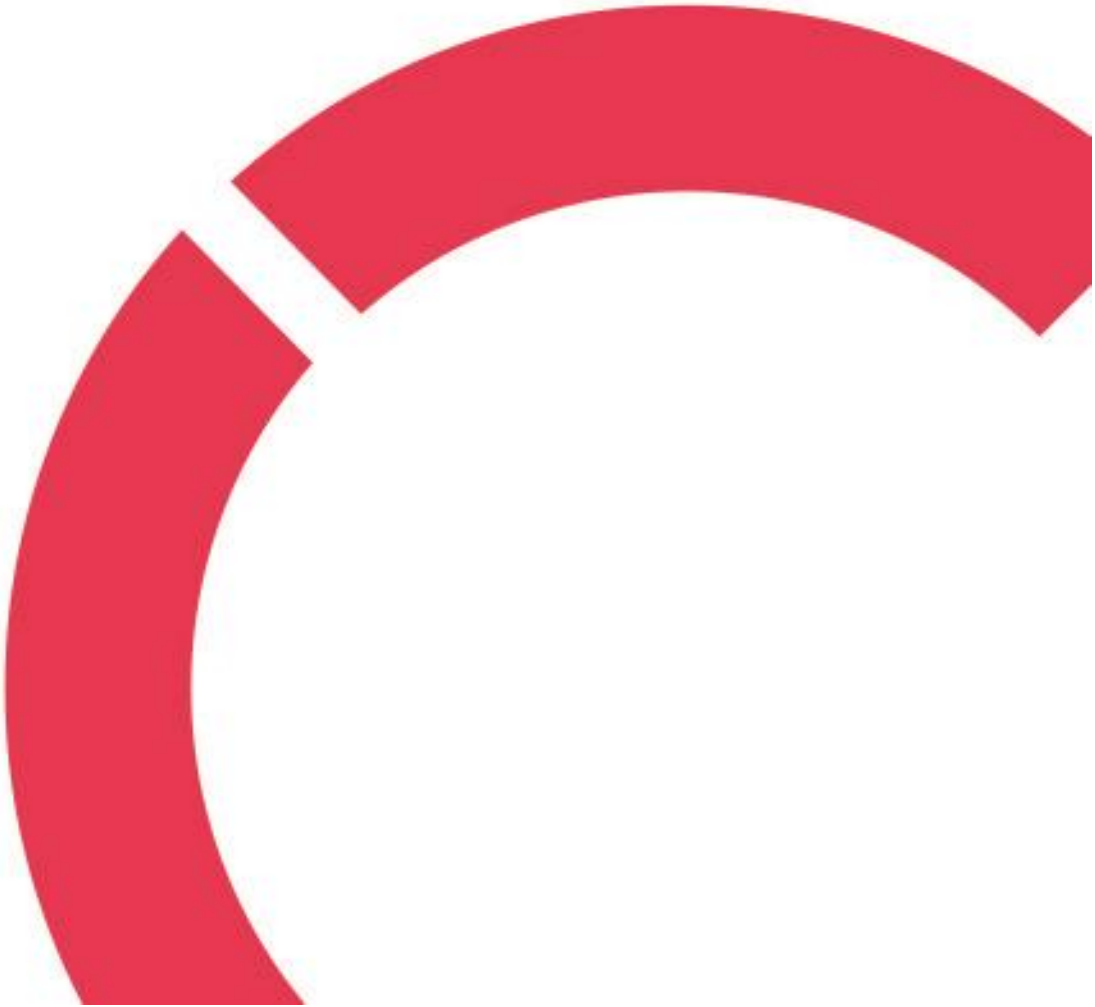
Markus Nummi

INDIE PELIKEHITYS

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikan koulutus
Syyskuu 2024



TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Syyskuu 2024	**Tekijä/tekijät** Markus Nummi
Koulutus Tieto- ja viestintätekniikan koulutus		☒ AMK ☐ YAMK
Työn nimi INDIE PELIKEHITYS		
Työn ohjaaja Sari Lipsanen		**Sivumäärä** 50 + 4
Työelämäohjaaja		

Opinnäytetyössä käsiteltiin indie-pelikehityksen erityispiirteitä ja haasteita, joita kohtaavat erityisesti pienet tiimit ja itsenäiset kehittäjät. Tutkimuksen kohteena oli demoprojekti, jonka kehitystä seurattiin ja analysoitiin kehityspäiväkirjan avulla. Tarkoituksena oli selvittää, kuinka paljon aikaa ja taloudellisia resursseja tarvitaan indie-pelin kehittämiseen alfaversiosta julkaisukelpoiseksi tuotteeksi. Opinnäytetyössä arvioitiin myös, miten realistista on saavuttaa kaupallista menestystä indie-peliprojektilla, kun otetaan huomioon pelimarkkinoiden nykytilanne ja vertailukohtina käytetyt samankaltaiset pelit.

Opinnäytetyön tulokset osoittivat, että indie-pelikehitykseen liittyy merkittäviä haasteita, kuten rajalliset resurssit, pitkä kehitysaika ja vaikeus erottua kilpailijoiden joukosta. Pienen budjetin ja ajan puitteissa työskentely johti siihen, että pelin markkinoille saattaminen vaati huomattavia investointeja kehitykseen ja markkinointiin. Vaikka teknisesti laadukkaan pelin kehittäminen oli mahdollista, kaupallisen menestyksen saavuttaminen osoittautui haastavaksi.

Johtopäätöksenä todettiin, että indie-pelikehitystä kannattaa lähestyä ensisijaisesti intohimon ja harrastuksen näkökulmasta, sillä taloudellisesti kannattavan lopputuloksen saavuttaminen edellytti huomattavaa panostusta ja markkinointiosaamista. Työ tarjosi kattavan kuvan indie-pelikehityksen haasteista ja antoi arvokasta tietoa niille, jotka harkitsevat omaa peliprojektiaan.

Asiasanat

3D-pelit, pelikehitys, Unity-pelimoottori

ABSTRACT

Centria University of Applied Sciences	Date	Author
	September 2024	Markus Nummi
Degree programme Information and Communications Technology		
Name of thesis INDIE GAME DEVELOPMENT		
Centria supervisor Sari Lipsanen		**Pages** 50 + 4
Instructor representing commissioning institution or company		

The thesis addressed the unique aspects and challenges of indie game development, particularly those faced by small teams and independent developers. The research focused on a demo project, which development was tracked and analyzed through a development diary. The purpose was to determine how much time and financial resources are required to develop an indie game from an alpha version to a market-ready product. The thesis also evaluated how realistic it is to achieve commercial success with an indie game project, considering the current state of the gaming market and using similar games as benchmarks.

The results of the thesis indicated that indie game development involves significant challenges, such as limited resources, long development time, and difficulty in standing out among competitors. Working within a small budget and limited time led to the realization that bringing a game to market requires substantial investment in both development and marketing. Although developing a technically high-quality game is possible, achieving commercial success proved to be challenging.

In conclusion, it was suggested that indie game development should be approached primarily from the perspective of passion and hobby, as achieving a financially profitable outcome requires considerable investment and marketing expertise. The thesis provided an overview of the challenges associated with indie game development and offered valuable insights for those considering their own game project.

Key words

3D games, game development, Unity game engine

KÄSITTEIDEN MÄÄRITTELY

AAA
AAA on pelialalla käytetty termi, jolla kuvataan korkealaatuisia ja korkeabudjettisia videopelejä.

Ambienssivarjostuskartta
Ambienssivarjostuskartta määrittää, kuinka paljon ympäristön valo pääsee tiettyyn kohtaan mallin pinnalla.

Avoin lähdekoodi
Avoimella lähdekoodilla tarkoitetaan ohjelmistokehityksessä mallia, jossa ohjelman lähdekoodi on vapaasti saatavilla kenen tahansa nähtäväksi ja muokattavaksi.

Copyleft -lisenssi
Avoimen lähdekoodin lisenssi, joka vaatii, että johdannaisteokset ovat myös copyleft -lisenssin alla.

C#
Microsoftin kehittämä ohjelmointikieli.

Diffuusikartta
Diffuusikartta, toisilta nimiltään myös värikartta tai albedo-kartta, määrittää 3D-mallin pinnan värin, kun siihen osuu tasainen valo.

GNU GPL -lisenssi

GNU GPL eli GNU General Public -lisenssi on avoimen lähdekoodin ohjelmistoissa käytetty copyleft -lisenssi, jonka alla lisensoiduista ohjelmista pitää aina olla lähdekoodi vapaasti saatavissa ja ohjelmaa voi vapaasti käyttää mihin tahansa tarkoitukseen sekä sitä voi muokata ja levittää vapaasti edelleen. GNL-lisenssi vaatii, että ohjelmaa ja siihen perustuvia ohjelmia levitetään GPL: n alaisuudessa.

IDE
Lyhenne sanoista Integrated Development Environment, eli ohjelmointiympäristö.

Immersio
Peliin syventyminen niin että keskittyminen on kokonaan pelissä ja ulkopuolisen maailman tiedostaminen heikkenee.

Indie-pelit
Yksittäisten pelintekijöiden tai pienempien ryhmien tekemiä pelejä ilman julkaisijan rahoitusta

Normaalikartta
Normaalikartta on tekstuuri, joka sisältää tietoa pinnan normaalien suunnista, joita käytetään simuloimaan pieniä yksityiskohtia, kuten kohoumia tai uurteita, ilman että 3D-malliin tarvitsee lisätä

geometriaa. Normaalikartta muuttaa valon käyttäytymistä pinnalla, jolloin saadaan aikaan hienostuneempia valon ja varjon efektejä.

Pelijami
Pelijami on tapahtuma, jossa kehitetään pelejä nopealla aikataululla, tyypillisesti noin 24–72 tunnin sisällä.

Spekulaarikartta
Spekulaarikartta määrittää harmaasävyjen avulla kuinka kiiltävä tai heijastava pinta on tietyissä kohdissa. Spekulaarikartan tummat alueet edustavat matalaa heijastavuutta, kun taas vaaleat alueet edustavat korkeaa heijastavuutta. Spekulaarikartta mahdollistaa erilaisten materiaalien, kuten metallin tai nahan, realistisen renderöinnin.

UI
Lyhenne englannin kielen sanoista user interface. Suomeksi käyttöliittymä.

TIIVISTELMÄ

ABSTRACT

KÄSITTEIDEN MÄÄRITTELY

SISÄLLYS

KUVAT

TAULUKOT

1 JOHDANTO

Indie-pelikehitys on viime vuosikymmeninä kasvanut merkittäväksi osaksi peliteollisuutta, tarjoten mahdollisuuden yksittäisille kehittäjille ja pienille tiimeille toteuttaa omia visioitaan ilman suurten julkaisijoiden vaikutusvaltaa. Teknologian kehityksen ja digitaalisten jakelualustojen, kuten Steamin ja Epic Games Storen, ansiosta indie-pelit voivat tavoittaa laajan yleisön ympäri maailmaa. Tämä vapaus ja saavutettavuus ovat johtaneet indie-pelien monipuoliseen ja innovatiiviseen tarjontaan, mutta samalla ne ovat asettaneet kehittäjät uusien haasteiden eteen, erityisesti resurssien ja rahoituksen osalta.

Vaikka indie-pelikehityksessä on potentiaalia luoda menestyviä ja kaupallisesti kannattavia pelejä, monille kehittäjille todellisuus on usein karumpi. Pienen budjetin ja rajallisen ajan puitteissa työskentely saattaa johtaa siihen, että pelin kehitys venyy, ja markkinoille saattaminen vie huomattavasti enemmän resursseja kuin alun perin arvioitiin. Kilpailu on kovaa, ja vaikka peli olisi teknisesti laadukas, se ei välttämättä erottuisi joukosta ilman merkittäviä markkinointiponnisteluja tai poikkeuksellista sisältöä.

Tässä työssä tarkastellaan indie-pelikehityksen erityispiirteitä ja haasteita, joita kohtaavat erityisesti pienet tiimit ja itsenäiset kehittäjät. Kehityspäiväkirjan avulla analysoidaan demoprojektin etenemistä ja sen vaatimuksia ajallisesti ja taloudellisesti. Lisäksi pyritään arvioimaan, kuinka realistista on saavuttaa kaupallinen menestys indie-peliprojektilla, ottaen huomioon nykyiset markkinat ja vertailukohtana käytetyt esimerkkipelit.

2 PELIKEHITYS TIIMISSÄ

Pelikehitys tiimissä on monivaiheinen prosessi, joka edellyttää saumatonta yhteistyötä eri alojen ammattilaisten välillä. Pelikehitystiimit voivat vaihdella kooltaan ja koostumukseltaan projektin laajuuden ja käytettävissä olevan budjetin mukaan. Tyypillisesti tiimiin kuuluu pelisuunnittelijoita, jotka vastaavat pelin konseptin ja mekaniikkojen kehittämisestä, sekä ohjelmoijia, jotka toteuttavat nämä ideat käytännössä. Lisäksi taiteilijat luovat pelin visuaalisen ilmeen, ja äänisuunnittelijat vastaavat pelin äänimaailmasta, joka parantaa pelikokemusta ja syventää pelaajan immersiota. (Gregory 2015, 5.)

Tuottajat puolestaan koordinoivat koko kehitysprosessia, varmistavat aikataulujen pitävyyden ja huolehtivat, että projekti etenee suunnitelmien mukaisesti. Koska jokaisen tiimin jäsenen työ liittyy läheisesti muiden tekemiseen, jatkuva viestintä ja yhteistyö ovat avainasemassa projektin onnistumisessa. Näin varmistetaan, että peli valmistuu ajallaan ja vastaa alkuperäistä visiota. (Gregory 2015, 7.)

2.1 Pelisuunnittelija

Pelisuunnittelijan tehtävänä on suunnitella pelin kulku sekä määritellä pelin kentät ja juoni. Suunnittelija vastaa myös pelin hahmojen ja visuaalisen ilmeen kehittämisestä. Lisäksi pelisuunnittelija voi osallistua pelin tuotantoon muissa rooleissa, kuten ohjelmoijana tai testaajana. (Video Game Designer Career.)

Peliprojektissa suunnittelijoita voi olla useita, erilaisilla toimenkuvilla. Suunnittelijaksi lasketaan myös esimerkiksi käsikirjoittajat, joiden työnkuva voi vaihdella yksittäisten dialogien käsikirjoittamisesta kokonaisen tarinan käsikirjoittamiseen. (Gregory 2015, 7.)

2.2 Tuottaja

Tuottaja ohjaa ja valvoo pelin kehittämistä sekä koordinoi koko prosessia. Hänen vastuullaan on varmistaa, että kehitystyö etenee sovitun aikataulun mukaisesti. Lisäksi tuottaja huolehtii siitä, että projekti pysyy ennalta määritellyn budjetin rajoissa. (Vuorela 2007, 65.)

Tuottajan rooli voi myös vaihdella studioiden välillä. Joissain studioissa tuottajan tehtävä on hallinnoida pelikehitystä kokonaisuutena, kun taas toisissa studioissa tuottajan pääasiallinen tehtävä on toimia linkkinä tuotantotiimin ja rahoittajan välillä. Pienemmissä studioissa tuottajaa ei välttämättä ole ollenkaan. (Gregory 2015, 7.)

2.3 Ohjelmoija

Ohjelmoijan rooli pelikehitysprojektissa on keskittyä pelin ohjelmointiin. Hän vastaa siitä, että pelin koodi toimii suunnitellusti ja toteuttaa tarvittavat tekniset ratkaisut. Ohjelmoijan tehtäviin voi kuulua myös pelin testaaminen, jotta koodi toimii virheettömästi ja lopputulos on laadukas. (Vuorela 2007, 65.)

Ohjelmoijat jaetaan yleensä kolmeen pääkategoriaan: junior-, mid- ja senior-ohjelmoijiin. Tämä jako perustuu ensisijaisesti ohjelmoijien työkokemukseen alalta. Junior-ohjelmoijilla on yleensä vähemmän kokemusta, kun taas mid- ja senior-ohjelmoijat ovat kerryttäneet enemmän kokemusta ja osaamista. Lisäksi ohjelmoijat voidaan jakaa tarkemmin osaamis- tai tehtäväalueen mukaan, mikä määrittää heidän erityisosaamisensa ja työtehtävänsä. (A Quick Guide to Software Developer Hierarchy.)

2.4 Graafikot

Graafiset taiteilijat ovat olennaisessa roolissa pelin visuaalisten osien elävöittämisessä. Heidän päävastuullaan on suunnitella ja toteuttaa pelin visuaalinen ilme, johon kuuluvat muun muassa hahmot, ympäristöt, valaistus ja animaatio sekä usein erilaiset pelaajan käyttämät käyttöliittymät. Graafikot tekevät tiivistä yhteistyötä taidejohtajien, pelisuunnittelijoiden ja muiden tiimin jäsenten kanssa varmistaakseen visuaalisen tyylin, taiteellisen ilmeen ja tarinankerronnan yhtenäisyyden pelikehitysprosessin aikana. (What is a Video Game Artist?)

Graafikot osallistuvat usein myös pelikehityksen tekniseen puoleen, kun peliin luodut visuaaliset elementit tulee optimoida pelimoottoria varten suorituskyvyn parantamiseksi. Lisäksi graafikoiden tehtävänä voi olla esimerkiksi markkinointimateriaalin tuottaminen peliprojektin markkinointiprojektia varten. Kaiken kaikkiaan graafikkojen määrä sekä työtehtävät vaihtelevat kuitenkin peliprojektin mukaan. (What is a Video Game Artist?)

2.4.1 Konseptitaiteilija

Konseptitaiteilijat tuottavat vedoksia siitä, miltä pelin tulisi näyttää. Konseptitaiteilijat aloittavat projektissa työskentelyn tyypillisesti hyvin aikaisessa vaiheessa ja heidän työskentelynsä yleensä jatkuu koko peliprojektin elinkaaren ajan. (Gregory 2015, 6.)

Konseptitaiteilijan tehtäviin kuuluu hahmojen ja ympäristön suunnittelu. He luovat pelin alkuperäiset visuaaliset ideat ja määrittelevät, miltä pelin maailma ja hahmot näyttävät. Työskentelynsä kautta konseptitaiteilijat valitsevat pelin visuaalisen tyylisuunnan ja auttavat luomaan pelin yleisen tunnelman. (Concept artist (Games).)

2.4.2 3D-mallintaja

3D-mallintaja vastaa pelimaailman kolmeulotteisten mallien tuottamisesta. Heidän työnsä on keskeisessä roolissa pelin visuaalisen laadun luomisessa. 3D-mallintajat tuovat konseptitaiteen ja suunnittelumääritelmät eloon, mikä lisää pelin immersiivisyyttä ja tekee pelikokemuksesta vaikuttavamman. (What does a 3D modeler do: Understanding the Role and Responsibilities of a 3D Modeler.)

3D-mallintajat jaetaan yleensä kahteen eri ryhmään, joista toinen ryhmä mallintaa pelimaailman eri objektit, eli esimerkiksi aseet, hahmot sekä ajoneuvot ja toinen ryhmä mallintaa pelimaailman taustat, kuten esimerkiksi rakennukset ja erilaiset maanpinnan muodot. (Gregory 2015, 6.)

2.4.3 Tekstuuritaiteilija

Tekstuuritaiteilijat luovat pelimaailmaan kaksiulotteisia kuvia, joita kutsutaan tekstuureiksi. Tekstuureja käytetään 3D-mallien pintoina tuomaan malleille yksityiskohtia (Gregory 2015, 6). Tekstuuritaiteilijat voivat luoda erilaisia tekstuurityyppejä, kuten diffuusikarttoja väritiedolle, spekulaarikarttoja pinnan kiiltoa varten, normaalikarttoja pinnan yksityiskohtia varten ja ambienssivarjostuskarttoja varjoefektien luomiseksi. (Gregory 2015, 462.)

Tarkasti hiotuilla tekstuurien luomisella tekstuuritaiteilijat parantavat pelin realistisuutta ja visuaalista laatua, edistäen kokonaisvaltaista immersiivistä kokemusta pelaajille (Gregory 2015, 6). Tekstuuritaiteilijat ovat myös ratkaisevassa roolissa tekstuurien optimoinnissa pelimoottorissa, tasapainottaen visuaalista laatua muistin käytön ja renderöinnin tehokkuuden kanssa. (How Do You Optimize Game Art Assets on Blender?)

2.4.4 Valaisija

Valaisija suunnittelee ja toteuttaa peliprojektissa käytettävän valomaailman. Valaisija määrittää pelissä käytettävien valojen suunnan, määrän sekä niiden värikylläisyydet (Gregory 2015, 6). Valaisija on peliprojektissa keskeisessä roolissa, kun pelimaailmaan luodaan visuaalista tunnelmaa ja ilmapiiriä. Valaisijan rooliin kuuluu ottaa huomioon erilaisia tekijöitä, kuten vuorokauden aika, sääolosuhteet ja kunkin kohtauksen emotionaalinen sävy luodessaan valaistusasetteluja, jotka herättävät halutun tunnelman ja lisäävät pelin immersiota. (The Creator of Lights in Games.)

Valaisija sijoittaa pelimaailmaan valonlähteitä ja säätelee niiden ominaisuuksia, kuten intensiivisyyttä, väriä ja suuntaa. Valaisijan tehtävänä on myös hienosäätää varjojen laatua ja pintojen heijastuksia. Näin valaisija luo pelin visuaalista ilmettä ja tunnelmaa. (Gregory 2015, 6.)

2.4.5 Animaattori

Animaattori saa pelimaailman hahmot ja muut objektit liikkumaan. Kuten tietokoneanimaatioelokuvatuotannossa, peliprojektissa animaattorit toimivat ikään kuin näyttelijöinä. Pelianimaattorin tulee kuitenkin näyttelijöihin verrattuna omata hyvin uniikki taitosetti luodakseen

animaatioita, jotka sulautuvat saumattomasti pelimoottorin teknologisiin perustuksiin. (Gregory 2015, 6.)

Animaattorit varmistavat, että kaikki pelin liikkeet ovat yhtenäisiä ja tukevat pelin visuaalista tyyliä sekä teknisiä vaatimuksia. Animaattorit luovat työllään immersiivisen pelikokemuksen, jossa hahmot reagoivat realistisesti ja pelaaja tuntee painon ja voiman välittyvän liikkeistään, mikä on erityisen tärkeää pelaajan sitoutumisen ja pelin tarinankerronnan kannalta. (What is a Video Game Artist?)

2.4.6 Liikkeenkaappaaja

Liikkeenkaappaajat käyttävät liikekaappaustyökaluja tallentamaan liikettä luodakseen karkean datasetin siitä, miten hahmot liikkuvat peleissä. Tämä karkea datasetti tyypillisesti siistitään ja hienosäädetään animaattoreiden toimesta ennen sen integroimista peliin. (Gregory 2015, 6.)

Liikkeentallennus tarjoaa erinomaisen tavan saada luonnollisia ja uskottavia liikkeitä ilman, että animaattorien täytyy luoda niitä alusta alkaen itse. Liiketallennusdatan pohjalta animaattorit pystyvät erilaisten animointityökalujen ja ohjelmistojen avulla muokata ja hienosäätää tallennettuja liikkeitä tarpeen mukaan. Tämä voi sisältää liikkeiden yhdistämistä, lisäyksiä tai poistoja sekä niiden ajoituksen ja nopeuden säätämistä. Tällä tavoin liiketallennus tarjoaa tehokkaan ja realistisen tavan tuottaa laadukkaita animaatioita lyhyemmässä ajassa verrattuna täysin käsin animoituun lähestymistapaan. (Animation Techniques.)

2.5 Äänimaailma

Äänimaailma peliprojektissa ja peleissä on keskeinen osa immersiota. Immersion lisäksi äänimaailma tarjoaa pelaajalle muun muassa palautetta pelissä tapahtuvista asioista, helpottaa navigointia ja orientoitumista pelissä sekä tuo pelihahmoihin lisää syvyyttä ja luo pelihahmoille tarvittaessa omat identiteetit. Onnistunutta äänimaailmaa voidaan käyttää hyväksi myös markkinoinnissa. Esimerkiksi helposti tunnistettava ääniraita on omiaan luomaan pelille brändiä. (The Importance of Sound Design in Video Game Development.)

Äänimaailman suunnitteluun ja luomiseen tarvitaan tyypillisesti monen eri osa-alueen osaajia. Tyypillisesti äänimaailman suunnitteluun ja toteutukseen osallistuvat henkilöt työskentelevät tiiminä

saavuttaakseen mahdollisimman yhtenäisen äänimaailman musiikkeineen ja ääniefekteineen sekä täyttääkseen riittävät laatustandardit. (Horowitz & Looney 2014, 2.)

Projektin mukaan peliprojekti voi sisältää erilaisia määriä musiikkia, puhetta, ääniefektejä tai laulua. Tällöin peliprojektiin voi kuulua myös omat säveltäjänsä, sanoittajansa, soittajansa sekä esimerkiksi miksaajansa. (Horowitz & Looney 2014.)

2.5.1 Äänisuunnittelija

Äänisuunnittelija suunnittelee ja toteuttaa pelin äänimaailman. Äänisuunnittelu sisältää ääniefektien, musiikin ja ääninäyttelyn suunnittelua, joilla täydennetään pelin visuaalisia elementtejä sekä kerronnallisia teemoja. (Horowitz & Looney 2014, 93.)

Yksi äänisuunnittelijan päätehtävistä on luoda äänitehosteita, jotka elävöittävät pelimaailman ja tarjoavat auditiivista palautetta pelaajille. Nämä äänitehosteet voivat vaihdella askeleista ja aseiden äänistä ympäristön tunnelmaan ja käyttöliittymän äänitehosteisiin. (Horowitz & Looney 2014, 99-101.)

Äänitehosteiden lisäksi äänisuunnittelijat vastaavat myös musiikin luomisesta ja integroinnista peliin. Musiikilla on keskeinen rooli eri pelikohtausten sävyn, tunnelman ja ilmapiirin luomisessa. (Horowitz & Looney 2014, 47, 99.)

Pelin äänillä ja musiikilla voidaan vaikuttaa huomattavasti onnistuneeseen pelikokemukseen. Äänisuunnittelijan tehtävät vaihtelevat äänimaailman ideoinnista ja suunnittelusta toteutukseen ja aina oman musiikkiteoksen säveltämiseen asti. (Pelin äänisuunnittelija.)

2.5.2 Ääninäyttelijä

Ääninäyttelijä tuottaa nimensä mukaisesti pelimaailman hahmoille puheäänen. Pelin luonteen mukaan ääninäyttelijöitä saatetaan tarvita myös muihin kuin hahmojen puheäänien näyttelemiseen, esimerkiksi tarinankertojaksi. (Horowitz & Looney 2014, 109.)

Ääninäyttelijät näyttelevät tärkeää roolia hahmojen ominaispiirteiden ja persoonallisuuksien luonnissa. Vaikka hahmojen luonnin onnistumiseen vaikuttavat muutkin seikat kuin ääninäyttely itsessään, on puheen sisältö ja ilmaisutapa keskeinen osa hahmosta saatavaa vaikutelmaa. Näin ollen onnistunut ääninäyttely on keskeisessä osassa immersion luomisessa. (Horowitz & Looney 2014, 112)

2.6 Testaaja

Nimensä mukaisesti testaajan tehtäviin pelikehityksessä kuuluu testaaminen. Testauksessa pelitestaajat käyttävät erilaisia testaustekniikoita, kuten tutkimuksellista testausta, regressiotestausta ja yhteensopivuustestausta, jolla varmistetaan, että peli toimii eri alustoilla ja laitteilla sekä etsitään ja raportoidaan muulle tuotantotiimille pelistä löytyvät bugit ja muut virheet. (Vuorela 2007, 65.)

Pelitestaajilta edellytetään muun muassa hyvää keskittymiskykyä, havainnointikykyä ja pitkäjännitteisyyttä sekä hyviä kirjallisia viestintätaitoja. Mikäli testaaminen on luonteeltaan teknistä, testaajalta edellytetään ohjelmoinnin osaamista, sekä testaukseen liittyvien työkalujen ja laitteiston tuntemusta. (Pelitestaaja.)

2.7 Muu henkilöstö

Onnistuneen projektin toteuttamiseen tarvitaan luonnollisesti muutakin henkilöstöä. Peliprojektiin voi olla tarpeen perustaa erillinen IT-osasto, joka huolehtii laitteiston ja ohjelmistojen hankinnasta, hallinnoinnista ja asentamisesta. IT-osasto tukee tuotantotiimiä varmistamalla, että kaikki tekniset resurssit ovat kunnossa ja käytettävissä. (Gregory 2015, 6.)

Hallinnollinen henkilöstö tyypillisesti huolehtii päivittäisistä toimistotehtävistä, kuten mahdollisista matkajärjestelyistä, kokouskoordinoinnista ja dokumentoinnista. Lisäksi projektilla voi olla muun muassa erillinen markkinointiosasto, joka vastaa pelin markkinoinnista ja viestinnästä. (Gregory 2015, 7.)

3 PELIKEHITYKSEN TYÖKALUT

Pelikehityksen työkalut ovat keskeisiä elementtejä, jotka mahdollistavat pelien suunnittelun, kehittämisen ja optimoinnin. Yksi tärkeimmistä työkaluista on pelimoottorit, jotka tarjoavat kehittäjille valmiin alustan, jolla peli voidaan rakentaa ilman tarvetta luoda kaikkea alusta alkaen. Pelimoottorit sisältävät työkaluja grafiikan renderöimiseen, fysiikan simuloimiseen, ja skriptaukseen, mikä nopeuttaa kehitysprosessia ja mahdollistaa monimutkaisten pelimekaniikkojen luomisen. (Gregory 2015, 11.)

Toinen keskeinen ryhmä työkaluja on graafiset suunnittelu- ja 3D-mallinnustyökalut, joita käytetään pelin visuaalisen ilmeen luomiseen, kuten hahmojen, ympäristöjen ja muiden grafiikan elementtien suunnitteluun. Graafiset suunnittelijat ja 3D-mallinnusasiantuntijat käyttävät näitä ohjelmia luodakseen korkealaatuisia visuaaleja, jotka rikastuttavat pelikokemusta ja tukevat pelin tarinaa ja teemaa. (Gregory 2015, 6.)

Äänityökalut ovat myös olennainen osa pelikehitystä, sillä pelin äänimaailma vaikuttaa merkittävästi pelaajakokemukseen. Äänityökalut mahdollistavat äänitehosteiden, musiikin ja dialogin luomisen ja muokkaamisen. Äänisuunnittelijat käyttävät näitä ohjelmia luodakseen äänimaiseman, joka tukee pelin tunnelmaa ja intensiivisyyttä, ja jotka voivat parantaa pelaajan immersiota pelimaailmaan. Näiden työkalujen avulla voidaan myös optimoida äänten laatua ja varmistaa, että ne sulautuvat hyvin pelin visuaaliseen ja pelimekaaniseen kokonaisuuteen. (Gregory 2015, 6, 49.)

3.1 Pelimoottori

Pelimoottorilla tarkoitetaan ohjelmistokehystä, jonka päälle peli rakennetaan. Nimitys pelimoottori tulee 1990-luvun puolivälissä vakiintunut termi, joka juontaa juurensa Id Softwaren 1993 kehittämään Doom -peliin. Doomissa oli tarkkaan määritetty ja eroteltu ohjelmiston ydinkomponentit kuten törmäyksen tunnistimet (engl. collision detection) sekä kolmiulotteisen grafiikan renderöintijärjestelmä sen pelimaailmasta, pelin yleisestä kulusta ja visuaalisesta ilmeestä. Erottelun myötä tuotantotiimin oli mahdollista julkaista uusi peli uusilla maailmoilla, objekteilla ja tapahtumilla, mutta vain minimaalisilla muutoksilla itse pelimoottoriin. (Gregory 2015, 11.)

Pelimoottorin avulla peliin saadaan siis luotua haluttu sisältö ja pelimaailman säännöt, mutta pelimoottori itsessään hoitaa kaiken taustatyön. Pelimoottori voi olla kolmannen osapuolen tuottama valmis tuote, tai pelimoottori voidaan rakentaa tuotettavaa peliä varten itse. Kustannustehokkaassa indie-pelikehityksessä kuitenkin yleensä päädytään käyttämään valmista pelimoottoria. (Gregory 2015, 27-32.)

3.2 Yleisesti käytössä olevia pelimoottoreita

Pelimoottorit ovat ohjelmistoalustoja, jotka tarjoavat kehittäjille työkalut pelien luomiseen ja hallintaan. Ne tarjoavat valmiit rakenteet ja ominaisuudet, jotka helpottavat pelikehitystä ja mahdollistavat erilaisten pelien luomisen alusta alkaen. Pelimoottorit voivat vaihdella ominaisuuksiltaan ja käyttötarkoituksiltaan, mutta niiden pääasialliset tavoitteet ovat tarjota tehokkaita kehyksiä, grafiikan ja fysiikan hallintaa, ja skriptauksen mahdollisuuksia. (Gregory 2015, 27-32.)

Vertailu eri pelimoottorien välillä auttaa kehittäjiä valitsemaan oikean työkalun projektin tarpeiden mukaan, riippuen siitä, tarvitsevatko he monipuolista käyttöliittymää, avoimen lähdekoodin joustavuutta vai huipputason grafiikkakykyjä. Jokainen moottori tuo mukanaan ainutlaatuisia etuja, jotka voivat vaikuttaa merkittävästi projektin kehitykseen, aikatauluun ja lopulliseen laatuun. Pelimoottoreista Unity, Godot ja Unreal Engine ovat kolme erityisen suositeltua moottoria, jotka tarjoavat omat ainutlaatuiset ominaisuutensa ja edut. (Gregory 2015, 27-32.)

3.2.1 Unity

Unity on vuonna 2005 julkaistu järjestelmäriippumaton pelimoottori, jolla voi kehittää pelejä usealle eri alustalle. Unityn vahvuuksiin kuuluu sen helppokäyttöisyys, monipuoliset ominaisuudet sekä tuki 2D- ja 3D-pelituotantoon. Unityn skriptikielenä toimii C# -ohjelmointikieli. (Gregory 2015, 29–30.)

Unitylle löytyy myös oma verkkokauppa nimeltä Unity Asset Store, joka tarjoaa laajan valikoiman valmiita resursseja ja työkaluja, kuten animaatioita, tekstuureja, äänitiedostoja ja skriptejä, jotka helpottavat peliprojektien kehitystä. Lisäksi Unityn ympärille on rakentunut kookas yhteisö, joka

tarjoaa runsaasti oppimateriaalia, dokumentaatiota ja muita tukiresursseja. (What is the Unity Asset Store and how do I purchase Assets?)

Unitystä on olemassa viittä eri lisenssiä, Unity Student, Unity Personal, Unity Pro, Unity Enterprise ja Unity Industry -lisenssit. Student -lisenssi on ilmainen ja tarkoitettu yli 16-vuotiaille opiskelijoille opetukselliseen käyttöön. Myös Personal -lisenssi on ilmainen, mikäli sen käyttäjä on saanut voittoa tai rahoitusta alle 100 000 Yhdysvaltain dollaria viimeisen 12 kuukauden aikana. Pro -lisenssi maksaa 1877 euroa käyttäjää kohden vuodessa. Enterprise -lisenssiin on neuvoteltavissa tarjous ja Industry -lisenssi maksaa 4554 euroa käyttäjää kohden vuodessa. (Plans and pricing.)

3.2.2 Unreal Engine

Unreal Engine on Epic Gamesin vuonna 1998 kehittämä pelimoottori, joka tukee useaa eri käyttöjärjestelmää. Unreal Enginellä voi kehittää pelejä usealle eri alustalle ja Unreal Enginestä löytyy tuki sekä 2D- että 3D-pelituotantoon. Unreal Enginen skriptaus tapahtuu visuaalisesti blueprintejä käyttämällä tai vaihtoehtoisesti C++ -ohjelmointikielellä. (Gregory 2015, 27–28.)

Unreal Engine on tunnettu erityisesti korkealaatuisesta grafiikastaan ja tehokkaista visuaalisista efekteistään, mikä tekee siitä houkuttelevan vaihtoehdon kehittäjille, joiden tavoitteena on kehittää visuaalisesti vaikuttavia pelejä. Unreal Engineltä löytyy oma verkkokauppa, josta löytyy pelikehitykseen erilaisia valmiita resursseja, kuten 3D-malleja, tekstuureja, animaatioita, äänitiedostoja ja skriptejä. (Gregory 2015, 27–28.)

Unreal Enginestä on olemassa kolmea erilaista lisenssityyppiä, Standard -, Enterprise ja Custom -lisenssi. Näistä Standard -lisenssi on ilmainen, Enterprise -lisenssi maksaa 1500 Yhdysvaltain dollaria käyttäjää kohden vuodessa ja Custom -lisenssin hinta on Unreal Enginen verkkosivujen mukaan neuvoteltavissa. (Licensing.)

3.2.3 Godot engine

Godot on ilmainen avoimen lähdekoodin pelimoottori. Godotista löytyy tuki 2D- ja 3D-pelituotantoon. Godotin skriptauskielenä toimii sen oma GDScript -skriptikieli, joka muistuttaa syntaksiltaan hieman Python-ohjelmointikieltä. Näin ollen GDScript tarjoaa Godotin käyttäjille mahdollisuuden toteuttaa nopeaa ja tehokasta pelilogiikkaa mahdollisimman loivalla oppimiskäyrällä. Godotin laaja tuki, pelimoottorin lähdekoodin avoimuus ja yhteisön aktiivisuus ovat tehneet siitä suositun valinnan indie-pelikehittäjille. (Godot Docs.)

Godotille löytyy oma verkkoarkisto nimeltä Godot Asset Library, jonka ominaispiirteitä on, että siellä tarjottavat resurssit ovat ilmaisia ja lisensoitu avoimen lähdekoodin lisenssien alle. Godot Asset Library tarjoaa laajan valikoiman erilaisia resursseja ja työkaluja, kuten 3D-malleja, animaatioita ja skriptejä. (Godot Docs.)

Alun alkujaan Godot Engine oli kaupallinen ja suljettu lisenssi, mutta se julkaistiin vuonna 2014 avoimen lähdekoodin lisenssillä (Machado, 2017). Godot on lisensoitu MIT lisenssin alle, joten sen käyttäjän on mahdollista käyttää Godot Engineä mihin tahansa käyttötarkoitukseen ja jopa tarpeen tullen muokata pelimoottoria (License).

3.3 Ohjelmointiympäristö

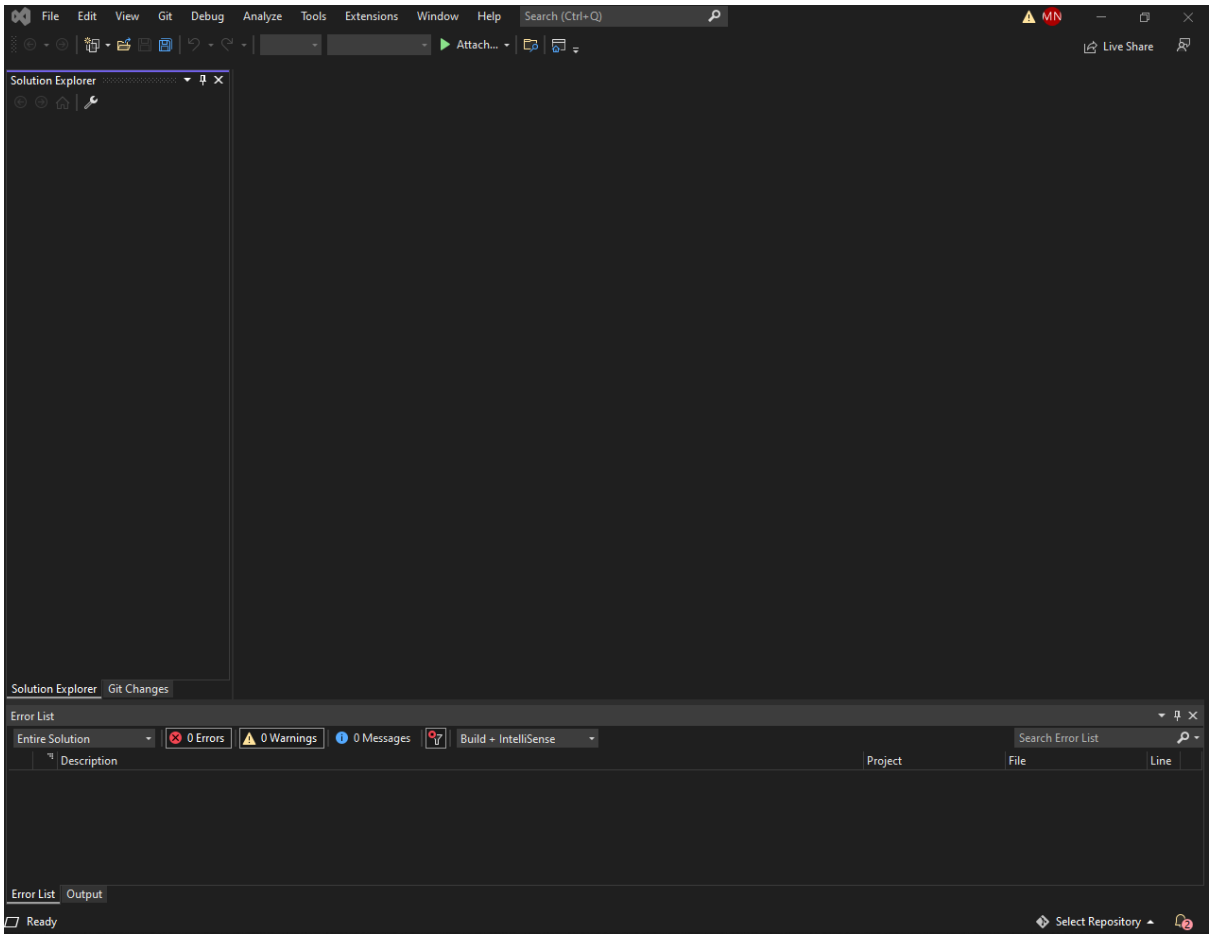
Ohjelmointiympäristö, on ohjelma tai joukko ohjelmia, jolla ohjelmoija toteuttaa ohjelmistotuotantoa. Tällaisia ohjelmia ovat muun muassa, tekstieditori, kääntäjä, koodianalysoija ja erilaiset virheenetsintä- sekä testaustyökalut. Ohjelmointiympäristöstä käytetään tyypillisesti nimitystä IDE, joka on lyhenne ohjelmointiympäristön englanninkielisestä vastineesta integrated development environment. (Gregory 2015, 73.)

IDE:t ovat vakiinnuttaneet asemansa nykypäivän modernissa ohjelmistokehityksessä, sillä ne tehostavat kehitysprosessia ja helpottavat laadukkaiden ohjelmistojen tuottamista nopeammin ja tehokkaammin verrattuna siihen, että kaikki ohjelmistokehitykseen käytettävät työkalut olisivat erillisinä työkaluinaan. Usein IDE:t tarjoavat myös laajan valikoiman lisäosia ja laajennuksia, joilla mahdollistetaan tarvittavat räätälöinnit kehittäjän ja ohjelmistoprojektin tarpeiden mukaisiksi. (What is an IDE (Integrated Development Environment)?)

3.3.1 Visual Studio

Visual Studio (KUVA 1.) on Microsoftin luoma ohjelmointiympäristö. Visual Studiosta löytyy tuki useille eri ohjelmointikielille, kuten C#, C++, Visual Basic, F#, Python ja JavaScript. Tämä tekee Visual Studiosta monikäyttöisen työkalun, joka soveltuu useisiin erilaisiin ohjelmistokehitysprojekteihin. (Gregory 2015. 73–75.)

Visual Studiolla on työpöytäohjelmistojen lisäksi mahdollista kehittää esim. web-ohjelmia sekä mobiiliohjelmia. Visual Studiosta löytyy tuki Microsoft Windows -käyttöjärjestelmille. Visual studiosta on olemassa ilmainen Community edition sekä maksulliset Professional - ja Enterprise edition -versiot. (What is Visual Studio?)

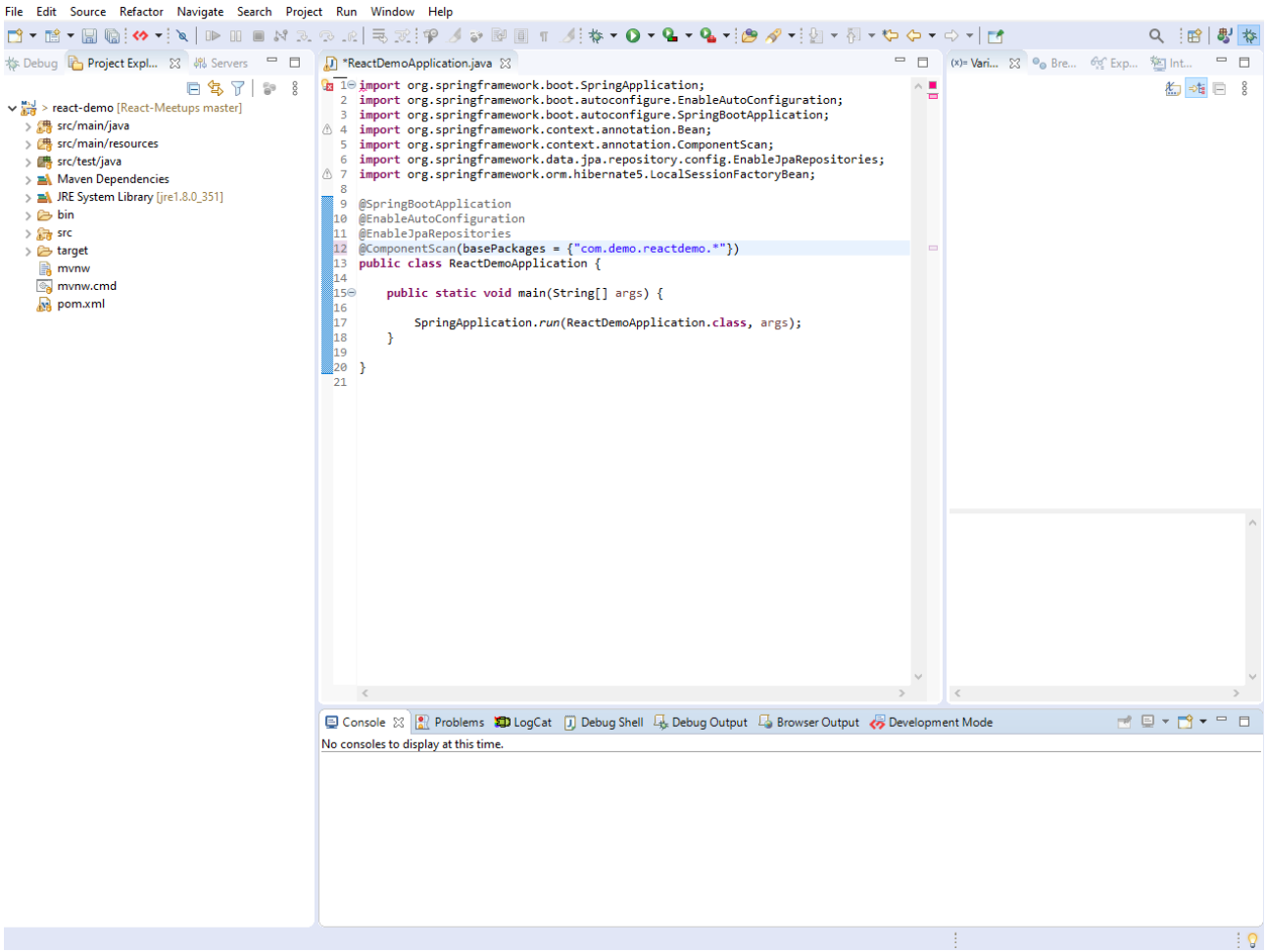


Kuva 1. Visual Studio

3.3.2 Eclipse

Eclipse (KUVA 2.) on erittäin suosittu ohjelmointiympäristö, josta löytyy tuki muun muassa Javalle, PHP: lle, Pythonille sekä C\C++ -ohjelmointikielille. Eclipse on saatavilla Microsoft Windows, macOS ja Linux -käyttöjärjestelmille. (Eclipse documentation.)

Eclipse on lisensoitu Eclipse Public License (EPL) -ohjelmointilisenssin alle, joka on copyleft-lisenssi. Eclipsen lisenssi on kuitenkin huomattavasti sallivampi kuin muut copyleft-lisenssit kuten GNU GPL. Näin ollen Eclipseä voi käyttää ja muokata vapaasti ilman että sen käyttöön liittyisi copyleft-lisensseille tyypillisiä rajoituksia. (Eclipse Public License.)

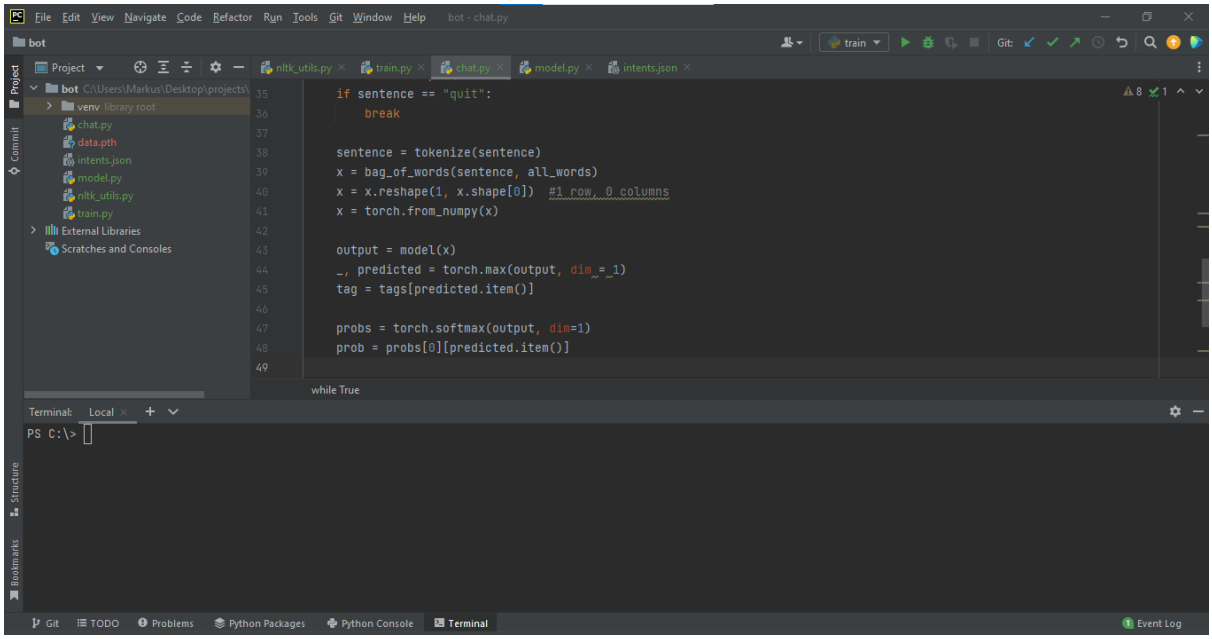


Kuva 2. Ecplise

3.3.3 PyCharm

PyCharm (KUVA 3.) on JetBrainsin Python -ohjelmointikieltä varten suunniteltu ohjelmointiympäristö. PyCharm on saavuttanut Python-kehittäjien keskuudessa suurta suosiota sen tehokkaiden ominaisuuksien myötä, jotka tarjoavat kehittäjille mahdollisuuden keskittyä luovaan työhön ja koodin laatuun samalla kun PyCharm hoitaa monimutkaisemmat ja toistuvat tehtävät automaattisesti. (The Python IDE for data science and web development.)

PyCharmista on olemassa kaksi eri versiota, ilmainen avoimen koodin Community edition sekä maksullinen Pro edition. PyCharmista löytyy tuki Microsoft Windows, macOS ja Linux -käyttöjärjestelmille. (The Python IDE for data science and web development.)



Kuva 3. PyCharm.

3.4 Grafiikka ja animaatiot

Grafiikkojen ja animaatioiden luomiseen on olemassa lukuisia työkaluja, jotka tarjoavat erilaisia ominaisuuksia ja toiminnallisuuksia erityyppisten projektien tarpeisiin. Maksullisia työkaluja ovat muun muassa Spine jolla voi tehdä 2D-animaatioita, Maya LT jolla voi tehdä 3D-mallinnusta sekä animaatioita ja Aseprite jolla voi tehdä pikseligrafiikkaa ja animaatiota. (Gregory 2015, 54.)

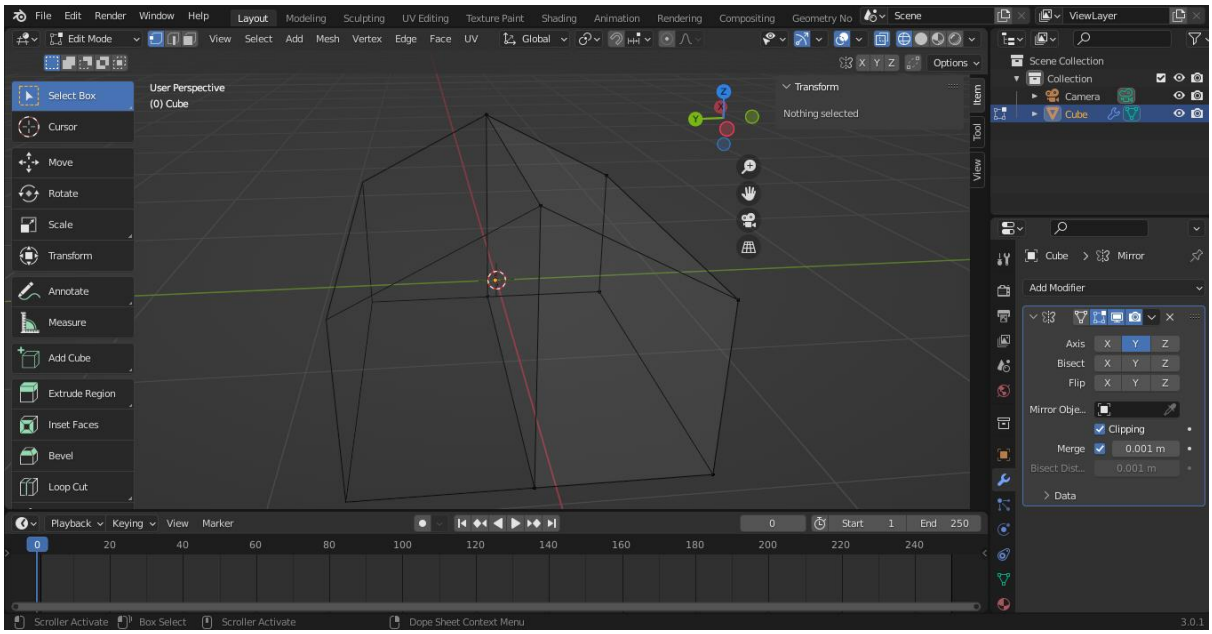
Ilmaisia työkaluja grafiikan ja animaatioiden tekemiseen ovat muun muassa Blender, jolla voi tehdä animaatioita sekä 2D ja 3D -assetteja, Asepriten kaltainen LibreSprite, jolla voi tehdä pikseligrafiikkaa ja animaatioita sekä esimerkiksi kuvankäsittelyohjelma GIMP. Tässä luvussa keskitytään esittelemään keskeisiä ilmaisia työkaluja grafiikan ja animaatioiden tekemiseen. (Blender; LibreSprite; GIMP.)

3.4.1 Blender

Blender (KUVA 4.) on monipuolinen ja käyttäjäystävällinen 3D-grafiikan tuottamiseen suunniteltu ohjelmisto, joka tarjoaa laajan valikoiman työkaluja erilaisiin tarkoituksiin. Sen avulla käyttäjät voivat luoda animaatioita, visuaalisia efektejä ja monimutkaisia 3D-malleja eri projekteihin. Blenderin käyttömahdollisuudet eivät rajoitu pelkästään 3D-mallinnukseen ja animointiin, vaan siinä on myös ominaisuuksia liikkeenkaappaukseen sekä tuki Python-skripteille, mikä avaa mahdollisuuden automaatioon ja lisäosien kehittämiseen. (Blender.)

Blenderin avulla käyttäjät voivat toteuttaa monenlaisia projekteja, oli kyse sitten ammattimaisesta animaatiosta, peligrafiikasta tai visuaalisista efekteistä. Sen käyttö ei vaadi kalliita lisenssimaksuja, mikä tekee siitä houkuttelevan vaihtoehdon niin harrastajille kuin ammattilaisillekin. Lisäksi Blenderin avoin lähdekoodi mahdollistaa yhteisön osallistumisen kehitykseen ja lukuisien ilmaisten lisäosien saatavuuden, mikä rikastuttaa ohjelmiston toiminnallisuutta entisestään. (Blender.)

Blenderin käyttöönottaminen saattaa vaatia jonkin verran opiskelua ja harjoittelua, mutta sen monipuoliset ominaisuudet ja laaja tuki tekevät siitä varteenotettavan vaihtoehdon graafisen alan ammattilaisille ja harrastajille ympäri maailmaa. Ohjelmiston jatkuvan kehityksen ansiosta Blenderistä löytyy aina uusia ominaisuuksia ja parannuksia, mikä tekee siitä yhä vahvemman kilpailijan muihin kaupallisiin 3D-grafiikan tuottamiseen tarkoitettuihin ohjelmistoihin. (Blender.)



Kuva 4. Blender.

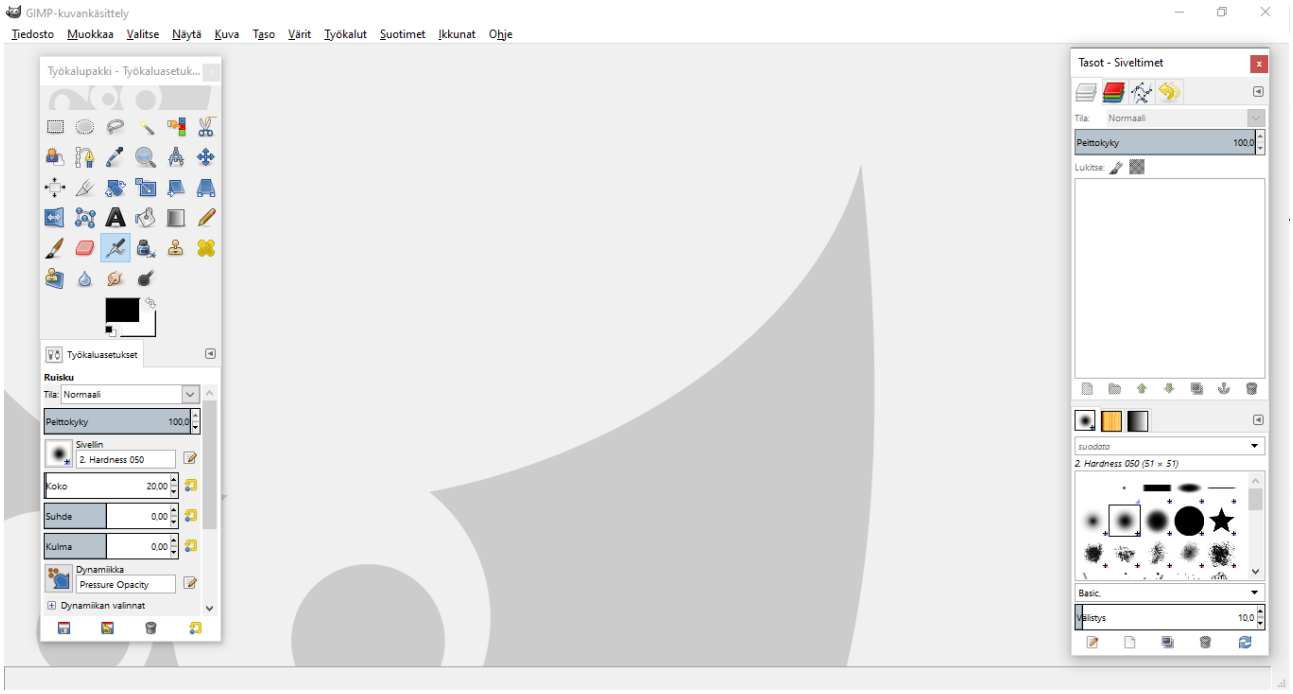
3.4.2 GIMP

GIMP (KUVA 5.), lyhenne sanoista GNU Image Manipulation Program, on monipuolinen ja ilmainen kuvankäsittelyohjelma, joka tarjoaa laajan valikoiman työkaluja erilaisten kuvien muokkaamiseen ja manipulointiin. Se on suosittu valinta sekä ammattilaisten että harrastajien keskuudessa, sillä se tarjoaa monia ominaisuuksia ja toimintoja, jotka kilpailevat monien kaupallisten kuvankäsittelyohjelmien kanssa. GIMP on saatavilla useille eri käyttöjärjestelmille, kuten macOS, Linux ja Windows, mikä tekee siitä käyttäjäystävällisen ja helposti saatavilla olevan vaihtoehdon kaikille käyttäjille. (GIMP.)

GIMPin avoimen lähdekoodin lisenssi, GNU GPL, mahdollistaa sen ilmaisen käytön ja kehityksen yhteisön avulla. Tämä on houkutteleva ominaisuus monille käyttäjille, jotka arvostavat avoimuutta ja yhteisön osallistumista ohjelmiston kehitykseen. Lisäksi avoin lähdekoodi antaa käyttäjille mahdollisuuden mukauttaa ohjelmistoa omiin tarpeisiinsa ja lisätä siihen uusia toimintoja tarvittaessa. (GIMP.)

GIMPin käyttöliittymä voi vaikuttaa aluksi monimutkaiselta, mutta sen monipuoliset ominaisuudet ja työkalut tarjoavat käyttäjille runsaasti mahdollisuuksia luoda laadukkaita kuvia ja grafiikkaa. Ohjelmisto sisältää muun muassa erilaisia maalaustyökaluja, kuvan säätö- ja korjaustoimintoja,

tekstityökaluja ja suodattimia, jotka mahdollistavat monipuolisen kuvankäsittelyn erilaisiin tarkoituksiin. (GIMP.)



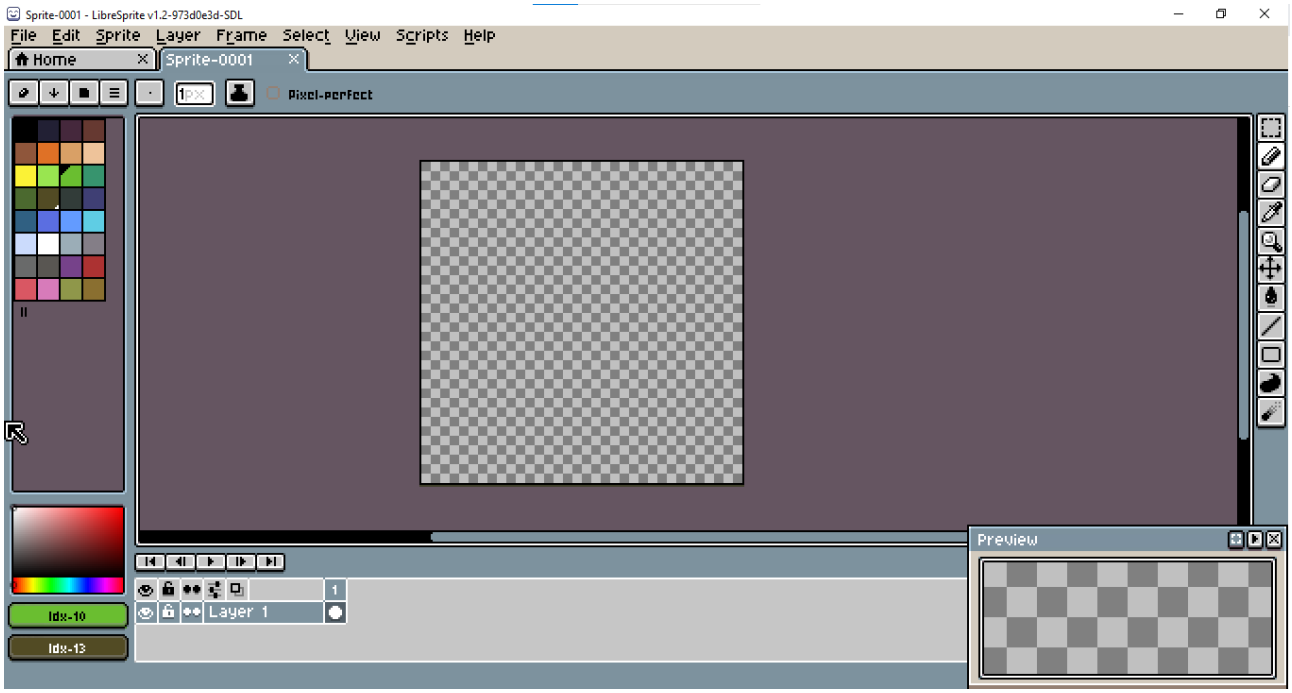
Kuva 5. GIMP.

3.4.3 LibreSprite

LibreSprite (KUVA 6.) tarjoaa monipuolisen ja ilmaisen vaihtoehdon 2D-pikseligrafiikan ja animaatioiden luomiseen avoimen lähdekoodin periaatteiden mukaisesti. Ohjelman juuret juontavat alunperin Aseprite-kuvankäsittelyohjelmaan, mutta LibreSprite on sittemmin kehittynyt omaksi itsenäiseksi projektikseen, joka tarjoaa käyttäjilleen runsaasti työkaluja ja toimintoja grafiikan luomiseen. LibreSprite on saatavilla ilmaiseksi ja sen avoimen lähdekoodin lisenssi, GNU GPL, mahdollistaa yhteisön osallistumisen ohjelmiston kehitykseen ja laajentamiseen. (LibreSprite.)

LibreSprite tarjoaa käyttäjilleen monia työkaluja ja ominaisuuksia, jotka helpottavat pikseligrafiikan ja animaatioiden luomista. Sen käyttöliittymä on selkeä ja helppokäyttöinen, mikä tekee siitä houkuttelevan vaihtoehdon niin ammattilaisille kuin harrastajillekin. Ohjelmisto sisältää muun muassa erilaisia piirtotyökaluja, animaatioiden kehitys- ja editointityökaluja sekä mahdollisuuden tuoda ja viedä erilaisia tiedostomuotoja, mikä tekee siitä monipuolisen ja käyttäjäystävällisen vaihtoehdon graafisen sisällön luomiseen. (LibreSprite.)

LibreSprite on saavuttanut suosiotaan erityisesti pelikehittäjien ja animaattoreiden keskuudessa, jotka arvostavat ohjelmiston monipuolisia ominaisuuksia ja sen ilmaisuutta. Sen avulla käyttäjät voivat luoda laadukasta pikseligrafiikkaa ja animaatioita erilaisiin projekteihin, olivatpa ne sitten pelejä, animaatioita tai muita visuaalisia teoksia. LibreSprite tarjoaa vaihtoehdon kaupallisille ohjelmistoille ja mahdollistaa luovien projektien toteuttamisen avoimen lähdekoodin ympäristössä. (LibreSprite.)



Kuva 6. LibreSprite.

3.5 Äänimaailma

Äänimaailma on keskeinen tekijä pelin immersiossa ja tunnelman luomisessa. Se auttaa pelaajaa uppoutumaan pelimaailmaan ja eläytymään pelin tapahtumiin. Äänimaailman luomiseen on saatavilla monia erilaisia työkaluja ja menetelmiä, joilla voidaan tuottaa erilaisia ääniefektejä, musiikkia ja ääninäyttelyä. Näiden elementtien yhdistelmä luo monipuolisen ja elävän äänimaiseman, joka tukee pelin visuaalista kokemusta. (Schell 2008. 351–352.)

Äänimaailman luominen voi tapahtua joko kokonaan digitaalisesti tai osittain digitaalisesti. Osittain digitaalisessa äänituotannossa käytetään usein studiossa äänitettyä materiaalia, kuten hahmojen dialogia tai tarvittavaa musiikkia. Tällöin äänet tallennetaan ja käsitellään jälkikäteen erilaisilla äänityökaluilla ja ohjelmistoilla. Digitaalisesti luotu äänimaailma mahdollistaa suuremman hallinnan ja

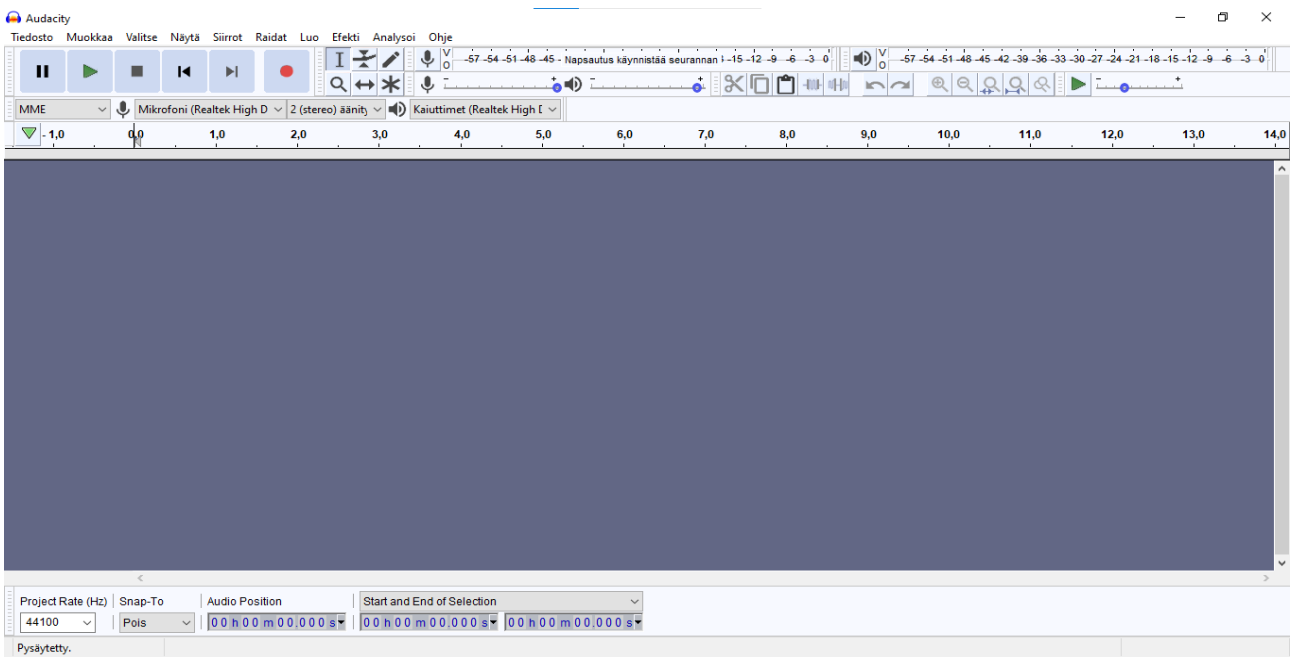
muokattavuuden ääniefektien suhteen, mikä voi tarjota pelin kehittäjille enemmän luovaa vapautta ja joustavuutta äänisuunnittelussa. (Soundscaping - The Art Of Soundscape Creation.)

Monipuolisten äänityökalujen ja ohjelmistojen avulla äänimaailman luominen voi olla kiehtova ja monipuolinen prosessi pelinkehityksen aikana. Pelaajille tarjotaan näin erottuva ja mukaansatempaava äänimaisema, joka vahvistaa pelin tunnelmaa ja tarinaa. (Soundscaping - The Art Of Soundscape Creation.)

3.5.1 Audacity

Audacity (KUVA 7.) on järjestelmäriippumaton avoimen lähdekoodin äänieditori, joka tarjoaa laajan valikoiman toimintoja äänitykseen ja äänen muokkaukseen. Audacityn avulla käyttäjät voivat tallentaa ääntä, suorittaa erilaisia editointitoimenpiteitä sekä hyödyntää erilaisia efektejä äänitiedostoihin. (About Audacity.)

Audacity tukee useita eri ääniformaatteja, kuten WAV, FLAC, AIFF ja MP3, mikä tekee siitä monipuolisen ja käyttäjäystävällisen vaihtoehdon äänityökaluksi. Lisäksi Audacity tarjoaa mahdollisuuden käyttää kolmansien osapuolien plug-ineja, mikä laajentaa sen toiminnallisuutta entisestään ja mahdollistaa monipuolisemman äänieditoinnin. (About Audacity.)

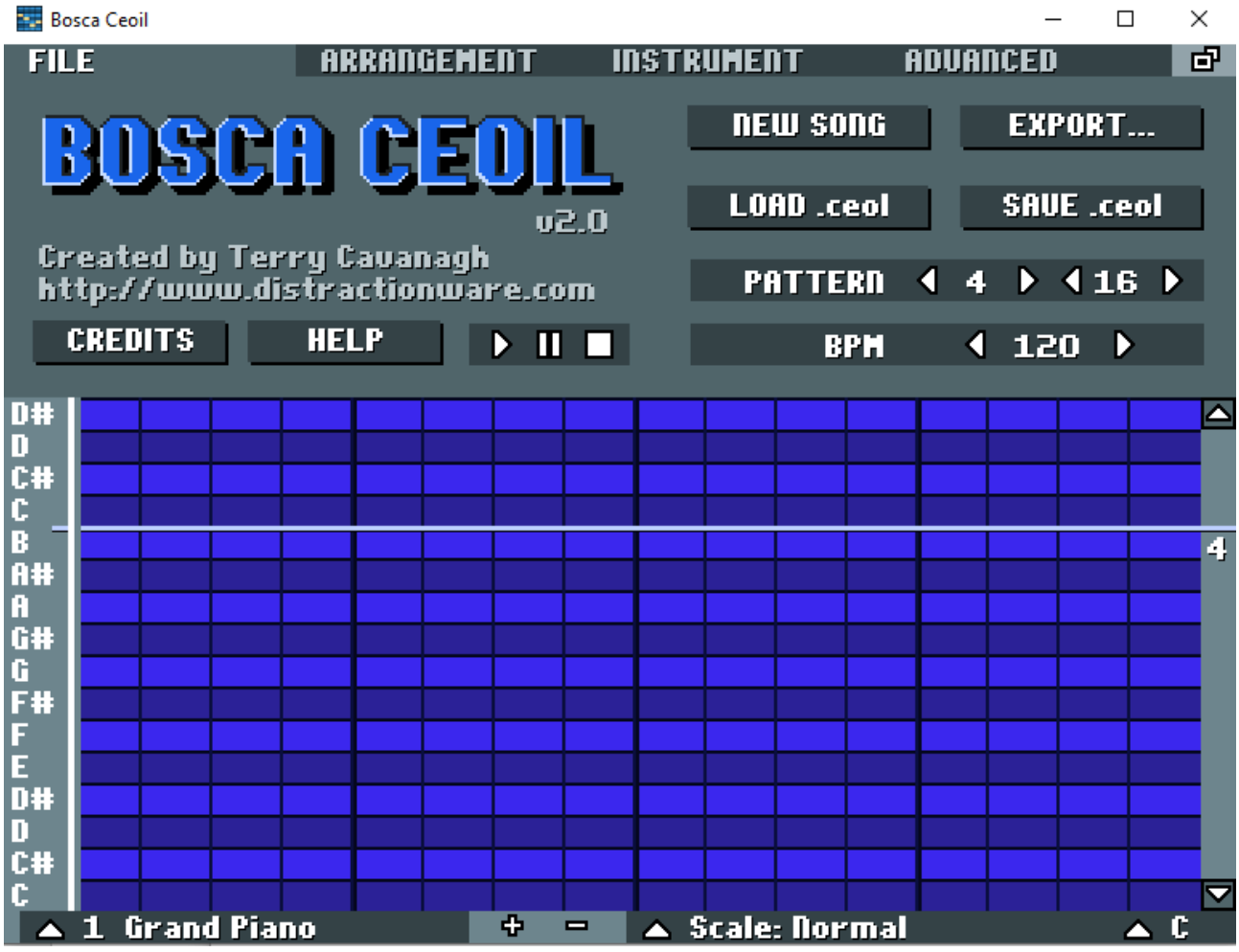


Kuva 7. Audacity.

3.5.2 Bosca Ceoil

Bosca Ceoil (KUVA 8.) on indie-pelikehittäjä Terry Cavanaghin luoma ilmainen ohjelmisto, joka on suunniteltu erityisesti 8-bittisen musiikin tuottamiseen. Bosca Ceoil on suunniteltu mahdollisimman aloittelijaystävälliseksi ja se tarjoaa käyttäjilleen mahdollisuuden luoda retrotyylisiä musiikkikappaleita ilman syvällistä musiikin tuntemusta tai teknistä osaamista. (Bosca Ceoil.)

Bosca Ceoil tarjoaa valmiita ääniraitoja ja melodiaideoita, jotka helpottavat musiikin luomista ja antavat käyttäjille inspiraatiota omien kappaleiden luomiseen. Lisäksi ohjelmisto mahdollistaa kappaleiden tallentamisen eri ääniformaatteihin, jotta niitä voidaan helposti käyttää peliprojekteissa tai muissa musiikin tuottamiseen liittyvissä projekteissa. (Bosca Ceoil.)



Kuva 8. Bosca Ceoil.

3.6 Versionhallintajärjestelmä

Versionhallintajärjestelmä on työkalu, joka mahdollistaa useamman henkilön työskentelyn samojen tiedostojen parissa yhtäaikaisesti. Versionhallintajärjestelmä sisältää historian tiedostoihin tehdyistä muutoksista, jolloin ongelmien ilmetessä tietyssä tiedostoversiossa voidaan helposti palata aikaisempaan, ongelmattomaan tiedostoversioon. (Gregory 2015. 63–64.)

Versionhallintajärjestelmä ei ole pelkästään tiedostojen tallennuspaikka, vaan se on myös tärkeä työkalu tiimin kommunikoinnin ja yhteistyön edistämisessä. Sen avulla tiimin jäsenet voivat helposti jakaa työtään, tarkastella toistensa tekemiä muutoksia ja tarvittaessa tehdä yhteistyötä samojen tiedostojen parissa. Tämä tehostaa työskentelyä ja auttaa varmistamaan, että kaikki tiimin jäsenet pysyvät synkronoituina projektin edistymisen suhteen. (Gregory 2015. 63–64.)

3.6.1 Git

Git on Linus Torvaldsin kehittämä hajautettu versionhallintajärjestelmä, joka on tullut erittäin suosituksi monissa ohjelmistokehitysprojekteissa, mukaan lukien peliprojekteissa. Sen suosio johtuu sen nopeudesta, joustavuudesta ja tehokkuudesta, jotka tekevät siitä ihanteellisen työkalun monenlaisiin versionhallintatarpeisiin. Gitin avulla tiimit voivat hallita, seurata ja jakaa muutoksia ohjelmistoprojekteihin tehokkaasti. (A Short History of Git.)

Gitin hajautettu rakenne mahdollistaa sen, että jokainen tiimin jäsen voi pitää oman kopionsa versionhallintajärjestelmästä ja tehdä muutoksia paikallisesti. Tämä mahdollistaa työskentelyn ilman jatkuvaa internet-yhteyttä ja antaa tiimin jäsenille suuren vapauden ja joustavuuden työskennellä missä ja milloin tahansa. Hajautettu luonne myös vähentää riippuvuutta yhdestä keskitetystä palvelimesta, mikä tekee Gitistä luotettavan ja skaalautuvan ratkaisun suurille tiimeille. (Intro to Github for version control.)

3.6.2 Subversion

Subversion, joka usein lyhennetään muotoon SVN, on avoimen lähdekoodin versionhallintajärjestelmä. Subversion tarjoaa keskitetyn lähestymistavan versionhallintaan, mikä mahdollistaa useiden käyttäjien yhteistyön samoissa tiedostoissa ja kansioissa. Toisin kuin hajautetut versionhallintajärjestelmät, Subversion perustuu keskitettyyn tietovarastoon, johon tallennetaan kaikki tiedostojen versiot ja niiden historia. (Gregory 2015. 66–66.)

Subversionin keskeisistä ominaisuuksista on sen yksinkertaisuus ja helppokäyttöisyys, mikä tekee siitä sopivan sekä pienille että suurille ohjelmistoprojekteille. Subversionia on laajasti hyödynnetty ohjelmistokehitysyhteisössä sen vakauden, luotettavuuden ja laajan dokumentaation ansiosta. (Gregory 2015. 66–66.)

4 PELIKEHITYKSEN VAIHEET

Tyypillisesti pelikehitys jaetaan kolmeen eri vaiheeseen, joita ovat esituotanto, tuotanto ja jälkituotanto. Esituotannossa kehitetään pelin konsepti, suunnitellaan sen rakenne ja valmistellaan tarvittavat resurssit. Tuotanto-vaiheessa toteutetaan kaikki pelin sisällöt ja mekaniikat, kun taas jälkituotannossa keskitytään pelin viimeistelyyn, virheiden korjaamiseen ja julkaisukelpoisuuden varmistamiseen. (Video Game Development Process.)

Konseptointivaiheessa kehitystiimi hioo pelin ideaa ja tavoitteita, määrittelee pelin visuaalisen ilmeen ja mekaniikat sekä tekee tarvittavat suunnitelmat pelin kehityksestä. Tämä vaihe voi sisältää ideoiden kokoamista, konseptikuvien ja prototyyppien luomista sekä markkina-analyysiä pelin kohdeyleisön ja kilpailijoiden selvittämiseksi. (Vuorela 2007, 43–46.)

Esituotantovaiheessa kehitystiimi syventyy suunnitelmiin ja valmistelee kaiken tarvittavan ennen varsinaisen tuotannon aloittamista. Tämä vaihe sisältää pelin suunnitteludokumentaation laadintaa, hahmojen ja ympäristöjen konseptisuunnittelua, äänisuunnittelua sekä prototyyppien kehitystä. Esituotantovaiheen tavoitteena on luoda vankka pohja pelin kehitykselle ja varmistaa, että koko tiimi ymmärtää selkeästi pelin visiot ja tavoitteet. (Vuorela 2007, 40–43.)

Tuotantovaiheessa varsinaiset pelin resurssit luodaan ja peli rakennetaan. Tässä vaiheessa kehittäjät toteuttavat suunnitelmat käytännössä, kuten koodaavat pelin mekaniikat, luovat grafiikat ja animaatiot, äänittävät äänimaailman ja suunnittelevat pelin tasoja. Tuotantovaiheessa työskentely on usein intensiivistä ja vaatii tiivistä yhteistyötä eri tiimin jäsenten välillä varmistaakseen, että peli etenee suunnitellusti ja aikataulussa. (Vuorela 2007, 66–68.)

Jälkituotantovaiheessa pelin kehitys lähestyy loppuaan, ja painopiste siirtyy viimeistelyyn, testaukseen ja julkaisuvalmisteluihin. Tässä vaiheessa pelin eri osa-alueita hienosäädetään ja optimoidaan, jotta lopputulos olisi mahdollisimman laadukas ja pelikokemus saumaton. Jälkituotantovaiheeseen voi kuulua myös pelin markkinointi- ja julkaisusuunnitelman viimeistelyä. (Vuorela 2007, 72, 78.)

4.1 Esituotanto

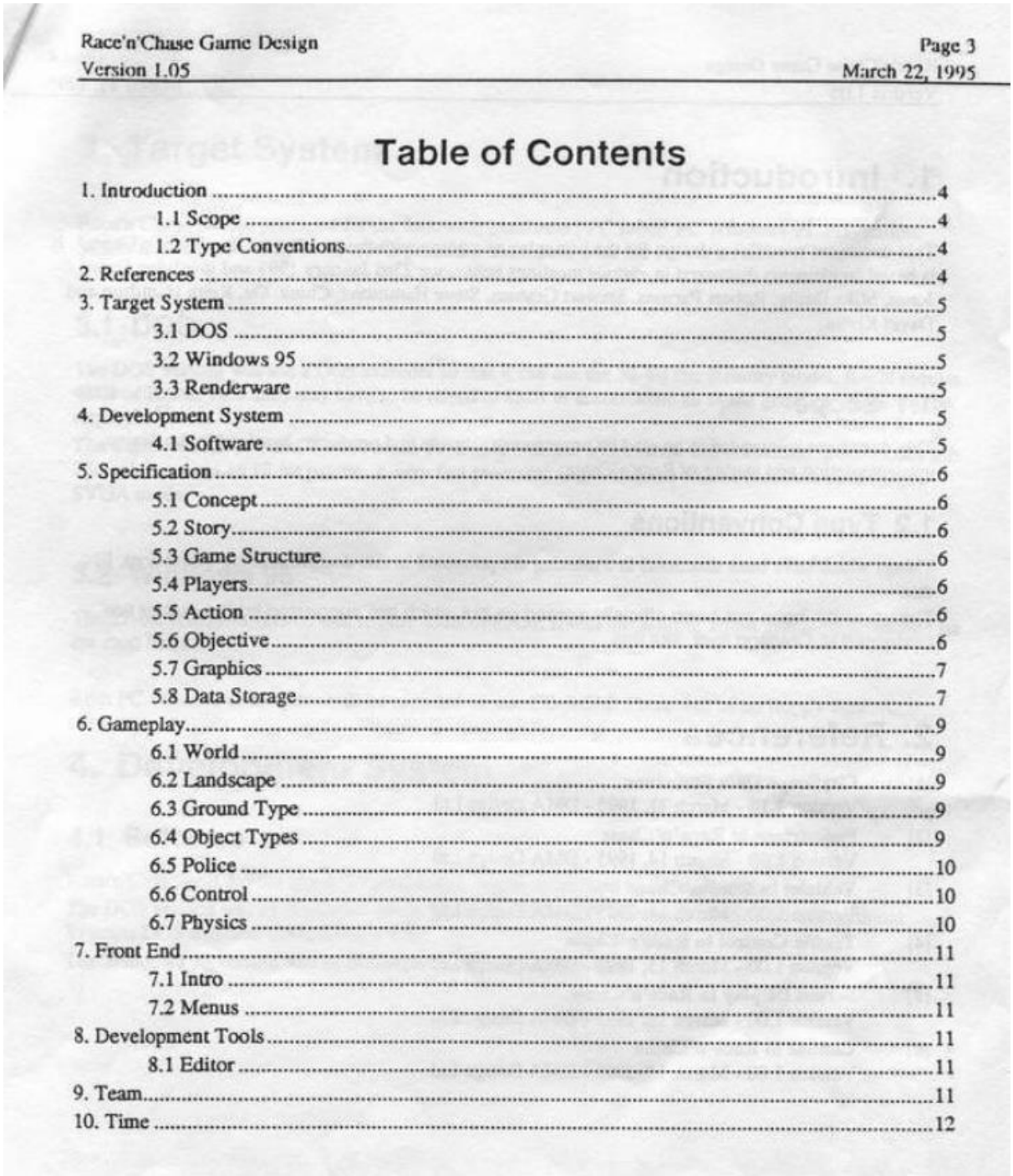
Peliprojektin esituotantovaiheessa valmistutaan varsinaiseen tuotantovaiheeseen. Peliprojektin esituotantovaiheessa tehdään pelille vaatimusmäärittely, sekä tuotetaan pelisuunnitelma. Esituotantovaiheessa myös määritellään tarvittavat resurssit, kuten budjetti, aikataulut sekä sisäisen ja ulkoisen työvoiman tarve. (Video Game Development Process.)

Peliprojektin esituotantovaihe alkaa, kun päätös pelin tekemisestä tehdään ja päättyy, kun pelin rakenne ja sisältövaatimukset ovat selvillä. Esituotantovaiheen ja varsinaisen tuotantovaiheen välinen vuorovaikutus kuitenkin kestää koko peliprojektin ajan, mutta kuitenkaan varsinaisessa tuotantovaiheessa ei peli-ideaan kannata tehdä kovin suuria muutoksia, sillä tämä tarkoittaa yleensä sitä, että tuotanto joudutaan aloittamaan enemmän tai vähemmän alusta asti uudelleen. (Vuorela 2007, 42.)

4.1.1 Konseptointi

Pelikehityksessä konseptoinnilla tarkoitetaan pelin hahmottelemista ja sen dokumentointia. Konseptointivaiheessa pelistä tuotetaan siis konseptointidokumentti, joka tyypillisesti sisältää pelin perusidean ja kuvauksen, sekä sen että mille alustoille peli tuotetaan. Dokumentti voi myös sisältää esimerkkejä visuaalisesta tyylistä, hahmoista ja maailmasta. Huomioitavaa on, että konseptidokumentti ei ole sama asia, kuin esituotantovaiheessa tuotettava pelisuunnitelma -dokumentti. (Vuorela 2007, 42-43.)

Konseptin tulee olla muistettava, mutta konseptin esitystapa riippuu siitä kenelle sitä pitää esittää. Tuotantotiimille konseptiksi voi riittää esimerkiksi pääpointit kirjoitettuna ranskalaisilla viivoilla paperille, mutta ammattimaisessa pelisuunnittelussa konsepti on usein useita sivuja pitkä sekä sisältää kuvia jokaisesta aiheesta ja kilpailevista tuotteista. (Vuorela 2007, 55.)



Race'n'Chase Game Design
Version 1.05

Page 3
March 22, 1995

Table of Contents

Kuva 9. Grand Theft Auto -pelin konseptointidokumentin sisällysluettelo Vuodelta 1995. Pelin alkuperäinen nimi oli Race'n'Chase (Race’n’Chase Game Design).

4.1.2 Resurssointi

Resurssi on mikä tahansa pelin tuottamiseen sijoitettava aineellinen tai henkinen voimavara. Ennen tuotantovaihetta tulisi olla tiedossa mitä resursseja tuotantotiimillä on käytettävissä, joten resurssoinnissa nuo resurssit tulee saada vastaamaan pelikonseptia. Ihannetapauksessa resurssien rajallisuus on otettu huomioon jo konseptointivaiheessa. (Vuorela 2007, 56.)

Nyrkkisääntönä peliprojekteissa voidaan pitää, että resursseja ei ole koskaan riittävästi. Kun resursseja ei ole riittävästi, on vaihtoehtona joko resurssien lisääminen tai vaihtoehtoisesti konseptin supistaminen. Resurssoinnissa keskeistä on, että koska aikaa ja voimavaroja on rajallisesti, ne tulee osata keskittää oikeisiin asioihin. (Vuorela 2007, 56.)

4.2 Tuotespeksi

Tuotespeksi on toiminallinen kuvaus, jonka tarkoituksena on, että tuotespeksin luettuaan koko tekijäryhmällä on sama käsitys pelissä tapahtuvista asioista. Tuotespeksissä on tarkoitus kuvailla tuote niin hyvin, että mahdolliset henkilöstövaihdokset tai tuotantoon liittyvät unohdukset eivät johda tahattomiin muutoksiin. Tuotespeksin tarkoituksena on myös varmistaa, että näkemys pelistä ei muutu sen kehittämisen aikana. (Vuorela 2007, 57.)

Pelin tuotespeksistä tulisi käydä ilmi se mitä pelaaja voi tehdä pelimaailmalle sekä se että mitä pelimaailma voi tehdä pelaajalle (Vuorela 2007, 57). Jokaisessa tuotespeksin kohdassa tulisi käydä toiminnallisuuden lisäksi ilmi kohdan vaatimat komponentit, kuten 3D-mallit, animaatiot, äänet sekä muut tarvittavat komponentit. Lopuksi tuotespeksistä tehdään yhteenveto, josta on helppo tarkastaa mitä pelistä jo löytyy, mitä pelistä puuttuu ja mitä peliin tulisi vielä lisätä. (Vuorela 2007, 58.)

4.2.1 Konsepti

Tuotespeksi kannattaa aloittaa esittelemällä konsepti sellaisena kuin se on määritelty resurssien jälkeen. Vaikka tuotespeksi on tarkempi ja yksityiskohtaisempi kuvaus pelin sisällöstä, sen tavoitteena on kuvata alkuperäistä konseptia ja sen herättämiä mielikuvia alusta loppuun. Konseptiosio

tuotespeksissä toimii ikään kuin johdantona pelimaailmaan, tarjoamalla yleiskuvan pelin ideasta ja sen tarjoamasta kokemuksesta. (Vuorela 2007, 58.)

Konseptiosio voi sisältää lyhyen kuvauksen halutusta genrestä ja pelin sisällöstä, sekä teknisiä tietoja erilaisista pelimuodoista. Tämä osa dokumenttia auttaa määrittämään pelin suuntaa ja varmistaa, että kaikki tiimin jäsenet ovat samalla sivulla pelin visioista ja tavoitteista. Tavoitteena on luoda selkeä ja houkutteleva kuva pelin konseptista, joka ohjaa kehitystyötä ja viestii projektin ydinajatuksesta. (Vuorela 2007, 58.)

4.2.2 Pelin aloitus

Pelin aloitus -kohdassa kuvataan, mitä pelaaja kokee ja näkee, kun peli alkaa. Tämä osa dokumenttia sisältää kaikki aloittamiseen liittyvät toimenpiteet, kuten pelihahmon luontiprosessin tai pelin taustatarinankerronnan. Tavoitteena on varmistaa, että pelaajalle tarjotaan selkeä ja mukaansatempaava aloituskokemus. (Vuorela 2007, 58.)

Pelin aloitus -kohta toimii myös kuvauksena niistä ruuduista, valikoista ja tilanteista, joita pelaaja kohtaa pelin alussa. Tämä osio auttaa hahmottamaan pelin käyttöliittymän ja aloitusprosessin, tarjoten yksityiskohtaisia tietoja siitä, miten peli käynnistyy ja miten pelaaja pääsee alkuun. (Vuorela 2007, 58.)

4.2.3 Käyttöliittymä

Tuotespeksin käyttöliittymä -kohdasta käy ilmi millaista tietoa pelistä välittyy pelaajalle sekä miten tieto välitetään pelaajalle. Tiedonvälittiminä voivat toimia esimerkiksi erilaiset ruudulla näkyvät osoittimet ja mittaristot tai esimerkiksi pelimaailmaa kuvaava kartta. Jokaisesta tietoa välittävästä komponentista on tehtävä erillinen tarkka kuvaus niin että graafikot osaavat piirtää ne, sekä ohjelmoijat saavat ohjelmoitua ne toimimaan halutulla tavalla. (Vuorela 2007, 58.)

Pelaajalle tulee myös välittyä tarpeen mukaan käyttöliittymästä tieto siitä, miten pelimaailma ja mittareiden käyttämät lukuarvot suhtautuvat toisiinsa niin, että pelaaja tietää edustaako näytetty

lukuarvo pelissä hyvää vai huonoa tilannetta. Käyttöliittymästä tulisi käydä myös ilmi, miten pelimaailmassa tehty päätös vaikuttaa valloillaan olevaan tilanteeseen. (Vuorela 2007, 58.)

4.2.4 Pelaajan toiminnot

Pelaajan toiminnot ovat toimintoja, jotka vaativat pelaajalta tietoisia tekoja ja päätöksiä pelin aikana. Tässä kohdassa määritellään, miten pelaaja voi syöttää peliin tietoa ja miten tämä syötetty tieto vaikuttaa pelin tapahtumiin ja pelimaailmaan. Tähän sisältyvät kaikki pelaajan mahdollisuudet, kuten liikkuminen, vuorovaikutus hahmojen kanssa ja pelimaailman muokkaaminen. (Vuorela 2007, 59.)

Pelaajan toiminnot -kohta auttaa varmistamaan, että kaikki pelaajalle tarjotut toiminnot on määritelty selkeästi ja ohjelmoitu peliin asianmukaisesti. On tärkeää päättää erikseen, mitä pelaajan halutaan kykenevän tekemään pelissä, jotta näiden toimintojen integrointi pelin sisälle on sujuvaa ja toimivaa. (Vuorela 2007, 59.)

4.2.5 Pelaajasta riippumattomat toiminnot

Pelaajasta riippumattomat toiminnot -kohta tuotespeksissä sisältää erilaiset sattumanvaraiset tilanteet, joita pelissä voi tapahtua, mutta ne eivät johdu pelaajan toiminnasta. Tällaisia toimintoja voivat esimerkiksi olla sääilmiöiden tapahtuminen, bonuskertoimen antaminen pelaajalle sekä muut sattumanvaraisiksi tarkoitetut toiminnot. (Vuorela 2007, 60.)

Rajan vetäminen pelaajan toimintojen ja pelaajasta riippumattomien toimintojen välille voi olla haastavaa. Esimerkiksi, jos pelaaja suorittaa toiminnon, joka ei ole heti ilmeinen, saattaa tämä toiminto aiheuttaa uuden tapahtuman, josta pelaaja ei ole tietoinen. Tämä uusi tapahtuma voi puolestaan laukaista lisää toimintoja, mikä vaikeuttaa selkeää erottelua pelaajan ja peliin liittyvien automaattisten toimintojen välillä. (Vuorela 2007, 60.)

4.2.6 Viholliset, vastustajat ja haasteet

Tuotespeksin "viholliset, vastustajat ja haasteet" -kohdassa kuvataan pelaajan pelissä kohtaamat vastukset. Tämä osa dokumenttia määrittelee, millaisia vihollisia, vastustajia ja haasteita peli tarjoaa, ja miten nämä vaikuttavat pelaajan kokemukseen. (Vuorela 2007, 60.)

Videopeleissä vihollisiksi tai vastustajiksi lasketaan konkreettiset hahmot ja yksiköt, jotka toimivat pelaajan etuja vastaan. Nämä vastustajat käyttäytyvät itsenäisesti tekoälyohjelmointinsa perusteella, mikä tarkoittaa, että heillä on omat toiminta- ja reaktiomallinsa, jotka vaikuttavat pelin kulkuun ja haasteiden tasoon. (Vuorela 2007, 60.)

4.2.7 Pelin kulku

Pelin kulku -kohta tuotespeksissä kuvaa pelin normaalin etenemisen ja tapahtumat. Tämän osion tärkeys voi vaihdella eri pelien välillä, mutta tyypillisesti videopeleissä on tärkeää selostaa pelin tapahtumat tarkasti. Pelin tapahtumien seikkaperäinen kuvaus on oleellista sekä teknisten että pelisuunnittelullisten syiden vuoksi. (Vuorela 2007, 60.)

Tarkka pelin kulun suunnittelu auttaa varmistamaan, että peli etenee loogisesti ja sujuvasti. Pelin kulku on yleensä suunniteltu vaiheittain, jolloin pelaaja siirtyy kentästä tai tehtävästä toiseen. Tämä vaiheittainen rakenne auttaa pitämään pelaajan kiinnostuneena ja ohjaa heitä kohti pelin tavoitteita ja haasteita. (Vuorela 2007, 60.)

4.2.8 Pelimaailman tarkennus

Jos pelimaailmaa ei ole kuvailtu konseptissa riittävän tarkasti, tai jos tarkempi kuvaus voi tarjota lisäarvoa pelin markkinoinnin ja imagon kannalta, tehdään yksityiskohtaisempi kuvailu yleensä tuotespeksissä. Tämä tarkempi kuvaus voi sisältää pelimaailman visuaalisen ilmeen, sen rakenteen ja erityispiirteet, jotka voivat vaikuttaa pelin houkuttelevuuteen ja markkinointiin. (Vuorela 2007, 61.)

Tuotespeksissä annettu yksityiskohtaisempi pelimaailman kuvaus auttaa varmistamaan, että kaikki kehitystiimin jäsenet ymmärtävät pelin ympäristön ja sen roolin pelin kokonaisuudessa. Se tarjoaa

myös arvokasta materiaalia markkinointiin ja brändäykseen, kun peliä esitellään mahdollisille pelaajille ja sidosryhmille. (Vuorela 2007, 61.)

Tuotespeksin pelimaailman tarkennuksen tarkoituksena on, että sen luettuaan lukija tietää mistä pelissä on kyse, mitä arvoja se käyttää, miten peli toimii, miten pelaajan odotetaan toimivan sekä millaisia mahdollisia vastuksia, vihollisia ja juonenkäänteitä pelissä voidaan odottaa. Parhaimmillaan huolella tehty pelimaailman kuvaus tarjoaa käyttäjälle uusia näkökulmia, ideoita ja mielikuvia, joita välttämättä kirjoittajakaan ei ole tullut ajatelleeksi. Pelimaailman tarkennukseen turvaudutaan usein myös siksi, että peli edustaa kuluttajalle kokonaista brändiä mahdollisine jatko-osineen ja oheistuotteineen. (Vuorela 2007, 61.)

4.2.9 Yhteenveto tarvittavista komponenteista

Videopeleissä komponenteilla tarkoitetaan erilaisia pelin elementtejä, kuten hahmomalleja, rakennustyyppejä, maastotyyppejä, sekä ruudulla näkyviä pelin ohjaamiseen ja toimintoihin liittyviä osia. Lisäksi komponentteihin kuuluvat äänitehosteet ja musiikki. Yhteenvedon ei tarvitse olla täydellinen tai muuttumaton, mutta sen tulisi olla riittävän tarkka, jotta jokainen tiimin jäsen voi käyttää tuotespeksiä pohjana omien tehtäviensä suunnittelussa. (Vuorela 2007, 62.)

Mikäli tuotespeksissä pyydetään esimerkiksi kolmea erilaista rakennusta, ei ole ongelma, että työntekijä luo kuusi rakennusta, kunhan ne edustavat kolmea haluttua rakennustyyppiä. Tämä joustavuus mahdollistaa luovuuden ja laajentaa vaihtoehtoja, kunhan pääelementit ja -vaatimukset säilyvät ohjeiden mukaisina. (Vuorela 2007, 62.)

4.2.10 Tuotespeksin loppuarviointi

Kun tuotespeksi ja listat tarvittavista komponenteista ovat pääpiirteittäin valmiit, voidaan projektissa siirtyä seuraavaan vaiheeseen. Vaikka tuotespeksi ei koskaan ole täysin valmis ennen pelin julkaisua, myöhemmät muutokset eivät saisi vaikuttaa alkuperäiseen konseptiin, joka on määritelty speksin alussa. (Vuorela 2007, 62.)

Yleisenä nyrkkisääntönä voidaan pitää, että jos peliin on tarpeen tehdä muutos, se tulisi ensin päivittää tuotespeksiin ja vasta sitten pelin työn alla olevaan versioon. Tämä käytäntö auttaa pitämään tuotespeksin ajan tasalla, ja sen avulla voidaan aina tarkastaa, miten pelin tulisi toimia. Tuotespeksiä arvioidessa tulee muistaa, että oleellista ei ole tuotespeksin pituus vaan sen sisältö. (Vuorela 2007, 62.)

4.2.11 Tehtäväjako

Viimeinen vaihe esituotannossa on tuotespeksin lukeminen ja tehtävien jako. Vaikka tuotantotiimi kattaisikin vain yhden ihmisen, on kannattavaa tehdä lista siitä mitä peliprojektissa tulee tehdä missäkin vaiheessa. Mikäli peli tehdään ryhmässä, on ryhmän jäsenillä hyvä olla etukäteen sovitut vastuualueet. Kun tuotespeksi on luettu, vastuu tuotespeksin kohtien toteuttamisesta jaetaan kaikkien ryhmän jäsenien kesken. (Vuorela 2007, 63.)

Kaupallisessa pelikehityksessä ryhmän jäsenet voivat lisäksi kirjoittaa omia speksejään siitä, miten kunkin vastuualue tullaan toteuttamaan. Mikäli resurssointivaiheessa on ryhmän jäsenille annettu mahdottomia tehtäviä, asia kannattaa korjata ennen tuotannon alkamista, sillä tuotespeksin muuttaminen ennen tuotannon alkamista on vielä verrattain helppoa. (Vuorela 2007, 63.)

4.3 Tuotanto

Tuotanto on pelikehityksen pisin ja tärkein vaihe. Tuotantovaiheessa käytetään suurin osa ajasta ja rahasta joka peliprojektiin on varattu, ja tuotantovaihe on projektin vaihe, jossa peli alkaa todella herätä eloon. (Bramble 2023.) Tuotantovaihe on projektissa myös se vaihe, joka määrittää pelin lopullisen laadun ja valmiuden julkaistavaksi. (Vuorela 2007, 46.)

Tuotannon lopputuloksena ovat kaikki materiaalit siihen peliin, jonka asiakas ostaa. (Vuorela 2007, 66.) Tämä tarkoittaa, että tuotantoprosessissa kehitetään ja hiotaan kaikki pelin osat valmiiksi tuotteeksi. Tuotannon etenemistä seurataan työvaiheiden kautta. Tiivistetysti työvaiheita ovat runko, sisältö ja testaus. (Vuorela 2007, 67.)

4.3.1 Tuotantosuunnitelma

Esituotanto ja tuotanto yhdistyvät tuotantosuunnitelmassa, jossa on koottu kaikki tehtävälistat ja aikataulutettu ne pelin tuotannon eri vaiheiden mukaan. Tuotantosuunnitelma tarjoaa selkeän kuvan siitä, mitkä tehtävät ovat meneillään ja milloin ne on määrä suorittaa. Ammattimaisessa pelituotannossa tuotantosuunnitelma mahdollistaa sen, että voidaan tarkasti nähdä, milloin henkilön X vaihetta Y koskevat työtehtävät päättyvät ja milloin hän on käytettävissä muihin tehtäviin. (Vuorela 2007, 67.)

Harrasteprojekteissa uusien tehtävien jakaminen voi olla haastavampaa, mutta projektin vetäjän on silti tärkeää seurata, kuka tekee mitä ja missä vaiheessa tuotantosuunnitelmaa ollaan. Tämä varmistaa, että työskentely etenee suunnitellusti ja että resurssit käytetään tehokkaasti. Tuotantosuunnitelman avulla projektin johtaminen on järjestelmällisempää, mikä edesauttaa projektin onnistumista ja aikataulujen pitämistä. (Vuorela 2007, 67.)

4.3.2 Runko

Peliprojektissa rungosta käytetään usein nimitystä alfaversio, joka viittaa ensimmäiseen pelattavaan versioon. Se mitä alfaversiolla tarkoitetaan voi vaihdella suuresti, mutta tyypillisesti siihen kuuluu vähintäänkin ruudulla liikkuva kuva, joka reagoi ohjaimiin. Alfaversio edustaa pelin ydintä, joka yleensä sisältää alkeellisen pelimoottorin. (Vuorela 2007, 68.)

Runko on pelin ydin, tyypillisesti alkeellinen pelimoottori, johon lisätään pelissä vaadittuja toimintoja sekä grafiikkaa. Rungon ei katsota olevan valmis, ennen kuin kaikki tiimin jäsenet ovat saaneet omat osuutensa valmiiksi alfaversiovaiheessa. (Vuorela 2007, 68.)

4.3.3 Sisältö

Peliprojektissa sisältövaihe on usein vaihe, jossa viimeistään kohdataan aikataulullisia ongelmia. Koska sisältövaiheessa tekeminen on iteratiivista, käy usein niin että kun jotain tehdään, huomataan että sitä pitää muuttaa tai parannella. Parannuksien ja lisäyksien jälkeen huomataan, että ollaan taas pisteessä, jossa tarvitaan lisäyksiä ja parantelua aiempiin ominaisuuksiin. (Vuorela 2007, 69.)

Sisältövaiheessa törmätään usein myös siihen ongelmaan, että kaikkea haluttua sisältöä ei voidakaan syystä tai toisesta tehdä. Tämän takia alkuperäinen sisältö voi kuihtua olemattomiin, kun jokin keskeinen muihin sisällöllisiin elementteihin vaikuttava tekijä joudutaan aikataulullisista syistä jättämään pelin ulkopuolelle. (Vuorela 2007, 69.)

4.3.4 Testaus

Kun pelissä alkaa olla sisältöä sen verran, että peli voidaan pelata alusta loppuun, siitä käytetään nimitystä betaversio. Betaversion eri toiminnallisuuksia testataan mahdollisimman paljon pelin laadun ja toimivuuden varmistamiseksi. (Vuorela 2007, 69.) Tyypillisesti pelitestaus jaetaan useaan eri vaiheeseen, joita ovat muun muassa:

- Funktionaalinen testaus, engl. functional testing, jolla tarkoitetaan testausta, jossa varmistetaan, että pelin mekaniikka sekä kaikki sen ominaisuudet toimivat kuten on haluttu, eikä se sisällä pelaamista estäviä bugeja. (Exploring the Different Types of Game Testing for Quality Assurance.)

- Yhteensopivuustestaus, engl. compatibility testing, jolla tarkoitetaan sitä, että varmistetaan että peli toimii virheettä eri käyttöjärjestelmillä, sekä eri alustoilla, kuten PC: llä, konsoleilla ja mobiililaitteilla. (Exploring the Different Types of Game Testing for Quality Assurance.)

- Suorituskykytestaus, engl. performance testing, jolla tarkoitetaan pelin suorituskyvyn testausta. Suorituskyvyn testauksessa tarkastellaan pelin ruudunpäivitysnopeutta, latausaikoja ja resurssien, kuten muistin ja suorittimen käyttöä. (Exploring the Different Types of Game Testing for Quality Assurance.)

- Käytettävyystestaus, engl. usability testing, jolla tarkoitetaan pelin käyttöliittymän, kontrollien ja yleisen käyttäjäkokemuksen testausta. (Exploring the Different Types of Game Testing for Quality Assurance.)

- Lokalisaatiotestaus, engl. localization testing, jolla tarkoitetaan testausmenetelmää, jossa peliä testataan eri kielillä ja alueilla käännösten toimivuuden ja kulttuurillisten aspektien korrektiuden varalta. (Exploring the Different Types of Game Testing for Quality Assurance.)

- Regressiotestaus, engl. regression testing, jolla tarkoitetaan testausmenetelmää, jossa testataan uudelleen aiemmin pelissä ilmenneiden bugien korjaukset. (Exploring the Different Types of Game Testing for Quality Assurance.)

- Pelitestaus, engl. playtesting, jolla tarkoitetaan testausmenetelmää, jossa kohdeyleisöstä ennalta valittu joukko testaa peliä pelaamalla sitä. Tältä joukolta kerätään palaute pelin sisällöstä, tasosuunnittelusta, pelin vaikeudesta sekä ylipäätään tyytyväisyydestä peliin. (Exploring the Different Types of Game Testing for Quality Assurance.)

4.4 Jälkituotanto

Kun pelin sisältö on lyöty lukkoon ja pelitestaus on päättynyt, pelistä puhuttaessa puhutaan gold-versiosta. Gold-versiolla tarkoitetaan siis valmista peliä, joka julkaistaan. (Vuorela 2007, 72.) Jälkituotannon vaiheeseen kuuluu julkaisun lisäksi myös pelin paketointi, monistus, jakelu sekä mahdollisesti vikakorjausten tekeminen sekä mahdollisen uuden sisällön luominen peliin esimerkiksi ladattavan lisäsisällön muodossa. (Vuorela 2007, 74–84.)

Olennainen osa jälkituotantoa ovat myös myynnin seuraaminen ja palautteen kerääminen sekä tarkastaminen. Jälkituotannon ensimmäisten kuukausien aikana seurataan pelin myyntilukuja, tarkastellaan asiakaspalautetta ja analysoidaan pelin suorituskykymittareita. Näin varmistetaan, että peli on toimiva ja käyttäjäystävällinen laajalle yleisölle. (Indeed Editorial Team 2024.)

5 PELIKEHITYKSEN RAHOITUS

Pelikehityksen rahoitukseen on olemassa useita erilaisia malleja, jotka vaihtelevat riippuen pelin tyypistä ja kehitystiimin resursseista. AAA-peleissä yleensä julkaisija hoitaa pelin rahoituksen kokonaan, sisältäen esimerkiksi kehitysbudjetin, markkinointikustannukset ja muut tuotantoon liittyvät menot. AAA-peleillä julkaisijalla on merkittävä rooli pelin kehityksen ja markkinoinnin tukemisessa, ja he ottavat yleensä suuremman osan taloudellisesta riskistä. (What is AAA game development.)

Indie-peleissä tuotantotiimi hyvin usein hankkii rahoituksen itse. Tämä voi tarkoittaa esimerkiksi joukkorahoituskampanjan käynnistämistä, jossa pelaajat voivat lahjoittaa rahaa pelin kehityksen tukemiseksi. Lisäksi indie-pelien kehittäjät voivat hakea apurahoja, stipendejä tai muita tukimuotoja esimerkiksi kulttuurirahastoilta tai pelialan järjestöiltä. (Financing indie games.)

5.1 Joukkorahoitus

Joukkorahoituksella tarkoitetaan rahoitusmallia, jossa laaja yksityishenkilöiden - tai yritysten joukko lahjoittaa rahaa rahoitettavan kohteen toteutukseen. Joukkorahoitus voi myös auttaa luomaan vuorovaikutusta ja yhteydenpitoa fanien kanssa sekä toimia markkinointikeinona tuleville peleille. Joukkorahoituksen keräämiseen on olemassa useita erikoistuneita sivustoja, esimerkiksi Kickstarter. (Crowdfunding For Video Games — Essentials.)

Joukkorahoitukseen on olemassa erilaisia malleja, kuten esimerkiksi:

- Lahjoituspohjainen joukkorahoitus, jossa projektin tukijat lahjoittavat rahaa tukeakseen projektia ilman odotuksia saada mitään konkreettista vastineeksi. (Four Types of Crowdfunding.)
- Palkkioperustainen joukkorahoitus, jossa tukijat osallistuvat projektin rahoittamiseen ja saavat vastineeksi projektin valmistuessa jonkinlaisen projektiin liittyvän edun tai palkinnon. (Four Types of Crowdfunding.)
- Osakeperusteinen joukkorahoitus, jossa sijoittajat antavat rahoitusta vastineeksi omistusosuuksista projektissa tai yrityksessä ja mahdollistaa projektin tukijoille osakkeenomistajuuden kautta taloudelliset tuotot projektin onnistuessa. (Four Types of Crowdfunding.)

- Velkapohjainen joukkorahoitus, jossa tukijat lainaavat rahaa projektille ja projekti sitoutuu maksamaan varat takaisin korkojen kanssa. (Four Types of Crowdfunding.)

5.2 Apurahat

Pelikehitykseen on haettavissa erilaisia apurahoja, useista eri lähteistä. Esimerkiksi voittoa tavoittelemattomat organisaatiot ja säätiöt voivat myös myöntää apurahoja pelikehitykseen, usein keskittyen tiettyihin teemoihin, kuten koulutuspeliin, yhteiskunnallisiin vaikutuksiin tai pelien terveyshyötyihin. (Financial Support and Funding.) Valtion myöntämät apurahat keskittyvät usein kulttuuriteollisuuden, teknologian kehittämisen ja taloudellisen kasvun edistämiseen. Esimerkiksi kansalliset tai alueelliset taideneuvostot voivat myöntää apurahoja peliprojekteille, joilla on kulttuurista merkitystä tai jotka tukevat paikallista taloutta. (How to Get Grant Funding: A Step-by-Step Guide.)

Apurahoja hakiessa kehittäjien on tyypillisesti toimitettava yksityiskohtaiset ehdotukset, joissa kuvataan peliprojektin tavoitteet, budjetti, aikataulu ja pelin mahdollinen vaikutus. Apurahahakemuksissa vaaditaan usein osoittamaan, miten projekti vastaa rahoittajan tavoitteita ja miten varat käytetään projektin päämäärien saavuttamiseksi. (How to Get Grant Funding: A Step-by-Step Guide.)

5.3 Omakustanne

Omakustanteisuus voi olla myös osittaista, sisältäen sekä kehitystiimin omia varoja, että muita erilaisia rahoitusmalleja. Tyypillisesti peliprojektin omakustanteisella rahoittamisella kuitenkin tarkoitetaan sitä, että kehittäjät rahoittavat projektinsa itse ilman ulkopuolisia sijoittajia tai rahoituslähteitä. Tämä voi tapahtua henkilökohtaisilla säästöillä, omalla varallisuudella tai muilla henkilökohtaisilla resursseilla. Omakustanteinen rahoittaminen antaa kehittäjille täyden hallinnan projektin kaikista osa-alueista, mutta se tuo mukanaan myös omat haasteensa. (The Pros and Cons of Self-Funding Your Small Business.)

Tilanteesta riippuen omakustanteinen rahoittaminen voi olla taloudellisesti haastavaa. Pelikehitys vaatii usein merkittäviä investointeja, ja omakustanteinen rahoittaminen voi kuluttaa runsaasti

kehittäjien henkilökohtaisia säästöjä tai varoja. Omakustanteisuus voi myös rajoittaa peliprojektin laajuutta ja aikarajoja, jos resursseja ja rahoitusta ei ole riittävästi. Omakustanteisessa projektissa kehittäjien onkin erityisen tärkeää laatia realistinen budjetti ja aikataulu sekä varautua mahdollisiin ylimääräisiin kustannuksiin. (Top Tips For Getting Game Development Funding.)

6 DEMOPROJEKTI - RACING GAME

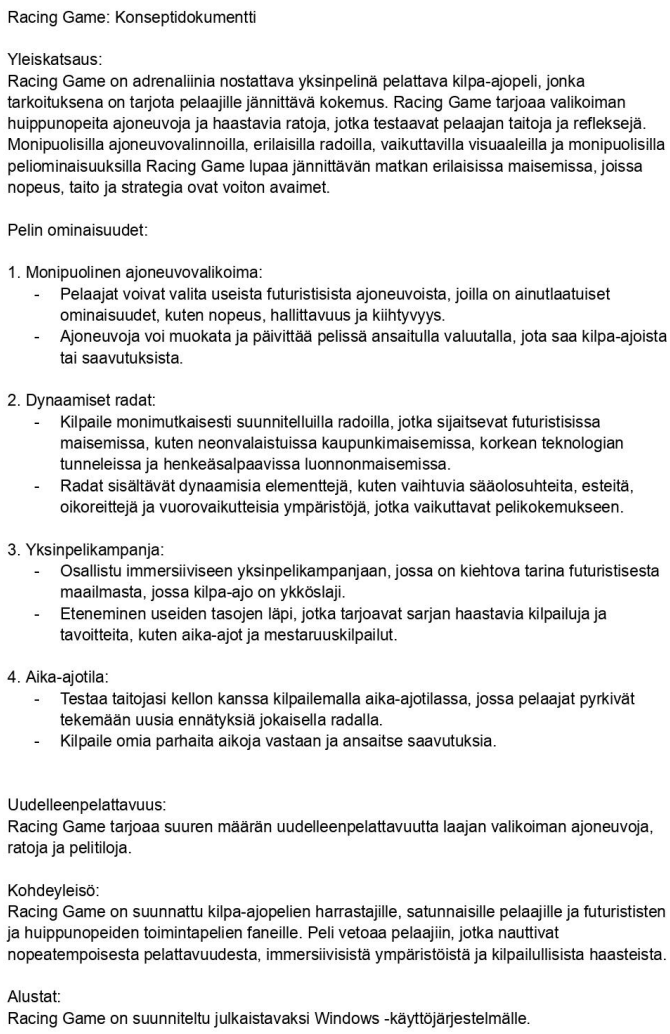
Tässä demoprojektissa on käytössä kaksi karttaa ja päävalikko, jotka tarjoavat pelaajalle perustason pelikokemuksen. Pelissä ei ole tarkoitus julkaista, vaan sen pääasiallinen tavoite on toimia indie-pelikehityksen kannattavuuden mittarina. Demoprojekti on suunniteltu niin, että se mahdollistaa pelikehityksen keskeisten näkökohtien arvioimisen ja analysoimisen, ilman tarvetta laajentaa tai julkaista peliä kaupallisesti.

Demoprojektin avulla pyritään demonstroimaan pelinkehityksen prosesseja ja haasteita, jotka ovat keskeisiä indie-pelikehityksen kannattavuuden arvioimisessa. Päävalikko ja kartat on valittu tarjoamaan yksinkertainen mutta riittävä alusta, jolla voidaan tutkia pelin perusmekaniikkoja ja -toimintoja. Tämä lähestymistapa mahdollistaa projektin keskittyvän olennaisiin kehitysvaiheisiin ja tutkimusaiheisiin, jotka tarjoavat arvokasta tietoa indie-pelikehityksen käytännöistä ja taloudellisista näkökohdista.

6.1 Konsepti

Pelin perusidea on geneerinen kilpa-autopeli (KUVA 10.), jossa pelaaja kilpailee kolmea tietokoneen ohjaamaa vastustajaa vastaan. Tyypillisesti kilpa-autopeleissä yksinkertainen mutta koukuttava pelimekaniikka tarjoaa pelaajalle kilpailullisen kokemuksen, jossa keskiössä ovat ajotaito ja strateginen suunnittelu.

Vaikkakaan opinnäytetyötä varten tehtävä demoprojekti ei tule olemaan täysimittainen peli, on demoprojektin konseptoinnissa ja toteutuksessa on kuitenkin otettu huomioon skaalautuvuus, jotta se voisi toimia pohjana täysimittaiselle pelille. Tämä tarkoittaa, että vaikka projekti on demoversio, sen kehittämisessä on pyritty luomaan perusta, joka mahdollistaa mahdollisten laajennusten ja lisäominaisuuksien sisällyttämisen tulevaisuudessa.



Racing Game: Konseptidokumentti

Yleiskatsaus:
Racing Game on adrenaliinia nostattava yksinpelinä pelattava kilpa-ajopeli, jonka tarkoituksena on tarjota pelaajille jännittävä kokemus. Racing Game tarjoaa valikoiman huippunopeita ajoneuvoja ja haastavia ratoja, jotka testaavat pelaajan taitoja ja refleksejä. Monipuolisilla ajoneuvovalinnoilla, erilaisilla radoilla, vaikuttavilla visuaaleilla ja monipuolisilla peliominaisuuksilla Racing Game lupaa jännittävän matkan erilaisissa maisemissa, joissa nopeus, taito ja strategia ovat voiton avaimet.

Pelin ominaisuudet:

1. Monipuolinen ajoneuvovalikoima:
   - Pelaajat voivat valita useista futuristisista ajoneuvoista, joilla on ainutlaatuiset ominaisuudet, kuten nopeus, hallittavuus ja kiihtyvyys.
   - Ajoneuvoja voi muokata ja päivittää pelissä ansaitulla valuutalla, jota saa kilpa-ajoista tai saavutuksista.

2. Dynaamiset radat:
   - Kilpaile monimutkaisesti suunniteltuilla radoilla, jotka sijaitsevat futuristisissa maisemissa, kuten neonvalaistuissa kaupunkimaisemissa, korkean teknologian tunneleissa ja henkeäsalpaavissa luonnonmaisemissa.
   - Radat sisältävät dynaamisia elementtejä, kuten vaihtuvia sääolosuhteita, esteitä, oikoreittejä ja vuorovaikutteisia ympäristöjä, jotka vaikuttavat pelikokemukseen.

3. Yksinpelikampanja:
   - Osallistu immersiiviseen yksinpelikampanjaan, jossa on kiehtova tarina futuristisesta maailmasta, jossa kilpa-ajo on ykköslaji.
   - Eteneminen useiden tasojen läpi, jotka tarjoavat sarjan haastavia kilpailuja ja tavoitteita, kuten aika-ajot ja mestaruuskilpailut.

4. Aika-ajotila:
   - Testaa taitojasi kellon kanssa kilpailemalla aika-ajotilassa, jossa pelaajat pyrkivät tekemään uusia ennätyksiä jokaisella radalla.
   - Kilpaile omia parhaita aikoja vastaan ja ansaitse saavutuksia.

Uudelleenpelattavuus:
Racing Game tarjoaa suuren määrän uudelleenpelattavuutta laajan valikoiman ajoneuvoja, ratoja ja pelitiloja.

Kohdeyleisö:
Racing Game on suunnattu kilpa-ajopelien harrastajille, satunnaisille pelaajille ja futuristisen ja huippunopeiden toimintapelien faneille. Peli vetoaa pelaajiin, jotka nauttivat nopeatempoisesta pelattavuudesta, immersiivisistä ympäristöistä ja kilpailullisista haasteista.

Alustat:
Racing Game on suunniteltu julkaistavaksi Windows -käyttöjärjestelmälle.

Kuva 10. Racing Game konseptidokumentti.

6.2 Käytettävät työkalut

Työkaluiksi demoprojektille on valittu Unity3D -pelimoottori, jolloin pelin skriptit kirjoitetaan C# -ohjelmointikielellä. Ohjelmointiympäristöksi esimerkkitapaukseen on valittu Visual Studio Community Edition sen nopeuden sekä helppokäyttöisyyden vuoksi. Unity3D tarjoaa kattavat työkalut pelin kehittämiseen, ja C# -ohjelmointikieli on hyvin tuettu sekä dokumentoitu, mikä tekee siitä sopivan valinnan.

Koska pelissä ei tule olemaan hahmojen välistä vuorovaikutusta, Unity Asset Storesta saatavat valmiit äänipaketit riittävät pelin toteutukseen. Tämä vähentää merkittävästi ajankäyttöä sekä kustannuksia, sillä valmiit äänipaketit voivat kattaa kaikki pelin tarpeet ilman tarvetta luoda tai ostaa erikseen räätälöityjä äänitehosteita.

Myös demoprojektin grafiikkaan käytettävät assetit on hankittu taloudellisista ja ajankäytöllisistä syistä ilmaiseksi Unity Asset Storesta. Tämä lähestymistapa mahdollistaa projektin toteuttamisen ilman suuria investointeja ulkoisiin grafiikkakustannuksiin tai lisenssimaksuihin, mikä auttaa pitämään projektin kustannukset kurissa. Lisäksi ilmaiset assetit tarjoavat laajan valikoiman vaihtoehtoja, jotka ovat riittäviä demoversion tarpeisiin.

6.3 Kehityspäiväkirja

22.05.2023 – 2 tuntia

Luotu karkea terrainkartta muutamalla puulla, johon lisätty auto ja lisätty autolle controlleri.

23.05.2023 - 2 tuntia

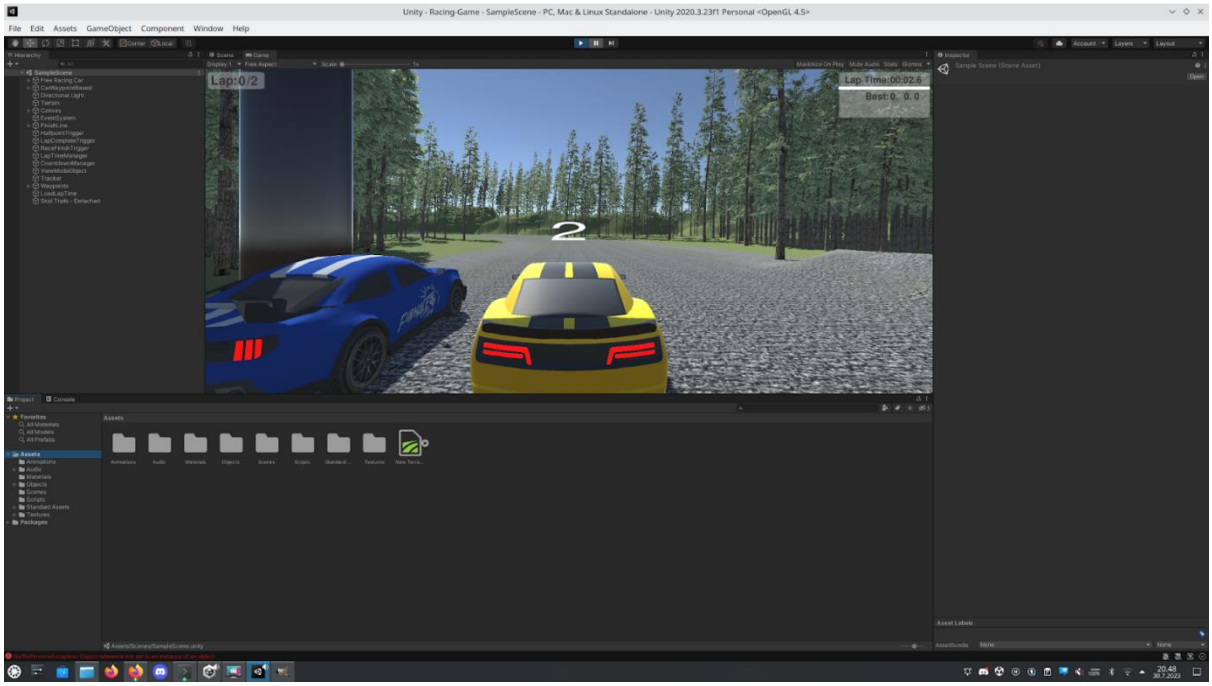
Vaihdettu aiemmat demoassetit uusiin, Unity Storesta hankittuihin, lisenssointikysymysten vuoksi.

24.05.2023 - 5 tuntia

Lisätty “AI” vastustaja. Tässä päädyttiin käyttämään pohjana 2018 standard asseteista waypoint car -objektia, jolle voi antaa radalta erilaisia waypointeja. Lisätty myös kamerakulman vaihto -ominaisuus.

05.06.2023 - 4 tuntia

Lisätty musiikkia. Lisätty Lap time counter, parannettu ohjausta hieman ja tehty kisan päättyminen



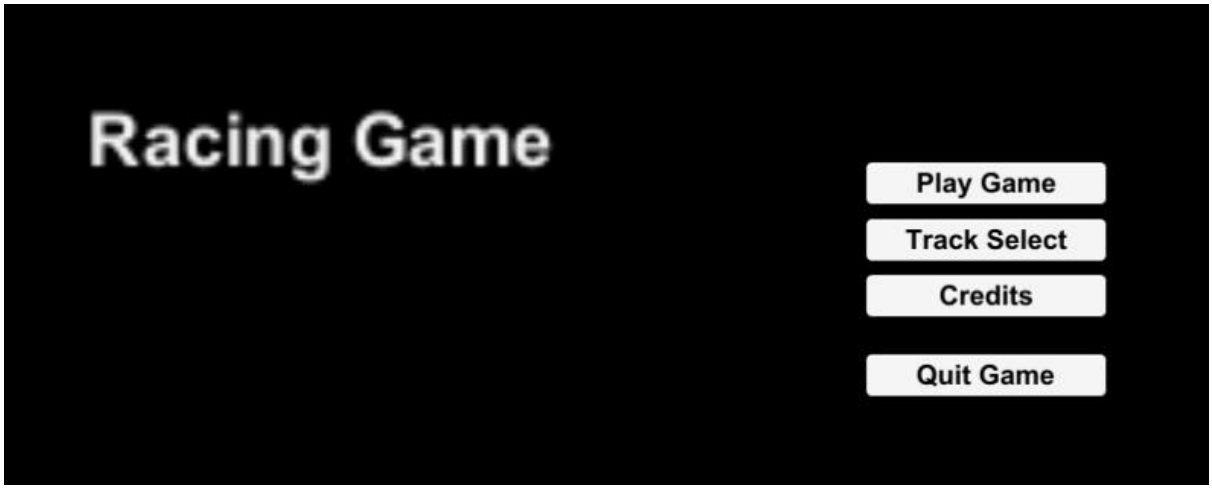
Kuva 11. Demoprojekti Unity Editorissa.

08.08.2023 - 1 tunti

Korjattu auton ohjausta.

08.08.2023 - 1 tunti

Lisätty aloitusvalikko ja radan valitsemisvalikko.



Kuva 12. Aloitusvalikko.

Kuva 13. Radan valitsemisvalikko.

09.08.2023 - 2 tuntia

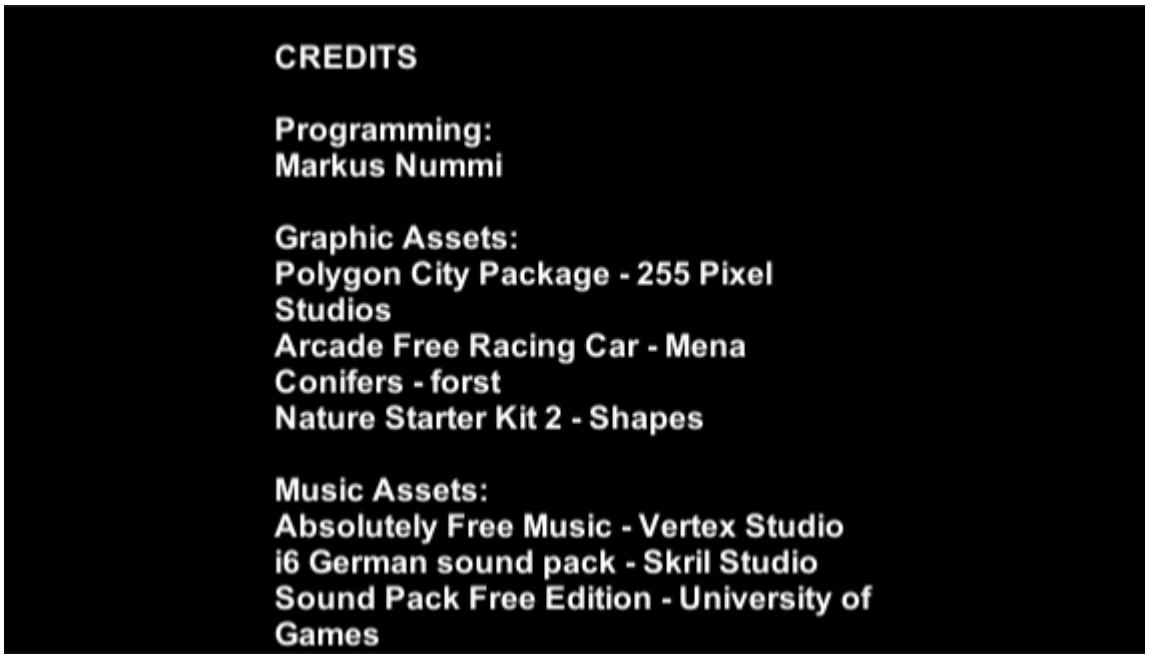
Toisen radan suunnittelu editorissa

16.08.2023 - 2 tuntia

Toisen radan suunnittelu editorissa

27.08.2023 – 1 tunti

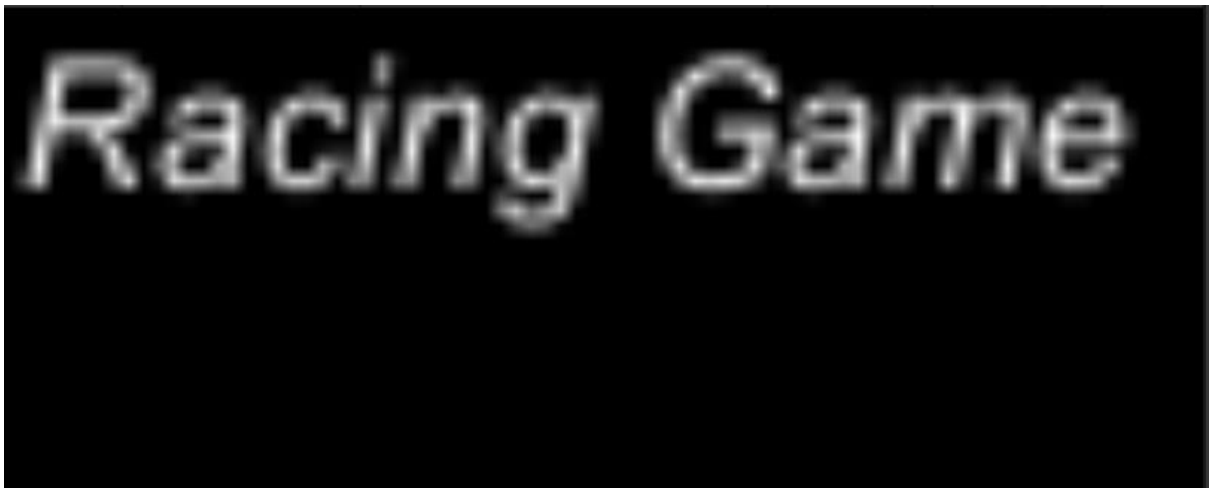
Luotu lopputekstit -kohtaus ja sille animaatio.



Kuva 14. Lopputekstit.

27.08.2023 – 1 tunti

Luotu Splash-scene ja sille animaatio. Lisäksi tehty build settingsit, lisätty iconit buildille ja buildattu development build.



Kuva 15. Splash -kohtaus.

6.4 Tiedossa olevia ohjelmointivirheitä

Projektia kehittäessä on vastaan tullut useita erilaisia ohjelmointivirheitä. Oheisessa taulukossa käy alfaversiosta löytyvät suurimmat ongelmat ja karkea arvio siitä, kuinka paljon niiden korjaamiseen menisi aikaa.

Taulukossa 1 on lueteltu ongelmat, kuten virheelliset skriptit, jotka aiheuttavat pelin kaatumisen, grafiikkavirheet, jotka vaikuttavat pelin visuaaliseen laatuun, sekä ääniongelmat, jotka heikentävät pelin äänenlaatua. Jokaisen ongelman kohdalla on myös arvioitu korjaamiseen tarvittava aika, mikä auttaa projektin tiimiä priorisoimaan tehtäviä ja suunnittelemaan työvaiheita tehokkaammin. Näiden arvioiden avulla voidaan myös varmistaa, että aikarajoja ja budjettia voidaan säilyttää realistisina ja että kehitystyö etenee hallitusti.

TAULUKKO 1. Tiedossa olevat bugit

Bugi	Arvioitu korjaukseen menevä aika
Ongelmia ohjauksessa	5h
Kaupunkikartassa on mahdollista ajaa radalta ulos	5h
Autojen äänet käyttäytyvät miten sattuu	4h
Lukuisia puutteita AI :n käyttäytymisessä, törmäilyä yms.	12h

6.5 Jatkokehityksen vaatimat resurssit

Demoprojektin jatkokehittäminen valmiiksi tuotteeksi vaatisi runsaasti erinäisiä resursseja. Taulukossa 2 on tehty alustavia arvioita siitä, kuinka paljon aikaa ja rahaa vaatisi pelin kehittäminen alfaversiosta betaversioon. Taulukossa on eriteltävä kunkin kehitysvaiheen resurssitarpeet, mukaan lukien bugikorjaukset, ajoneuvopakettien hankinta ja lisääminen, ajoneuvoluokkien ohjelmointi, sekä uusien kenttien suunnittelu, toteutus ja ohjelmointiosuus.

Esimerkiksi bugikorjaukset arvioidaan vievän tässä vaiheessa yhteensä vähintään 100 tuntia, mikä korostaa ohjelmointivirheiden korjaamisen merkittävää osuutta projektin kokonaisajasta. Ajoneuvopakettien hankinta ja lisääminen vie noin 8 tuntia, kun taas ajoneuvoluokkien ohjelmointi

arvioidaan vievän 12 tuntia. Uusien kenttien suunnittelu ja toteutus sekä niiden ohjelmointiosuus yhdessä vaativat huomattavan määrän aikaa, yhteensä 54 tuntia. Näiden arvioiden avulla voidaan suunnitella ja aikatauluttaa kehitystyötä tehokkaasti, varmistaa resurssien optimaalinen käyttö ja valmistella tarvittavat budjetointitoimenpiteet. Koska alfaversiosta betaversioon kehittämisen ajalliseksi määräksi arvioidaan noin 192 tuntia, betaversiosta julkaisukelpoisen version kehittämiseen voitaneen karkeasti arvioiden sanoa menevän noin 500 tuntia.

TAULUKKO 2. Alfaversiosta betaversioon kehittämisen vaatimat resurssit.

Resurssi	Aika
Bugikorjaukset ja testaus	100h
Ajoneuvopakettien hankinta ja lisääminen	8h
Ajoneuvoluokkien ohjelmointi	12h
Asset -paketit uusiin kenttiin	30h
Uusien kenttien suunnittelu ja toteutus	12h
Uusien kenttien ohjelmointiosuus	30h

7 PÄÄTELMÄT

Kuten kehityspäiväkirjasta on nähtävissä, on demoprojektin kehittämiseen mennyt aikaa tähän mennessä noin 21 tuntia. Kehityspäiväkirjasta on nähtävissä myös, että koska projektille ei ole rahoitusta ja se on toteutettu päivätöiden ja muiden askareiden ohessa, on kehitykseen mennyt ajallisesti huomattavan kauan aikaa.

Pelin kehittäminen alfaversiosta betaversioon ottaa arvion mukaan noin 192 työtuntia, ja pelin saaminen betaversiosta julkaisukelpoiseksi ja kilpailijoidensa kanssa samalle tasolle vaatisi karkean arvion mukaan vähintään 500 työtuntia.

Mikäli kehitystyölle laskee hintaa 25 e/h, tulee karkean arvion mukaan pelkästään työn hinnaksi betaversiolle noin 4 800 euroa ja julkaisukelpoiselle versiolle noin 12 500 euroa. Yhteensä tämä tekee noin 692 työtuntia ja 17 300 euroa. Pikainen vilkaisu Unity Asset Storen assetteihin kertoo, että musiikkipakkauksiin, eri automalleihin sekä ratasuunnittelutyökaluihin pitäisi varata vähintään 500 euroa ja tällöinkin puhutaan vain yksinpelinä pelattavasta pelistä, joka ei nykypäivän standardeilla ole kovinkaan houkutteleva, ellei kehitystiimi onnistu luomaan peliin jotain poikkeuksellista sisältöä.

Vastaavia tuotteita markkinoilla on runsaasti, kuten esimerkiksi Milwoo Studion kehittämä Joyride. Joyriden hinta on verrattain matala, 3,99 euroa ja se sisältää pitkälti samat ominaisuudet kuin demoprojektille konseptidokumentissa on määritelty. Mikäli ajatellaan että esimerkiksi julkaisualusta Steamille maksetaan jokaisesta myydystä tuotteesta 30 % provisio, tämä tarkoittaa, että peliin menevät kulut kattaakseen pelistä tulisi myydä 3,99 euron hintaan vähintään 5 800 kopiota. Steam ei julkaise peleistään myyntilukuja, mutta esimerkkinä käytetty Joyride on saanut kirjoitushetkellä Steamissa vain 21 arvostelua, joten lienee turvallista olettaa, että sitä ei ole myyty 5800 kopiota, tai mikäli on, pelillä ei juurikaan ole tehty voittoa. Epic Games Storessa provisiopalkkio on 12 % myydystä tuotteesta mikä tarkoittaa, että pelin kehityksen kulut kattaakseen pelistä tulisi myydä noin 5 000 kopiota. Epic Games Storen laadunvalvonta on kuitenkin tarkempaa, joten hyvin todennäköisesti Joyriden ja esimerkkinä käytetyn Racing Gamen laatu ei yltäisi siihen, että pelit voisi siellä julkaista. Tuloksista voidaan siis todeta, ettei pienen budjetin indie-pelikehitystä ainakaan ensi alkuun kannata lähteä harjoittamaan niinkään liiketoiminnalliset ja voittoa tavoittelevat tavoitteet mielessä, vaan ehkä pikemminkin huvin ja harrastuksen vuoksi.

LÄHTEET

A Quick Guide to Software Developer Hierarchy. Saatavissa: https://www.howdy.com/blog/redefining-seniority-from-ticket-closers-to-problem-framers. Viitattu 28.10.2024.

About Audacity. Saatavissa: https://www.audacityteam.org/FAQ/. Viitattu 29.03.2023.

Animation Techniques. Saatavissa: https://www.linkedin.com/pulse/6-animation-techniques-frank-govaere-xyhaf. Viitattu 29.03.2023.

Blender. Saatavissa: https://www.blender.org/. Viitattu 29.03.2023.

Bosca Ceoil. Saatavissa: https://yurisizov.itch.io/boscaceoil-blue. Viitattu 29.03.2023.

Bramble R. 2023. The Seven Stages Of Game Development. Saatavilla: https://gamemaker.io/en/blog/stages-of-game-development#production. Viitattu: 03.05.2024.

Concept artist (Games). Saatavissa: https://www.screenskills.com/job-profiles/browse/games/art/concept-artist-games/. Viitattu 29.03.2023.

Crowdfunding For Video Games — Essentials. Saatavissa: https://medium.com/icopartners/crowdfunding-for-video-games-essentials-4b2d5f35a681. Viitattu 28.10.2024.

Eclipse Public License. Saatavissa: https://www.linux.fi/wiki/Eclipse_Public_License. Viitattu 29.03.2023.

Eclipse documentation. Saatavissa: https://help.eclipse.org/latest/index.jsp. Viitattu 29.03.2023.

Exploring the Different Types of Game Testing for Quality Assurance. Saatavissa: https://www.pixelqa.com/blog/post/types-of-game-testing-for-quality-assurance. Viitattu 29.03.2023.

Financial Support and Funding. Saatavissa: https://www.playfinland.fi/financial-support-and-funding. Viitattu 28.10.2024.

Financing indie games. Saatavissa: https://www.futurelearn.com/info/courses/introduction-to-indie-games/0/steps/96363. Viitattu 28.10.2024.

Four Types of Crowdfunding. Saatavissa: https://legalvision.co.uk/corporations/types-crowdfunding/. Viitattu 28.10.2024.

Godot Docs. Saatavissa: https://docs.godotengine.org/en/3.0/about/introduction.html. Viitattu 29.03.2023.

GIMP. Saatavissa: https://www.gimp.org/. Viitattu 29.03.2023.

Gregory, J. 2015. Game Engine Architecture. 2., painos. Boca Raton, FL: Taylor & Francis Group, LLC. Viitattu 29.03.2023.

Horowitz, S & Looney, S 2014. The Essential Guide to Game Audio. Focal Press, Burlington, MA. Viitattu 29.03.2023.

How Do You Optimize Game Art Assets on Blender? Saatavissa: https://vagon.io/blog/how-to-optimize-game-assets-on-blender. Viitattu 29.03.2023.

How to Get Grant Funding: A Step-by-Step Guide. Saatavissa: https://www.lindushealth.com/blog/how-to-get-grant-funding-a-step-by-step-guide. Viitattu 28.10.2024.

Indeed Editorial Team. 2024. Complete Guide to the 7 Gaming Development Stages. Saatavilla: https://www.indeed.com/career-advice/career-development/gaming-development-stages. Viitattu 09.09.2024.

Intro to Github for version control. Saatavissa: https://ourcodingclub.github.io/tutorials/git/. Viitattu 29.03.2023.

Licensing. Saatavissa: https://www.unrealengine.com/en-US/license. Viitattu 29.03.2023.

License. Saatavissa: https://godotengine.org/license/. Viitattu 29.03.2023.

LibreSprite. Saatavissa: https://libresprite.github.io/#!/. Viitattu 29.03.2023.

Machado, D. 2017. Why we choose Godot Engine. Saatavilla: https://medium.com/rock-milk/why-godot-engine-e0d4736d6eb0. Viitattu 29.03.2023.

Plans and pricing. Saatavissa: https://unity.com/pricing. Viitattu 29.03.2023.

Pelin äänisuunnittelija. Saatavissa: https://tyomarkkinatori.fi/ammattitieto/ammatit/pelin-aanisuunnittelija. Viitattu 29.03.2023.

Pelitestaaja. Saatavissa: https://tyomarkkinatori.fi/ammattitieto/ammatit/pelitestaaja. Viitattu 29.03.2023.

Race'n'Chase Game Design. Saatavilla: https://www.gamedevs.org/uploads/grand-theft-auto.pdf. Viitattu 29.03.2023.

Schell, J 2008. The Art of Game Design. Morgan Kaufmann Publishers. Burlington, MA. Viitattu 29.03.2023.

Soundscaping - The Art Of Soundscape Creation. Saatavissa: https://www.audiocube.app/blog/soundscaping. Viitattu 29.03.2023.

The Creator of Lights in Games. Saatavissa: https://about.ncsoft.com/en/news/article/theoriginality-kch. Viitattu 29.03.2023.

The Importance of Sound Design in Video Game Development. Saatavissa: https://daracrawford.com/audio-blog/game-audio-project-management-part-3-sound-design. Viitattu 29.03.2023.

The Pros and Cons of Self-Funding Your Small Business. Saatavissa: https://rauva.com/blog/self-funding-pros-and-cons. Viitattu 28.10.2024.

The Python IDE for data science and web development. Saatavissa: https://www.jetbrains.com/pycharm/. Viitattu 29.03.2023.

Top Tips For Getting Game Development Funding. Saatavissa: https://www.perforce.com/blog/vcs/game-development-funding. Viitattu 28.10.2024.

Video Game Development Process. Saatavissa: https://www.nuclino.com/articles/video-game-development-process. Viitattu 28.10.2024.

What does a 3D modeler do: Understanding the Role and Responsibilities of a 3D Modeler. Saatavissa: https://www.coohom.com/article/what-does-a-3d-modeler-do-9690. Viitattu 29.03.2023.

What is AAA game development. Saatavissa: https://ejaw.net/what-is-aaa-game-development/. Viitattu 28.10.2024.

What is an IDE (Integrated Development Environment)? Saatavissa: https://aws.amazon.com/what-is/ide/. Viitattu 29.03.2023.

What is a Video Game Artist? Saatavissa: https://www.tealhq.com/career-paths/video-game-artist. Viitattu 28.10.2024.

What Is a Video Game Designer? Saatavissa: https://www.wgu.edu/career-guide/information-technology/video-game-designer-career.html. Viitattu 29.03.2023.

What is Visual Studio? Saatavissa: https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022. Viitattu 29.03.2023.

LIITE 1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CarController : MonoBehaviour
{
    public float enginePower = 150.0f;
    public float power = 0.0f;
    public float brake = 0.0f;
    public float steer = 0.0f;
    public float maxSteer = 80.0f;
    public WheelCollider Wheel_LF;
    public WheelCollider Wheel_RF;
    public WheelCollider Wheel_LR;
    public WheelCollider Wheel_RR;
    public static bool CanControl = false;

    //AI Variables
    public GameObject AiCar01;

    void Start()
    {
        GetComponent<Rigidbody>().centerOfMass = new Vector3(0f, -0.5f, 0.3f);
    }
```

Koodi 1. Auton ohjauksen julkiset muuttujat ja start -funktio.

LIITE 2/1

```
void Update()
   {
      if (!CanControl)
      {
         brake = GetComponent<Rigidbody>().mass * 10.0f;
         Wheel_LR.motorTorque = 0.0f;
         Wheel_RR.motorTorque = 0.0f;
         Wheel_LF.brakeTorque = brake;
         Wheel_RF.brakeTorque = brake;
         Wheel_LR.brakeTorque = brake;
         Wheel_RR.brakeTorque = brake;
      }

      if (CanControl)
      {
         //Activating AI Car control as well
         AiCar01.GetComponent<UnityStandardAssets.Vehicles.Car.CarAIControl>().enabled = true;

         if (Input.GetKey(KeyCode.W))
         {
            power = enginePower * Time.deltaTime * 650.0f;

            Wheel_LR.motorTorque = power;
            Wheel_RR.motorTorque = power;
            Wheel_LF.brakeTorque = 0;
            Wheel_RF.brakeTorque = 0;
            Wheel_LR.brakeTorque = 0;
            Wheel_RR.brakeTorque = 0;
         }
         else if (Input.GetKey(KeyCode.S))
         {
            power = -enginePower * Time.deltaTime * 650.0f;
            Wheel_LR.motorTorque = power;
            Wheel_RR.motorTorque = power;
            Wheel_LF.brakeTorque = 0;
            Wheel_RF.brakeTorque = 0;
            Wheel_LR.brakeTorque = 0;
            Wheel_RR.brakeTorque = 0;
         }
         else if (Input.GetKey(KeyCode.Space))
         {
            brake = GetComponent<Rigidbody>().mass * 15.0f;
            Wheel_LF.brakeTorque = brake;
            Wheel_RF.brakeTorque = brake;
            Wheel_LR.brakeTorque = brake;
```

LIITE 2/2

```
            Wheel_RR.brakeTorque = brake;
            Wheel_LR.motorTorque = 0.0f;
            Wheel_RR.motorTorque = 0.0f;
        }
        //else if (Input.GetKey(KeyCode.A))
        //{
        //    Wheel_LF.steerAngle = -maxSteer;
        //}
        //else if (Input.GetKey(KeyCode.D))
        //{
        //    Wheel_RF.steerAngle = maxSteer;
        //}
        else
        {
            Debug.Log("ELSE!");
            brake = GetComponent<Rigidbody>().mass * 2.0f;
            Wheel_LF.brakeTorque = brake;
            Wheel_RF.brakeTorque = brake;
            Wheel_LR.brakeTorque = brake;
            Wheel_RR.brakeTorque = brake;
            Wheel_LR.motorTorque = 0.0f;
            Wheel_RR.motorTorque = 0.0f;
            Wheel_LF.motorTorque = 0.0f;
            Wheel_RF.motorTorque = 0.0f;
            power = 0;
        }
        steer = Input.GetAxisRaw("Horizontal") * maxSteer;
        Wheel_LF.steerAngle = steer;
        Wheel_RF.steerAngle = steer;
    }
}
```

Koodi 2. Auton ohjauksen update -funktio

LIITE 3

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class ButtonOptions : MonoBehaviour
{
    public void MainMenu()
    {
        SceneManager.LoadScene(1);
    }
    public void PlayGame()
    {
        SceneManager.LoadScene(3);
    }
    public void TrackSelection()
    {
        SceneManager.LoadScene(2);
    }
    public void Track01()
    {
        SceneManager.LoadScene(3);
    }
    public void Track02()
    {
        SceneManager.LoadScene(4);
    }
    public void Credits()
    {
        SceneManager.LoadScene(5);
    }
    public void QuitGame()
    {
        Application.Quit();
    }
}
```

Koodi 3. C# -luokka jolla hoidetaan UI-toiminnot.

LIITE 4

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Credits : MonoBehaviour
{
    void Start()
    {
        StartCoroutine(FinishCredits());
    }
    IEnumerator FinishCredits()
    {
        yield return new WaitForSeconds(16);
        SceneManager.LoadScene(0);
    }
}
```

Koodi 3. Luokka, jolla hoidetaan lopputekstien animaation jälkeinen kohtauksen vaihto