Videopelin luonnin perusteet tarinaan keskittyen

Joni Miettinen

Haaga-Helia ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Toiminnallinen opinnäytetyö
2023

Tiivistelmä

Tekijä(t)
Joni Miettinen

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Videopelin luonnin perusteet tarinaan keskittyen

Sivu- ja liitesivumäärä
22

Tämä raportti sisältää videopelien tuotannon perusteita tarinakeskeisen pelin näkökulmasta. Käydään läpi keskeisiä taitoja yksinpelinkehittäjälle, joihin kuuluu taide, animaatio, musiikki, äänet, ohjelmointi, kirjoitus, ja tietenkin itse pelin suunnittelu.

Tässä raportissa käydään myös läpi ilmaisia ohjelmia pelin kehitykseen, mukaan lukien pelimoottorit, taideohjelmat, sekä musiikinluontiohjelmat. Näitä seuraa pelin suunnittelu tarinaan keskittyen, ja itse tarinan kirjoitus, jossa luon tarinan, joka ei vaadi konfliktia taikka kamppailua.

Lopulta käyn läpi pelin luonnin, käyden läpi Unityn perusteet, hahmon luonnin, pelaajahahmon luonnin, Unityn tasosysteemin nimeltä näkymät, dialogisysteemin, välianimaatiot pelaamisen ja dialogin välillä, sekä musiikin ja ääniefektit.

Asiasanat
Videopeli, ohjelmointi, kuvataide, pelimusiikki

Sisältö

1 Johdanto

Opinnäytetyöprojektissani tulen tekemään yksinkertaisen videopelin käyttäen Unity-pelimoottoria ja Ink-dialogisysteemiä. Peli tulee olemaan pelattavuudeltaan yksinkertainen 2D-kävelysimulaattori, jossa pelin vetovoima tulee sen tarinasta ja kahdesta päähahmosta. Tämä on hyvä tapa oppia tekemään pelejä Unitylla ja tähän liittyvillä ohjelmilla rutiininomaisesti.

Olen valinnut kyseisen aiheen opinnäytetyökseni, koska syyni IT-alalla opiskelemiseen on aina ollut lopulta työskennellä videopelialalla. Isoin hyöty on siis itselleni, tämä työ ollen näyte osaamisestani, joka liittyy haluamaani alaan. Työni myös auttaa itseäni paremmin ymmärtämään kyseistä alaa teknisellä tasolla.

Olen tehnyt tämän raportin vetoketjumallilla, jokaisen kohdan tietoperustan ja tulokset löydät alta. Lähtötilanteella ei ole erillistä tarvetta molemmille, joten se on pidetty yhtenä osana.

Teema	Tietoperusta (Luku/Sivu)	Tulokset (Luku/Sivu)
Lähtötilanne	2/	2/
Hyödyllisiä aikaisempia taitoja	3.1/	3.2/
Pelinteko ohjelmia nollabudjetilla	4.1/	4.2/
Pelin suunnittelu	5.1/	5.2/
Pelin tarinan kirjoitus	6.1/	6.2/
Pelin tuottaminen		7.1–6/

2 Lähtötilanne

Suunnittelin opinnäytetyöni aiheen itse, eli minulla ei ole ollut erillistä toimeksiantajaa. Tämä tuotos oli oppimiskokemus itselleni, missä opin pelienteon perusteet käyttäen Unitya. Tällä tuotoksella ei ole tiettyä asiakaskuntaa, mutta muut, jotka haluavat oppia näitä samoja asioita, pääsevät tarkistelemaan tämän projektin avointa lähdekoodia GitHubissa. Vuorostaan ne, jotka haluavat pelata tätä peliä, voivat löytää sen GitHubista ja Itch.iosta

Projektin suurimpina haasteina olivat oma taitoni, aikani ja täysin nollabudjetti. Minulla oli projektin alkaessa aiempaa kuvataide- ja musiikkitaustaa. Tämän lisäksi olin ollut osana isompaa peliprojektia ohjelmoijana, mutta tässä projektissa astuin sisään vain osa-aikaisesti tekemään näitä hommia. Minulta puuttuivat tiedot ja taidot, jotka liittyvät Unityn syvempiin järjestelmiin, esim. välikohtauksiin ja dialogisysteemiin.

En ollut itsekään varma, milloin tiedän, onko lopputuotos ’hyvä’, pelithän ovat aika subjektiivisia asioita. Päätin, että olen tyytyväinen omaan tuotokseeni, jos saan tehtyä yhden täyden pelin, joka ei ole täynnä vikoja, ja jota voi olla miellyttävä pelata läpi kerran tai kaksi.

3 Hyödyllisiä aikaisempia taitoja

3.1 Tietoperusta

Pelien tuotannossa tarvitaan monta eri taitoa. Ohjelmointi ja suunnittelu ovat ensisijaisia mielessä, mutta tarvitaan myös kuvataidetta ja musiikkia. Kun lähtee yksin tekemään peliä, pitää pelintekijällä olla nämä kaikki hallussa tavalla taikka toisella.

Kuva 1. Fleischer-tyylinen karhun naama, käyttäen Gameboyn tyylistä väripalettia.

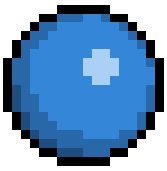
Yksi tapa, jolla ohjelmointitaustaiset yksinpelintekijät paikkaavat artistin paikkaa, on käyttämällä helposti opittavaa pikselitaidetta. Kuva 1 näyttää esimerkin 32 kertaa 32-kokoisesta pikselitaiteesta. Pikselitaiteessa kuva tuodaan pienelle tasolle, jossa ei ole tilaa käyttää värigradientteja eli värien tasaista siirtymistä yhdestä toiseen, sumennusta taikka reunan pehmennystä. Tämän taiteen taiteen pieni koko ja siitä syntyvät rajoitukset tekevät tästä tekniikasta helpon oppia ohjelmointitaustaisille pelintekijöille. (Kotaki 12.07.2012)

Helposti erottuvat värit, yhdenpaksuiset ääriviivat ja erikseen muokattavat pikselit mahdollistavat sen, että ei tarvitse siveltimien kanssa temppuilla. Kunhan on hyvä silmä muodoille ja väreille. Tämä on yksi syy, miksi pikselitaide on helppo oppia.

Pikselitaiteessa käytetään usein myös rajoitettuja väripaletteja, joissa kunkin värin valinta on tärkeä askel. Kuva 1 näyttää esimerkin neljän värin paletista, käyttäen neljää vihreän sävyä. Kun valitsee värejä, on kolme keskeistä muuttujaa: Värisävy, esim. sininen, punainen, vihreä. Kylläisyys, eli kuinka voimakas väri on. Ja kirkkaus, eli kuinka vaalea tai tumma väri on. (Pixel Overload 1.1.2021, 0:24-1:27)

Näitä värejä valittaessa, on käytössä monta pientä keinoa. Kun vaaditaan monta tasoa samaa väriä, esimerkiksi varjoja ja kohokohtia varten, kannattaa käyttää kaikkea kolmea muuttujaa. Varjojen kannattaa olla kylmempiä värejä, lähempänä sinistä, kun kohokohdat vuorostaan ovat lämpimämpiä värejä, lähempänä keltaista. Varjojen kannattavaa olla myös tummempia ja kohokohtien kirkkaampia verrattuna toisiinsa. Yksi keino on ottaa samanlaisia värejä paletista ja tehdä niistä yksi väri. Tämä parantaa paletin yhtenäisyyttä. (Pixel Overload 1.1.2021, 1:28-1:27)

Seuraavaksi hahmot animoidaan. Hahmoilla pitää olla vähintään kävelysyklit, elleivät ne ole liitäviä kuutioita. Tässä vaiheessa on hyvä ottaa käyttöön animaation kaksitoista periaatetta. Seuraavassa on kerrottu omasta mielestäni pikselitaiteeseen liittyen tärkeimmät periaatteet.

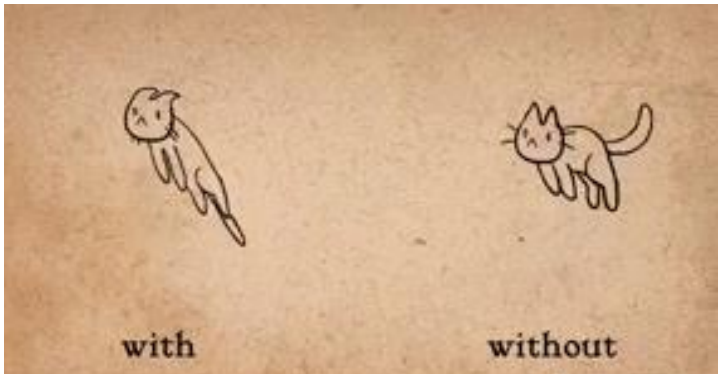


Kuva 2. Pomppiva sininen pallo.

Ensimmäinen periaate on ’squash and stretch’, eli litistys ja venytys. Tämän periaatteen mukaan animoitujen esineiden kannattaa litistyä ja venähtää kun ne liikkuvat, litistyen pysähtyessä ja venähtäen nopeuden lisääntyessä. Näiden asioiden täytyy pitää sama tilavuus, joten litistyessä ne leviävät ja venähtäessä ne ohentuvat. Tämän periaatteen avulla korostetaan esineen liikettä. (Ritchie 2017.) Kuvassa 2 nähdään esimerkki, kun pikselöity pallo litistyy osuessaan maahan ja kapenee ylös noustessaan.

Toinen periaate on timing, eli ajoitus. Ajoitus tarkoittaa sitä, kuinka monta ruutua menee kuhunkin yksittäiseen animaation osaan. Tällä tarkoitetaan esimerkiksi, kuinka kauan hahmo on liikkeessä verrattuna tätä edeltävään ja seuraavaan asentoon. Näiden asentojen annetaan yleensä olla näytillä pari ruutua enemmän liikkeeseen verrattuna. (Ritchie 2017.) Kuva 2 antaa tästäkin esimerkin. Kuvasta on nähtävillä se, kuinka vähän aikaa pallolla menee maan ja hypyn huipun välillä näihin kahteen kiinteään asentoon verrattuna.

Kolmannes periaate on anticipation, eli ennakointi. Ennakoinnissa tuodaan hahmo ennakoivaan asentoon ennen seuraavaa liikettä. Jos hahmo meinaa huitaista pesäpallomailalla, on hänen ensin vedettävä maila taaksepäin ja vakauttaa asentonsa. Täten seuraava isku vaikuttaa voimakkaammalta. (Ritchie 2017.) Kuva 2 antaa vielä tästäkin esimerkin, kun pallo litistyy hieman ennen hyppyä.



Kuva 3. Esimerkki läpiviennistä ja päällekkäisestä toiminnasta. (AlanBeckerTutorials 2015)

Viimeinen periaate on follow-through ja overlapping action, eli läpivienti ja päällekkäinen toiminta. Tämä periaate kertoo, että jonkin asian osat seuraavat itse asiaa hieman jäljessä, kun tämä asia lähtee liikkeelle. Nämä osat jatkavat pienen matkan asiaa pidemmälle, kun esine pysähtyy paikalleen. Periaate on nähtävillä muun muassa pitkien hiuksien, vaatteiden ja varsinkin viittojen liikkeissä. Tätä voi myös käyttää isoissa liikkeissä, esim. antaen hahmon mennä hetkeksi kyykkyyn ennen pystyyn nousemista, ja silloin kun hahmo laskeutuu maahan hypyn jälkeen. (Ritchie 2017.) Kuva 3 näyttää tästä esimerkin, vasemmalla olevan kissan häntä ja korvat seuraavat hieman perässä, ja koko kissa tippuu kyykkyyn laskeutuessaan maahan.

3.2 Tulokset

Pikselitaide on ollut osa videopelejä näiden alkuajoista lähtien, ja olen itse luonut pikselitaidetta pidemmänkin ajan oman kiinnostuksen pohjalta. Täten päädyin tekemään pelin visuaalisen puolen pikselitaiteella.

Olin päättänyt käyttää valmista Adigun Polackin (2017) AAP-64 väripalettia, joka tuli käyttämäni taideohjelman Asepriten kanssa. Tämä kyseinen värilajitelma oli itselleni tuttu, ja se on myös viidenneksi ladatuin paletti pikselitaide resurssivarastosivusto lospec.com:in palettilistoilla. AAP-64 on ladattu 41,015 kertaa kun tätä kirjoitan, ja tämä on vain lospec.com:ista tehdyt lataukset.

Aloitin pelin tekemisen päähahmon taiteen luonnilla ja sen animoinnilla. Tähän käytin pohjana taidetta, jonka olin jo tehnyt ennen projektin virallista aloittamista, koska olin tiennyt haluavani tehdä tämän tyyppisen projektin jonain päivänä. Pelin päähahmosta tuli Jonne, jota käytin mittana pelin pikselitaiteen mittakaavaa varten. Myös muut pelin hahmot perustuivat tähän Jonnen hahmoon.

Tietenkin animoin nämä hahmot, käyttäen avuksi aiemmin mainittuja animaatiotekniikoita. Animaatiot keskittyivät vain kahteen hahmoon, jotka saivat kävelysyklit sekä muita liikkeitä. Animaatioiden ohella tuli tehtyä pelin taustat ja paikat. Nämä piti tehdä osittain samaan aikaan kuin hahmojen animaatiot, jotta nämä saataisiin sopimaan toisiinsa ilman virheitä.

Tämän jälkeen tein pelin neljä pelialuetta. Pelialueet ovat Jonnen koti, lähellä oleva tori, viereisen kahvilan sisätila, sekä kieleke, jossa on vanha ilmatorjuntatykki. Näihin käytin omaa kotikaupunkiani Tuusulaa referenssinä.

4 Pelinteko-ohjelmia nollabudjetilla

4.1 Tietoperusta

Onneksi nykymaailmassamme on saatavilla runsaasti ilmaisia pelinteko-ohjelmia. Muuten olisi vaikea tehdä pelejä ilman budjettia.

Aloitetaan pelimoottorilla. Pelimoottori on rakenne, joka mahdollistaa pelin nopean ja sujuvan luonnin tekemällä renderöinnin, pelien fysiikan laskemisen, äänten soittamisen, ja monen muun asian valmiiksi. Täten pelientekijöiden ei aina tarvitse lähteä tekemään näitä systeemejä nollasta jokaista uutta peliä varten. (Loveridge, S. 23.12.2018)

Pelimoottoreista on paljon mahdollisia vaihtoehtoja, mutta keskityn seuraavaksi kahteen isoimpaan saatavilla olevaan pelimoottoriin, Unity ja Unreal. Nämä kaksi pelimoottoria tekevät suunnilleen samat asiat, ne käyttävät jo olemassa olevia ohjelmointikieliä ja ovat siksi erittäin joustavia. (Trainor-Fogleman 2021)

Trainor-Fogleman (2021) tuo esille sen, että näistä Unity on helpompi käyttää, koska se käyttää helpompaa C#-kieltä. Unreal vuorostaan käyttää C++-kieltä, joka on vaikeampi oppia. Tämä asia pätee myös yleisemmin molempiin ohjelmiin. Unity on helpompi käyttää, mutta Unreal-moottorilla vuorostaan pystytään tekemään paremman näköisiä pelejä helpommin ja nopeammin, ja se pystytään myös optimoimaan niitä paremmin.

Seuraavaksi tarvitaan pelin taide. Tulen keskittymään työkaluihin, joilla luodaan pikselitaidetta, kun olen jo päättänyt käyttää tätä tiettyä tyyliä.

Greer (2019) tuo esille kaksitoista eri vaihtoehtoa, joista perehdymme kolmeen. Ensimmäinen on Aseprite, saatavissa Steam-kauppapaikassa hintaan 16,79 € kun tätä kirjoitan. (Kuva 4.) Sen saa myös koota sen avoimesta lähdekoodista. Aseprite on luotu juuri pikselitaidetta varten, jonka näkee jo sen pikselöidystä käyttöliittymästä. Aseprite tekee animaatioiden teosta helppo sen sisältämällä aikajanalla, lisäksi sisältää väripaletteja, joita usein käytetään pikselitaiteessa sekä vanhoissa pelikonsoleissa.

Kuva 4. Asepriten käyttöliittymä. (Igara Studio 2016.)

Toisena vaihtoehtona on Pyxel edit hintaan 9 $ omalla sivustollaan. Pyxel edit keskittyy tekemään laattasettejä, joilla voidaan esim. tehdä pelikarttoja toistamalla laattoja. Pyxel edit myös sisältää animaatiotyökalut, mutta sen käyttöliittymä on Asepriteen verrattuna yksinkertaisempi. (Greer 2019.)

Täysin ilmaisena vaihtoehtona suositellaan ohjelmaa nimeltä GraphicsGale. Tämä on freewarena levitettävä ohjelma, joka myös keskittyy pikselitaiteeseen. GraphicsGale sisältää väripaletteja kuten Asepritessa ja Pyxel editissä, sekä animaatiotyökalut ja tähän liittyvät aputyökalut. (Greer 2019.)

Seuraavana on tarve tehdä musiikkia peliä varten. Monella pelintekijäksi haluavalla ei todennäköisesti ole minkäänlaista kokemusta musiikin tekemisestä, eikä budjettia tai aikaa hankkia ja oppia soittamaan soittimia ja käyttämään isoja musiikinluonti ohjelmia. Onneksi on saatavilla monia ilmaisia ja helposti käytettäviä vaihtoehtoja.

Baxter ja Clark (2022) esittelevät viisi vaihtoehtoa musiikinteko-ohjelmista, joista perehdymme seuraavaksi kolmeen: Waveform Free, GarageBand ja CakeWalk.

Waveform Free on ilmaiseksi saatavissa oleva vanhempi versio Tracktionin Waveform-ohjelmasta, joka on yleensä kaksi päivitystä maksetun version takana. Ilmaisessa versiossa on kuitenkin saatavilla kaikki tärkeimmät ominaisuudet. Niitä ovat esimerkiksi loputon määrä midi- ja audioraitoja, skaalattava ja muokattava käyttöliittymä sekä, yhteensopivuus mm. VST ja audio unit-liitännäisten kanssa.

Garageband on Apple-laitteille tehty ilmainen, monipuolinen ja tehokas musiikinluonti-ohjelma. Sillä ei voi ohjata ulkoisia soittimia midillä tai tallentaa musiikkia midi-muotoisena, ja siinä on vain 255 midiraitaa. Garagebandilla pääsee silti pitkälle mm. Drummer-lisäosalla, jolla on helppo tehdä rumpurytmejä. Mikäli ohjelmalta tarvitaan enemmän ja rahaa on käytettävissä, tästä ohjelmasta on helppo siirtyä Applen Logic-musiikinluontiohjelmaan.

Cakewalk on yksi ensimmäisistä musiikinteko-ohjelmista. Ohjelman alkuperäinen tekijä Gibson, lakkautti päivitykset aikoinaan, mutta BandLab-niminen alusta otti ohjelman käyttöönsä ja teki siitä ilmaisen. Cakewalkilla pystyy tekemään kaiken mitä musiikinteossa tarvitaan, siinä on loputtomasti midiraitoja, miksaus- ja masterointityökalut, sekä työkaluja musiikin kirjoitukseen ja soittimien käyttöön.

4.2 Tulokset

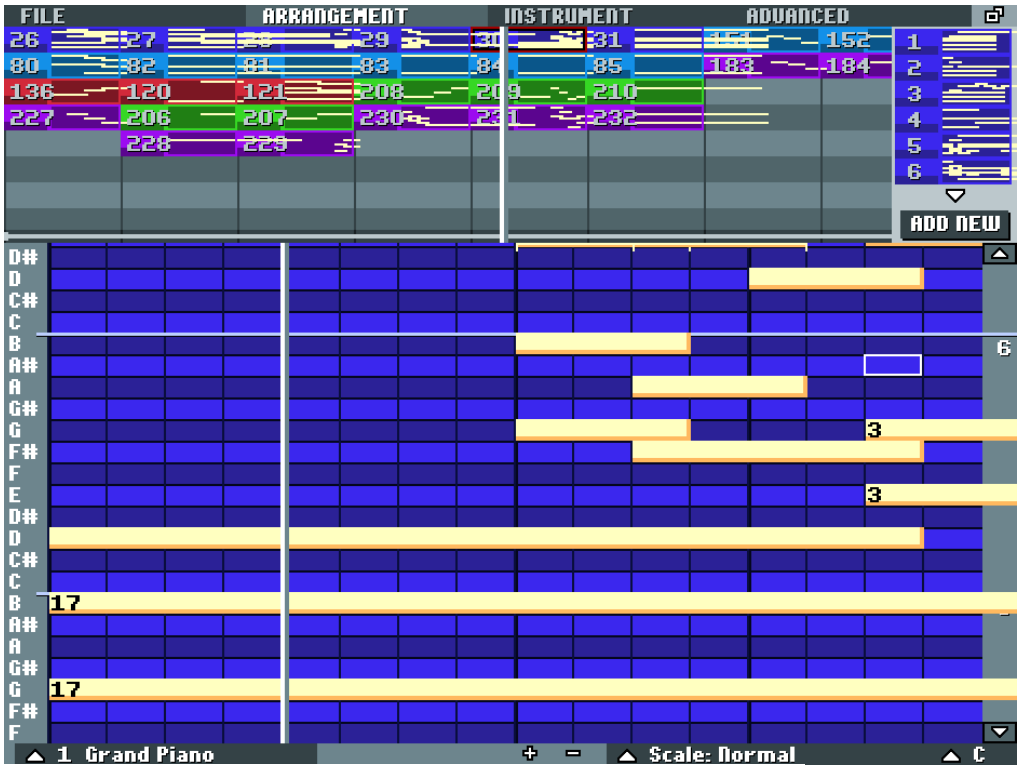
Olin valinnut Unityn pelin tekemiseen jo ennen projektin aloitusta, koska minulla oli jo aikaisempaa kokemusta ohjelman käytöstä omien projektien kanssa, sekä työssäoppimisjaksoista ammattikoulussa ja ammattikorkeakoulussa. Tiesin myös, että sille löytyy hyviä ohjeita internetistä, sekä lisäosia, jotka tekevät muun muassa dialogin tekemisestä helpompaa.

Pelissä käytin inklen ink-dialogisysteemiä. Kyseisellä systeemillä tehdään yksinkertaisia dialogipohjaisia pelejä, ja sille löytyy myös Unityyn liittämiseen mahdollistava lisäosa. Näiden kahden sekä inklen Inky-tekstieditorin kanssa pystyn tuottamaan pelin vaatiman dialogin helposti ja nopeasti.

Sitten tuli tarve ohjelmalle, jolla tehdä pelin taide. Kun olin päättänyt, että tekisin tämän pikselitaiteella, oli minulla aikaisemmin hankittu ohjelma Aseprite tätä varten valmiina. Normaalisti kyseisen ohjelman voi ostaa, mutta sen lähdekoodi on avoin, ja sen kokoamiseen voi löytää netistä ohjeet, jotka mahdollistavat sen ilmaisen hankkimisen.

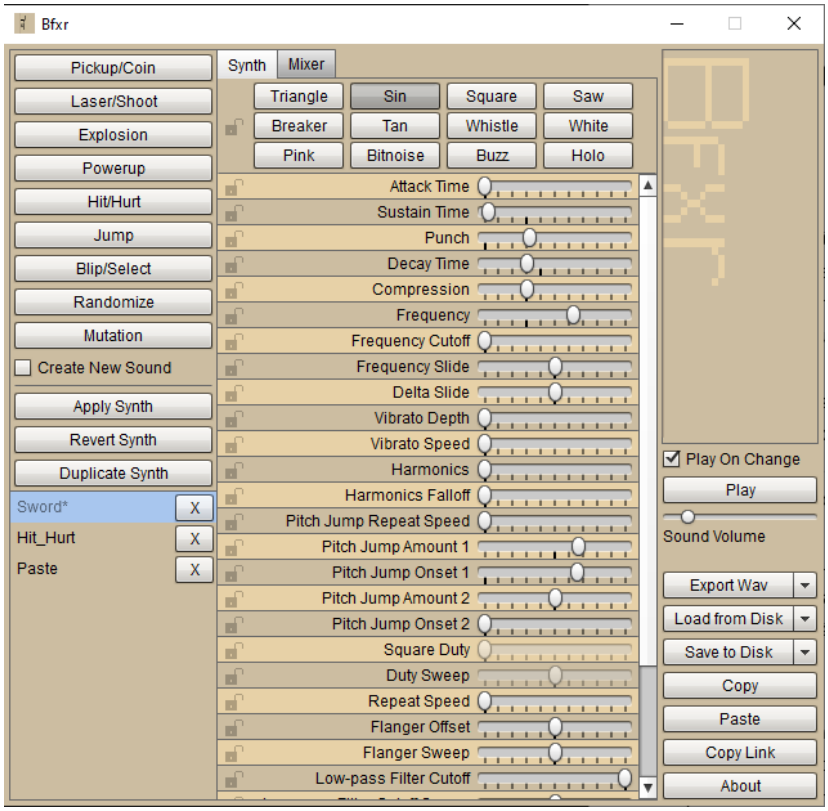
Seuraavaksi tarvitsin ohjelmia pelin musiikkia ja äänitehosteita varten. Näihin molempiin olin myös aikaisemmin löytänyt sopivat ohjelmat omia tarpeitani varten.

Ensimmäinen näistä on Bosca Ceoil, ilmainen ja avoimen lähdekoodin omaava musiikinteko-ohjelma, joka on suunnattu aloittelijoille ja varsinkin pelintekijöille, joilla ei ole aikaisempaa kokemusta musiikinteosta. Ohjelma on helppo käyttää ja se sisältää työkaluja, jotka helpottavat musiikin tekoa. Esimerkkinä käytän sävellaji- ja sointutyökalua, jotka rajaavat käytössä olevat sävelet pelaajan valitseman sävellajin mukaan. (Kuva 5.)



Kuva 5. Bosca Ceoilin käyttöliittymä (Cavanagh)

Toinen näistä on Bfxr-ohjelma. Kuten Bosca Ceoil, tämänkin ohjelman lähdekoodi on avoin. Kyseinen ohjelma on luotu tekemään yksinkertaisiä ääniefektejä, käyttäen yksinkertaisia aaltomuotoja ja monenlaisia muuttujia ja suodattimia. (Kuva 6.)



Kuva 6. Bfxr:n käyttöliittymä

5 Pelin suunnittelu

5.1 Tietoperusta

Pelien suunnittelussa on hyvä lähteä liikkeelle mekaniikoista. Jos aloittaa tarinalla, tulee nopeasti huomattua, että tämä rajoittaa suuresti, mitä voi tehdä. Valmis tarina estää tylsien ja rikkinäisten mekaniikoiden tiputtamisen, ja lopputulos on muodoton massa mekaniikoita, jota kukaan ei halua pelata. (Extra Credits 08.02.2013, 0:00–1:17 min.)

On myös erittäin helppoa kirjoittaa enemmän pelin tarinaa paperille verrattuna siihen, että kykeneekö luoda pelin tarinan näyttämiseen vaadittua taidetta. Tämän takia joko projekti juuttuu paikalleen tai tarinasta joutuu leikkaamaan isoja osia pois. (Extra Credits 08.02.2013, 1:54–2:11 min.)

Yksi tärkeä menetelmä, jolla parantaa pelin suunnittelua on tehdä ensimmäisestä pelistä minimum viable product', eli pienin totetuttamiskelpoinen tuote. Tällä tarkoitetaan peliä, jossa on vain tärkeimmät perusominaisuudet. Pelistä ei siis enää voi poistaa ominaisuuksia ilman koko pelin rikkomista. Tämän avulla pystyy löytämään, mitkä tehdyistä mekaniikoista ovat vetävimmät, ja mistä voi löytää reunatapauksia, jossa mekaniikat hajoavat. (Extra Credits 28.01.2015, 0:42–1:27 min.)

5.2 Toteutus

Kun aloitin suunnittelemaan peliäni, törmäsin tietoperustassa ensimmäisenä mainittuun ongelmaan. Aloitin tarinalla, pitäen mielessä vain suuntaa antavan idean haluamistani mekaniikoista. Tästä tuli ongelma, kun aloitin peliäni suunnittelemaan ja tuottamaan, jonka tajusin liian myöhässä.

Olin jo kuitenkin itseni saanut tähän tilanteeseen, ja nyt piti yrittää tehdä sillä parhaani. Olin onneksi tehnyt mekaniikoistani yksinkertaiset, keskittyen vain dialogiin ja yksinkertaiseen liikkumiseen, antaen pelin tarinan olla pelin suurin vetovoima.

Pelin suunnittelun alkupuolella oli mielessäni, että hahmo pystyisi kävelemään, juoksemaan ja hyppimään. Tarinan leikkauksien jälkeen kuitenkin tajusin, että hahmolla ei missään vaiheessa olisi tarvetta hypylle, tämän ominaisuuden ei sopisi pelin tarinaan tai hahmon luonteeseen. Juoksua en ollut ehtinyt toteuttaa, mutta tämä tuli myös pudotettua suunnitelmista samasta syystä. Täten hahmo tuntuisi hitaalta ja hieman joutilaalta, kun tämä vain kävelee paikasta toiseen ja seuraa tarinaa.

Näiden leikkauksien avulla pystyin myös vähentämään, ominaisuuksien määrää lopullisesta pelistä. Tällä siis toin pelin lähemmäksi pienimpään totetuttamiskelpoiseen tuoteeseen Tietenkin on aika vaikea arvioida, kuinka paljon dialogia tarvitaan peliin, jonka vetovoima perustuu tarinaan.

6 Pelin tarinan kirjoitus

6.1 Tietoperusta

Kun pelaaja pelaa peliä, on hänellä valtuus päättää kuinka nopeasti edetä pelissä. Vaikka ei ole mahdollista täysin ohjata rytmitystä jokaisen sanan ja sekunnin kohdalta, on pelin tekijöillä monta tapaa ohjata rytmitystä koko kokemuksen näkökulmasta.

Yksi tapa on tasapainottaa hektiset ja rauhalliset hetket tarinassa sekä pelissä, käyttäen eri toimintatiloja eri hetkiin. Kun tarina alkaa kuumenemaan, on pelin aika tuoda esille kamppailua, juoksua, tai vaikkapa ajastettu dialogi valinta. Kun on aika viilentää ja ottaa henkeä, voi pelaajalle antaa hetken aikaa sisäistää mitä hänen ympärillänsä on, ja rentoutua hitaan tekemisen parissa. (Snow 2015.)

Peleissä on myös pidettävä mielessä, että rytmitys ei päde pelkästään pelin tarinaan. Jokaisessa hetkessä ja toimessa on oma rytmitys, se voi olla vaikka kohtaus kahden hahmon kanssa, ottelu kaksinkamppailu pelissä, tai jopa kertalaukaus ampumapelissä. (Extra Credits 18.05.2018.)

Pelin rytmitys ei kuitenkaan ole ainoa asia mitä pelaaja ohjaa. Se on oleellisin asia minkä kanssa pitää pelissä työskennellä, mutta pelejä tehdessä helposti ilmestyy muita ongelmia ja mahdollisuuksia. Ensiksi tulee yksi isoimmista, mutta harvemmin puhutuista ongelmista, ludonarratiivinen dissonanssi.

Lyhyesti selitettynä, ludonarratiivisella dissonanssilla kuvataan sitä tilannetta, kun pelin tarinankerronta ja pelin mekaniikat eroavat huomattavasti toisistaan ja tulevat ristiriitaan keskenään. Tämä luo katkaisun pelaajan ja tarinan hahmojen välillä, ja hahmot voivat vaikuttaa tekopyhiltä. Tämä taas haittaa pelaajan samastumista tarinaan, ja hän voi kokonaan irrottaa itsensä siitä. Näin ollen pelaaja joko jatkaa pelaamista välittämättä tarinasta, tai lopettaa pelaamisen kokonaan. Yksi helpoimmista tavoista mitä on löydetty tämän estämiseen, on tehdä pelaajasta tarinan rikollinen. “Miksi tämä hahmo haluaa ryöstää pankin? Koska hän haluaa rahaa.” (Game Revo 2017.)

Yksi pelien suurimmista vahvuuksista tarinankerronnassa, ovat haarautuvat tarinat, joissa pelaaja voi päättää usean valinnan välillä. Se voi olla vain pieni muutos, joka muuttaa dialogia tai pieniä yksityiskohtia myöhemmin pelissä. Muutos voi olla myös keskikokoinen, joka muuttaa monen hahmon suhtautumista pelaajaa kohden, täten muuttaen pelaajan myöhempiä valintamahdollisuuksia pelissä. Muutos voi olla myös huomattavan suuri, mahdollistaa esimerkiksi valinnan kahden tai kolmen erilaisen tarinan välillä. (Boast 2019.)

Mikäli tekee haarautuvaa tarinaa, on hyvä aloittaa tekemällä yhden varsinaisen tarinan ääriviivat, ja sitten katsoa minne ja miten voi lisätä haarautumia. Tarinaa tehdessä täytyy myös pitää mielessä valinnan illuusio, jossa kaksi valintaa vie samaan tulokseen. Tällaiset valinnat ovat hyviä syventämään pelaajan tarinaan, mutta niitä on paras välttää isompien valintojen kanssa. Tarinoiden haarautumia ei kannata yrittää pakottaa sinne, minne ne eivät sovi. Tällöin on joko jätettävä haarautumat pois tai muuttaa tarinaa siten, että ne sopivat. Tämä on varsinkin totta, jos mielessä on yritys miellyttää kaikkia pelaajia. Ei ole mahdollista miellyttää kaikkia pelaajia, joten joku pelaajista jää ilman haluamaansa valintaa. (Boast 2019.) Omasta mielestäni on parempi pitää tarina, niin kuin se on ja tehdä tarinan haarautuma jossain muualla.

Tarinoissa, on usein valmiita rakenteita, joita ihmiset ovat käyttäneet tarinoissaan ajan alusta asti. Yksi näistä on kishotenketsu, suomennettuna esittely, kehitys, mutka ja loppu. Tässä tarinamuodossa länsimaissa usein vaadittu konflikti sivuutetaan, sen tilalle ilmestyy yllättävä mutka taikka kierre tarinan kohokohdassa, joka kontekstualisoi tätä edeltävän tarinan. (Kerr 2020.) Olin päättänyt käyttää tätä tyyliä omaa tarinaani varten.

6.2 Toteutus

Kun aloitin tarinaa suunnittelemaan, piti ensin valita sen struktuuri. Olisiko se rakenne suoranainen ilman yhtään tilaa pelaajan valinnoille, vai lähtisikö se haarautumaan moneen suuntaan? Päätin lopulta antaa tarinan tehdä haaraumia pelkästään dialogissa, jotka tulisivat takaisin tarinan keskeiseen runkoon. Täten pelaaja pääsee vaikuttamaan tarinaan, mutta sen runko pysyy yksinkertaisena.

Sen jälkeen oli päätettävä, minkälaiset mekaniikat haluaisin peliini. Olisin voinut yrittää tehdä taso hyppelyä tai tappelupeliä. Kuitenkin päädyin tekemään kaksiulotteisen kävelysimulaattorin lempipelini Night in the Woodsin inspiroimana. Täten minun ei myöskään tarvinnut tehdä suurta määrää taidetta hahmoille tai tasoille, joka mahdollistaa työmääräni hallittavuuden. Tämän avulla pääsin myös testaaman japanilaista kishotenketsu tarinarakennetta, joka ei vaadi konfliktia, toisin kuin suurin osa länsimaisista tarinoista.

Seuraavaksi oli mietittävä mitä tarina lopulta sisältää. Olin jo aikaisemmin päättänyt käyttää Jonne-hahmoa, jonka olin aikaisemmin tehnyt, pelin päähahmona. (Kuva 7.) Sen jälkeen jatkoin tarinaa kahdella uudella hahmolla. Hahmoiksi tuli Jonnen äiti, joka pelin

Kuva 7. Jonne-hahmosta animaatio, tehty ennen peliä.

alussa auttaa Jonnen surkean tilanteen selventämisessä pelaajalle, sekä Eeva, jonka ilmestyminen Jonnen elämään aloittaa tarinan ja suuntaa sen kulkua.

Lopulta oli ratkaistava se, miten tarina kulkee, eli tarinan esittely, kehitys, kierre ja loppu. Olin päättänyt tehdä Eevasta enkelin, joka on salassa lähtenyt jumalan seuraajaa etsimään, kun edellinen jumala oli hävinnyt. Tarinan alkutilanne siis esittelisi Jonnen, joka elää normaalia, joskin surkeaa elämäänsä. Tarina alkaa kehittymään, kun hän löytää Eevasta ystävän, ja sitten tarina tekee käänteen, kun Eeva paljastaa oikean muotonsa. Tarina tulisi loppuun yhdessä kolmesta mahdollisesta lopusta, riippuen siitä olisiko Jonne hyvä valinta jumalaksi, tai edes hyvä ystävä Eevalle.

Aloitin kirjoittamaan tarinaa sillä ajatuksella, että siitä tulisi suhteellisen pieni, jotta saisin sen tehtyä ajallaan. Tarina, joka kuitenkin oli pieni paperilla, oli toteutuksessa haastava tehdä annetussa ajassa. Varsinkin, koska opettelin vielä käyttämään uusia valitsemiani työkaluja. Tätä tarinaa täytyi leikata viiden päivän tarinasta kolmeen päivään, ja sitten lopulta kolmeen kohtaukseen yhden päivän sisällä.

Pelin päähahmo on Jonne, vanhempiensa kanssa asuva ja yksinäinen opiskelija. Pelin päivän alussa hän saa uuden henkilön elämäänsä. Tämä henkilö on Eeva, maan päälle saapunut kuolevaiseksi naamioitunut enkeli. Hän etsii uutta kandidaattia jumalan seuraajalle, kun aikaisemmin viran omistanut henkilö on hävinnyt. Pelin aikana Jonne tietämättään todistaa Eevalle, että hänestä voisi olla seuraavaksi jumalaksi, tai hyväksi kaveriksi samalla tavoin yksinäiselle enkelille.

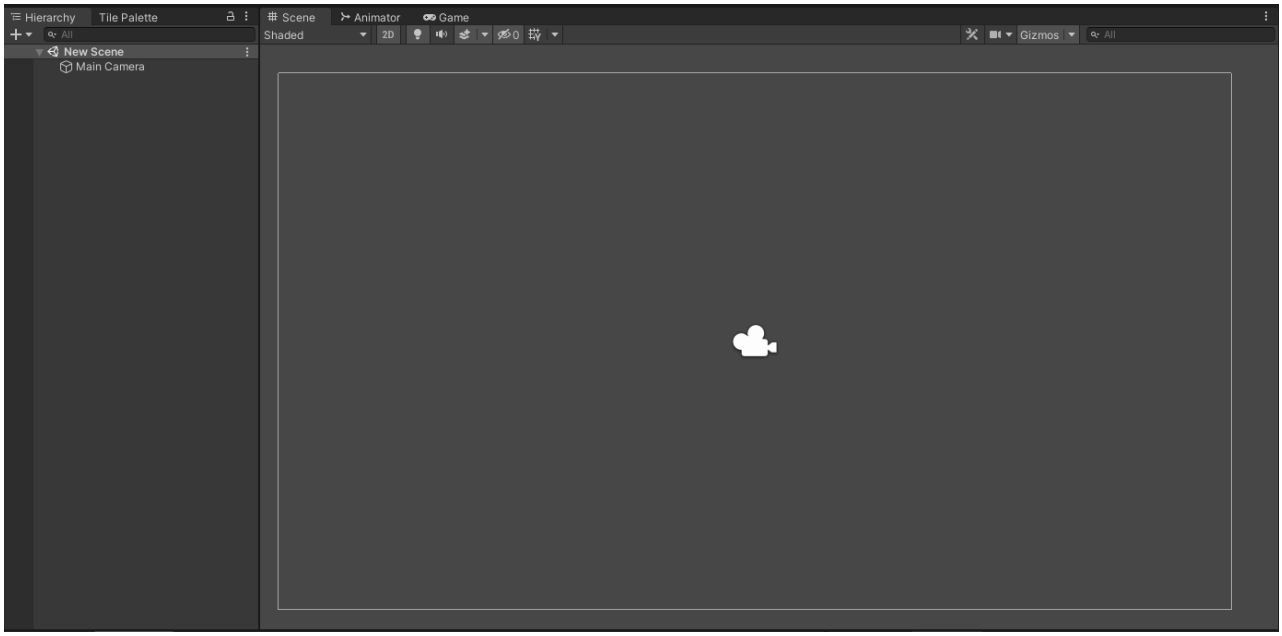
Jonne aloittaa kotonaan, josta hän pian siirtyy torille, jossa hän tapaa Eevan. Sieltä juttelemisen jälkeen he siirtyvät läheiseen kahvilakonditoriaan aamupalalle. Tämän jälkeen he lopulta siirtyvät lähelle vanhaa ilmatorjuntatykkiä kaupungin laidalla katsomaan kaupungin menoa. Täällä juttelun jälkeen peli sitten päättyy, jolloin pelaajan valituista dialogivaihtoehdoista lasketaan pistemäärä, jolla päätetään pelaajan saama loppu hyvän, keskivertoisen, ja huonon lopun välillä.

Hyvässä lopussa Eeva paljastaa Jonnelle oikean muotonsa, ja nostattaa hänet uuden jumalan paikalle. Keskivertoisessa lopussa Eeva ei paljasta oikeaa muotoansa Jonnelle, mutta hän päättää pysyä tämän ystävänä, kun hän jatkaa jumalan seuraajan etsintää. Huonossa lopussa Eeva on saanut tarpeekseen Jonnesta, ja ei halua olla hänen kanssaan tekemisissä enempää. Jonne täten palaa kotiin, ja hänen elämänsä jatkuu sen entiseen malliin.

7 Pelin toteuttaminen

7.1 Unityn käytön perusteet

Nyt kun kaikki valmistelut oli tehty ja tarina kirjoitettu, oli aika aloittaa tekemään itse peliä. Unity oli itselläni jo asennettuna, joten piti vain aloittaa uusi Unity-projekti. Täten pääsemme Unityn käyttöliittymään, josta pääsemme käsiksi kaikkeen tarvitsemaamme.



Kuva 8. Esimerkki Unityn käyttöliittymästä.

Kuva 8 antaa esimerkin osasta juuri avattua käyttöliittymää. Oikealla näkyy tällä hetkellä melkein tyhjä Unity-näkymä. Vasemmalla näkyy tämän näkymän hierarkia, jossa on yksi objekti nimeltä ’Main Camera’. Kaikki pelin sisältö, eli hahmot, esineet, taustat, ja skriptit, menevät näihin näkymiin. (Unity Documentation 2023a). Yksinkertaiset pelit yleensä toimivat vain yksittäisellä näkymällä, mutta tiedän että tulen itse tarvitsemaan yhden näkymän pelin aloitusnäytölle, yhden näkymän kullekin eri alueelle, mikä on siis neljä yhteensä, ja yhden näkymän, joka sisältää pelin loppukohtaukset. Tämä tuo näkymien kokonaismäärän kuuteen.

Nämä näkymät täytetään eri Game Objecteilla, eli objekteilla, jotka muodostavat hahmot, taustat, ja toimivat myös skriptien syöttöinä ja ulostuloina. Näitä eri käyttötarkoituksia varten nämä objektit vaativat komponentteja, joihin kuuluu mm. valolähteet, törmäyslaatikot, kamerat, ja tehdyt skriptit (Unity Documentation 2023b). Esimerkiksi, aiemmin mainittu ’Main Camera’-objekti vaatii kamerana toimiakseen Camera-komponentin. Se myös sisältää Transform-komponentin, jonka avulla sen sijaintia voidaan seurata ja muuntaa X- ja Y-koordinaatistosysteemin avulla. Unityssa on myös Z-akseli, mutta pelini ei käytä sitä koska se on vain kaksiulotteinen.

7.2 Pelaajahahmon luonti

Seuraavaksi aloitin pelaajahahmon luonnin. Ensimmäinen tehtävä oli luoda hahmon ulkonäkö ja animaatiot. Kuten aiemmin mainitsin, minulla oli valmis hahmo, ja tälle olin jo tehnyt seisomisanimaation. Annoin näille pienen ehostuksen peliä varten, ja tämän jälkeen loin animaatiot istumiselle ja kävelylle, koska tiesin näiden olevan pelin keskeisimmät tapahtumat pelaajahahmon suhteen.

Animaatioiden valmistuttua oli aika luoda pelaajahahmo Unityssa. Aloitin tekemällä objektin, joka sisältää Transform-komponentin hahmon paikan seuraamista ja koon määrittämistä varten. Seuraavaksi lisäsin Sprite Renderer-komponentin, jolla lisäsin objektiin tekemäni hahmon taiteen, ja Animator-komponentin, jolla animoin tämän Sprite Rendererin. Lopuksi lisäsin Rigid Body 2D-komponentin, jolla lasketaan ja sovelletaan hahmoon vaikuttavat fyysiset voimat, ja Box Collider 2D-komponentin, joka muitten Collider-komponenttien avulla estää hahmoa menemästä lattian taikka seinien läpi.

Kun hahmon Unity-puoli oli tehty, siirryin hahmon skriptin koodaamiseen. Unityn skriptit kirjoitetaan C#-kielellä, mikä on yksinkertainen yleiskäyttöön tarkoitettu oliosuuntautunut kieli (Ecma-International 2022). Unity on tietenkin lisännyt omia vaatimuksiaan C#:iin. Näistä ensimmäinen, joka näkyy uutta C#-tiedosta tehtäessä ovat Start- ja Update-funktiot. Start-funktio suoritetaan kerran, kun objekti, johon skripti on liitetty, ladataan ensimmäistä kertaa (Unity Documentation 2023c). Update-funktio vuorostaan suoritetaan joka kerta kun objekti on ladattu ja pelin kuva päivitetään, yleensä 30 tai 60 kertaa sekunnissa (Unity Documentation 2023d).

Nämä skriptin palat tulevat käyttöön pelaajahahmon kanssa. Start-funktion avulla voidaan napata pelaajahahmosta vaadittavat komponentit, joita manipuloidaan skriptin muilla osilla. Updatessa voidaan aloittaa tämän manipulointi. Ensimmäisenä voidaan Updaten pyöriessä tarkistaa, jos jompikumpi hahmon liikkumisnäppäimistä on painettu alas. Jos on, lähetetään hahmon Rigid Body 2D-komponentille arvo, jolla hahmoa työnnetään annettuun suuntaan, lisäten nopeutta tiettyyn pisteeseen asti kitkan takia. Jos ei ole yhtään nappia painettuna alas, hahmo hidastuu ja lopulta pysähtyy itsestään kitkan ansiosta. Samalle tarkistetaan hahmon nettonopeus, jonka avulle käsketään Animator-komponenttia animoimaan hahmon liike, kävellen vasemmalle ja oikealle riippuen siitä, onko tällä nopeutta kummassakaan suunnassa.

Tästä valmiista hahmosta voidaan tehdä Prefabi. Unityn Prefab-systeemillä varmistetaan, että jokaisen pelin näkymän versio pelaajahahmosta on sama, jotta erilaisuuksien takia ei ilmesty ongelmia. (Unity Documentation 2023e).

7.3 Unity-näkymät

Seuraavaksi tarvitaan alue, jossa pelaaja voi pelata tätä peliä, ja olla vuorovaikutuksessa tätä ympäröivän maailman kanssa. Tähän käytetään Unityn Näkymiä, kuten mainitsin kohdassa 7.1. Ensin oli luotava tähän liittyvä taide, sisältäen pelaajahahmon makuuhuoneen, eteisen, ja keittiön, jossa tämän äiti istuu tämän näkymän välianimaatiota varten. Nämä kaikki tehtiin mittakaavaan, joka määräytyi niiden koosta verrattuna pelaajahahmoon.

Loin näkymässä objektin, johon lisäsin Sprite Renderer-komponentin, jossa on nämä aiemmin mainitut huoneet yhtenä kuvana, sekä Box Collider 2D-komponentin, jolla rajataan alueen lattia. Tämä törmää pelaajan Collider-komponentin kanssa, ja täten estää sitä tippumasta lattian läpi. Seuraavaksi lisäsin kaksi korkeaa ja ohutta objektia, jotka vaativat vain Box Collider 2D-komponentit. Näiden avulla määritin oikean- ja vasemmanpuoleiset seinät, jotta hahmo ei kävele pelialueelta ulos.

Samalla myös loin skriptin pelaajaa seuraavalle kameralle, jotta tämä seuraa pelaajaa näyttämättä mitä pelialueen ulkopuolella on. Tämä tehdään luomalla muuttujat sisältäen enimmäis- ja vähimmäisarvot kameran X- ja Y-koordinaateille. Nämä muuttujat voidaan asettaa kunkin pelin näkymän koon mukaan.

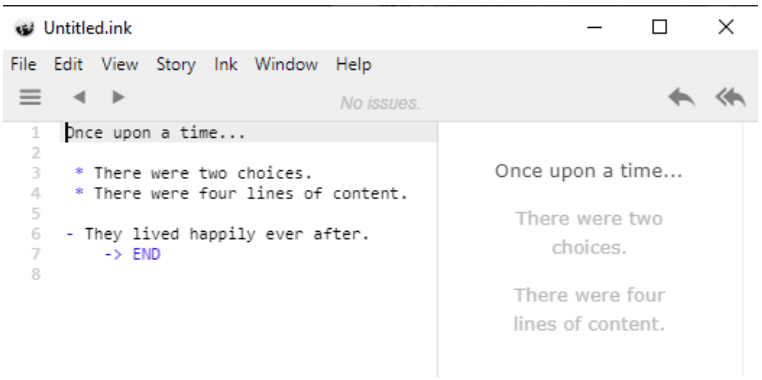
Seuraavaksi loin objektin, joka toimii ovena ulos asunnosta. Tässä objektissa on Box Collider 2D-komponentti ja skripti. Tässä Box Collider-komponentissa on asetettu boolean ’isTrigger’ todeksi, mikä tarkoittaa, että tämä Box Collider ei enää estä pelaajan liikkumista sen läpi. Sen sijaan sitä voidaan käyttää skriptissä tapana tarkistaa, jos pelaaja on tämän objektin sisältämällä alueella. Tämä tehdään kutsumalla skriptissä Collider.OnTriggerStay-funktio, joka palauttaa pelaajahahmo-objektin, jos pelaaja on tämän alueen sisällä, ja vain tyhjää, jos se ei ole (Unity Documentation 2023f). Kun tarkistus antaa pelaajan komponentin ja pelaaja painaa näppäintä käyttääkseen ovea, kutsuu tämä SceneManager.LoadScene-funktion, joka vaatii näkymän nimen taikka indeksin (Unity Documentation 2023g).

Pelin jokaisella näkymällä on indeksi, joka on numero nollan ja pelissä käytettyjen näkymien määrän miinus yksi välillä. Näkymien indeksit voidaan järjestää haluttuun järjestykseen, ja täten lisäämällä yhden nykyisen näkymän indeksiin saada pelin seuraava näkymä. Tämän luvun syöttämällä SceneManager.LoadScene-funktioon ladataan haluttu näkymä. Tämä keino toimii vain, jos pelaaja menee näkymien läpi lineaarisesti. Pelini kuitenkin on suoraviivainen, joten tämä keino sopii minulle.

7.4 Dialogisysteemi

Seuraavaksi oli aika luoda pelin keskeisin ominaisuus, dialogi. Kuten mainitsin kohdassa 4.2, olin päättänyt käyttää Inkle Studiosin ink-dialogisysteemiä Inky ohjelman ja tätä systeemiä tukevan Unity-lisäosan avulla. Ensin oli siis hankittava nämä asiat.

Inkyn löytää Inkle Studiosin omalta sivulta. Sen tarkoitus on toimia tekstieditorina, joka mahdollistaa Ink merkintäkielen käytön haarautuvien tarinoiden luomisessa. Tämän avulla pystyy luomaan dialogin, joka sisältää haarautumispisteet sekä merkit, joilla voi mm. vaihtaa kuvia ja nimiä, jotka näkyvät dialogikäyttöliittymässä puhujan paikalla. Tämän jälkeen hankin vielä Unityyn edellä mainitun lisäosan, joka automaattisesti kääntää Inkyn ink-tiedostot muotoon, jonka Unity pystyy lukemaan, sekä lisää C#-funktioita, joiden avulla näitä pystyy lukemaan ja käyttämään pelin aikana.



Kuva 9. Esimerkki Inkyn käyttöliittymästä.

Nämä hankittuani oli aika aloittaa itse dialogin kirjoittaminen. Dialogia kirjoittaessani lisäsin merkinnät, jotka kertovat puhujan nimen ja sisältävät halutun puhujan muotokuvan nimen. Tietyin paikoin nämä merkinnät myös antavat pisteitä riippuen siitä, jos valitaan haarautumassa hyvä, keskivertoinen tai huono valinta. Nämä merkinnät tehdään luettaviksi Unityn skripteissä dialogin kanssa, jolloin voin lukea näiden sisältämät tiedot ja muuttaa asioita dialogikäyttöliittymässä sekä kulissien takana niiden mukaan.

Sitten kun oli saatu valmis ja ohjelman luettavaksi parsittu dialogi Unityyn, oli aika pistää tästä tuleva tieto näkyville. Ensin oli luotava Unityn näkymässä dialogikäyttöliittymä, joka sisältää tekstilaatikon, puhujan kuvan ja tekstin sisältävät laatikot, sekä kolme painiketta kolmea valintaa varten. Seuraavaksi oli tehtävä skripti, joka aloittaisi tämän dialogin. Tätä varten pystyin käyttämään kohdassa 7.3 luotua ovea pohjana, muuttamalla tuloksen näkymän vaihdosta dialogin aloittamiseen objektiin liitetyn dialogitiedoston kanssa.

Ennen dialogikäyttöliittymää täytyi vielä tehdä järjestelmä, joka pitää kirjaa dialogista saaduista pisteistä oikean loppuanimaation valintaa varten. Tätä varten löysin Unityn Scriptable Object-ominaisuuden. Tämän avulla pystyy tekemään skriptejä, joita ei tarvitse liittää mihinkään objektiin näkymässä, kunhan sille luo nimetyn instanssin pelin tiedostoissa. Näkymässä olevat skriptit pystyvät

vaikuttamaan tähän skriptiin niin kuin mihin tahansa muuhun, mutta koska tämä ei ole koskaan näkymässä, sen sisältämä tieto pysyy samana näkymien välissä. (Unity Documentation 2023h). Täten tallentamalla pisteet Scriptable Objectiin, pystyn varmistamaan, että pelaajan keräämät pisteet eivät häviä missään vaiheessa.

Saatuani nämä tehdyksi oli aika aloittaa tätä kokoelmaa ohjaavan ja dialogitiedoston lukevan skriptin kirjoittaminen. Aloitin tämän skriptin kirjoittamisen luomalla kokoelman muuttujia, sisältäen listan, johon hain painikkeet korkeimmasta alimpaan, sekä erilliset muuttujat dialogin tekstikenttää, puhujan kuvaa, ja puhujan nimen tekstikenttää varten. Kun dialogitiedostosta luetaan sen tieto, voidaan tämä syöttää oikeaan paikkaan näkymässä. Käyttämällä listaa painikkeita varten, voidaan ladata vain ne painikkeet, joihin tulee sisältöä valinnan aikana. Ja kun dialogitiedostosta tulee pisteitä, ne tallennetaan aiemmin mainittuun Scriptable Objectiin. Näin on tehty toimiva dialogisysteemi.

Päätin myös tätä tehdessäni siirtää näkymien välissä liikkumisen välianimaatiota seuraavan dialogin loppuun. Muutin oviskriptin siten, että kun se on aktivoitu näkymässä, se tarkistelee dialogiskriptiltä, onko sillä dialogia päällä. Mikäli dialogi ei ole päällä, voidaan vaihtaa näkymää. Kun näkymä ladataan, tämän skriptin objekti on deaktivoitu, ja pystyn aktivoimaan sen vasta kun kohtauksen jälkeinen loppudialogi on jo käynnistynyt. Näin tämä skripti odottaa kyseisen dialogin loppua ennen vaihtoa.

7.5 Välianimaatiot

Seuraavaksi tarvitsin nämä aiemmin mainitut välianimaatiot. Välianimaatioissa pelaaja ei pysty itse kontrolloimaan pelaajahahmoa, joka tekee tiettyjä ennalta määrättyjä asioita. Oman pelini kohdalla välianimaatioiden pääasiallinen tarkoitus on asettaa pelaajahahmo oikeaan kohtaan, kun tämä istahtaa tuoleille tai muuten aloittaa tärkeän dialogin.

Unity hoitaa välianimaatiot käyttäen Playable Director-komponenttia, johon sijoitetaan timeline-ominaisuus. Nämä timeline-ominaisuudet sisältävät aikajanan, johon lisätään ratoja. Jokaisella radalla on oma tapahtuma, joka animoi, aktivoi tai muuten ohjaa objekteja ja niiden komponentteja. Kun timeline on lisätty näkymään, näkymän objektit voidaan asettaa timelinen radoille, ja näiltä objekteilta halutut toimet taas asetetaan tiettyihin kohtiin kullakin radalla. (Unity Manual 2023).

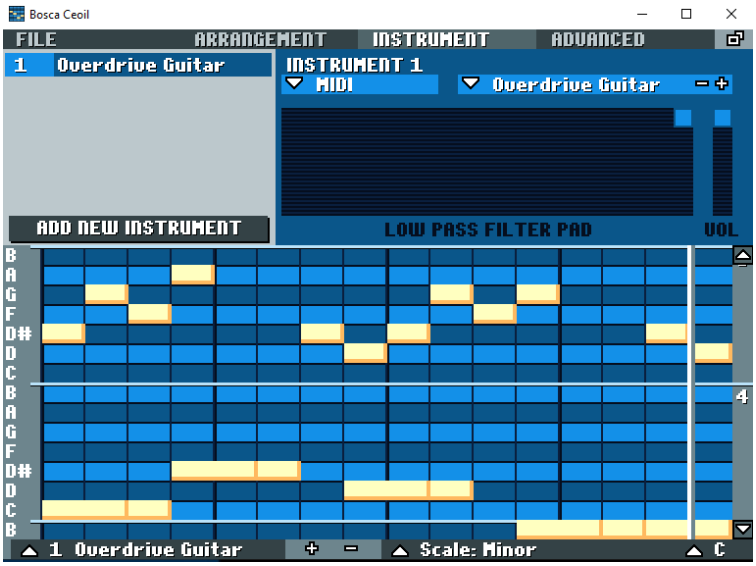
Aktivoidakseni välikohtauksen käytän taas kohdassa 7.3 luotua ja kohdassa 7.4 sovellettua koodin tyyliä, jossa Collider-komponentti asetettuna triggeriksi toimii tutkana pelaajan etäisyydestä kohteeseen. Tässä vaiheessa tajusin myös, että tarvitsisin tavan kertoa pelaajalle, jos tämä on satunnaisen dialogin taikka tarinaa jatkavan välianimaation aktivoijan edessä. Tätä varten lisäsin tähän koodiin pätkän, joka tuo esille indikaattoriobjektin, jos pelaaja pystyy siinä kohdassa aktivoimaan tämän dialogin taikka välikohtauksen. Tein myös erimuotoiset indikaattorit välianimaatioille ja dialogeille, jotta pelaajan ei tarvitse epäröidä, onko jokin asia vain lisää tekstiä, jonka hän voi lukea ennen tarinassa etenemistä, vai juuri tämän tarinassa etenemisen kohta.

Esimerkkinä välianimaatiosta on kohtaus pelaajahahmon kotona, jossa pelaajahahmo istuu tuolille puhumaan äitinsä kanssa. Tämän animaation objektiin on lisätty lapsiobjekti, jossa on vain yksi kuvakomponentti. Tämä komponentti sisältää mustan laatikon, joka peittää koko näytön. Välianimaation alussa musta laatikko tulee esille, pelaajan paikka asetetaan lyhyen etäisyyden päähän tuolista, ja kamera asetetaan ennalta määritettyyn kohtaan. Seuraavaksi musta laatikko haalistuu pois. Yksi rata sisältää hahmon liikkumisen tuolin eteen, toinen taas animoi tämän ensin kävelemään ja sitten istumaan tuolille. Yksi rata aktivoi välianimaation loputtua objektin, joka sisältää seuraavan dialogin käynnistävän skriptin. Toinen rata tekee saman objektille, jonka sisältämä skripti siirtää hahmon seuraavaan näkymään, kun dialogi päättyy.

7.6 Musiikki ja ääniefektit

Viimeinen asia, jonka tarvitsen pelissä, on jonkinlaista ääntä. Vaikka ääniä ei tarvita pelin toimintoja varten, on täysin hiljainen peli omasta mielestäni lähes pelikelvoton.

Musiikin tekemiseen tarvitsin ohjelman. Ottaen huomioon, että minulla ei ole tässä vaiheessa ollut minkäänlaista kokemusta taikka koulutusta musiikin teossa, haluamani ohjelma pitäisi olla helppo käyttää ja sisältää työkaluja, jotka helpottavat työn tekoa osaamattomalle. Olin vuosieni aikana netissä löytänyt ohjelman juuri tätä tarkoitusta varten nimeltään BoscaCeoil.



Kuva 10. Bosca Ceoilin käyttöliittymä

Tämän ohjelman avulla sain tehtyä seitsemän musiikkikappaletta peliäni varten. Neljä näistä on lyhyitä pätkiä, jotka on tehty soimaan alusta loppuun uudelleen ja uudelleen saumattomasti. Nämä kappaleet soivat pelin aikana, yksi kussakin pelin neljässä pelattavassa näkymässä. Kolme viimeistä ovat pidempiä sävelmiä, joista kukin soi pelin eri loppuanimaatioiden kohdalla. Unityssa nämä saadaan soimaan käyttämällä AudioSource-komponenttia, johon liitetään haluttu musiikki. (Unity Documentation 2023i.)

Lopuksi halusin vielä lisätä ainakin ääniefektin pelaajahahmon askelille. Näihin ääniin olin löytänyt ohjelman netistä nimeltä BFXR. Tämä ohjelma toimii muuntamalla eri aaltomuotoja monin eri tavoin. Tämän avulla sain tehtyä pari askeläänttä, joita voin toistaa vaihdellen hahmon kävelyanimaatiossa. Unityssa sain liitettyä pelaajahahmon kävelyanimaatioon kytkimet, jotka kytkeytyessään lähettää pelaajan skriptille käskyn toistaa nämä äänet tähän objektiin lisätyn AudioSource-komponentin avulla. Näin ollen pelaajan kävellessä kuuluu askelääni.

8 Loppupohdinta

Kirjoitan tätä lukua opinnäytetyöseminaarin jälkeen, nyt kun on melkein kaikki asiat tehty ja mieli selkeä tämän projektin stresseistä ja paineista.

Peli tuli lopulta tehtyä, joskin vuoden myöhässä. Olin rajannut kaikesta välittämättä aika suuren projektin yhdelle henkilölle. Varsinkin, kun otetaan huomioon samalla aikaa kohtaamani mielenterveysongelmat. Pandemian aikainen kotona opiskelu ei toiminut hyvin, ja tämän aikana kasvanut stressi tuli romahtaen päälle lopputyöni aikana. Työn tempo hidastui suuresti, ja koko projekti venähti niin pitkäksi, kun laki salli.

Peli kuitenkin tuli tehtyä, ja tämä raportti kirjoitettua. Näistä jälkimmäinen oli huomattavasti vaikeampi itselleni, sillä en ole kirjoittajaksi millään tavalla suuntautunut. Tässä ei myöskään auttanut se, että en saanut tätä kirjoittaa englanniksi, kun asiasta kysyin. Englanti kun on kieli, jossa olen paremmin tottunut pitkään kirjoittamiseen. Isot kiitokset haluan antaa oikolukijoilleni perheessä ja paikallisella ohjaamolla, jonka ansiosta sain aikaan luettavaa tekstiä.

Pelinteon aikana tuli kuitenkin paljon opittua. Miten välianimaatiot ja äänet toimivat Unityssa, miten luoda musiikkia ja ääniefektejä, ja tärkeimmillään miten tärkeää kunnon suunnittelu ja tämän suunnittelun seuraaminen on. En ole järjestelmällisin ihminen, ja oman diagnosoidun ADHD:n ja mahdollisen autismin ansiosta työni tulee suoritettua pienissä purkauksissa, joka teki minkäänlaisen puolipäisen suunnitelman seuraamisesta lähes mahdotonta.

Sain kuitenkin tehtyä haluamani pelin, kuten näette, ja pääsette kokemaan liitteistä löytyvässä Itch.io linkissä. Peliä voisi hioa ja parannella siellä sun täällä, ja mahdollinen uudelleen alusta luonti voisi tehdä sen pelin, jonka alun perin suunnittelin tämän projektin alussa.

Eli, mitä olen oppinut?
Yksi, projektit on suunniteltava kunnolla alusta loppuun, sillä hyvin suunniteltu projekti onnistuu puolet nopeammin kokoaikaiseen hatusta vetoon verrattuna.
Kaksi, mielenterveys on merkittävä osa työkykyä, ja sen kanssa on hyvä olla yhtä tarkkaavainen kuin fyysisen terveyden kannalta.
Kolme, on hyvä pyytää apua muilta, ei väliä kuinka pieni tai suuri asia.
Ja Neljä, videopelin luonnin perusteet tarinaan keskittyen, käyttäen Unitya, Ink-dialogisysteemiä, Asepritea, Bosca Ceoilia, ja BFXR:ää.

Lähteet

AlanBeckerTutorials 17.02.2015. 5. Follow Through & Overlapping Action - 12 Principles of Animation. Video. Katsottavissa: https://www.youtube.com/watch?v=4OxphYV8W3E. Katsottu: 17.10.2022.

Baxter, D. Clark, S. 2022. Best free music-making software in 2022. Tech Radar. Luettavissa: https://www.techradar.com/best/free-music-making-software. Luettu: 11.08.2022.

Boast J. 23.04.2019. Game Developer. The Branches of a story – How to give players options in a narrative? Luettavissa: https://www.gamedeveloper.com/design/the-branches-of-a-story-how-to-give-players-options-in-a-narrative-. Luettu: 21.2.2022.

Brackeys. 21.10.2018. Video. Basic Principles of Game Design. Katsottavissa: https://www.youtube.com/watch?v=G8AT01tuyrk. Katsottu: 2.5.2022.

Dustin, T. 27.09.2021. GameDesigning Game Design Principles. Luettavissa: https://www.gamedesigning.org/learn/game-design-principles/. Luettu: 2.5.2022.

Ecma-International. 2022. C# Language Specification. Haettavissa: https://www.ecma-international.org/wp-content/uploads/ECMA-334_6th_edition_june_2022.pdf. Luettu: 20.03.2023.

Extra Credits. 08.02.2013. Video. How to Start Your Game Narrative – Design Mechanics First. Katsottavissa: https://www.youtube.com/watch?v=22HoViH4vOU. Katsottu: 3.5.2022.

Extra Credits. 14.01.2015. Video. Making Your First Game: Basics – How To Start Your Game Development. Katsottavissa: https://www.youtube.com/watch?v=z06QR-tz1_o. Katsottu: 2.5.2022.

Extra Credits. 18.05.2018. Video. Pacing - How Games Keep Things Exciting. Katsottavissa: https://youtu.be/5LScL4CWe5E. Katsottu 21.2.2022.

Extra Credits. 28.01.2015. Video. Making Your First Game: Minimum Viable Product - Scope Small, Start Right Katsottavissa: https://youtu.be/UvCri1tqIxQ. Katsottu: 27.10.2022.

Greer, B. 28.12.2019. Video. What Program to use for Pixel Art? (Paid and Free Software) Katsottavissa: https://www.youtube.com/watch?v=90BghUX7SD0. Katsottu: 1.8.2022.

Igara Studio, 22.02.2016. Aseprite. Löydettävissä: https://store.steampowered.com/app/431730/Aseprite. Löydetty: 17.10.2022.

Kerr J. 08.07.2020. Kishōtenketsu: Exploring The Four Act Story Structure. Luettavissa https://artofnarrative.com/2020/07/08/kishotenketsu-exploring-the-four-act-story-structure/. Luettu: 20.10.2022.

Kotaki, G. 12.07.2012. Introduction to Pixel Art for Games. Raywenderlich.com Luettavissa: https://www.raywenderlich.com/2888-introduction-to-pixel-art-for-games. Luettu: 12.07.2022.

Loveridge, S. 23.12.2018. The most crucial part of video-game development explained - and how it powered Fortnite's runaway success. Luettavissa: https://www.gamesradar.com/what-is-a-game-engine-and-what-does-it-do/. Luettu 24.11.2022.

Pixel Overload 1.1.2021. How to choose good Colour Palettes (Pixel Art Tutorial). Video. Katsottavissa: https://youtu.be/uUdMb8Bb2II. Katsottu: 04.08.2022.

Polack, A. 10.10.2017. Hello to you, and a most wonderful day to you - -. Twitter-viesti @Adigun-Polack. Luettavissa: https://twitter.com/adigunpolack/status/917804769405792257. Löydetty: 27.10.2022.

Ritchie, J. 07.19.2017. The 12 Principles of Animation. Luettavissa: https://idearocketanimation.com/13721-12-principles-of-animation-gifs/. Luettu: 20.8.2022.

Snow N. 02.09.2015. Wordpress Pace Yourself: How Developers Control Pacing in a Video Game. Luettavissa: https://atimeforgames.wordpress.com/2015/09/02/pace-yourself-how-developers-control-pacing-in-a-video-game/. Luettu 8.2.2022.

Trainor-Fogleme & E. Evercast. 30.09.2021. Unity vs Unreal Engine: Game engine comparison guide for 2021 Luettavissa: https://www.evercast.us/blog/unity-vs-unreal-engine. Luettu: 1.8.2022.

Unity Documentation. 2023a. Scenes. Luettavissa: https://docs.unity3d.com/Manual/CreatingScenes.html. Luettu: 13.03.2023.

Unity Documentation. 2023b. GameObjects. Luettavissa: https://docs.unity3d.com/Manual/GameObjects.html. Luettu: 13.03.2023.

Unity Documentation. 2023c. Monobehaviour.Start(). Luettavissa: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html. Luettu: 20.03.2023.

Unity Documentation. 2023d. Monobehaviour.Update(). Luettavissa: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html. Luettu: 20.03.2023.

Unity Documentation. 2023e. Prefabs. Luettavissa: https://docs.unity3d.com/Manual/Prefabs.html. Luettu: 24.03.2023.

Unity Documentation. 2023f. Collider.OnTriggerStay. Luettavissa: https://docs.unity3d.com/ScriptReference/Collider.OnTriggerStay.html. Luettu: 24.03.2023.

Unity Documentation. 2023g. SceneManager.LoadScene. Luettavissa: https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.LoadScene.html. Luettu: 24.03.2023.

Unity Documentation. 2023h. ScriptableObject. Luettavissa: https://docs.unity3d.com/Manual/class-ScriptableObject.html. Luettu: 03.04.2023.

Unity Documentation. 2023i. AudioSource. Luettavissa: https://docs.unity3d.com/ScriptReference/AudioSource.html. Luettu: 10.04.2023.

Unity Manual. 2023. Timeline overview. Luettavissa: https://docs.unity3d.com/Packages/com.unity.timeline@1.8/manual/tl_about.html. Luettu: 03.04.2023.

Liitteet

Liite 1. Linkit pelin lataamiseen.

Tässä siis kaksi linkkiä, ensimmäinen on itch.io mistä pelin saa ladattua valmiina pelattavassa muodossa, ja toinen on Github, missä pelin lähdekoodi on luettavissa ja ladattavissa omaan käyttöön.

Itch.io: https://jonnemanni.itch.io/tassulan-tarina

Github: https://github.com/Jonnemanni/BearGame