# Pelinkehitys itsenäisenä kehittäjänä

**Mikael Huovinen**

2022 Laurea

**Laurea-ammattikorkeakoulu**

# Pelinkehitys itsenäisenä kehittäjänä

Mikael Huovinen

Tietojenkäsittely

Opinnäytetyö

Marraskuu, 2022

**Laurea-ammattikorkeakoulu**

Tietojenkäsittely

Tradenomi (AMK)

Mikael Huovinen

**Tiivistelmä**

**Pelinkehitys itsenäisenä kehittäjänä**

Vuosi 2022 Sivumäärä 29

Tämän opinnäytetyön päämääränä on ollut kehittää ja julkaista verkossa ensimmäinen toimiva versio 2D-videopelistä ja dokumentoida matka suunnittelusta pelattavaan versioon. Peli kehitettiin Unity-pelimoottorilla ja ohjelmoitiin C#-ohjelmointikielellä Visual Studio Code -ohjelmistossa. Päämääränä oli saavuttaa lopputulos hyödyntämättä valmiiksi tehtyjä julkisesti saatavilla olevia resursseja. Opinnäytetyö on tarkoitettu pelinkehityksestä ja ohjelmoinnista kiinnostuneille. Tarkoituksena on ollut luoda opinnäytetyö, jota voi käyttää oppaana oman peliprojektin kehityksessä.

Pelin sprite-grafiikat tehtiin GraphicsGale nimisellä sovelluksella. GraphicsGalella tehtiin pelimaailma, pelaaja- ja vihollishahmot. Musiikit pelille tehtiin MPC Beats -sovelluksella.

Opinnäytetyössä esitetään aluksi peliprojektissa käytettyjä menetelmiä ja sovelluksia sekä näiden teoriaa. Tässä osuudessa muun muassa käydään läpi, miten sovellukset asennettiin ja miksi juuri nämä sovellukset valittiin. Opinnäytetytön lopussa on pelinkehitysosio, jossa kuvaillaan jokaisen kehitysvaiheen iteraatiota. Pelin rakentaminen tehtiin iteratiivisesti.

Aiempaa osaamista opinnäytetyön laatijalla oli olemassa Unityn käytössä, ohjelmoinnissa ja pikselitaiteessa. Opinnäytetyö on kuitenkin ensimmäinen projekti tällä skaalalla. Musiikin tekeminen tuli uutena kokemuksena projektia tehdessä. Projekti oli kokonaisuudessaan haastava ja vaati paljon aikaa, mutta haluttu lopputulos saavutettiin.

Asiasanat: pelinkehitys, ohjelmointi, unity

**Laurea University of Applied Sciences**

Degree Programme in Business Information Technology

Bachelor's degree

Mikael Huovinen

**Abstract**

**Video game development as an independent developer**

Year 2022 Pages 29

The goal of this Bachelor's Thesis was to develop and release the first working version of a 2D-video game and document the journey from the design to the playable version. The game was developed using Unity game engine and was programmed using C# programming language in Visual Studio Code. The goal was to reach a result without using already made publicly available resources. The thesis is intended to those who are interested in game development and programming. The intention was to create a thesis which could be used as a guide in your developing your own game project.

The games Spirte graphics were made using a program called GraphicsGale. GraphicsGale was used to create the game world, player, and enemy characters. The music for the game were made using MPC Beats software.

There was previous experience in using Unity, programming, and pixel art. The thesis was still the first project of this scale. Creating music came as a new experience while creating the project. The project overall was challenging and required a lot of time, but the desired outcome was still reached.

Keywords: video game development, programming, unity

Sisällys

1 Johdanto ..... 6
2 Työn Lähtökohdat ..... 6
3 Tavern Crawl -peliprojekti ..... 7
3.1 Peliprojektin aloitus ..... 7
3.1.1 Unity ..... 7
3.1.2 GitHub ..... 8
3.1.3 Suunnittelu ..... 10
3.2 Käsitteitä ..... 10
3.3 Grafiikat ..... 11
3.4 Musiikki ..... 14
4 Pelin rakentaminen ..... 14
4.1 Ensimmäinen iteraatio: Pelimaailma ja liikkuminen ..... 16
4.2 Toinen Iteraatio: Käyttöliittymä ..... 18
4.3 Kolmas Iteraatio: Animaatiot ..... 20
4.4 Neljäs Iteraatio: Musiikki ..... 21
4.5 Julkaisu GitHub Pages -palvelussa ..... 23
5 Jatkokehitysideat ..... 24
6 Yhteenveto ..... 25
Lähteet ..... 28
Kuviot ..... 29

# 1 Johdanto

Lähestyn tätä projektia tee se itse -asenteella, joten en tule käyttämään valmiita ilmaisia grafiikkoja tai vastaavia. Dokumentoin suunnittelun, versionhallinnan (GitHub), työympäristön (Unity) ja itse pelin tekemisen. Haluan, että työntekoni antaisi kuvaa ihmisille siitä, miten pelin tekeminen sujuisi yhden henkilön ”indie” -pelifirmassa. Opinnäytetyö on itsenäinen projekti, eikä sillä ei ole toimeksiantajaa.

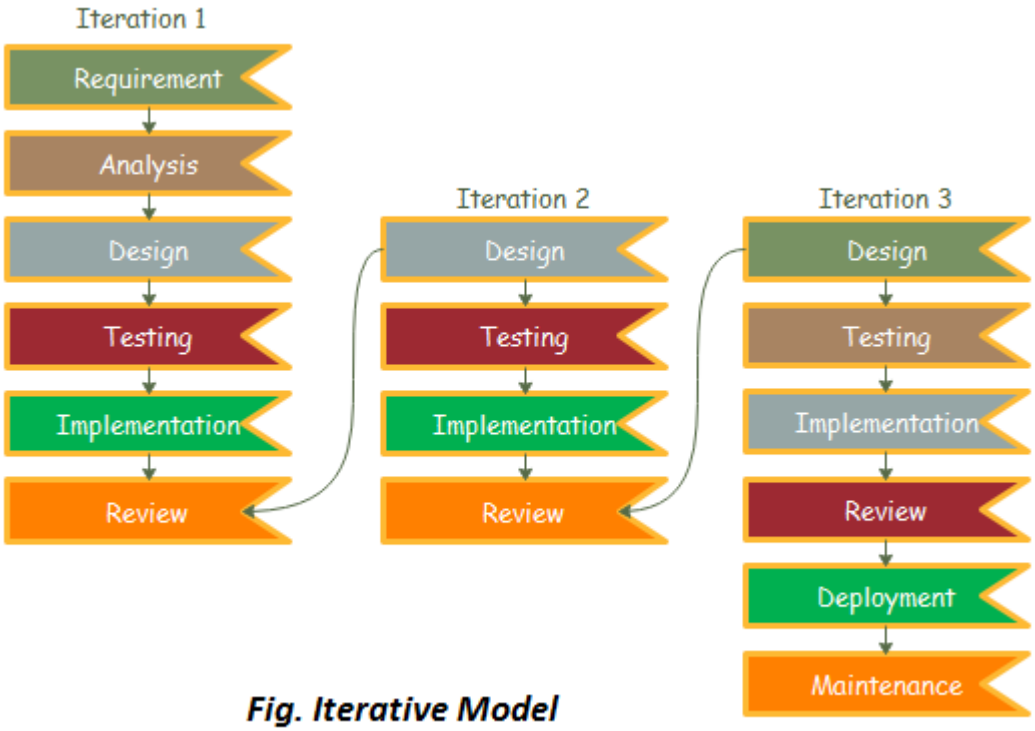
Tein pelin Unityllä käyttäen versiota 2021.3.1f1. Tarkoituksenani oli tehdä pelattava versio, jossa on itse tehdyt pikseligrafiikat, musiikit ja käyttöliittymä. Ensimmäisen julkaisun tavoittena oli luoda aloitus- ja loppuruutu, toimiva pelimaailma, jossa voi liikkua ruudukossa musiikki pelimaailmaa sekä aloitusruutua varten ja lopuksi mahdollista voittaa pelin kukistamalla vihollishahmo. Käyn läpi kaikki eri design päätökset läpikohtaisesti. Tarkoituksena oli, että valmista opinnäytetyötä voi käyttää apuna oman pelin, grafiikan sekä musiikin tekemisessä.

Kehitin projektin iteratiivisesti. Tässä opinnäytetyössä käyn jokaisen iteraation lävitse. Opinnäytetyössä en kuitenkaan perehdy pelinkehitykseen aiheena.

# 2 Työn lähtökohdat

Idea peliprojektia varten syntyi halusta tehdä itselle sopiva opinnäytetyö, jonka voi liittää CV:n työnhakua varten. Peliprojekti mahdollistaa monien eri taitojen ja niiden hallitsemisen esittelyn. Keskityn opinnäytetyössä eniten pelin ohjelmointiin ja grafiikoiden luomiseen, koska niistä minulla on aiempaa osaamista. Pelinkehityksen toteutin iteratiivisesti, joskin melkein kaikki käytetyistä sprite-grafiikoista oli aiemmin tehtyjä. Usein iteraation lopussa on arvosteluvaihe, mutta tällä projektilla ei ole toimeksiantajaa, joten se osan jätin opinnäytetyöstä pois.

Opinnäytetyössä käyn eri iteraatioita yksityiskohtaisesti läpi suunnittelusta implementointiin ja kuvailen ongelmia, jotka hidastivat edistystä pelinkehityksessä. Erilaiset ongelmat pelinkehityksessä ovat usein aikaa vieviä, joten kaikki uudet ongelmat venyttävät tarvittavaa työmäärää.




Kuvio 1: Iteraatiivinen kehitys Software Development Life Cycle (SDLC) (Javatpoint, 2022)

# 3 Tavern Crawl -peliprojekti

## 3.1 Peliprojektin aloitus

Pelin tekemiseen liittyy monia asioita. Tämä tulee nopeasti selväksi varsinkin, mikäli pelin tekee yksin. Ensimmäisessä vaiheena valitsin sovellukset, joilla rakensin kaikki eri osat. Valitsemani pelimoottori on Unity (Unity 2022), koska minulla on hieman aiempaa kokemusta tämän kanssa. Unityssä käytän C# -ohjelmointikieltä. Yhdistän peliprojektin versionhallintajärjestelmän kanssa, joka on GitHub (GitHub 2022) tätä projektia varten. Pelin taidetyyliksi valitsin pikseligrafiikat ja tämä teen GraphicsGale -sovelluksella.

### 3.1.1 Unity

Unity on erinomainen sovellus opiskelijoille ja harjoittelijoille, sillä se on ilmainen ei-kaupallisille projekteille. Peliteollisuudessa Unity -sovellus on myös yleisesti käytetty. Tämän sovelluksen opetteleminen on kannattavaa heille, jotka unelmoivat urasta pelialalla (Takahashi 2022). Unityn asennus tehdään sovelluksen omien sivujen kautta. Asennuksen yhteydessä valitaan kunkin haluama versio. Tätä projektia varten käytän versiota 2021.3.1f1. Valitsin tämän version, koska se löytyy jokaisesta laitteestani, joka pyörittää Unityä. Tällä varmistan, että

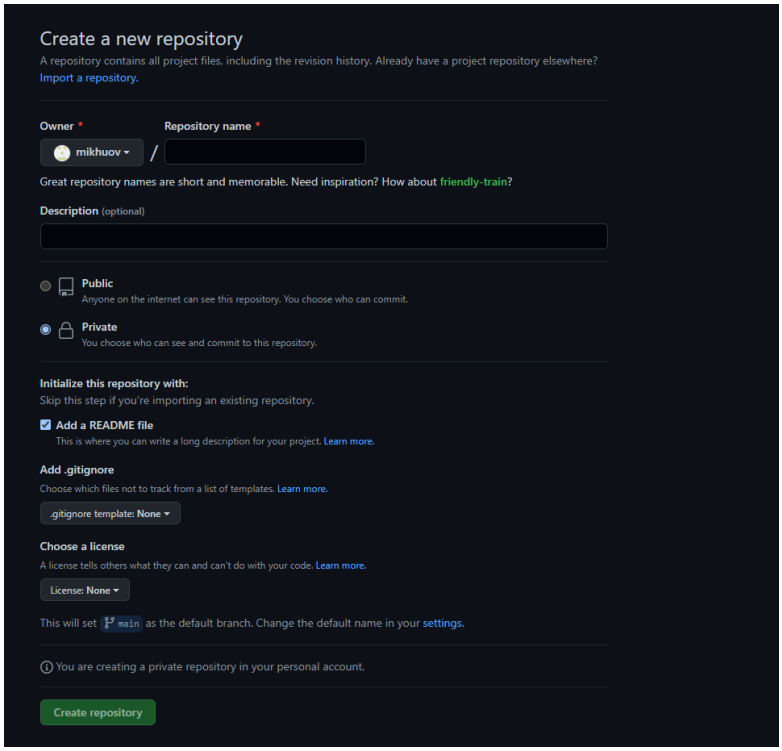
tulevaisuudessa en törmää mahdollisiin ristiriitoihin, mikäli joudun vaihtamaan laitetta projektin aikana. Ohjelmistoversio on tärkeä, koska projekti ei välttämättä ole yhteensopiva uuden päivityksen kanssa. Siksi on kannattavaa pitää Unityn version samana projektin aikana. Ohjelmiston käyttöä varten tarvitaan myös Unity käyttäjätili.

Latauksen jälkeen luon 2d-projektin Unityssä. Projektit voivat olla joko 2d tai 3dsei. Kaksiulotteiset pelit luodaan usein 3d-projekteina. En itse koe sitä tarpeellisena tätä projektia varten, joten luon sen 2d-projektina sovelluksessa. 2d- ja 3d-pelien teko eroavat toisistansa huomattavasti, joten on oleellista projektin tekijälle valita projektille sopivin vaihtoehto. Itselleni 2d on hyvä vaihtoehto, koska haluan käyttää pelissäni pikseligrafiikoita. Vaikka pelillä olisi 3d grafiikat, tämä ei välttämättä tarkoita, että se olisi 3d peli. Ulottuvuudet määräävät, millä tavalla pelaaja voi vaikuttaa pelimaailmaan. Kaksiulotteisessa pelissä pelaaja on lukittu x- ja y-akseleihin, joka luonnostaan määrää minkälaisen pelin pystyt luomaan. (Docs Unity 2022.)

### 3.1.2 Github

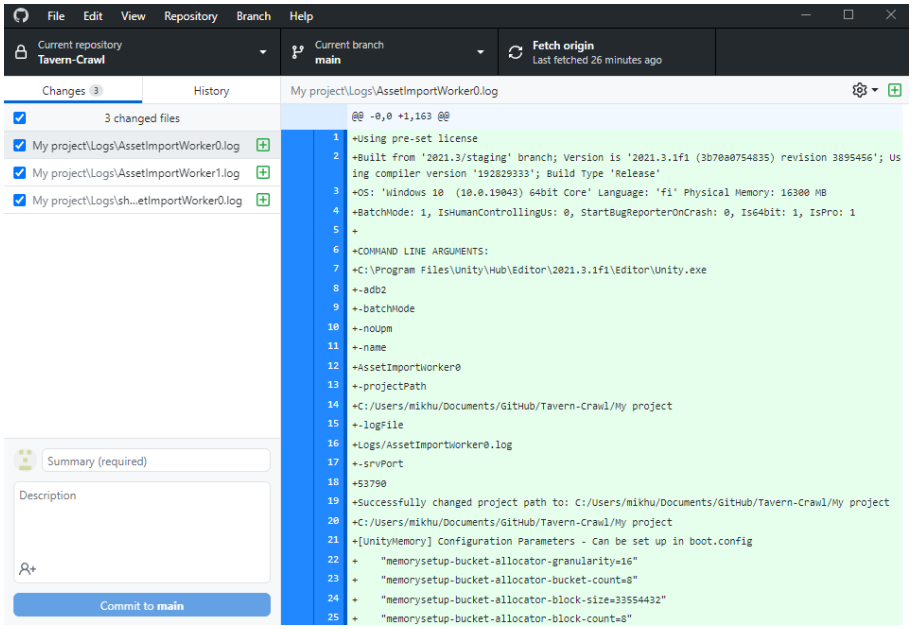
Github on verkkosivusto, joka on tarkoitettu projektien versionhallintaa varten. Git itsessään on open-source versionhallintajärjestelmä. Sen tarkoituksena on luoda ympäristö, jossa voi rakentaa samaa projektia eri haaroissa (branch). Valmiit haarat yhdistetään lopuksi pääprojektiin. Git:iä käytetään komentorivillä. Github antaa tälle käyttöliittymän, jossa voit tarkastella eri tiedostojen muutoksia kätevästi. Ilman Githubia muutosten katsominen olisi hankalampaa ja vaatisi kokenutta käyttäjää. Github auttaa käyttäjiä arkistoimaan muutokset liittyen projektiin. (Kinsta 2022)

Aloitin tämän prosessin luomalla uuden repositorion Githubissa. Github -käyttäjän tekeminen on helppoa, ilmaista ja kannattavaa jokaiselle ihmiselle, joka opiskelee tietojenkäsittelyä. Laitoin repositorion yksityiseksi siksi ajaksi, kun tein sen tätä raporttia varten. Tässä kohtaa pitää ainoastaan antaa projektille nimi sekä päättää, haluaako README-tiedoston ja lisätä Gitignore-asetukset. README on yleensä vain esittely projektista pähkinänkuoressa. Gitignore on tärkeä tässä projektissa, koska sen avulla voin pakottaa mitä, tiedostoja GitHub ei huomioi tiedostojen luomisen ja muokkauksen yhteydessä. Tämä tiedosto yleensä laitetaan projektiin juureen (Tutorialspoint 2022). Varsinkin Unityä käyttäessä projektin aikana tulee kaikenlaisia tiedostoja, jotka ovat erilaisia asetuksia ja kompilaation yhteydessä luotuja tiedostoja. Nämä lisäävät vain projektiin kokoa, joten se on hyvä käytäntö lisätä gitignore projekteihin. Oheisessa kuviossa 2 näkyy repositorion luonnin näkymä.




Kuvio 2: Repositorion luonti (GitHub, 2022)

Käytin tässä projektissa GitHubin omaa sovellusta nimeltä GitHub Desktop. Tämä on erinomainen käyttöliittymä, johon voin laittaa peliprojektini kansion. Aina, kun tiedostoissa tapahtuu muutos, voin puskea muutoksen suoraan projektin repositorioon. Käyttöliittymän yläpalikassa Repository -kohdasta löytyy push- ja pull- toiminnot. Pull-komennolla ladataan nykyinen repositorion tila laitteelle, ja push-komento päivittää repositorion laitteella tehdyillä muutoksilla. Oheisessa kuviossa 3 näkyy GitHub Desktopin käyttöliittymä.



Kuvio 3: GitHub Desktop käyttöliittymä

### 3.1.3 Suunnittelu

Olen halunnut tehdä jo pitkään itseäni varten peliprojektin. Nyt suunnitelma on toteutunut, ja projektin nimi on Tawern Crawl. En halunnut tehdä 2d-scroller peliä kuten monilla on tapana tehdä ensimmäisenä projektinaan. Päätin aloittaa projektin vuoropohjaisesta 2d-pelistä, johon teen pikseligrafiikat. Pelin ulkonäkö tulisi muistuttamaan shakkilaudan kaltaista ruudukkoa, jossa pelaaja taistelee tietokoneen ohjaamaa vastustajaa vastaan. Sain inspiraation tätä projektia varten erilaisista peleistä, joita olen pelannut ja jotka on luotu samankaltaisilla tekniikoilla.

Pelin taustatarinana on, että pelaajan ohjaama hahmo Martini Steve menee tavernasta tavernaan taistellen portsareita vastaan. Pelin ulkoasu on tarkoitus olla hyvin leikkisä ja sarjakuvamainen. Tämä tulee esille hahmojen ja maailman olomuodoista ja värimaailmasta. Pelaajan hahmo hymyilee ja näyttää positiivisilla vaaleilla sävyillä värjätyltä piirroshahmolta, joka on asennoitunut taistelemaan. Kontrastina vastustaja on tehty hieman tummemmilla sävyillä, joka ei muistuta ihmistä. Tämä kontrasti hahmoissa luo sanattomasti tunteen, että kyseessä on selvästi fantasiamaailma. Ammattilaistaiteilija videopelejä varten Jon Neimeister sanoo blogikirjoituksessaan, että on kaksi suurta komponenttia hyvää hahmonsuunnittelua varten: idea itse ja visuaalinen kommunikointi kyseisestä ideasta (Neimester 2022). Suunnittelin hahmot ottaen mallia Neimeisterin ideoista.

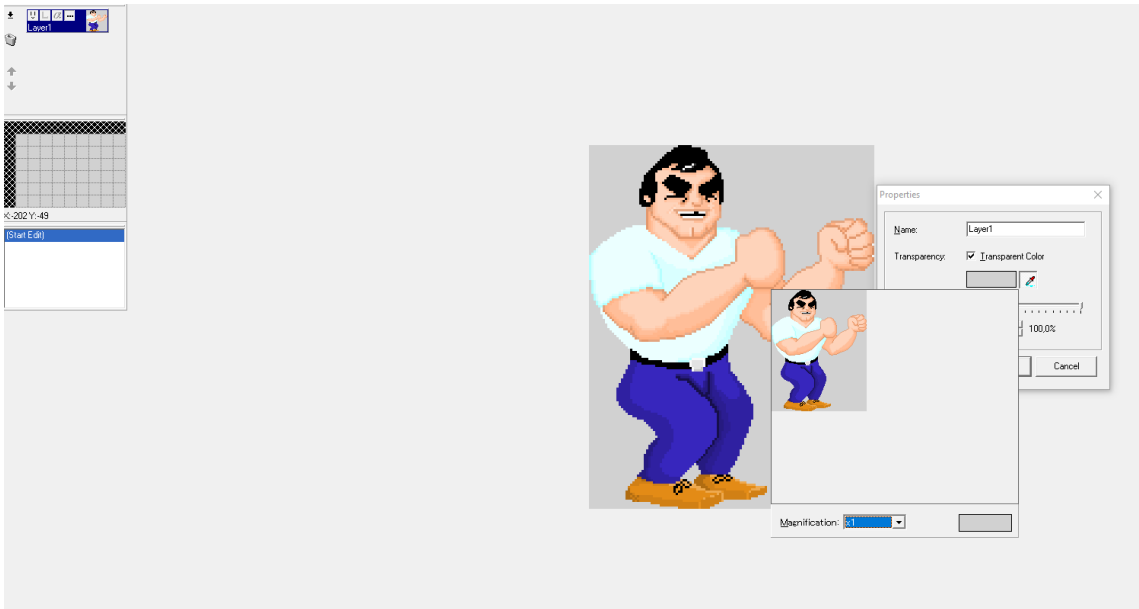
## 3.2 Käsitteitä

| | |
|---|---|
| Grid | Unityn komponentti, joka auttaa sijoitella objekteja ruudukossa. |
| Commit | Toiminta GitHubissa, joka lähettää lähdekoodin muutokset projektin repositorioon. |
| Repositorio | Sisältää projektin tiedostot ja näiden historian. Repositorioita käytetään säilyttämään tiedostoja ohjelmointiprojektien versiohallintaa varten. |
| Collider | Unity -komponentti, joka auttaa peliobjekteja tunnistamaan toisensa. |
| Sprite-grafiikka | Koostuu pikseleistä ja on kaksiulotteinen bittikartta. Käytetään yleensä 2D-peleissä. |
| Frame | Animaatiossa käytetty termi, joka viittaa yksittäiseen kuvaan kuvasarjassa, joista animaatio koostuu. |

## 3.3 Grafiikat

Tein projektini piskeligrafiikoilla ja tähän käytin GraphicsGale -nimistä ohjelmaa. Käytin aiemmin tehtyjä piirustuksia pelissäni nopeuttaakseni kehitystä. Nämä olivat alunperinkin suunniteltu peliprojektia varten. Valitsin tämän ohjelman, koska se on ilmainen ja minulla on aikaisempaa kokemusta sen kanssa. GraphicsGale on erinomainen työkalu animointiin ja peliprojekteihin. Siihen löytyy myös paljon tutoriaaleja, mikäli tulee tilanne, ettei tiedä mitä tehdä. Lataus tapahtuu nopeasti ja ongelmitta GraphicsGalen omilta sivuilta (GraphicsGale 2022).

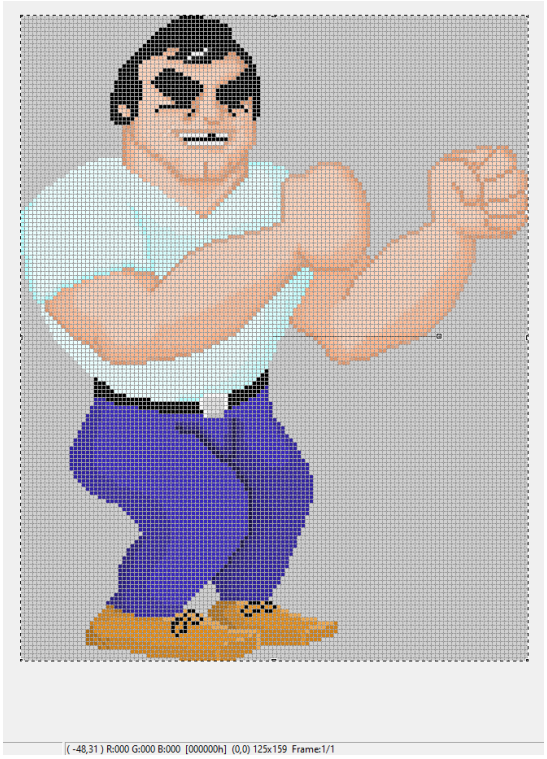
GraphicsGalessa tarkoituksenani on maalata piirustusalusta vaalean harmaaksi. Tämä johtuu siitä, että GraphicsGalella ei ole varsinaisesti ”pyyhekumia” poistaakseen töherryksiä, vaan käyttäjän pitää valita väri, jonka ohjelma määrää läpinäkyväksi. Tämä on mielestäni tärkein asia ymmärtää GraphicsGalella työskentelystä, koska tämä on olennainen osa digitaalista piirtämistä. Tämän vaihtoehdon löydät menemällä layeriin, painamalla painiketta, jossa on kolme pistettä. Tämä avaa Properties-valikon, jossa heti näkyy Transparency-valikko. Paina Transparent Color kohtaa ja sitten valitse väri piirustuksestasi, jonka haluat olevan läpinäkyvä color picker -työkalulla, joka on visualisoitu pipetillä. Paina ok. Nyt sinun pitäisi pystyä käyttämään valitsemaasi väriä pyyhekumina poistamaan tapahtuneita virheitä. Oheisessa kuviossa 4 visualisoidaan transparentin värin valinta.




Kuvio 4: Transparentin värin valinnan prosessi

Pidän aina muutamia konsepteja mielessä, kun piirrän. Pikselitaiteessa anti-aliasing on tärkeää antaa kuville pehmeät reunat ja paremman muodon. Värit sekoittuvat toisiinsa paremmin (Medeiros 2018). Tätä ei tarvitse välttämättä pitää piirtämisen ”lakina” tai pakollisena

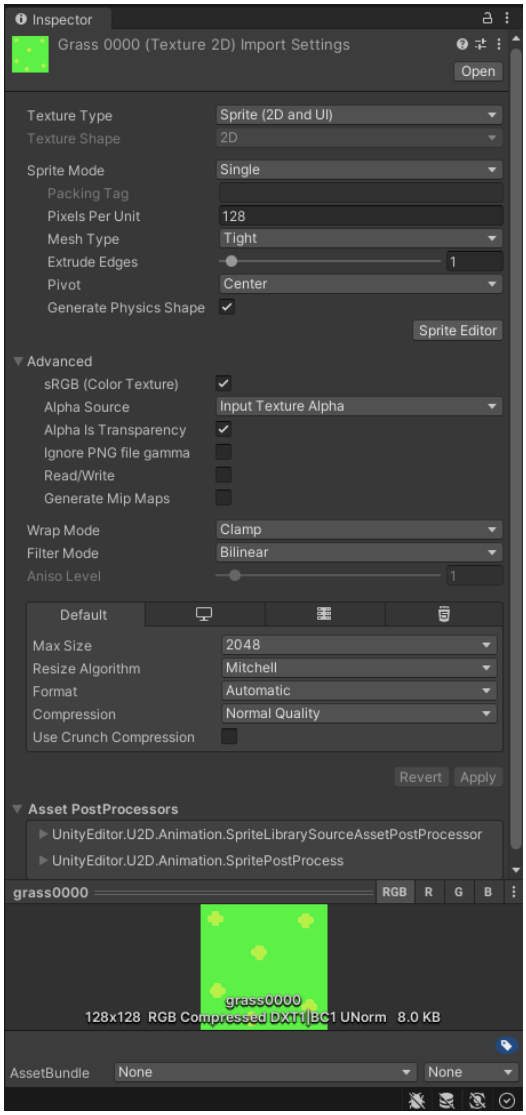
asiana, koska taide loppujen lopuksi perustuu henkilökohtaiseen näkemykseen. Tämä on silti suositeltava käytäntö, koska anti-aliasing auttaa tekijää kumoamaan pikselien mukana tulleet rajoitukset (Lux 2020). Toisin kuin vektoripiirtämisessä, jossa saat täysin luonnolliset ja pehmeät muodot, pikseleitä käyttäessä on yritettävä replikoida luonnolliset ja pyöreät muodot eri tekniikoilla. Pikseleitä käyttäessä pitää ymmärtää se konsepti, että jokainen pikseli on olemassa ruudukon päällä. Luotaessa uusi piirustus sille määritellään korkeus ja leveys. Esimerkiksi oma pelihahmoni on 125x159 pikseliä. Hahmoja luodessa kannattaa olla tietoinen pikselien määrästä, että kaikki piirustukset ovat järkevässä suhteessa toisiinsa. Piirsin itse aika vapaasti välittämättä hahmon lopullisesta koosta, koska en ole erityisen taitava pikselitaiteessa ja halusin hahmostani juuri sen näköisen kuin suunnittelin. Kokeneempi taiteilija pystyy tiivistämään piirustukset pienemmiksi ilman, että menettää yksityiskohtia. Oheisessa kuviossa 5 näkyy projektini pelihahmo.




Kuvio 5: Hahmon ulottuvuudet

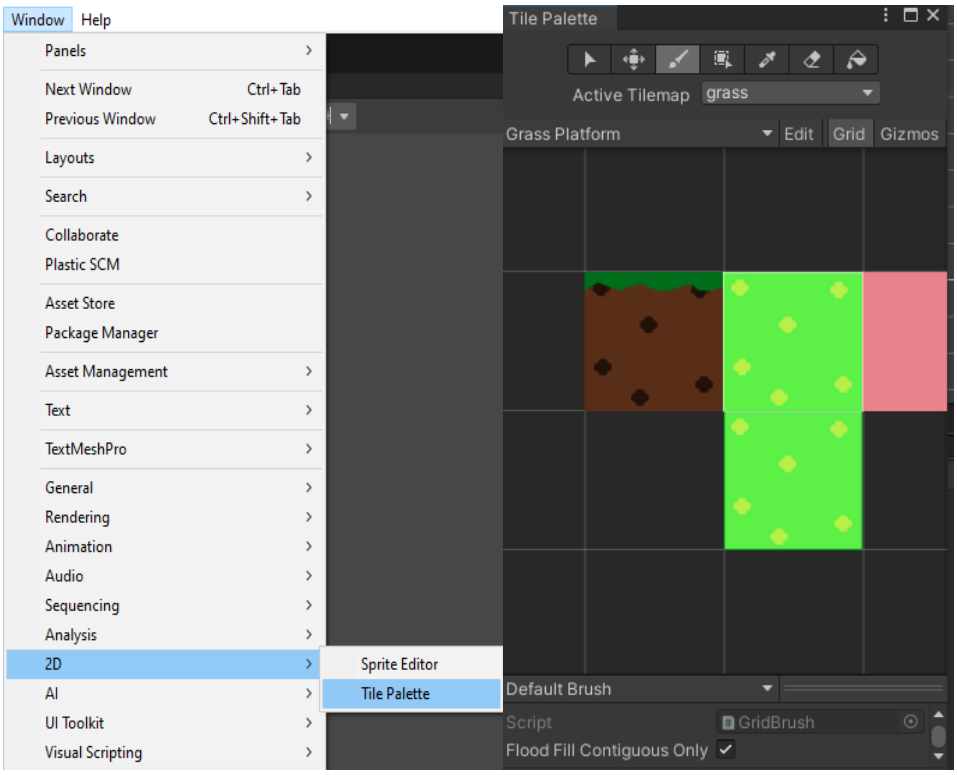
Tilemap viittaa pelimaailmassa käytettyihin ”palikoihin”, jotka muodostavat pelissä näkyvän maiseman. Näitä palikoita suunnitellessa pitää muistaa oikeat ulottuvuudet, että ne käyvät järkeen pelimaailmassa. Itse koitin luoda tarpeeksi ison palikan, joka sopii näppärästi pelihahmon jalkojen alle. Päädyin itse 128x128 kokoisiin Tilemap-palasiin. Tilemappia suunnitellessa kannattaa myös ottaa huomioon, miten se vaikuttaa hahmoihin ja itse pelaamiseen. Pelini on vuoropohjainen peli, jossa hahmot liikkuvat ruudukossa. Yksi Tilemap-palanen edustaa liikkumisetäisyyksiä, joten pelin kannalta on tärkeää, että pelaaja tunnistaa jokaisen ruudukon. Jos

näitä ruutuja ei erotella toisistansa, on pelaajalla huomattavasti vaikeampaa hahmottaa mihin hän voi liikkua. Tämän ongelman voi ratkaista monella tapaa, mutta minä päätin korjata sen heti piirustusvaiheessa. Lisäsin jokaiselle liikkumiskelpoiselle palikalle ääriviivan, jonka myötä lopullisessa tuloksessa pelaaja hahmottaa ruudukon. Pixels Per Unit on myös muutettava oikean kokoiseksi, jos se ei ole automaattisesti oikein. Palaseni ovat 128x128 kokoisia, joten muutan kaikkien kohdalla Pixels Per Unit edustamaan tätä. Oheisessa kuviossa 6 näkyy kyseinen Tilemap Unityssä.




Kuvio 6: Tilemap Unityssä

Unity tarjoaa Tile Palette -työkalun, johon voi syöttää kaikki tekemäsi eri Tilemap-palikat ja tämän jälkeen voi maalata haluamansa pelimaailman. Tile Palette löytyy yläpalkista valitsemalla Window, 2d ja viimein Tile Palette. Tile Paletin käyttö on yksinkertaista. Valitaan se palikka, jonka haluaa maalata pensselityökalulla ruudukossasi. Oheisessa kuviossa 7 näkyy Tile Palette -työkalun sijainti ja käyttöliittymä.

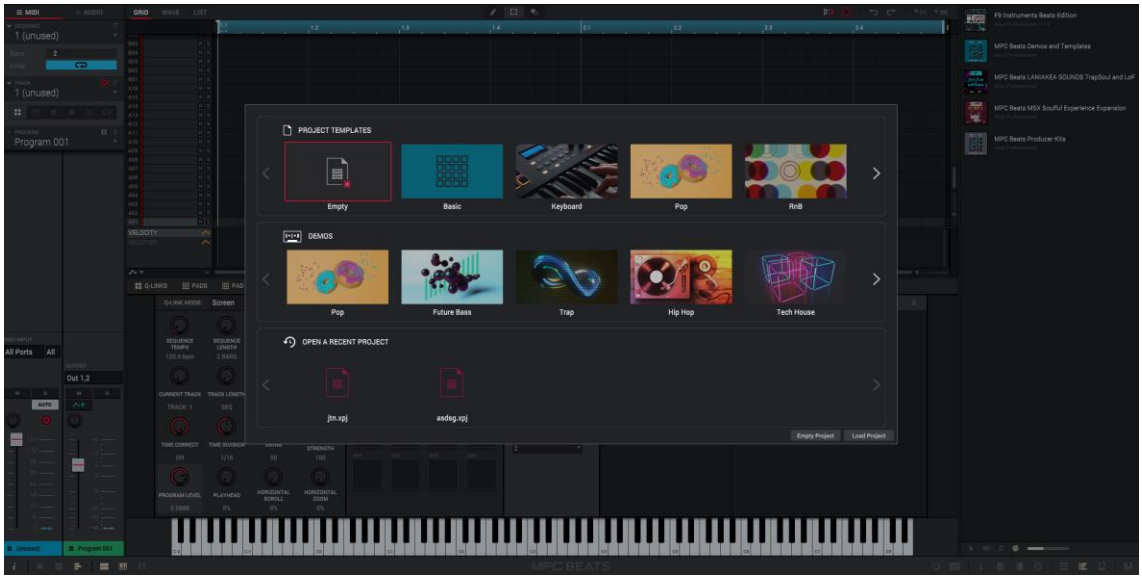



Kuvio 7: Tile Palette-työkalu Unityssä

## 3.4 Musiikki

Musiikki oli pieni osa tätä projektia. Tein kaksi pientä kappaletta peliä varten: yksi aloitusruutua varten ja toinen itse peliä varten. Käytin MPC Beats -sovellusta tekemään lyhyet kappaleet. Kokemukseni musiikin teon kanssa oli melkein olematon ennen tätä projektia. Olen harjoitellut hieman pianon soittoa MIDI-koskettimella, mutta en kokenut tämän auttaneen opinnäytetyössä. MPC Beats on ilmaiseksi saatavilla oleva sovellus Mac sekä Pc -alustoille ja sen asentaminen on yksinkertaista, mutta se vaatii käyttäjätilin luomisen AKAI:n sivuilla (Akaipro 2022).

MPC Beats on digital audio workstation (DAW), joka tarkoittaa sovellusta musiikin tekemistä ja editointia varten. Musiikin lisäksi DAW-sovelluksia voi käyttää erilaisiin tarpeisiin, johon liittyy audio. Nämä sovellukset toimivat lataamalla digitaalisia instrumentteja. Nuotteja voi asetella itse käyttöliittymään, missä voi hallita muun muassa volyymiä ja säveltä.




Kuvio 8: MPC Beats -sovelluksen käyttöliittymä

## 4 Pelin rakentaminen

Lähestyin pelin tekemistä iteratiivisella kehityksellä. Iteratiivinen ohjelmistokehitys on prosessimalli, jossa ohjelmisto kehitetään useammissa peräkkäisissä iteraatioissa. Tämä mahdollistaa luoda ohjelmiston askel askeleelta, jonka avulla kehityksen aikana huomataan erilaiset puitteet tai ongelmat jo alkuvaiheissa. Näissä iteraatioissa on mahdollista saada palautetta mahdollisilta asiakkailta tai testaajilta, mutta tässä projektissa en kerännyt käyttäjäkokemuksia. (Eastern Peak 2022).

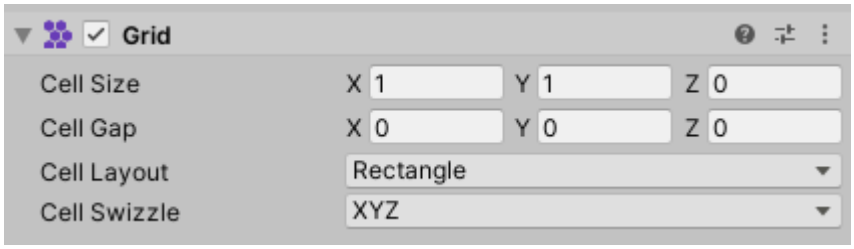
Koodasin pelin Unityssä C#-ohjelmointikieltä käyttäen Visual Studio Codea. Lähestyin tätä taakkaa ensimmäiseksi suunnittelemalla kaikki liikkuvat osaa ensimmäistä pelattavaa versiota varten, joka täyttää haluamani ehdot. Tarvitsin tähän versioon pelaajan, vastustajan, pelimaailman, pienet UI-elementit hahmoja varten ja tavan voittaa tai hävitä. Tarkoituksena ei ollut nyt tehdä täydellistä peliä loppuun, vaan toteuttaa peli iteratiivisesti lisäten uusia elementtejä uusissa iteraatioissa. Martini Steve ja Tim ovat pelihahmot, Level1Grid on tämän iteraation pelimaailma, Healthbar-Ui edustaa hahmojen kestävyyttä ja CombatSystem hallinnoi pelaamista. Kuviossa 9 näkyy ensimmäiset objektit, joita aloin työstämään. Oheisessa kuviossa näkyy lopullinen rakenne peliobjekteista.

Kuvio 9: Peliobjektien lopullinen rakenne

## 4.1 Ensimmäinen iteraatio: Pelimaailma ja liikkuminen

Päämääränä oli luoda vuoropohjainen peli, jossa pelaaja liikuttaa hahmoa ruudukossa tietokone vastustajaa vastaan. Ensimmäisen iteraation tehtävänä oli luoda pelimaailma, lisätä hahmot ja niiden toiminnallisuudet. Ennen kuin aloin koodaamaan luin läpi Unityn omaa dokumentaatiota. Se auttoi minua tutustumaan siihen ja antoi idean, miten edetä. Unity on niin laajasti käytetty pelimoottori, että siihen liittyvää hyödyllistä materiaalia löytyy paljon, mutta itse koen virallisen dokumentaation tutustumisen olevan rakentavampaa. Ensimmäisenä askeleeni oli luoda pelimaailma ja toimiva ruudukko, missä pelihahmot oleskelevat.

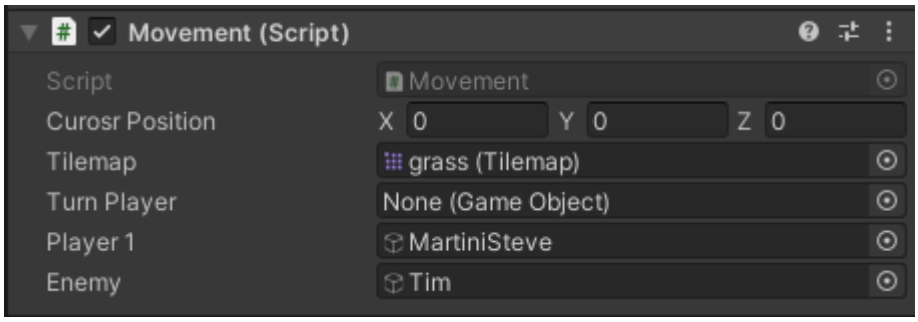
Grid on yksi Unityn monista komponenteista, joka auttaa asettamaan peliobjekteja paikallensa. Grid luo ruudukon, josta voi ottaa koordinaatteja solujen perusteella. Tällä komponentilla voi valita gridin solujen muodon ja koon. Käytin tätä luomalla peliobjektin ja annoin sille tämän komponentin, niin se loi ruudukon peliobjektin koordinaatteihin. On mahdollista luoda samankaltainen menetelmä koodaamalla itse, mutta tämä on kätevä ja toimiva vaihtoehto hyödyntäen Unityn valmiita komponentteja. (Docs Unity 2022.)




Kuvio 10: Grid-komponentti Unityssä

Luon tiedoston hahmojen liikkumista varten, jossa erotellaan pelaaja ja vastustajat. Kaikki hahmot ovat peliobjekteja ja pelimaailma tulee olemaan gridin tilemap. Tässä versiossa tulee olemaan pelaajahahmo ja yksi vastustaja. Player 1 -objekti edustaa pelaajan ohjaamaa hahmoa ja Enemy-objekti vastustajaa. Liitän Movement-skriptin pelikenttään kiinni ja liitän sen

jälkeen kaikkiin valikkoihin haluamani objektit. Tämän jälkeen pelimaailma hallitsee liikkumista määräävää logiikka. Oheisessa kuviossa 11 näkyy Movement-skripti pelikenttään liittämisen jälkeen.



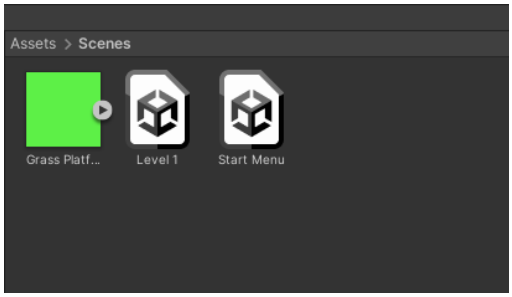

Kuvio 11: Movement Scripti liitettynä pelikenttään

Movement skriptissä aluksi Start -funktiossa määritetään hahmot ja vuoropelaajaksi valitaan pelaaja. Update -funktio tapahtuu aina kun ohjelma tunnistaa pelaajan tehneen jotain. Liikkuminen suoritetaan katsomalla mitä ruutua pelaaja painoi kursorilla. Tämä katsotaan kutsumalla ScreenToWorldPoint funktiota, johon syötetään hiiren painalluksen positio. Tämä tieto tallennetaan cursorPosition nimiseen Vector3-muuttujaan ja sen z-koordinaatit nollataan, koska tässä pelissä sitä ei tarvita. Vector3Int tilemapPos muuttujaan laitetaan WorldToCell funktion palauttama data. Tämä data ottaa cursorPositionin hankkimat koordinaatit ja muuttavat ne solun positioksi. Tämä varmistaa, että pelihahmot ovat olemassa oikeissa koordinaateissa pelimaailman yläpuolella ja pystyvät liikkua gridissä tarkoituksenmukaisesti. GetCellCenterWorld-funktio palauttaa gridin solun keskustan. Hahmojen liikkuessa tämä funktio varmistaa sen, että pelihahmot löytävät paikkansa solun keskeltä. Muuten hahmo liikkuu juuri siihen pikseliin mihin pelaaja painaa kursorillaan, eikä siihen Gridin osioon mihin pelaaja haluaa liikkua ruudukossa. Joka kerta kun jokin hahmo tekee siirron, niin pelaaja vaihtuu. Hahmojen liikkuminen on rajoitettu maxMoveDistance muuttujalla. (Docs Unity 2022.)

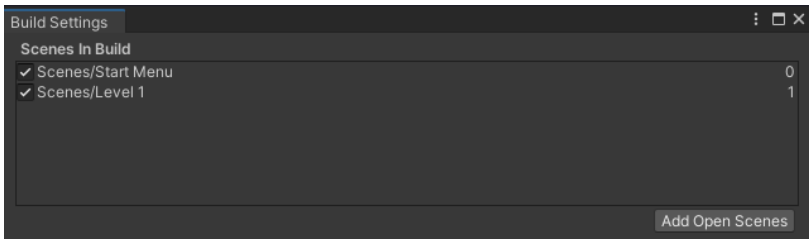
Tässä kohtaa ensimmäinen iteraatio oli valmis, joten tein ensimmäisen commitin projektin githubiin ja aloitin seuraavan iteraation suunnittelun ja toteutuksen. Seuraavan iteraation on tarkoitus sisältää UI-elementtejä, jotka mahdollistavat pelin aloittamisen ja pelin päättymisen.

## 4.2 Toinen Iteraatio: Käyttöliittymä

Jos peli alkaa varoituksetta, tuntuu se luonnottomalta. Siksi loin alkuun hyvin alkeellisen käyttöliittymän pelin aloitusta varten. Toteutin luomalla uuden ”kohtauksen”, Unityssä tätä varten käytetään sanaa scene. Uuden scenen loin Scenes-kansioon ja Build Settings-valikossa varmistin, että Start Menu-scene on ensimmäisenä eli indeksinä 0. Tämä rakenne näkyy kuvioissa 12 ja 13.




Kuvio 12: Level 1 ja Start Menu

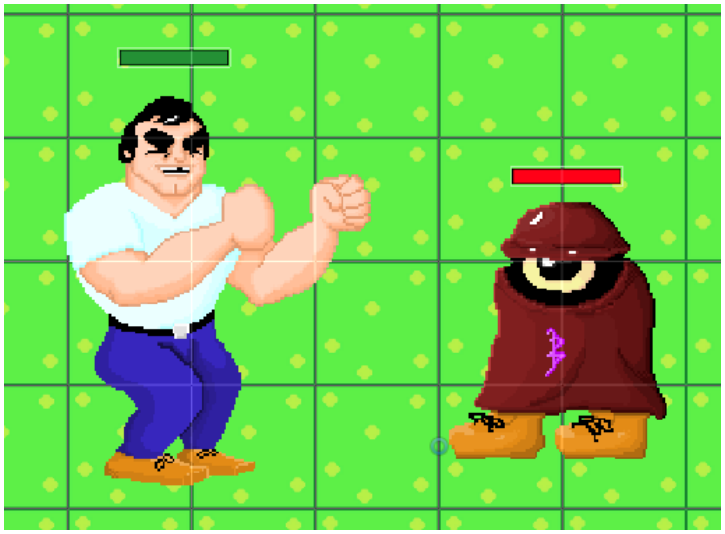



Kuvio 13: Build Settings

Ideana on se että, pelin aloittaessa pelaaja näkee värillisen ruudun, jossa on keskellä Play-nappi. Tätä painaessa peli alkaa ja pelaaja siirtyy varsinaiseen pelin näkymään. Teen samanlaisen näkymän sitä varten, kun peli loppuu, jolloin pelaaja voi heti aloittaa uuden pelin. Oheisessa kuviossa 14 esitetään tämä näkymä.

Kuvio 14: Play-nappi

Viimeinen UI-elementti, jonka lisäsin, oli palkit, jotka edustavat hahmojen kestävyyttä. Kun tämä palkki menee arvolta nollaan, niin peli loppuu joko pelaajan voittoon tai häviöön riippuen kumman hahmon kestävyys loppui. Annoin tälle objektille nimen unityssä Healthbar-Ui ja se koostuu kahdesta elementistä: graphicsgale sovelluksessa piirretty musta ääriviiva ja värillinen täyttö, joka vähenee sitä mukaan, kun hahmo ottaa vahinkoa. Ensimmäiseksi loin Unityssä Canvas-alueen, joka sisältää Ui-elementit. Nimesin tämän Healthbar-Ui:ksi ja lisäsin sinne Healthbar-nimisen objektin, joka sisältää ääriviivan ja värillisen täytön. Skaalasin nämä elementit samankokoisiksi ja asetin ne päällekkäin. Lisäsin myös Slider-komponentin, joka antaa toiminnallisuuden Healthbarin arvon vähentämistä varten, eli kun hahmo ottaa vahinkoa, niin täyttö vähenee numerollisen arvon mukaan. Lopuksi kiinnitän nämä elementit hahmoin siirtämällä objektit hahmojen sisälle hierarkiassa. Oheisessa kuviossa 15 näkyy lopullinen Healthbar-Ui.



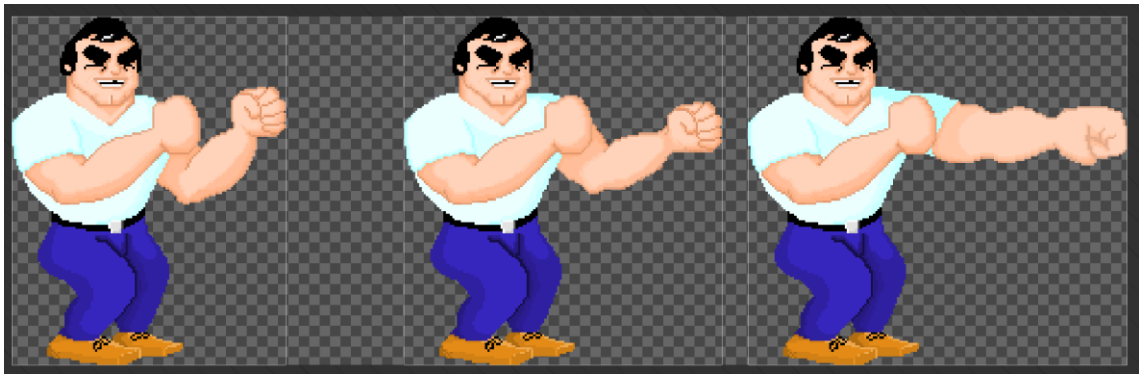
Kuvio 15: Healthbar-Ui

Slider-komponenttia käytetään Healthbar-skriptillä, joka liitetään hahmojen peliobjekteihin. Healthbarissa käytetään SetHealth-funktiota. Se asettaa arvon hahmojen health-muuttujalle, joka edustaa hahmojen kestävyyttä numerolla. Loin myös hahmoille omat skriptit, jossa kutsutaan Healthbarin funktioita ja asetetaan hahmoille omat arvot. Loin funktion nimeltä Attack, jossa asetan attackPoint-nimiseen muuttujaan AttackPoint peliobjektin. Se edustaa pelaajahahmon lyöntiä pelimaailmassa. Kiinnitin tämän Martini Steve -hahmon nyrkkiin ja tarkoituksena on seuraavassa iteraatiossa luoda animaatio Martini Steven lyöntiä varten. Attack-funktiossa loin Collider2D-nimisen taulukon. Se tunnistaa vihollisen, jos nyrkki osuu sen spriteihin. Tämän jälkeen funktio kutsuu vihollisen TakeDamage-funktiota. Se vähentää Healthbar-objektin arvoa määrätyllä luvulla. Koodi tunnistaa hahmoon osumisen Unityn omalla Physics2D funktiolla OverlapCircleAll, joka antaa listan kaikista collider-komponenteista asetetulla alueella (Docs Unity 2022). Lopuksi lisäsin vihollishahmolle Box Collider 2D -komponentin, jotta koodi toimisi. Box Collider 2D on Unityn komponentti. Se tunnistaa 2D-maailmassa interaktion muiden peliobjektien kanssa (Docs Unity 2022). Tässä kohtaa toinen iteraatio oli valmis testauksen jälkeen ja sopiva puskea repositorioon.

## 4.3 Kolmas Iteraatio: Animaatio

Tässä iteraatiossa oli tarkoituksena luoda animaatio pelaajahahmon iskulle. Animointi on hyvin aikaa vievä prosessi, joten päätin tässä iteraatiossa vain keskittyä mielestäni tärkeimpään animaatioon pysyäkseni halutussa aikataulussa. Lähestyin tätä prosessia miettimällä, mikä olisi luonnollisin asia, joka sopisi pelaajahahmon hyökkäykseksi. Tärkeää oli myös huomioida mitä pystyn tehdä ja mitä en. Kun alun perin piirsin hahmoani, olin jo ajatellut tätä tilannetta. Hahmo on pienessä kyykyssä ja pitää käsiänsä nyrkkeilyasennossa, joten nopea lyönti hahmon vasemmalla kädellä käy hyökkäyksestä.

Tehtävänäni oli siis suoristaa käsi ja antaa sille nopea animaatio, jotta se muistuttaa ”jabiä” nyrkkeilystä. Tätä animaatiota ei tehty ennen peliprojektin aloittamista, toisin kuin muut sprite-piirustukset. Suoritin valitsemalla animaation lähtökohdan, joka oli pelaajahahmon normaali asento. Asetin tämän kuvan graphicsgale-sovelluksessa alimmaiseksi kerrokseksi ja ylemmäksi kerrokseksi kopion saman kuvan, mutta leikkasin tästä irti käden ja piirsin sen uudessa asennossa. Laitoin tämän kerroksen opasiteetin noin puoleen, koska halusin nähdä alemman kerroksen kuvan, että pysyisin mahdollisimman paljon mallissa piirtäessäni kättä uudessa asennossa. Tämän kaltaista tekniikkaa kutsutaan Onion skinning:ksi (Kdanmobile 2022). Loppujen lopuksi tulin siihen tulokseen, että tarvitsen vain kaksi uutta piirustusta. Animaation kontekstissa käytetään sanaa frame näille animaation yksittäisille piirustuksille (Shuter 2020). Oheisessa kuviossa 16 näkyy lopullinen animaatio.

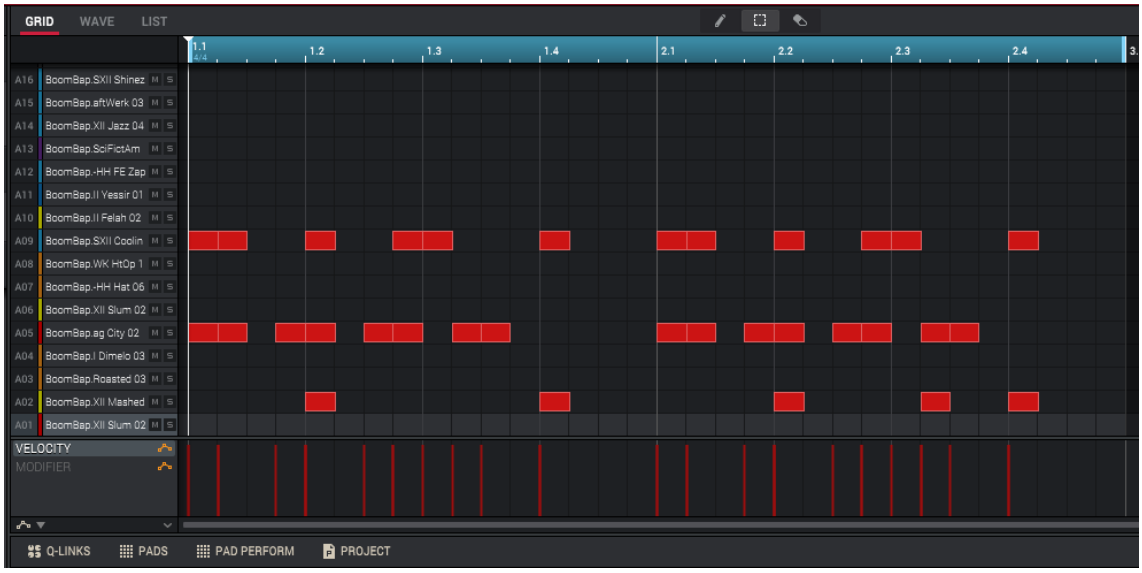


Kuvio 16: Jab-animaatio

Tässä kohtaa kehitystä törmäsin ongelmaan, joka johtui animaation kehysten (engl. frame) asetuksista. Näille uusille kuville piti antaa samat asetukset kuin hahmon normaalilla asenolla. Animaatio Unityssä toimii siirtymällä eri animaatioiden välillä. Kun hahmo seisoo paikallaan, sitä kutsutaan idle-animaatioksi ja on yleensä hahmon normaali tila (Couture 2018). Tässä on myös mahdollisuus antaa hahmolle enemmän personaalisuutta luomalla hahmon tyyliin sopivan animaation (Couture 2018). Tein hahmolleni idle-tilan, josta hän siirtyy jab-animaatioon ja takaisin. Tekemällä nämä transitiot hahmo ei pysy vain yhdessä tilassa, vaan näyttää luonnollisemmalta siirtyessään eri animaatioiden välillä. Lopuksi testasin, että se toimii halutusti ja pistin muutokset repositorioon.

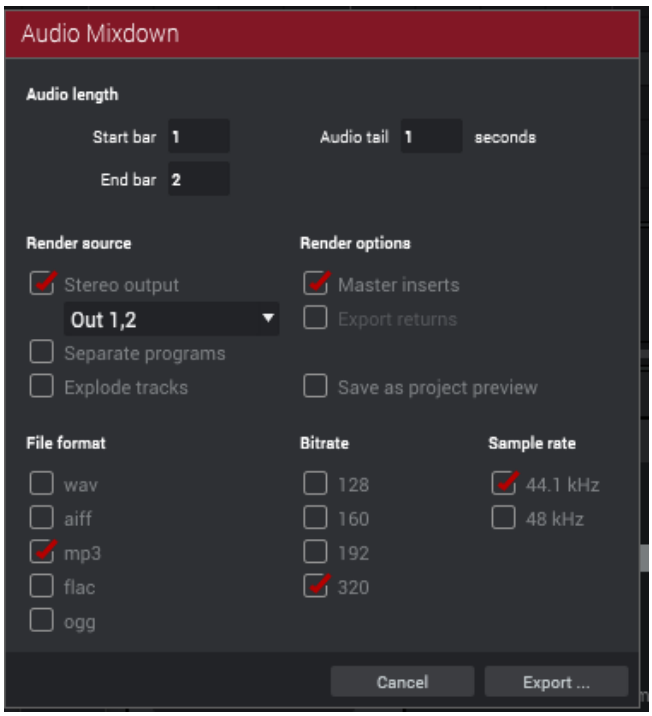
## 4.4 Neljäs Iteraatio: Musiikki

Etenin musiikin luomisessa kokeilemalla kaikkea mitä näin ja yritin luoda oman maun mukaan jotain sopivaa pelille. Suunnitelmani oli tehdä hieman nopeatempoinen muutaman iskun sävellys, jonka laittaisin toistumaan pelissä. Päädyin käyttämään peliä varten MPC Beatsin mukana tulleita BoomBap-rumpua, ja SynthWave-syntetisaattoria. Aloitusruutua varten käytin MPC Beats Producer Kits BassHouse-Kick ja Trap-Clap -nimisiä sampleja. Halusin käyttää molemmissa enimmäkseen rumpuja, koska ajattelin niiden tuovan videopelille sopivan tunnelman.




Kuvio 17: Musiikki peliä varten

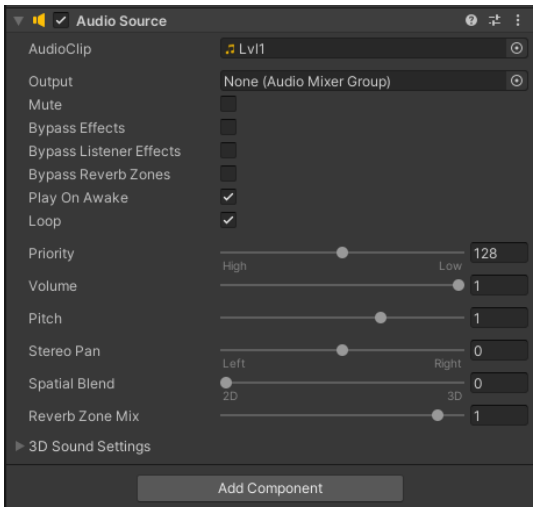
Siinä kohtaa, kun olin tarpeeksi tyytyväinen molempiin kappaleisiin, menin MPC Beats -sovelluksessa Export-valikkoon ja valitsin As Audio Mixdown -valinnan. Täällä valitsin tiedostotyypiksi mp3 ja Bitrate-valinnaksi 320. Oheisessa kuviossa on Audio Mixdown -ikkuna.




Kuvio 18: Audio Mixdown Export -ikkuna mp3-tiedostoa varten

Tämän jälkeen menin takaisin projektiini Unityssä ja valitsin pelimaailmani objektin Unityn hierarkiassa ja lisäsin Audio Source -komponentin objektiin. Kopioin mp3-tiedostoni projektiin audio-kansioon ja täältä liitin mp3-tiedoston AudioClip-kohtaan Audio Sourcessa. Laitoin myös

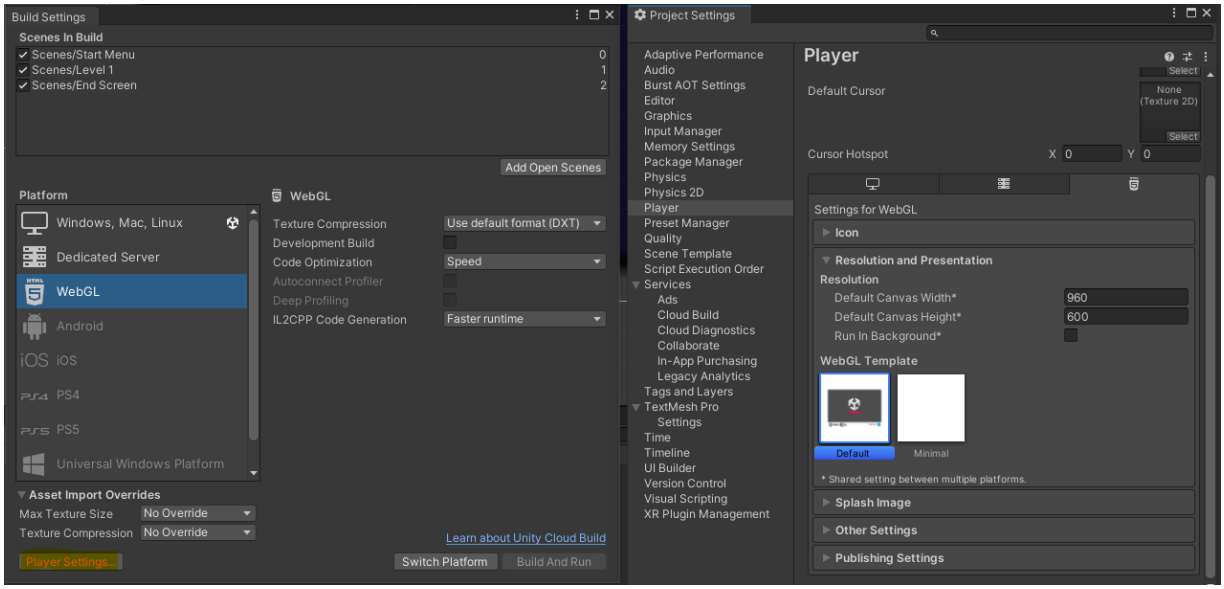
Loop-valinnan päälle, että kappale toistuisi tässä näkymässä niin kauan, kun pelaaja on siellä. Toistin samat asiat aloitusruudun kanssa, mutta liitin siihen oman mp3-tiedostonsa. Kuviossa 19 esitetään Audio Source-komponentti. Tämän jälkeen iteraation tehtävä oli valmiina puskettavaksi repositorioon ja olin valmis aloittamaan julkaisun.




Kuvio 19: Audio Source -komponentti Unityssä

## 4.5 Julkaisu GitHub Pages -palvelussa

Halusin tämän projektin olevan pelattavissa internetissä. Tämän suoritin GitHub Pages -toiminnalla. Tämän voi tehdä ilmaiseksi julkisella repositoriolla menemällä repositorion Settings-valikkoon ja navigoida Pages-kohtaan. Tässä kohtaa vaihdoin repositorioni julkiseksi. Tätä varten tarvitaan index-tiedosto, jonka voi luoda normaalisti repositorioon, mutta minun tapauksessani ensin pitää viimeistellä peli Unityssä. Build Settings -valikkossa valitsin WebGl-vaihtoehdon (Boutin 2019). WebGl (Web Graphics Library) on JavaScript API, joka renderöi grafiikoita kaikille yhteensopiville selaimille (Developer Mozilla 2022). Kuviossa 20 näkyy Build Settings -ikkuna.




Kuvio 20: Build Settings -ikkuna

Build Settingsistä menin Player Settingsiin ja laitoin Publishing Settingsissä Compression Formatin -valikon pois päältä. Tämä oli iso ongelma, johon törmäsin julkaisuvaiheessa. GitHub Pages ei toiminut gzip ja Brotli -valintojen kanssa. Yritin näillä molemmilla asetuksilla, mutta molemmilla kerroilla GitHub Pages -sivuni ilmoitti, että näitä tiedostoja ei voinut lukea. Tämän takia päädyin ottamaan kompressoinnin pois päältä. (Docs Unity 2022.)

Puskin viimeiset muutokseni läpi repositorioon ja lopuksi painoin Build-nappia Build Settingsissä. Tämän suoritettua sain Build-kansion ja index-tiedoston ja siirsin nämä repositorion juureen. Tämä tarvitsee tehdä, koska GitHub Pages katsoo juuresta index-tiedoston, jolla se suorittaa pelin (Medium 2019). Tämän jälkeen pelini on saatavilla GitHubissa (GitHub 2022).

Kuvio 21: Valmis julkaistu peli GitHubissa (GitHub, 2022)

## 5 Jatkokehitysideat

Tällä hetkellä peli sisältää tarpeeksi piirteitä, että sen voi tunnistaa peliksi. Pelaaja kykenee liikkumaan pelimaailmassa, lyömään vastustajaa ja voittamaan pelin. Lisäksi pelillä on musiikki aloitusruudussa ja itse pelin aikana. Vaikka nämä ominaisuudet ovat olemassa, niitä silti voi optimoida ja lisätä uusia ominaisuuksia. Isoin puuttuva osio on vihollisen toiminnallisuudet. Vihollinen kykenee liikkumaan samalla tavalla kuin pelaajahahmo, mutta se ei pysty taistelemaan vastaan. Pelimaailma tuntuu tyhjältä ja kaipaisi enemmän visuaalisia elementtejä. Asiat tuntuvat vielä hieman epäluonnollisilta. Peli alkaa ja loppuu hyvin äkkiä, joten peliin olisi hyvä lisätä hieman viivettä, kun asioita tapahtuu pelissä. Ohjelmointia vaativat kehitysideat tulevat tarvitsemaan enemmän suunnittelua ja parempaa Unity-pelimoottorin hallintaa.

Vaikka musiikki on olemassa pelissä, sen implementointi on huomattavasti vähemmän laadukasta, kuin esimerkiksi visuaalinen toteutus. Tämä tulee tarvitsemaan syvempää perehtymistä musiikin opettelemiseen. Tällä hetkellä luodut kappaleet peliä varten eivät välttämättä sovi pelin teemaan kovin luonnollisesti, joten jatkokehitysideana tässä voisi olla uusien kappaleiden luominen, ja samalla varmistaa niiden pysyvän pelin teemassa paremmin.

Viimeinen jatkokehitysidea on pelin laajentaminen. Tällä hetkellä on vain yksi vastustaja ja kenttä. Aiempien mainittujen ideoiden jälkeen voisi miettiä, olisiko vaivan arvoista luoda enemmän uusia piirteitä, jotka lisäävät pelattavuutta. Yksi vaihtoehto on myös hyödyntää opinnäytetyön aikana luotua ympäristöä ja jakaa projekti ystäville, jotka haluavat myös

kokemusta pelinkehityksessä. Peli on olemassa julkisessa repositoriossa, joten sen jakaminen on helppoa. Pystyn antamaan ihmisille, joihin luotan, oikeudet tekemään muutoksia repositorioon. Tämän kannalta olisi myös kannattavaa kirjoittaa repositorion ReadMe-tiedostoon ohjeistukset ympäristön luontia varten.

Oman kehittymisen kannalta voisi olla muun muassa miettiä parannuksia kehittämisympäristöön ja laajentaa osaamista tutustumalla uusin teknologioihin. Pelimoottoreita ja ohjelmointikieliä on useampia, niin tiedon laajentaminen aiheesta voisi olla seuraava askel.

# 6 Yhteenveto

Ensimmäinen julkaistu peliversio toteuttaa asetetut kriteerit ensimmäistä julkaisua varten. Jatkan tämän projektin kehitystä opinnäytetyön jälkeenkin oman harrastuksen vuoksi. Iteratiivinen kehitysmenetelmä auttoi segmentoimaan pelin ominaisuudet ja hahmottamaan, miten paljon työtä tarvitsee tehdä. Projekti on valmis uusia iteraatioita varten.

Pelin kehityksen aikana ilmeni useampia ongelmia, joista suurin osa johtui erilaisten Unity-asetuksista. Projekti eteni tämän takia ajoittain haluttua hitaammin. Ohjelmointi C#:lla oli myös alkuun hidasta, ennen kuin tottui sen syntaksiin. Kokemus Unityn käytössä oli rajoittava tekijä projektin aikana. Tämä vaati sen, että opin projektin aikana korjaamaan vieraita ongelmia ja lukemaan paljon dokumentaatiota pelinkehityksestä Unity-moottorilla.

Haasteena opinnäytetyössä oli ongelmien ratkaisu, kun niitä tuli eteen. Projektin aikana Unity useasti vaihtoi erilaisia asetuksia aavemaisesti, jotka rikkoivat peliprojektin useampaan kertaan. Korjaus näihin ongelmiin oli aina hyvin yksinkertainen, mutta nämä silti hidastivat kehitystä huomattavasti, koska olin tekemässä projektia yksin.

Opinnäytetyön aikana pelinkehityksen todellisuus tuli selväksi. Pelinkehitys on vaikeaa tehdä hyvin ja se vaatii taidon lisäksi todella paljon aikaa. Opinnäytetyön tavoite saavutettiin, toiminnallinen 2D-peli, joka on saatavilla internetissä dokumentoiden siihen tarvittavat askeleet. Jokainen osa projektia oli itsenäisesti tehty ja sain halutut ominaisuudet tehtyä, mutta kehityksen aikana ymmärsin, että aliarvioin vaaditun työnmäärän. Tämän työn aikana kehityin paljon pelinkehityksessä, joka on antanut minulle paremman ymmärryksen siitä, miten pelit toimivat ja miten ne tehdään.

Lähteet

Sähköiset

Microsoft. C# Documentation. Viitattu 22.11.2022. https://learn.microsoft.com/en-us/dotnet/csharp/

Unity. Unity Learn. Viitattu 22.11.2022. learn.unity.com

Docs Unity. 2D And 3D Mode Settings. Viitattu 22.11.2022. https://docs.unity3d.com/Manual/2DAnd3DModeSettings.html

Docs Unity. Grid. Viitattu 22.11.2022. https://docs.unity3d.com/Manual/class-Grid.html

Docs Unity. WebGl Server Configuration Code Samples. Viitattu 22.11.2022. https://docs.unity3d.com/Manual/webgl-server-configuration-code-samples.html

Docs Unity. Camera ScreenToWorldPoint. Viitattu 22.11.2022. https://docs.unity3d.com/ScriptReference/Camera.ScreenToWorldPoint.html

https://en.wikipedia.org/wiki/Game_design_document

Takahashi, D. 16.3.2022. Unity report: Number of games made with Unity grew 93% in 2021. Viitattu 22.11.2022. https://venturebeat.com/games/unity-report-number-of-games-made-with-unity-grew-93-in-2021/

Medeiros, P. 6.11.2018. How to start making pixel art #5. Viitattu 22.11.2022 https://medium.com/pixel-grimoire/how-to-start-making-pixel-art-4-ff4bfcd2d085

Lux. 14.9.2020. Anti-Aliasing Fundamentals for Pixel Artists. Viitattu 22.11.2022 https://pixelparmesan.com/anti-aliasing-fundamentals-for-pixel-artists/

Kinsta. 9.16.2022. What Is GitHub? A Beginner's Introduction to GitHub. Viitattu 22.11.2022 https://kinsta.com/knowledgebase/what-is-github/

Neimeister, J. 18.2.2022. Jon Neimeister offers an in-depth analysis of great character design. Viitattu 22.11.2022 https://visualartspassage.com/blog/principles-of-character-design/

Tutorialspoint. 2022. What is the purpose of the .gitignore file?. Viitattu 22.11.2022 https://www.tutorialspoint.com/what-is-the-purpose-of-the-gitignore-file

Shuter, G. 12.10.2020. Frame-By-Frame Animation: A Complete Guide. Viitattu. 22.11.2022 https://www.twine.net/blog/frame-by-frame-animation-complete-guide/

Animation Desk. 2022. What is Onion Skin and How to Use it in 2D Animation. Viitattu. 22.11.2022 https://www.kdanmobile.com/en/animation-desk/onion-skin-in-2D-animation

Boutin, A. 4.2.2019. Host Unity Games on GitHub Pages; For Free. Viitattu. 22.11.2022 https://medium.com/@aboutin/host-unity-games-on-github-pages-for-free-2ed6b4d9c324

Developer Mozilla. 2022. WebGL: 2D and 3D graphics for the web. Viitattu. 22.11.2022 https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API

Akai Professional. 2022. MPC BEATS. Viitattu. 22.11.2022 https://www.akaipro.com/mpc-beats

Javatpoint. 2022. Iterative Model. Viitattu. 22.11.2022 https://www.javatpoint.com/software-engineering-iterative-model

GitHub. Tavern-Crawl. Viitattu. 22.11.2022 https://mikhuov.github.io/Tavern-Crawl/

Couture, J. 21.5.2018. What makes a great idle animation? Devs share their favorites. Viitattu. 22.11.2022 https://www.gamedeveloper.com/art/what-makes-a-great-idle-animation-devs-share-their-favorites

Eastern Peak. 2022. Iterative Development. Viitattu. 22.11.2022 https://eastern-peak.com/definition/iterative-development/

Kuviot

Kuvio 1: Iteraatiivinen kehitys Software Development Life Cycle (SDLC) (Javatpoint, 2022) ... 7
Kuvio 2: Repositorion luonti (GitHub, 2022) ... 9
Kuvio 3: GitHub Desktop käyttöliittymä ... 9
Kuvio 4: Transparentin värin valinnan prosessi ... 11
Kuvio 5: Hahmon ulottuvuudet ... 12
Kuvio 6: Tilemap Unityssä ... 13
Kuvio 7: Tile Palette-työkalu Unityssä ... 14
Kuvio 10: Grid-komponentti Unityssä ... 16
Kuvio 11: Movement Scripti liitettynä pelikenttään ... 17
Kuvio 12: Level 1 ja Start Menu ... 18
Kuvio 13: Build Settings ... 18
Kuvio 14: Play-nappi ... 19
Kuvio 15: Healthbar-Ui ... 19
Kuvio 16: Jab-animaatio ... 21
Kuvio 17: Musiikki peliä varten ... 22
Kuvio 18: Audio Mixdown Export -ikkuna mp3-tiedostoa varten ... 22
Kuvio 19: Audio Source -komponentti Unityssä ... 23
Kuvio 20: Build Settings -ikkuna ... 24
Kuvio 21: Valmis julkaistu peli GitHubissa (GitHub, 2022) ... 25