyys, ja animaatio tärisyttää objektia. Random Shake-moodi oli helppo toteuttaa, mutta Color-moodin lisääminen oli varsin hankalaa. Color-moodin tulisi toimia 2D-, 3D- sekä UI-ympäristöissä, ja jokainen ympäristö käyttää hieman erilaisia keinoja ja komponentteja objektien näyttämiseen pelissä. Tästä syystä, heti kun peli alkaa, työkalu tarkistaa onko se asetettu 2D-, 3D- vai UI-objektiin, ja tästä riippuen muuttaa väri-arvoja vain tietyissä komponenteissa.

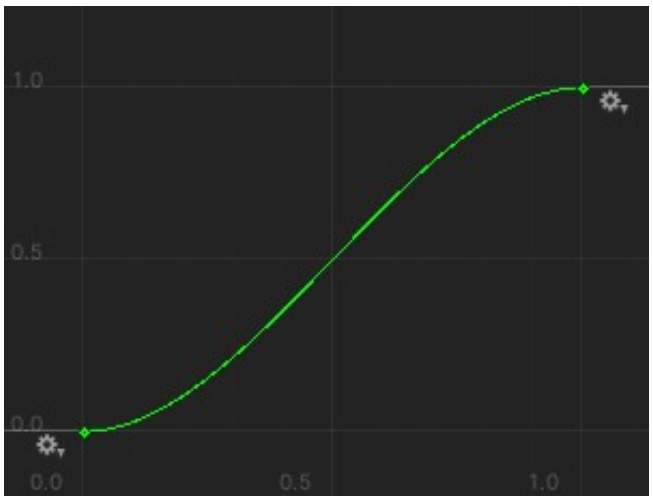
Tässä kohtaa työkalulla oli mahdollista luoda animaatio pisteestä A pisteeseen B, mutta animaatio oli erittäin kankea. Koko animaatio pyörii samalla nopeudella, joka saa aikaan erittäin mekaanisia animaatioita ilman minkäänlaista pehmennystä alussa tai lopussa. Tutkittuani aihetta hetken minulle selvisi tämän johtuvan siitä, etten käytä kurveja animaatiossani. Kurvi (eng. Curve) tarkoittaa graafia joka kertoo miten animaatio tulisi pyörittää. Graafissa kulkee viiva vasemmasta alanurkasta, eli (0, 0) pisteestä oikeaan ylänurkkaan, eli (1, 1) pisteeseen. Graafissa x arvo tarkoittaa animaation aikaa, eli kun x-arvo on nolla, on animaatio aivan alussa, ja kun x-arvo on 1, on animaatio päässyt loppuun. Y-arvon muutos taas kertoo animaatiolle kuinka suuri muutos A-pisteestä B-pisteeseen animaation tulisi sillä hetkellä olla. Eli kun y-arvo on 0, on animaatio täysin A-pisteessä, ja kun y-arvo on 1, on animaatio täysin B-pisteessä.

Seuraavassa kuviossa (Kuvio 3) näemme miten animaationi tällä hetkellä toimi. Se kulkee suoraa A-pisteestä B-pisteeseen samaa vauhtia, mikä luo erittäin jyrkän ja mekaanisen animaation. Vaikka tämä animaatio on hyvä joihinkin tapauksiin, ei se ole silti hyvä kaikkiin.



KUVIO 3. Suora, ”mekaaninen” animaation jana (0,0) pisteestä (1,1) pisteeseen.

Seuraavassa kuviossa (Kuvio 4) nähdään perinteisempi animaatio käyrä. Animaatio alkaa hitaasti, ja nopeuttaa vauhtiansa kunnes x-arvo pääsee yli puolenvälin, jolloin se alkaa hidastaa vauhtiansa samalla tavoin kuin nopeutti sitä. Tämä luo paljon luonnollisemman animaation jossa animaatiolle annetaan aikaa päästä vauhtiin ja hidastua ennen pysähdystä. Tämän ominaisuuden luominen oli helppoa, sillä Unity sisältää kaikki tarvittavat työkalut kurvien lisäämiseen työkaluun.



KUVIO 4. Kurvikas, ”luonnollinen” animaation jana (0,0) pisteestä (1,1) pisteeseen.

Tässä kohtaa animaatiot alkavat näyttää jo hyvältä. Niille on mahdollista asettaa kesto, nopeus, aloitus ja lopetus arvot, sekä kurvi joka kertoo miten animaation tulisi pyöriä. Seuraavaksi työkaluun lisättiin joitain perusominaisuuksia. Yksi oli animaatioiden automaattinen pyörittäminen ja uudelleenkäynnistys, jotka tarkistavat työkalun tullessa aktiiviseksi tuleeko sen heti tarkistaa käynnistys arvot, ja lähteä heti pyörittämään animaatiota vai ei. Toinen oli animaation looppaus, joka tarkistaa kuinka monta kertaa animaation tulisi pyöriä uudestaan ennen kuin se lopettaa, ja miten animaatio looppaa. Perinteisesti animaatioilla on kaksi eri tapaa loopata: Restart, jolloin animaatio looppaa tavallisesti aina aloittaen alusta loppuun, sekä Ping Pong, joka tarkoittaa animaation pyörivän ensin alusta loppuun, sitten lopusta alkuun, ja taas alusta loppuun edes takaisin. Ping Pong-looppaus saa nimensä pöytätenniksestä jota kutsutaan myös nimellä Ping Pong johtuen pallosta jota pelaajat lyövät edestakaisin, aivan kuten animaatiokin.

Seuraava suuri ongelma tuli kun huomattiin ettei animaatio pyöri oikein, jos se on jonkun muun objektin lapsi. Tämä johtui siitä, että animaatiot toimivat vain globaalissa ulottuvuudessa (global space) lokaalin ulottuvuuden (local space) sijaan. Unityssä objektit voivat olla toisten objektien lapsia, mikä tarkoittaa että lapsi-objekti perii vanhempi-objektin arvot, kuten positio, rotaatio ja skaalaus. Eli kun vanhempi-objekti on esim. positiossa (1, 1, 1) ja lapsi objekti on positiossa (0, 0, 0), tarkoittaa se että molemmat ovat oikeasti positiossa (1, 1, 1) koska lapsi objekti perii vanhempi-objektin positio arvot. Eli kun halusimme animaation

liikkuvan jonkun lapsena paikasta A paikkaan B, ei se välittänyt vanhempi-objektin positiosta. Tämä oli helppo korjata ottamalla laskussa huomioon vanhempi-objektin positio, lisäämällä metodi joka tarkistaa kumpaa ulottuvuutta käytetään, sekä käyttäjälle valikko, jossa käyttäjä voi päättää kumpaa ulottuvuutta hän haluaa animaation käyttävän.

Aloimme myös harkita monimutkaisempia käyttötapoja työkalulle. Tajusimme ettei käyttäjä välttämättä halua lisätä aloitus arvoja erikseen, vaan käyttää objektin senhetkisiä arvoja aloitus arvoina. Eli luotaisiin animaatio objektin senhetkisten arvojen ja käyttäjän asettamien arvojen välille. Toinen syvempi tarkoitus oli jos käyttäjä, sen sijaan että asettaisi animaation lopulliset arvot, hän asettaisi arvot jotka lisätään aloitus arvoihin. Eli jos aloitusarvot ovat (1, 1, 1) ja lopetus arvot ovat (2, 2, 2), animaatio kulkisikin (1, 1, 1) arvoista (3, 3, 3) arvoihin. Tämä animaatiotyyli sopisi varsin hyvin looppien kanssa, sillä työkalu pystyisi toistamaan animaatiota monta kertaa, aina aloittaen ja lopettaen uuteen paikkaan.

Tässä kohtaa työkalu alkaa sisältää jo kaiken olennaisen animaatioiden pyörittämiseen, mutta ei ole mitään tapaa kertoa muulle pelille että animaatio on alkanut tai loppunut. Tätä varten työkaluun lisättiin eventit kun animaatio käynnistetään ja pääsee loppuun. Eventit ovat Unityn versio delegate-muuttujasta. Delegate-muuttujat ovat siis muuttujia, joihin voidaan asettaa metodeita. Kun delegate-muuttujaa kutsutaan, se laukaisee kaikki metodit, jotka sen sisään on tallennettu. Delegaten ja Unityn Eventin erona on se, että Unityn Eventit voidaan asettaa Unityn editorissa työkaluun. Tämä tarkoittaa, että käyttäjä pystyy asettamaan haluamansa metodit suoraa Unityn editorissa työkaluun, ja työkalu laukaisee ne silloin kun animaatio käynnistyy ja loppuu. Tämä antaa käyttäjälle luoda monimutkaisempia animaatioita asettamalla ne toimimaan yksi toisensa jälkeen, tai luoda peliin ominaisuuksia asettamalla monia asioita tapahtumaan kun animaatio alkaa tai loppuu.

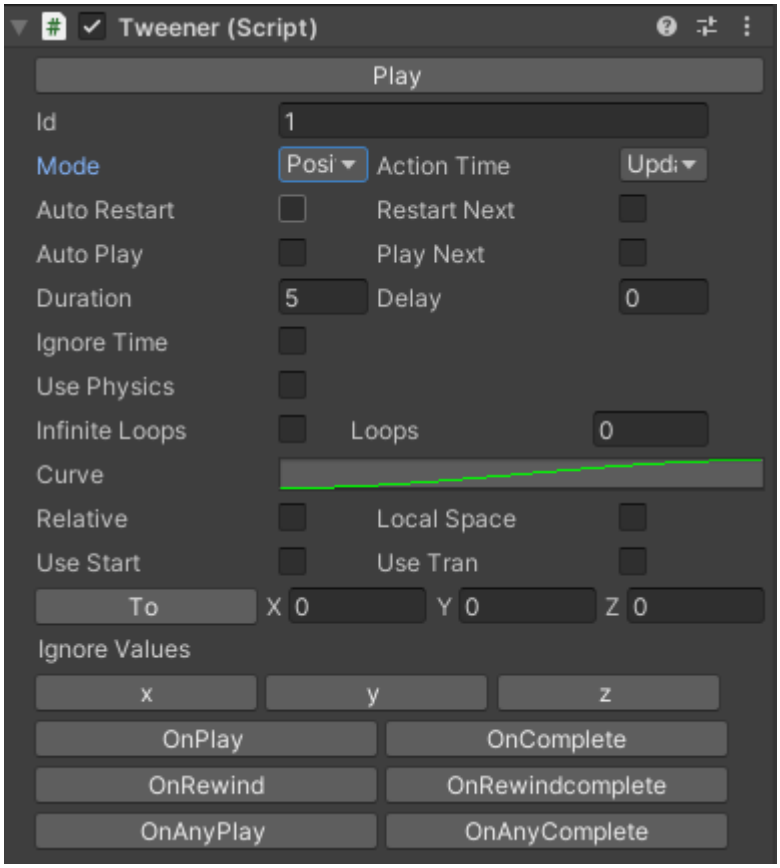
Yhdessä objektissa saattaa olla useampi tweener-työkalu kerrallaan kiinni, joten ongelmaksi syntyi, miten tiedetään, mikä on mikäkin työkalu. Tätä varten lisäsin työkaluun id-arvon, sekä tarvittavat metodit tämän löytämiseen eri työkaluista. Jokaisella tweenerillä on aluksi oma id:nsä joita käyttäjät voi muuttaa

halutessaan. Alunperin id otti vastaan pelkkiä numeraalisia arvoja, mutta pitemmän miettimisen jälkeen se vaihdettiin sanalliseksi koska käyttäjien on helpompi muistaa kokonaisia sanoja pelkkien numeroiden sijaan. Työkaluun lisättiin myös metodit työkalun käynnistämiseen id:n avulla, kuten tarkistamalla oikean työkalun objektissa ennen sen käynnistämistä, tai käynnistämällä kaikki työkalut, joilla on sama id-arvo, vaikka ne olisivatkin kiinni eri objektissa. Nämä mahdollistavat työkalujen tarkan käynnistyksen pelin aikana.

Suurimmaksi ongelmaksi kuitenkin paljastui työkalun testaaminen itse editorissa käynnistämättä peliä. Työkalu toistuu frame kerrallaan pelin ollessa käynnissä, mutta itse editorissa ei automaattisesti pyöri frameja. Pitkän tutkimisen jälkeen selvisi, että editorissa kaiken kaikkiaan mahdollista käyttää frameja. Tätä varten on vain lisättävä haluttu metodi delegaatioon, jota kutsutaan joka frame, joka editorissa tapahtuu. Tällöin on mahdollista edetä animaatio frame kerrallaan, että nähdään miltä se näyttää avaamatta itse peliä. Animaatiot alkoivat toimia, mutta erittäin huonosti. Animaatiot eivät ensinnäkään osanneet palata aloitusarvoihin testaamisen jälkeen. Tästä syystä työkalulle oli luotava omat muuttujat, joihin tallennetaan aloitus arvot ennen animaation käynnistämistä. UI ei myöskään päivittynyt animaation ollessa käynnissä, minkä takia oli lisättävä erillinen delegate-muuttuja, ja siihen metodi, jotka mahdollistaisivat UI:n päivittämisen silloin kun haluttiin. Pitkän kamppailun jälkeen animaatiot tuntuivat toimivan editorissa halutulla tavalla.

Työkalu ei silti ole vielä valmis. Jos katsoo liitettä 1, nähdään miltä työkalu tällä hetkellä näyttää. Se on sotkuinen, ja näyttää käyttäjälle aivan liikaa työkalussa käytettyjä muuttujia ja arvoja. Tämä johtuu siitä, että suurin osa muuttujista on public (julkinen) muuttujia, jotta käyttäjä voi tarvitessaan päästä niihin arvoihin käsiksi koodin kautta. Unity näyttää public muuttujat automaattisesti editorissa, ellei niihin ole erikseen merkitty ettei niitä näytettäisi. Koska näytettäviä muuttujia on vähemmän kuin kaikkia, ja koska haluamme muutenkin muokata työkalusta hienomman, luomme työkalulle editor-skriptin. Editor-skripti pitää sisällään informaatiota miltä työkalun tulisi oikeasti näyttää Unityssa. Sen lisäksi, että editor-skripti vain kertoo, miten muuttujat näytetään Unityssa, voimme myös lisätä nappeja ja valikoita joita emme pystyisi muuten lisäämään.

Kuvassa 3 näemme miltä lopullinen työkalu näyttää. Kaikki ylimääräiset muuttujat joita käyttäjän ei tarvitse miettiä on poistettu näkyvistä. Loin UI:n kohta kerrallaan miettien mitä käyttäjä tarvitsee milloinkin. Ylimpänä on tärkeimmät, eli Play-nappula jolla kokeillaan animaatiota, työkalun oma id-arvo, mode jota käyttäjä haluaa käyttää, sekä action time, eli missä kohtaa framea animaatio pyöritetään. Muut valinnat asettelin siten, että käyttäjä pystyy helposti kohta kohdalta käymään läpi, mitä hän tarvitsee työkaluun. Eniten tilaa vievät, eli aloitus ja lopetus kohdat, mitä arvoja jättää huomiotta, sekä mahdolliset eventit joita käyttäjä haluaa laukaista missäkin tilanteessa, asetettiin työkalun loppuun, etteivät ne vie turhaa tilaa muilta arvoilta.

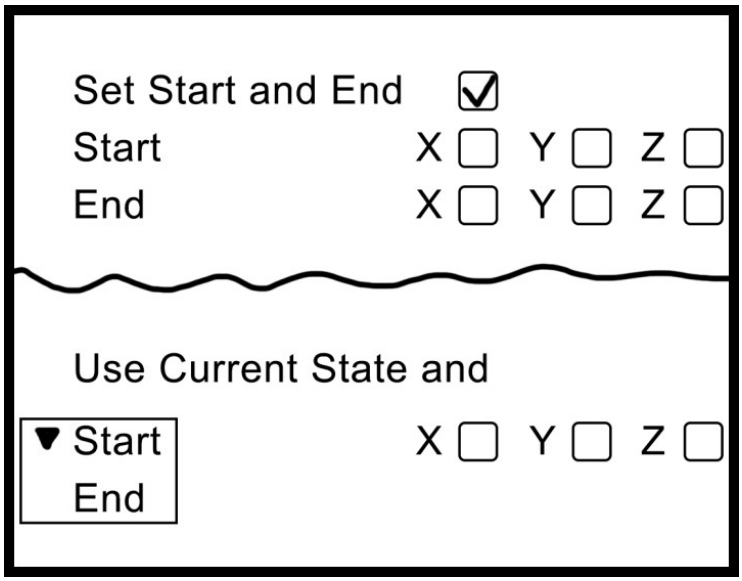


KUVA 3. Valmis työkalu

Piilotin myös kaikki arvot, joita ei kyseisessä modessa voi käyttää. Näitä olivat esim. Use Physics, joka toimii pääasiassa Position-modessa, UseQuaternion joka kertoo Rotation-modessa miten tarkasti objektia tulisi pyörittää. Suurin osa valinnoista ei hyödytä mitään Random Shake-modessa, minkä takia ne piilotetaan kyseisestä modesta kokonaan.

5.3 Testaus

Työkalua testattiin toimeksiantajan yrityksessä tehdyissä projekteissa, ja toimeksiantaja on ollut erittäin tyytyväinen työn laatuun. Yrityksen työntekijät laitettiin testaamaan työkalua ilman minkäänlaista opastusta, ja katsottiin miten hyvin työntekijät oppivat käyttämään työkalua. Yhdelläkään työntekijällä ei ollut suuria ongelmia työkalun käytössä, vaikka osalla heistä oli erittäin vähän kokemusta Unityn käytöstä. Ainoa ongelma työkalun käytössä oli pelkkien aloitus ja lopetus-arvojen asettaminen. Koska työkalu ei mainitse käyttävänsä objektin senhetkisiä arvoja hyödykseen animaatiossa, testikäyttäjille jäi epäselväksi miten pelkkä aloitus- ja lopetus-arvojen asettaminen toimii. Tätä varten ennen julkaisua aion tehdä kuvan 4 mukaisia muutoksia. Kun on aloitus- ja lopetusarvot näkyvillä, en aio muuttaa mitään. Mutta kun vain toinen arvoista on näkyvillä, laitetaan arvojen yläpuolella otsikko, joka kertoo työkalun käyttävän objektin tämän hetkisiä arvoja sekä arvoja jotka käyttäjä asettaa. Tällöin tulee selväksi mitä kaikkia arvoja työkalu käyttää.



KUVA 4. Ajatus miten työkalu kannattaisi parantaa.

Testien aikana paljastui muutama pieni virhe työkalun laskuissa, mutta ne korjattiin helposti pois. Työkalun testeissä kuitenkin paljastui joitain ongelmia, joiden korjaaminen vaatii enemmän työtä.

Kun työkalun luomia animaatioita testaa silloin kun peli ei ole käynnissä, animaatio ohittaa muutaman ensimmäisen framen. Muutama frame saattaa olla paljon riippuen animaation pituudesta, ja hyppy animaatiossa on selvästi huomattavissa. Ei ole selvää miksi tämä hyppy tapahtuu, mutta sen voisi mahdollisesti korjata viivästyttämällä animaatiota näiden muutama framen ajan. Tällöin animaatio aloittaa itsensä hieman myöhässä, mutta käyttäjä pystyisi näkemään koko animaation ongelmitta.

Toinen bugi tapahtuu vain värien kanssa. Kuten aiemmin tekstissä mainitsin, minulla oli jo työkalun tekovaiheessa ongelmia luoda työkalu toimimaan värien kanssa, koska kaikki, 2D-, 3D- ja UI- ympäristöt näyttävät visuaaliset objektinsa eri tavalla. Vaikka korjasin ominaisuuden itse peliin, se ei silti toimi pelin ulkopuolella. Kun käyttäjä yrittää katsoa miltä värin vaihtaminen näyttää UI-objektissa, animaatio hyppää suoraa viimeiseen väriin. Vielä kummallisemman ongelmasta tekee se, ettei animaatio edes muuta objektin väriä vaikka niin editori väittääkin. Ongelma johtuu todennäköisesti siitä, miten Unity piirtää UI-objektit näytölle. Kaikki UI-objektit ovat Canvas-objektin lapsia. Canvas-objekti tunnistaa kun sen lapsiin tehdään muutoksia, ja piirtää silloin kaikki sen lapsena olevat UI-objektit näytölle. Ilmeisesti kun peli ei ole päällä, Canvas-objekti ei tunnista automaattisesti mitään muutoksia mitä sen lapsiin tehdään, minkä takia se ei päivitä UI-objektien ulkonäköä, minkä takia objektien väri ei vaihdu. Tämä on kuitenkin vain spekulaatiota, sillä jos tämä olisi oikeasti ongelma, ei muidenkaan työkalun moodien tulisi toimia UI-objekteissa, mutta ne toimivat. Tämän ongelman korjaaminen siis vaatii hieman enemmän tutkimista ennen kuin työkalu on valmis.

Kolmas bugi ei ole niin vakava, mutta vaatii tutkimista korjaantuakseen. Unityn ikkunat eivät tunnista automaattisesti onko ikkunan sivussa scroll bar (vierityspalkki). Tästä syystä työkalussa näkyvät valinnat saattavat olla eri kohdassa kuin alun perin oli tarkoitus, ja kuvassa 5 näemmekin ettei työkalu näytä niin hyvältä kun haluttaisiin. Samaan ongelmaan liittyen huomattiin, että jos työkalua levittää, eri valinnat näyttävät olevan eri levyisiä riippuen siitä kuinka monta valintaa yhdellä rivillä on ja mitä kyseiset valinnat ovat. Tämä ei varsinaisesti haittaa työkalun käyttöä, mutta tekee työkalusta epämieluisan katsoa.

Kuva 5. Bugi UI:n leveyden kanssa.

Neljäs asia ei ole bugi, vaan pikemminkin pyyntö yhdeltä yrityksen graafikoista. Hän halusi tietää olisiko työkalua mahdollista käyttää kuvien kanssa, että joka frame kuva vaihtuisi. Teoriassa tämä on mahdollista. Tämä vaatisi että käyttäjä luo animaatio-tiedoston, johon käyttäjä asettaa haluamansa kuvat. Työkalu sitten lukisi kuvat tiedostosta samalla tavoin kuin Unityn oma Animator-komponentti. Tämä mahdollistaisi myös monimutkaisempien animaatioiden käyttämisen työkalun kanssa. Vaikka idea on hyvä, se tuntuu turhalta koska Animator-komponentti tekee jo kaiken tämän. En silti sano että idea on huono, mutta vaatii harkintaa, onko ominaisuus tekemisen arvoinen.

Työkalussa on siis vielä paljon parantamisen varaa. Jotkin esille nousseista ongelmista tulee korjata ennen työkalun myymistä markkinoille, mutta yrityksen projekteissa työkalua voidaan jo käyttää huoletta. Kun työkalua jatkokehitetään, on pidettävä mielessä että työkalu pysyy yksinkertaisena ja intuitiivisena. Jos työkalu mutkistuu liikaa, joudutaan se jakamaan pienempiin osiin, mutta tämä selviää vasta kun työkalu jatko kehitetään siihen pisteeseen.

6 POHDINTA

Opinnäytetyöni tavoitteena oli suunnitella ja toteuttaa monikäyttöinen työkalu toimeksiantajan, Mohavi Creativen, tuleviin projekteihin. Työkalulle asetettiin tavoitteeksi olla mahdollisimman intuitiivinen, joka saavutettiin käyttämällä hyödyksi UX-suunnittelun alkeita, jotta työkalu olisi helppo käyttää ilman ylimääräistä opastusta. Työkalun tuli myös olla mahdollisimman monikäyttöinen että se sopisi eri tilanteisiin, ja jotta sitä olisi helppo jatko kehittää edelleen. Toimeksiantaja oli tyytyväinen työkalun laatuun, ja suunnittelee työkalun julkaisua markkinoille ensivuoden puolella.

Uudella työkalulla on mahdollista luoda animaatioita muutamalla klikkauksella, ja niitä on helppo yhdistää ja muokata juuri käyttäjän haluamalla tavalla. Koska työkalun animaatiot ovat koodipohjaisia, toimivat ne millä tahansa FPS-nopeudella. Toisin kuin Unityn omat animaatiota, työkalun animaatiot saadaan toimimaan vaikka Unityn aikakerroin olisi asetettu nollaksi, mikä helpottaa valikoiden tekemistä. Työkalun käyttö ei myöskään vaadi uusien tiedostojen luomista, ja siten pitää projektin tiedosto koon pienempänä.

Työkalussa selvisi joitain bugeja testauksen aikana, jotka vaativat työkalun jatkokehittämistä. Työkalun käyttäjäkokemusta tulisi myös parantaa jatkossakin, jotta jokainen työkalun ostaja pystyy käyttämään työkalua ilman ylimääräistä opastusta. Työkalulle on myös luotava virallinen dokumentaatio joka pitää sisällään työkalun tärkeimmät ominaisuudet ja arvot joita kehittäjät voivat käyttää hyödykseen pelin luomisessa.

Jokainen peli on täynnä erilaisia animaatioita. Jotkut ovat pitkiä ja monimutkaisia, ja vaativat aikaa, miettimistä, ja suunnittelua, ja tähän Unityn tarjoama animaatiomekaniikka on hyödyllinen. Mutta kaikkien animaatioiden luominen ei tarvitse niin montaa työvaihetta mitä Unityn tarjoama animaatiosysteemi vaatii. Tästä syystä työkalu on hyödyllinen jokaisen pelikehittäjän kirjastoon.

LÄHTEET

Cooper, J. 2019. Game Anim: Video Game Animation Explained. 1. painos. Florida: CRC Press

Gregory, J. 2019. Game engine architecture. 3. painos. Florida: CRC Press.

Haas, J. 2014. A History of the Unity Game Engine. An Interactive Qualifying Project. Luettu 15.10.2020. https://digitalcommons.wpi.edu/cgi/viewcontent.cgi?article=4206&context=iqp-all

Itch.io. Top Engines. Luettu 20.10.2020. https://itch.io/game-development/engines

McKay, E. 2013. UI is Communication. 1. painos. Burlington: Morgan Kaufmann

Nabors, R. n.d. Static vs. Dynamic Animations. Luettu 22.10.2020. https://frontendmasters.com/courses/motion-design-css/static-vs-dynamic-animations/

Sito. T. 2013. Moving Innovation: A History of Computer Animation. 1. painos. Cambridge: MIT Press

Sung, K., Pavleas, J., Amez, F. & Pace, J. 2015. Build your own 2D Game Engine and Create Great Web Games Using HTML5, JavaScript, and WebGL. 1. painos. New York, Apress.

Toftedahl, M. 09.20.2019. Which are the most commonly used Game Engines? Luettu 20.10.2020. https://www.gamasutra.com/blogs/MarcusToftedahl/20190930/350830/Which_are_the_most_commonly_used_Game_Engines.php

Unity Docs. n.d.. Creating a new Animation Clip. Luettu 24.10.2020. https://docs.unity3d.com/Manual/animeditor-CreatingANewAnimationClip.html

Unity Technologies. Luettu 22.10.2020. https://unity.com/

LIITTEET

Liite 1. Kuva työkalusta ilman editor-skriptiä muokkaamassa ulkonäköä.