Kristian Kytönen

PELIN SUUNNITTELU JA TEKEMINEN

Tietojenkäsittelyn koulutusohjelma

2018

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

PELIN SUUNNITTELU JA TEKEMINEN

Kytönen, Kristian
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Marraskuu 2018
Ohjaaja: Nuutinen Petri
Sivumäärä: 41
Liitteitä:

Asiasanat: Peli, Pelimoottori, Unity

Tässä työssä tutkittiin pelin tekemistä ja suunnittelua teoreettiselta kannalta sekä tehtiin itse oma demo-peli. Olen itse pelannut pelejä useasti, mutta en aikaisemmin ollut perehtynyt pelien tekemiseen. En myöskään ollut tehnyt peliä itse tai käyttänyt pelimoottoria, joten siinä oli oma opettelunsa.

Erilaisia peligenrejä on monenlaisia ja ne vaativatkin erilaisia lähestymistapoja. Toiset voivat vaatia ison maailman, kun taas toisissa vaaditaan nopeaa reagointia pelaajalta. Työssä tutkittiin myös pelin tekemisen ja suunnittelun eri vaiheita. Näitä ovat muun muassa pelimekaniikat, hahmo ja tarinan suunnittelu sekä myöskin grafiikoiden ja äänien suunnittelu. Pelimoottori on ohjelmistokehys, jonka avulla pelejä on helpompi tehdä, sillä ne sisältävät paljon valmiita ominaisuuksia. Tutkin myös pelimoottoreiden rakennetta tarkemmin. Ne sisältävätkin oman fysiikkamoottorinsa ja paljon muita ominaisuuksia. Työssä selvitettiin myös kolmen erilaisen pelimoottorin tietoja tarkemmin.

Itse pelin tekemiseen käytin Unity-pelimoottoria, grafiikoihin Gimp-kuvankäsittelyohjelmaa, sekä ohjelmointiin Visual Studiota ja C#-ohjelmointikieltä. Pelistä tuli yksinkertainen tasohyppelypeli, jossa tarkoituksena on liikkua pelihahmolla kentässä eteenpäin ja varoa erilaisia vaaroja sekä edetä seuraavaan kenttään. Ottaen huomioon, että tämä on ensimmäinen pelini, olen tyytyväinen lopputulokseen.

GAME DESIGNING AND MAKING

Kytönen, Kristian
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Business information systems
November 2018
Supervisor: Nuutinen Petri
Number of pages: 41
Appendices:

Keywords: Game, Game engine, Unity

The purpose of this thesis was to study the development and design of the game making from a theoretical perspective and to make my own game. I have played games a lot but have not been familiar with the game development before. I have not done a game before or have not used any game engine before, so I had some learning to do.

There are many kinds of game genres and they require different kinds of approaches. Others may require large worlds and another may require quick reactions from the player. The thesis also studied the various stages of making and designing a game. These include for example game mechanics, character and story design and design of graphics and sounds. A game engine is a framework that makes making games easier. They include their own physics engine and many other features. The thesis also studied a bit more about three different game engines.

For the game I used the Unity game engine, for the graphics I used a GIMP graphics editor and for the programming Visual Studio and C# programming language. The game was going to be a simple platform game where the aim is to move around the level and watch out for different kinds of hazards and to reach the end of the level. Even thought this is my first game, I am happy with the result.

SISÄLLYS

1 JOHDANTO

Pelit ovat nykyään todella suuressa suosiossa ja kyseessä on isot markkinat. Tekijöitä vaaditaan tälläkin alalla paljon moniin erilaisiin tehtäviin. Myös yksittäiset ihmiset voivat tehdä pelejä ja niistä voi onnella tulla todella suosittujakin. Juuri erilaisten pelimoottoreiden avulla kuka tahansa voi tehdä oman pelinsä.

Esittelen tässä työssä ensin teoriaa peleistä ja niiden tekemisestä. Selitän ensin minkälaisia ja tyyppisiä pelejä voi olla, minkä jälkeen selvitän tarkemmin erilaisia työ- ja suunnitteluvaiheita, ja mitä pelien suunnittelu ja tekeminen vaatii. Miten muun muassa pelin mekaniikat toteutetaan ja mitä tulisi ottaa huomioon kenttä suunnittelussa. Teoriaosuuden lopuksi selitän vielä, mitä ovat pelimoottorit ja minkälaisia ominaisuuksia ne sisältävät. Kerron myös kolmesta erilaisesta pelimoottorista ja niiden ominaisuuksista sekä maksullisuuksista.

Varsinaisena työnä teen itse pelin käyttäen Unity-pelimoottoria. Kyseessä on ensimmäinen pelini, enkä myöskään ole käyttänyt pelimoottoria ennen, joten peli tulee olemaan melko yksinkertainen. Selitän tässä työssä pelin idean ja tavoitteet sekä toteutustavan. Esittelen myös eri työkalut, joita tarvitsin pelin tekemiseen. Näytän myös tekemäni grafiikat ja koodit, joita peli tarvitsee toimiakseen.

2 GENRET JA PELISUUNNITTELU

2.1 Genret

Peliä suunnitellessa on hyvä valita ensin, että minkälaista peliä lähtee tekemään. Peleistä onkin olemassa erilaisia genrejä/tyyppejä ja niiden alamuotoja. Pelin ei myöskään ole pakko noudattaa vain jotain tiettyä genreä, vaan niitä voidaan yhdistää keskenään. Käytännössä jokaisesta genrestä on myöskin mahdollista toteuttaa yksin- tai moninpeli.

2.1.1 Toimintapelit

Toimintapeleissä silmän ja käden yhteistyö on tärkeää. Pelaajat kohtaavat erilaisia haasteita ja niihin tarvitaan nopeaa reagointia. Useimmiten pelaaja liikkuu hahmolla tarkoituksenaan voittaa erilaisia haasteita ja vihollisia. Pelaaja voi joutua myös keräämään erilaisia esineitä ja ratkaisemaan pulmia ja arvoituksia. Toimintapelit sisältävät muun muassa ammuntapelit, tappelupelit ja tasohyppelypelit. (2012 Mitchell, 27-28.)

Toimintapelit ovat yksi suosituimmista peligenreistä ja ne sopivatkin erityisen hyvin pelaajille, jotka tarinan sijaan kaipaavat nopeatempoista toimintaa. Toimintapelejä ovat esimerkiksi Street Fighter, jossa tarkoituksena on tapella toista pelaajaa vastaan. Call of duty, jossa tarkoituksena on ampua viholliset. Myös Mario tasohyppelypelinä kuuluu toimintapeleihin. (2017 Mirillis Team.)

2.1.2 Seikkailupelit

Seikkailupelit perustuvat johonkin tarinaan ja maailmaan, jossa pelaajan on tarkoitus liikkua hahmolla ja suorittaa erilaisia tehtäviä ja pulmia. Pelimaailma on useimmiten laaja ja kompleksinen. Tarinaan on panostettu ja hahmot pelissä ovat mielenkiintoisia. Pelit voivat olla toimintapelien tyyppisiä, mutta sisältävät enemmän tarinaa ja

seikkailua. Seikkailupelit voivat olla myös vuoropohjaisia. Ensimmäiset seikkailupelit olivatkin tekstipohjaisia pelejä. (2004 Bates, 6.)

Seikkailupelit sopivatkin siis pelaajille, jotka haluavat keskittyä tarinaan sekä ratkaista erilaisia pulmia. Monesti pelaajat joutuvat myös miettimään seuraavia liikkeitään ja minne lähteä seikkailemaan seuraavaksi. Seikkailu pelejä ovat esimerkiksi Siberia ja Resident Evil. (2017 Mirillis Team.)

2.1.3 Roolipelit

Roolipeleissä hahmolla liikutaan pelimaailmassa ja pelkän voiman sijaan hahmolla on käytössään usein myös taikavoimia tai muita kykyjä. Pelaajan voi olla myös mahdollista valita pelihahmonsa useista erilaisista hahmoluokista. Pelissä voi taistelemalla tai erilaisia tehtäviä tekemällä saada kokemuspisteitä hahmolleen ja siten nostattaa hahmonsa tasoa paremmaksi, jonka avulla hahmosta tulee usein voimakkaampi. Peli sisältää usein jonkin tarinan ja pelin päästäkseen läpi pelaajan täytyy kehittää hahmoaan tarpeeksi hyväksi. Moninpeleissä pelaajat taistelevat tai toimivat yhteistyössä toistensa kanssa. (2008 Lecky-Thompson, 29-30.)

Roolipeleissä pelaaja käytännössä ottaa jonkun hahmon käyttöönsä ja pelaa sen roolia, päättäen näin sen kohtalosta ja tulevaisuudesta. Roolipelit ovat usein yksinpeleistä kaikkein pitkäkestoisimpia, jopa 120 tuntia tai enemmänkin pelattavaa. Roolipelejä ovat esimerkiksi The Witcher, Dark Souls ja World of Warcraft, joka on massiivinen monen pelaajan verkkopeli ja jossa pelaajat taistelevat joko yhdessä yhteisiä vihollisia vastaan tai toisiaan vastaan. (2017 Mirillis Team.)

2.1.4 Simulaatiopelit

Aikaisemmin simulaatioita kehitettiin lähinnä auttamaan ihmisiä kehittymään vaikeissa töissä ja esimerkiksi armeija voi harjoittaa sotilaitansa simulaatioiden avulla. Simulaatiot ja simulaatiopelit ovat toki kaksi eri asiaa. Simulaatiopelien tarkoituksena on simuloida oikeita asioita tai tapahtumia. Erilaisia simulaation kategorioita ovat muun muassa manageröinti ja rakentaminen esim. SimCityn kaltaiset kaupunginra-

kennuspelit, elämää simuloivat pelit ja erilaisten kulkuneuvojen käyttämisen simulointi. Simulaatiopelejä on käytännössä kaikesta mitä oikeassa elämässäkin on. (2012 Mitchell, 33-34.)

Varsinkin nykyään on simulaatiopeli lähes kaikesta oikeaan elämään liittyvästä asiasta ja työstä. Simulaatiopelejä ovat muun muassa kaikkien tuntema Sims-peli, jossa simuloidaan ihmisten elämää, Euro Truck Simulator, jossa tehdään töitä rekalla ja Football manager on peli, jossa nimensä mukaisesti manageroidaan jalkapallojoukkuetta. (2017 Mirillis Team.)

2.1.5 Urheilupelit

Urheilupeleissä pelataan käytännössä jotain oikean elämän urheilupeliä, joko urheilijana tai valmentajana/managerina. Pelin tarkoituksena on toteuttaa oikean urheilun säännöt ja taktiikat. Pelissä voi olla kyse yksittäisestä urheilutapahtumasta tai matsista, taikka kokonaisesta kaudesta. (2004 Bates, 9.)

Urheilupelit ovat nykyään todella realistisia ja niistä usein kehitetään ja tehdään uusia versioita joka vuosi. Urheilupelejä ovat mm. suosittu jalkapallopeli Fifa, autopeli Dirt Rally ja koripallopeli NBA 2K17. (2017 Mirillis Team.)

2.1.6 Strategiapelit

Strategiapelejä on kahdenlaisia: vuoropohjaisia ja reaaliaikaisia. Vuoropohjaisissa peleissä pelaajalla on aikaa miettiä ja suunnitella seuraavaa siirtoansa. Reaaliaikaisissa sen sijaan pelaaja joutuu tekemään päätöksiä pelin kuluessa ja tekemään mahdollisesti useitakin päätöksiä samanaikaisesti. Strategiapeleissä on usein rajattu määrä resursseja, joita pelaajaa kerää ja joidenka avulla pelaaja voi rakentaa tai tehdä erilaisia yksiköitä. Pelin tarkoituksena on voittaa vastustaja paremman strategian turvin. Vastustaja voi olla tietokoneohjattu tekoäly tai oikea ihmispelaaja. (2004 Bates, 8.)

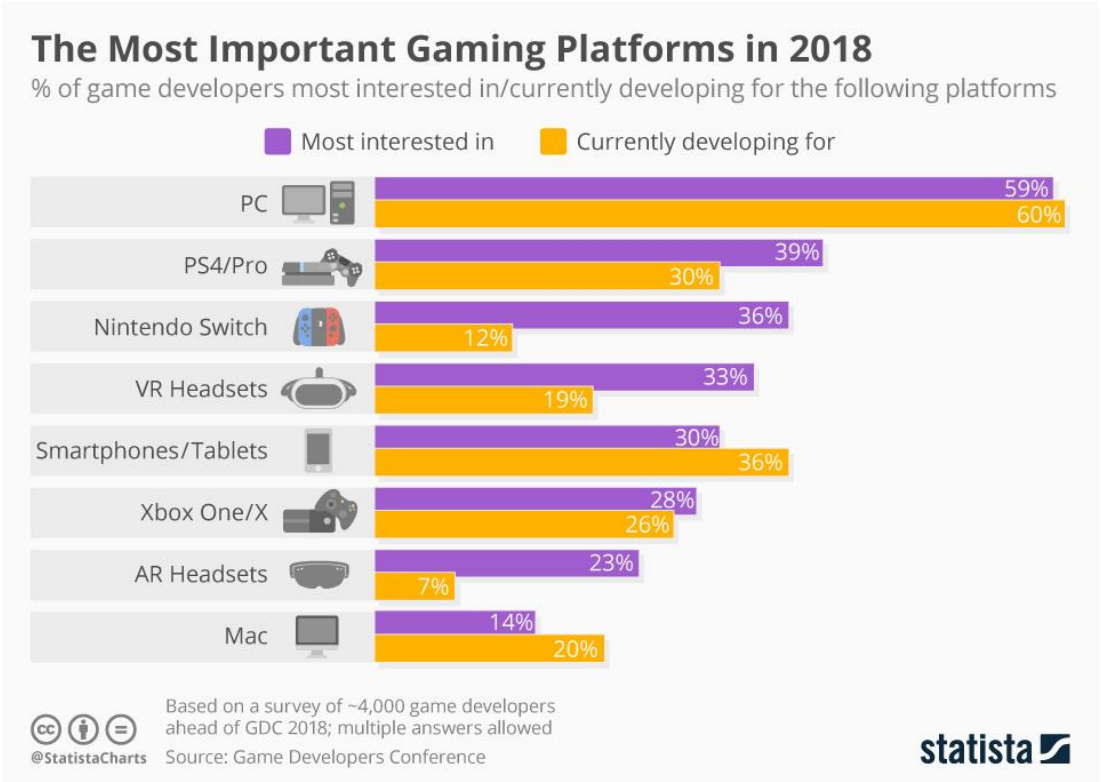
Starcraft on yksi reaaliaikaisista strategia peleistä. Siinä nopeus ja taktiikat ovat valttia. Toinen pelaaja saattaa voittaa taktiikan avulla, toinen taas nopeutensa avulla.

Starcraftia pelataan myös kilpailullisesti 1vs1-tyyppisesti. League of Legends on taas taisteluareenapeli, jota pelataan joukkueissa monella pelaajalla. Siinä vaaditaankin strategian ja nopeuden lisäksi myös hyvää tiimityöskentelyä.

Lopuksi huomattakoon, että on lukuisia muitakin erilaisia genrejä, eikä niitä ole tässä kaikkia käsitelty. Näitä ovat mm. pulmapelit, partypelit, oppimispelit, hyötypelit ja klassiset pelit. (2017 Mirillis Team.)

2.2 Alustat

Myös pelialustan valitseminen on tärkeää. Tehdäkö peli vain yhdelle alustalle vai monelle eri alustalle. Useampi alusta vaatii lisää työtä, mutta näin saadaan peli isommille pelaajamäärille. Pelientekijät ovat UBM:n toteuttamassa kyselyssä vastanneet, mitkä alustat ovat heille mielenkiintoisimpia ja mille alustoille he tällä hetkellä ovat tekemässä peliä (Kuva 1). Yksittäisistä alustoista PC 60 prosentin osuudella on selvästi suosituin. Seuraavaksi eniten tällä hetkellä tuotetaan pelejä pelikonsoleille, joista eniten PS4:lle. Kolmanneksi eniten pelejä tehdään mobiilialustoille. (Richter 2018.)



Kuva 1. Tärkeimmät pelialustat vuonna 2018 (Richter 2018)

On myös hyvä ajatella erilaisten pelialustojen tulevaisuutta. Miten pelejä tullaan pelaamaan tai mitkä alustat ovat suosituimpia tulevaisuudessa.

Yksi mahdollinen tulevaisuuden näkymä on konsolien poisjäänti markkinoilta. Tietokonepelit jatkavat kasvuaan ja mobiilipelit ovat yhä suositumpia ja jopa konsolien valmistajat ovat tehneet omia mobiilipelejään. Ehkä konsolit jäävät turhiksi. Toisaalta ehkä konsolit pystyvät vastaamaan kilpailuun omilla uutuuksillaan. Konsoleihin voitaisiin lisätä tietokoneiden kaltaista modulaarisuutta, jolloin konsolien käyttäjät voisivat päivittää konsoliaan tehokkaammaksi tietokoneen tapaan. Konsoleita myös pelataan eri tavalla kuin tietokonepelejä. Yksin tietokoneella pelaamisen sijaan konsolia pelataan usein sohvalla yksin tai kavereiden kanssa, mikä on monelle rennompaa esimerkiksi pitkän työpäivän jälkeen. Myös Nintendon Switch-konsoli on erittäin mielenkiintoinen uutuus eräänlaisena hybridikonsolina, sillä sitä voi pelata missä tahansa mobiilipelien tapaan. Myös Virtual Reality-pelit on syytä ottaa huomioon. Erilaiset VR-laitteet ovat toki vielä melko kalliita, mutta kehittyessään ja halvetessaan niistä voi tulla seuraava hitti. (Lozada 2018.)

2.2.1 Eri alustatyypit

Tietokone on pelialustana erittäin hyvä. Pelejä voi pelata sekä pöytäkoneella että kannettavalla tietokoneella. Tietokoneen osien kuten prosessorin, näytönohjaimen ja muistin tulisi kuitenkin olla tarpeeksi hyviä varsinkin raskaimmille peleille. Pelejä pelataan usein hiirellä ja näppäimistöllä, mutta myös erilaisilla peliohjaimilla on mahdollista pelata pelejä. (2016 PC dreams) Tietokoneella pelattavissa peleissä on usein parempi ruudunpäivitystahti kuin konsoleissa. Myöskin grafiikat ovat usein parempia johtuen tehokkaammista osista. Peleissä pystyy myös usein säätämään asetuksia, kuten resoluution koon vaihtaminen tai grafiikka-asetusten säätäminen, niin että myös huonommilla koneilla pystytään pelaamaan peliä. (2015 Williams)

Erilaiset pelikonsolit on tehty nimenomaan pelejä varten. Niitä pelataan normaalisti peliohjaimella ja konsoli on kytketty yleensä televisioon, mutta mikä tahansa muukin näyttö käy. Konsolit ovat erittäin suosittuja. Niitä on kuitenkin erilaisia, joista suosi-

tuimmat ovat Sonyn, Microsoftin ja Nintendon konsolit. Pelit ovat yksinkertaisia käyttää ja konsoli on muutenkin huolettomampi vaihtoehto tietokoneelle. Konsolia ei voi kuitenkaan päivittää kuten tietokonetta ja konsoleille tehdyt pelit rajoittuvat konsolille määrättyihin tehoihin ja muihin rajoituksiin. Myöskin pelissä käytetyt kontrollit rajoittuvat konsolilla käytettävään peliohjaimeen toisin kuin tietokoneessa. (2015 Williams)

Mobiilipelit ovat nykyään erittäin suosittuja. Niitä pelataan puhelimilla tai tableteilla. Suurimmat käyttöjärjestelmät näille laitteille ovat Android ja iOS. Mobiilipelit ovat monesti ilmaisia sisältäen mainoksia tai muita erillisiä vaihtoehtoisia maksuja. Pelit voivat toki olla myös täysin maksullisia. Pelit voivat olla myös sekä monimutkaisia että yksinkertaisia. (2016 PC dreams) Mobiilipelejä pelataan kosketusnäytöllä, mikä tekee pelin kontrolleista hieman erilaisia verrattuna konsoli- ja tietokonepeleihin. Mobiilipeleillä on todella suuret markkinat, sillä lähes jokaisella on nykyään oma mobiililaite, ja pelin asentaminen siihen on helppoa ja nopeaa. Monet pelit ovat erittäin addiktoivia ja yksinkertaisistakin peleistä voi tulla suosittuja, kuten Flappy Bird osoitti yli 80 miljoonalla latauksellaan kahdessa kuukaudessa. Mobiililaitteille ei voida kuitenkaan tehdä liian paljon tehoja vaativia pelejä. Alalla on myös paljon kilpailua ja vaihtuvuutta sekä pelien tekijöiden että laitteiden tekijöiden välillä. (2015 Williams)

2.3 Pelimekaniikat

Pelimekaniikat ja säännöt ovat erityisen tärkeitä peliä suunniteltaessa. On hyvinkin mahdollista, että peli on hieno grafiikoiltaan, ääniltään ja muutenkin näyttää hyvältä, mutta mekaniikat ovatkin huonot, minkä takia peli ei olekaan miellyttävä pelata. Mekaniikoilla ja pelin säännöillä käytännössä tarkoitetaan sitä, miten pelaaja voi liikkua pelissä, miten hän voi vaikuttaa peliin ja mitä rajoituksia siihen on. Mekaniikat ovat juuri se, mikä tekee pelistä uniikin ja määrittävät pelin tyypin. Pelaajat haluavat mekaniikoiden olevan ennustettavia ja johdonmukaisia, niin että pelattavat hahmot toimivat aina samalla tavalla ja pelaaja pystyy harjoittelun avulla tulemaan paremmaksi pelissä. (2008 Lecky-Thompson, 141-142.)

Toimivat pelit tarvitsevat erilaisia mekaniikoita. Toimiva peli siis syntyy monista erilaisista mekaniikoista, jotka yhdessä vaikuttavat pelattavuuteen. Mekaniikoiden on tärkeää sopia yhteen ja toimia yhdessä, sillä epämääräiset ja toisiinsa kuulumattomat mekaniikat tekevät vain ärsyttävän tai vaikean pelin. Luonnollisesti mitä enemmän mekaniikoita pelissä on, sitä monimutkaisempi se tulee olemaan. Jos pelistä on tulossa liian tylsä tai yksinkertainen, voi siitä helposti saada mielenkiintoisemman lisäämällä uusia mekaniikoita tai jo parantamalla nykyisiä. Järki on kuitenkin hyvä pitää mielessä, ja tehdä uusista mekaniikoista järkeviä ja peliin kuuluvia. (2018 Gamedesigning.org.)

Pelin sääntöjen määritteleminen on tärkeää, sillä ne määräävät pelaajan liikkumista pelissä ja mahdollistavat pelaajan mitata menestyksensä aikaisempiin kokemuksiin verraten. Säännöt määrittävät pelin tarkoituksen, ja sen mitä pelaajan täytyy tehdä päästäkseen pelin läpi tai saadakseen vaikka pisteitä. Ammuntapelissä pelihahmo voi kuolla yhdestä osumasta tai se voi ottaa montakin osumaa menettäen terveyspisteitä (HP). Näitä terveyspisteitä pelaajan voi olla mahdollista saada takaisin esimerkiksi jonkin esineen avulla, tai sitten peliä vaikeuttaakseen voi evätä mahdollisuuden saada niitä takaisin. Myös pelissä kuolemiseen voi vaikuttaa eri tavoilla. Pelaajalla voi olla käytössään tietty määrä elämiä, jotka kaikki menetettyään pelaaja häviää pelin tai kentän. Toinen vaihtoehto on, että pelaaja häviää pelin tai kentän jo ensimmäisestä kuolemasta. Pelille voi myös asettaa erilaisia vaikeusasteita, joiden avulla peli saadaan paremmaksi kaiken tasoisille pelaajille. (2008 Lecky-Thompson, 143-145.)

Myös pelin fysiikat ovat tärkeässä roolissa, sillä ne määrittävät pelin realistisuuden. Tähän on toki jo olemassa valmiita fysiikkamoottoreita, joten esimerkiksi painovoimaa ei tarvitse itse lähteä mallintamaan. Pelissä voi toki olla paljonkin yliluonnollista tai se voi esimerkiksi sijoittua avaruuteen, mutta pelaajat odottavat pelin toimivan johdonmukaisesti oman maailmansa mukaisesti. Pelissä käytettävän hahmon liikkuminen voi vaihdella ja sen liikkumismahdollisuuksista tulee päättää. Tehdäänkö hahmolla vain yksinkertaisia kävely-, juoksu- ja hyppyliikkeitä, vai onko sitä mahdollista liikuttaa paljon monimutkaisemminkin. (2008 Lecky-Thompson, 149-153.)

Joskus pelin mekaniikat voivat kuitenkin olla myös piilotettuja tai huomaamattomia. Ne ovat kuitenkin yhtä tärkeitä kuin muutkin mekaniikat. Esimerkiksi tasohyppely-

pelissä pelihahmo voi ihan pienesti olla jonkin tason reunan yli tippumatta, sillä tämä voi lisätä pelin pelattavuutta ja sujuvuutta, vaikka se olisikin vähemmän realistista. Pelien on tärkeää realistisuuden lisäksi myös tuntua hyvälle pelata ja joskus näiden välillä voidaan joutua tekemään kompromisseja. Tietokoneohjatun vihollisen voi tehdä helposti liian vaikeaksi tai helpoksi, mutta paljon vaikeampaa on tehdä vihollinen, joka on hauska ja sopivan haastava. Tämän kaltaisen vihollisen tekeminen on kuitenkin tärkeää, jotta peli pysyisi mielenkiintoisena. Pelaajalle voidaan antaa myös erilaisia etuja viholliseen nähden esim. vihollinen ampuu aina ensimmäisen ammuksen ohi pelaajasta. (2018 Wawro)

2.4 Grafiikat, äänet ja musiikit

Grafiikat määrittävät, miltä peli näyttää. Mitä paremmin grafiikat toteuttaa, sen paremmalta peli näyttää. Grafiikat olisi hyvä suunnitella niin, että lisäävät näyttävyyttä, mutta eivät häiritse. Jos pelin haluaa menestyä, tulisi kiinnittää huomiota kohderyhmään. Kenelle peliä ollaan tekemässä? Minkä ikäisille pelin on ensisijaisesti tarkoitettu? Myös sukupuolella voi olla väliä. Kun kohderyhmä on selvitetty, voidaan tutkia, mitä pelejä kohderyhmä pelaa. Erityisen tärkeää on kiinnittää huomiota kohderyhmän suosituimpiin peleihin. Suosituista peleistä voi saada selville mm. värit, tyylit ja käyttöliittymät. Esimerkkinä lapsille suunnatut pelit voisivat olla värikkäitä ja iloisia. (2012 Mitchell, 111-117.)

Peliä suunniteltaessa on hyvä päättää kamerakulmista. Tehdäänkö peli kenties sivulta kuvattuna 2D-pelinä, kuten Super Mario Bros? Aikaisemmin kaikki pelit olivatkin 2D-pelejä, ja niiden toteuttaminen onkin yksinkertaisempaa. 3D-pelit taas tuovat kompleksisuutta, ja kamera kulmatkin voivat vaihdella ja ovat mahdollisesti pelaajan liikuteltavissa. Peli voi olla myös pelihahmon näkökulmasta kuvattu tai niin, että kamera seuraa pelihahmoa. Yksi vaihtoehto on myös kuvata peliä ylhäältä päin. (2012 Mitchell, 118-121.)

Grafiikoiden tekoon käytettäviä erilaisia työkaluja on monenlaisia, joista jotkut ovat ilmaisia ja toiset maksullisia. Adobe Photoshop on yksi suosituin työkalu grafiikoiden tekemiseen. Se sisältää paljon erilaisia ominaisuuksia ja on muutenkin todella

monipuolinen ohjelma. Se on kuitenkin maksullinen. Yksi ilmainen, joskin hieman yksinkertaisempi kuvankäsittelyohjelma on GIMP. Se on kuitenkin tarpeeksi hyvä grafiikoiden tekemiseen, varsinkin jos haluaa pitää budjetin mahdollisimman pienenä. Maksullinen Marmoset Hexels on erikoistunut pikselitaiteen tekoon. Se on siis erityisen hyvä, jos haluaa pelin grafiikoita luoda pikselitaiteen avulla. (Crump 2018)

Pelissä on hyvä myöskin olla äänet. Ääniä tarvitaan muun muassa ääniefekteinä erilaisille tapahtumille, esimerkiksi kun pelissä käytettävä hahmo kävelee, tulisi siitä kuulua askelten ääniä. Pelissä voidaan tarvita myös puhetta, sillä hahmojen puhuminen lisää pelimukavuutta verrattuna siihen, jos puheen sijasta olisikin vain tekstiä. Pelissä olisi myös hyvä olla musiikkia, ja musiikin olisi hyvä sopia pelissä olevaan tapahtumaan, tilaan tai kenttään. (2008 Lecky-Thompson, 101-108.)

2.5 Kenttä- ja tehtäväsuunnittelu

Kun peliin lähtee suunnittelemaan uutta kenttää, pitää muistaa, että se on yksi osa suuremmasta kokonaisuudesta. On hyvä miettiä mitä varten kenttää ollaan tekemässä. Jos pelissä on jokin tarina, niin miten kenttä jatkaa sitä? Uudessa kentässä voi myös tulla vaikka jokin uusi vihollinen, tai pelattava hahmo voisi saada esimerkiksi uuden aseen käyttöön. Kentässä ideana voisi olla esimerkiksi hiiviskely, uuden asian oppiminen tai vaikka uusi visuaalinen ilme. Uutta kenttää suunniteltaessa olisi hyvä valita monen sijasta vain yksi idea kentälle, sillä liian monimutkaiset kentät saattavat saada pelaajan hämmennykseen. Kentän hahmottelu esimerkiksi paperille aluksi voisi olla hyvä idea, sillä voit tehdä useitakin luonnoksia ja tulla varmemmaksi siitä millaisen kentän haluat tehdä. (2004 Bates, 107-110.)

Uutta kenttää suunniteltaessa ei välttämättä tarvitse aloittaa kentän alusta, vaan voi miettiä mikä on kentän kohokohta tai loppu, ja lähteä siitä suunnittelemaan taakse päin. Tämä tyyli voi olla joillekin helpompi tapa. Pelaajan päästäkseen kentän loppuun tai ratkaistakseen jonkin pulman, on kuljettava kenties tiettyjen välietappien kautta. Pelaajan voi ohjailla näille välietapeille ohjeiden tai vaikka tehtävien avulla. Toki kenttä voi olla myös lineaarinen, ja pelaaja väistämättä eteenpäin liikkuessa tulee kohtaaman ne. Kentän olisi joka tapauksessa hyvä olla haastava, muttei liian tur-

hauttava pelaajalle. Yksi tapa miettiä uutta kenttää on pistää silmät kiinni ja uppoutua ajatuksiinsa. Voi kuvitella missä mikäkin paikka, hahmo, esine tai tehtävä on. Itsensä pelimaailmaan kuvitteleminen voi myös auttaa. Voi kuvitella kulkevansa pelimaailmassa ja vaikka miettiä, miltä siellä näyttää tai minne on menossa. Kentissä voisi olla myös salapaikkoja, joita on vaikea löytää, mutta joiden löytäminen on palkitsevaa pelaajalle. (2012 Mitchell, 166-168.)

Kentissä voi siis olla erilaisia tehtäviä ja niitäkin olisi hyvä suunnitella. On hyvä miettiä, missä järjestyksessä tehtävät suoritetaan ja onko pelaajan pakko suorittaa ne jonkinlaisessa järjestyksessä, voiko ne tehdä vapaasti missä järjestyksessä vain tai voisiko olla kenties jokin ideaali järjestys, jota ei ole pakko noudattaa, mutta siitä on pelaajalle eniten hyötyä. Tehtäviä voi olla eritasoisia ja ehkä kaikkein vaikeimmat voisivat olla vapaavalintaisia, joita suorittamalla saa erityispalkintoja. Tehtävät voivat sijaita eri paikoissa, ja jokaisen tehtävän ratkaistuaan pelaaja on lähempänä kentän loppua. Eri paikkoihin päästäkseen tai tehtäviä suorittaakseen pelaaja voisi tarvita jotain esinettä esimerkiksi avainta, jonka voi löytää vaikka jostain päin kenttää tai saada esimerkiksi palkinnoksi jostain toisesta tehtävästä. Yksinkertaisia tehtäviä kentässä voisivat olla esimerkiksi: kerää kaikki kolikot ja juokse maaliin, tapa kaikki viholliset tai vaikka pysy hengissä jokin tietty aika. (2008 Lecky-Thompson, 220-223.)

Erilaisilla värikoodeilla voidaan saada pelaajan huomio ja kertoa pelaajalle erilaisista tilanteista. Esimerkiksi vaaraa voidaan merkitä jollain värillä, tai erilaisilla väreillä voidaan merkitä objektien erilaista tilaa. Monet pelit käyttävätkin erilaisia värejä tai korostuksia auttamaan pelaajaa etenemään kentässä. Kentässä voi olla esimerkiksi paljon erilaisia esineitä, joita ei voi käyttää, mutta käytettävissä olevat esineet ovat korostettuja jollain tietyllä värillä, jolloin pelaaja tietää heti, että niitä voi käyttää. Myöskin valaistuksella ja erilaisilla kontrasteilla voidaan viestiä pelaajalle, minne pitää mennä. (Wilson 2018.)

2.6 Käyttöliittymä

Käyttöliittymän suunnittelu on kenties yksi aliarvostetuimmista tehtävistä peliä suunniteltaessa. Se on kuitenkin tärkeä, sillä se vaikuttaa siihen, kuinka pelaaja näkee

ja myöskin käyttää peliä. Käyttöliittymän tulisi olla mahdollisimman yksinkertainen, mutta kuitenkin sellainen, että siinä on kaikki tarvittavat ominaisuudet. Eli ei siis sellainen, josta saattaisi puuttua jotain pelaajalle tärkeää. Pelaajan tulisi kuitenkin pystyä näkemään tai löytämään kaikki tärkeät asiat. Asiat, joita pelaaja käyttää harvemmin, voivat olla useamman klikkauksen päässä, mutta vastaavasti useimmin käytetyt toiminnot pitää pystyä käyttämään näppäimistöllä, hiirellä tai ohjaimella. Muiden samantyyppisten pelien käyttöliittymistä voi katsoa mallia, sillä ne oletettavasti toimivat hyvin, varsinkin jos hyvin monet pelit käyttävät vastaavaa mallia. Myös käyttöliittymistä kannattaa tehdä luonnoksia ja käyttöliittymää kannattaa testauttaa muilla henkilöillä, sillä toiset saattavat löytää virheet ja huonon käytettävyyden paremmin kuin käyttöliittymän suunnittelijat. Käyttöliittymän tärkein asia on antaa pelaajan tehdä asioita vaivattomasti ja ajattelematta sitä liikaa. Pelaajan ei siis tarvitse ajatella, kuinka jokin asia tehdään vaan hän voi yksinkertaisesti vain tehdä sen. (2004 Bates, 26-28.)

Pelin aloitusruutu kannattaa tehdä mahdollisimman monitasoisille käyttäjille sopivaksi. Pelaaja voi olla jo kokenut, jolloin hän saattaa haluaa päästä mahdollisimman nopeasti pelaamaan, tai sitten kyseessä voi olla täysin uusi pelaaja. Asetuksista olisi hyvä tarjota pelaajalle mahdollisuus muokata ohjaimia ja oikeastaan kaikkea muutakin mahdollista esimerkiksi äänet ja grafiikat. Aloitusruudussa olisi hyvä olla ainakin seuraavat valinnat: (2004 Bates, 28.)

- Uusi peli
- Vanhan pelin jatkaminen
- Tutoriaali/harjoittelu
- Asetukset

GUI (graphical user interface), tarkoittaa käyttöliittymää, jota voi klikata, eli sillä voi tehdä asioita. HUD (heads-up display) puolestaan näyttää pelaajalle tärkeitä asioita esimerkiksi pelihahmon terveyspisteet, ajan tai vaikka aseet ja niiden panokset. HUD on yleensä näkyvillä koko ajan, ja pelaaja voi pelatessaan katsoa siitä tietoja. (2012 Mitchell, 143-144.)

Käyttöliittymän suunnittelu on usein haastavaa pelien kehittäjille, ja varsinkin monet indie-kehittäjät saattavat jättää sen pahimmillaan jopa tekemättä. Huono käyttöliittymä voi helposti ajaa uudet pelaajat pois pelin luota. Jos ensimmäisellä pelikerralla käyttöliittymä alkaa ärsyttää pelaajaa, niin saattaa hän helposti lopettaa koko pelin pelaamisen. Käyttöliittymää tulisikin miettiä jo ensimmäisistä pelin suunnittelun päivistä alkaen. Käyttöliittymästä ei siis saisi tulla liian monimutkainen. Kokeneet pelaajat voivat sitä toki arvostaa, mutta uusille pelaajille se on kauhu. Käyttöliittymää olisikin hyvä testata ihmisillä, jotka eivät sitä vielä ole käyttäneet. Hyvän käyttöliittymän tekemisen voisia sanoa olevan taidetta, sillä siinä tulee ottaa niin paljon asioita huomioon. Grafiikat, ja fontit, miten uudet pelaajat sen kokevat ja taas vastaavasti miten kokeneemmat pelaajat kokevat sen tarpeeksi hyväksi. (2018 Bycer)

2.7 Hahmo- ja tarinasuunnittelu

Hyvällä pelihahmolla voi olla suurikin vaikutus pelin tunnettavuuteen ja imagoon. Monet ihmiset todennäköisesti esimerkiksi tietävät Pikachun Pokémonista tai Nintendon pelihahmot Marion ja Luigin. Pelihahmon suunnittelun voi aloittaa miettimällä minkälainen hahmo on kyseessä. Pelattava hahmo voisi olla sankari, pahis, söpö eläinhahmo tai ihan mikä vaan. Kun hahmo on valittu, tulisi seuraavaksi kehitellä jonkinlainen taustatarina sille. Taustatarinat tekevät hahmosta heti paljon mielenkiintoisemman, ja siitä voi olla apua myös itse pelin tarinan kehittämiseen. Hahmon tarinan avulla voidaan myös paremmin selittää motiivit sille, mitä hän ikinä tekeekään pelissä. Lisäksi hahmolla olisi hyvä kehittää erilaisia luonteenpiirteitä ja mahdollisia suhteita muihin pelissä oleviin hahmoihin. Lopuksi tulisi suunnitella hahmon ulkonäkö, miltä hän näyttää tai miten hän pukeutuu. (2018 Gamedesigning.org.)

Pelihahmoa suunniteltaessa on hyvä miettiä myös minkälaisia taitoja ja ominaisuuksia sillä on. Nämä taidot tietysti riippuvat hyvin paljon siitä, mitä pelissä on tarkoitus tehdä. Ammuntapeleissä hahmo on oletettavasti hyvä ampumaan. Pelimaailmasta riippuen hänen on pystyttävä mahdollisesti esimerkiksi uimaan tai vaikka käyttämään erilaisia laitteita. Peliin on hyvä suunnitella myös erilaisia vastuksia. Ne ovat myös erilaisia hahmoja, joilla voi olla omat tarinansa. Nämä vastukset tulisi olla tarpeeksi

vahvoja voittaakseen pelaajan, mutta pelaajan opittua oikean taktiikan, hänen on mahdollista voittaa vastus. (2012 Mitchell, 69-70.)

Pelin tarinassa hyväksi havaittu tapa on jakaa tarina kolmeen osaan: alku-, keski- ja loppuosa. Pelin alku voisi lähteä siitä, että pelihahmolla on jokin ongelma, jota pitää lähteä ratkaisemaan, jolloin pelaaminen alkaa mahdollisimman nopeasti. Kaikkia taustatarinoita ja muita juonia ei siis kannata kertoa heti pelin alussa, vaan vasta pelin edetessä. Keskiosassa alun jälkeen voidaan taustat selittää paremmin. Hahmolla on siis jokin ongelma tai päämäärä, ja sinne päästäkseen hänen on kohdattava monia erilaisia haasteita. Pelissä voisi olla jonkinlainen päävihollinen, joka kenties hankaloittaa pelihahmon etenemistä. Tämän vihollisen kukistaminen on mahdollisesti pelihahmon päämäärä tai osa sitä. Lopussa kaiken koettelemuksen jälkeen pelihahmo taistelee tätä päävihollista vastaan ja ratkaisee ongelmansa tai pääsee päämääräänsä. (2004 Bates, 96-98.)

2.8 Pelisuunnittelu ja kehitysprosessi

2.8.1 Esituotanto

Pelin tekeminen alkaa peli-idean suunnittelulla. Se voi olla joko täysin uusi idea tai jo valmiiksi hyväksi havaitusta ideasta muunneltu versio. Peli-ideaa suunniteltaessa olisi hyvä tehdä erilaisia piirroksia hahmottamaan ideoita. Idean suunnittelu sisältää muun muassa pelin tarinan suunnittelua, pelimekaniikkojen suunnittelua, hahmojen suunnittelua ja ylipäätään pelin teeman suunnittelua. Myöskin pelin mahdollisia grafiikoita voi jo alkaa mallintaa. Esituotannon tarkoituksena on saada aikaan ”game design document”, joka sisältää pelin elementtejä sekä pelin projektisuunnitelman. Pelille on hyvä suunnitella myös hinnoittelustrategiaa. (2016 Khurramsamad.)

Esituotanto on erittäin tärkeä vaihe pelin suunnittelun ja tekemisen kannalta. Monet projektit ovat epäonnistuneet huonon esituotannon takia. Peli saattaakin esimerkiksi kasvaa isommaksi kuin mitä oli alun perin suunniteltu, tai vastaan voi tulla ongelmia, joita ei osattu havaita liian nopean esituotannon takia. Peliyritysten esituotantovaiheet peleistä riippuen voivat hyvin kestää 3-12 kuukautta. Esituotannon avulla pitäisi

pystyä selvittämään, mikä pelistä tekee hyvän sekä tehdä eräänlainen prototyyppi pelistä. Prototyypin ei tarvitse olla erityisen kaunis, mutta sen tulisi sisältää pelin pääominaisuudet. Prototyypin avulla voidaan vähentää tuotantovaiheessa tulevia riskejä. Mitä isommasta pelistä on kyse, sitä tärkeämpi vaihe esituotanto on. (2018 Tefler.)

2.8.2 Tuotanto

Tuotantovaiheessa varsinaista peliä voidaan alkaa tekemään. Tuotantovaihe sisältää suunnittelua, varsinaista tuotantoa ja testaamista. Peliä olisi hyvä tehdä pienissä erissä kerrallaan niin kutsutuissa sprinteissä, jotka kestävät noin kaksi viikkoa. Alussa tietenkin olisi hyvä tehdä pelin runko ja muut tärkeimmät ominaisuudet valmiiksi. Jokaisessa sprintissä olisi hyvä ensin suunnitella ja tehdä vaiheeseen tarvittavat grafiikat, minkä jälkeen ohjelmoidaan vaiheessa tarvittavat ominaisuudet. Viimeisenä vaiheena testataan tehdyt ominaisuudet. (2016 Khurramsamad.)

Tämän kaltaiset ohjelmistotuotannosta tutut ketterät menetelmät sopivat myös pelituotantoon. Sen avulla peliä voidaan suunnitella ja tehdä paremmin samaan aikaan, kun tehdään vain vähän kerrallaan. Yksi ketterä menetelmä on Scrum, joka koostuu pienistä noin 5-9 hengen tiimeistä, ja jossa yksi vaihe kestää 2-3 viikkoa. Pienen ryhmän etuna on parempi kommunikointi ja vähän kerrallaan tekeminen yksinkertaistaa asioita. (2014 Clinton.)

2.8.3 Jälkituotanto

Jälkituotannossa valmistaudutaan julkaisemaan valmis peli. Ennen kuin peli on valmis, on sitä hyvä vielä testata ennen lopullista julkaisua. Tämä voi tapahtua esimerkiksi beta-testauksella, eli annetaan käyttäjien pelata vielä testausvaiheessa olevaa peliä, jolloin käyttäjät huomaavat virheitä ja voivat raportoida niistä. Kun peliä on testattu tarpeeksi ja virheitä on korjattu, voidaan peli julkaista. Julkaisun jälkeen peliä täytyy myös markkinoida. Pelistä voi edelleen löytyä uusia virheitä, joita ei testausvaiheessa löydetty ja jotka täytyy korjata. Peliin voidaan myöskin haluttaessa lisätä ominaisuuksia. (2016 Khurramsamad.)

Jälkituotannon tärkeimpiä vaiheita ovat siis beta-testaus, jonka avulla löydetään virheitä pelistä ja testataan esimerkiksi pelin vaikeusastetta. Erilaisten medioiden tekeminen pelille on tärkeää, esimerkiksi sivustot tai sivut sosiaalisille medioille. Lopuksi tietysti tulee päättää varsinainen pelin julkaisupäivämäärä. Julkaisun jälkeenkin tarvitaan pelin ylläpitoa. Jotta pelaajat saadaan pidettyä mahdollisimman pitkään pelin parissa, tulisi peliin lisätä uusia lisäominaisuuksia, joista jotkin voivat olla ilmaisia kaikille ja toiset taas maksullisia. Maksullisilla lisäominaisuuksilla ja lisäosilla pelin tuottoa saadaan myöskin kasvatettua. (Anupama 2018.)

3 PELIMOOTTORIT

3.1 Yleisesti

Pelimoottori on ohjelmistokehys, jonka avulla pelinkehittäjien on helpompaa tehdä pelejä. Siinä on jo valmiiksi erilaisia ominaisuuksia pelien tekemiseen. Pelimoottoreissa on usein jo valmiina useita ominaisuuksia esimerkiksi grafiikan piirtäminen näytölle ja fysiikkamoottori, jonka avulla voidaan mallintaa vaikka painovoimaa. Pelimoottori tekee siis paljon asioita kulissien takana, ja pelinkehittäjien tarvitsee kirjoittaa paljon vähemmän koodia. (2012 Smiley.)

Pelimoottorit ovat kuin mitkä tahansa ohjelmistokehykset, mutta ne ovat täysin tehty pelien tekoa silmällä pitäen. Pelimoottorit sisältävät erilaisia komponentteja, jotka sisältävät monia ominaisuuksia helpottamaan pelien tekoa. (2013 Enger.)

3.1.1 Syöte

Yksinkertaisesti voidaan sanoa, että peli toimii toistosilmukassa, jossa ensin odotetaan käyttäjän syötettä. Kun tämä syöte on tapahtunut, niin seuraavaksi lasketaan syötteen aiheuttamat muutokset ja suoritetaan tarvittavat toimenpiteet. Lopuksi vielä päivitetään ruutua, jolloin pelaaja näkee tämän muutoksen (2012 Smiley.)

Pelimoottorin yksi ominaisuuksista on pelaajan syötteen käsitteleminen. Eli käytännössä miten peliä tullaan pelaamaan. Pelimoottorit voivat sisältää muun muassa seuraavia syötetyyppejä: näppäimistö, hiiri, peliohjain, näytön kosketus ja monia muita laitteita kuten ratti. Käytännössä syötteen hallinta tapahtuu niin, että ensin laite kuuntelee mahdollisten näppäinten painalluksia, ja kun jokin painallus on rekisteröity, tämä laukaisee tapahtuman, joka sisältää koodia. Esimerkiksi pelaaja on painanut näppäimistön välilyöntinäppäintä ja tämän johdosta pelissä käytettävä hahmo hyppää. Pelimoottorin avulla voidaan hallita myös esimerkiksi hiiren x- ja y-koordinaatteja tai peliohjaimen ohjaussauvan liikkeitä. (2013 Enger.)

3.1.2 Grafiikkamoottori

Pelimoottorit sisältävät myös grafiikan hallinnan. Pelimoottori piirtää pelissä käytettävät kuvat ruudulle. Se tehdään lataamalla kuvat yksitellen eri tiedostoista, mikä ei tosin ole kovin tehokasta. Parempi tyyli onkin laittaa kaikki kuvat yhteen isoon tiedostoon, josta pelimoottori voi tarvittaessa ottaa yksittäisiä kuvia ja piirtää ne näytölle. Tämän avulla voidaan tehdä erilaisia animaatioita, esimerkiksi pelihahmon jokaisesta kävelyliikkeestä on erillinen kuva samassa tiedostossa. Tästä pelimoottori voi sitten luoda animaation, jossa hahmo kävelee. (2012 Smiley.)

Grafiikkamoottorin avulla voidaan luoda myös erittäin laadukkaita 3D-pelejä. Näissä yleensä käytetään myös 3D-grafiikanmallinnusohjelmia kuten Blenderiä. Grafiikkamoottorissa on myös paljon muitakin erilaisia ominaisuuksia esimerkiksi valon ja varjojen mallintamista. (2013 Enger.)

3.1.3 Ääni

Hyvät pelit tarvitsevat myöskin ääniä, oli ne sitten peliin ja pelihahmoihin liittyviä erilaisia tilanneääniä tai musiikkia tuomaan pelin tilanteisiin sopivaa musiikkia. Pelimoottorit tekevät tämän helpoksi. Pelimoottorit voivat mahdollistaa myöskin musiikin suoratoiston, jolloin sitä ei tarvitse ladata keskusmuistiin. (2012 Smiley.)

Pelien äänillä on siis merkitystä ja varsinkin todella huonot äänet voivat pilata pahimmillaan koko pelin. Pelimoottori auttaa tässäkin. Äänien hallinta varsinkin 3D-peleissä on tärkeää, ja ilman pelimoottoria sen tekeminen olisikin hankalaa. Pelimoottorilla voi olla mahdollista vaikuttaa siihen miltä äänet kuulostavat pelaajalle. Esimerkiksi äänen tapahtuessa pelattavan hahmon takana, kuulee myös pelaaja sen hänen takanaan. Vastaavasti voidaan vaikuttaa vaikka äänen kaikuihin, esimerkiksi miekkataistelu kuulostaa hieman erilaisemmalta aukealla kuin sisätiloissa. Myös taustamusiikkien lisääminen onnistuu helposti. (2013 Enger.)

3.1.4 Fysiikkamoottori

Fysiikoiden mallintaminen ilman pelimoottorissa olevien valmiiden mallien hyödyntämistä olisi erittäin vaikeaa. Se vaatisikin hyvää matematiikan ja fysiikan tuntemusta. Pelimoottorissa nämä asiat ovat kuitenkin jo valmiiksi laskettu eikä pelinkehittäjän tarvitse vaivata päätään niiden tekemiseen. Kuten jo aikaisemmin sanottiin, pelimoottori sisältää painovoiman mallintamisen, mutta sen lisäksi se sisältää paljon muutakin fysiikan mallintamista. Sen avulla voidaan muun muassa havaita pelissä tapahtuvat törmäykset sekä laskea pelin hahmoihin tai esineisiin liittyvät voimat ja nopeudet. Pelihahmoihin liittyvien osumien tai vahinkojen mallintamisen lisäksi pelimoottori pystyy mahdollisesti mallintamaan esimerkiksi veden liikkeet ja aallot. (2012 Smiley.)

Pelimoottori sisältää siis erillisen fysiikkamoottorin, ja niitä onkin monia erilaisia olemassa, esimerkiksi Havok, PhysX ja Box2D. Fysiikkamoottorit toimivat yksinkertaistettuna siten, että pelissä oleva esineen graafisen muodon lisäksi esineellä on fyysinen muoto, jossa kaikki fysiikan mallintamiset tapahtuvat esimerkiksi esineeseen kohdistuvat voimat, esineen massa ja sen reagoiminen painovoimaan. Tämä fyysinen muoto ei välttämättä ole samanlainen kuin sen graafinen muoto (2013 Enger.)

3.1.5 Valmiit scriptit

Erilaisten mallinnuksien lisäksi pelimoottorit voivat sisältää erilaisia valmiita koodeja tai objekteja eri asioihin. Näitä voivat olla muun muassa kameran liikehdintä, va-

lojen manipulointi ja erilaisten tapahtumien laukaisu. Myös tekoälyn luominen esimerkiksi pelin vihollisille on helpompaa pelimoottorin avulla. (2013 Enger.)

Pelimoottorit voivat sisältää muitakin ominaisuuksia kuten käyttöliittymän luomisen. Pelimoottoreita on paljon erilaisia ja jotkin niistä soveltuvat 2D-pelien ja toiset taas 3D-pelien tekemiseen. Jotkut pelimoottorit voivat olla myös erikoistuneita johonkin tiettyyn peligenreen. (2013 Enger.)

Pelimoottoreissa voi siis olla valmiita scriptejä, joiden avulla voidaan kontrolloida pelin kameran lisäksi esimerkiksi pelissä käytettävien animaatioiden hallintaa. Pelimoottoreiden erikoistumisen takia olisikin hyvä miettiä, mikä sopii juuri omaan tarkoitukseen parhaiten, sillä niissä saattaa olla juuri omaan genreen sopivat scriptit. (2018 Unity3d.com)

3.2 Unreal Engine

Unreal Engine on Epic Games -yrityksen omistama pelimoottori ja se on yksi suosituimmista pelimoottoreista. Sitä kehitettiin alun perin vuonna 1998 Unreal-nimistä ammuntapeliä varten. Pelimoottorilla voi kuitenkin tehdä monenlaisia pelejä, vaikka se alun perin ammuntapeliä varten kehitettiinkin. Unreal Engine onkin todella käytetty monien pelinkehittäjien keskuudessa. Unreal Enginen avulla pelinkehittäjät saavat todella laajan pelien alustavalikoiman, koska sillä kehitettyjä pelejä voidaan julkaista monille eri alustoille, muun muassa Windows, Mac, Nintendo Switch, Xbox One, PlayStation 4, iOS ja Android. Unrealin avulla kehittäjien on mahdollista tehdä erittäin hyvännäköisiä ja laadukkaita pelejä ja sillä on myös hyvä optimoida pelejä paremmiksi. (2018 Newgenapps.com.)

Pelien kehittäminen Unreal Enginellä on ilmaista, mutta jos sillä kehitetyt pelit tuottavat tuloja yli 3000 dollaria, tulee pelinkehittäjän maksaa Epic Gamesille 5 prosenttia rojalteja. Unreal Engine sisältää muun muassa moninpeliominaisuuden, animaatiotyökalun, tekoälymoottorin, äänimoottorin ja reaaliaikaisen renderöinnin, ja läh-

dekoodina toimii C++. Unreal Engine on ihanteellinen 3D-pelien tekemiseen. (2018 Fram.)

3.3 GameMaker Studio 2

GameMaker Studio 2 on pelimoottori 2D-pelien kehittämiseen. Se on käyttäjäystävällinen ja helppo käyttää. GameMaker Studiolla peliä tehtäessä tarvitaan vain vähän varsinaista ohjelmointia. Pelin ominaisuudet ja logiikka tehdäänkin drag and drop -menetelmällä. Pelimoottori ei ole kuitenkaan ilmainen ja kaikkein halvin versio maksaa 39 dollaria. Developer-versio maksaa 99 dollaria ja se vaaditaan, jos haluaa julkaista pelin PC:lle. Maksuluokkia on muitakin erilaisia, joista kallein on 12 kuukauden lisenssi kaikille alustoille 1500 dollarilla. Ohjelmointikielenä käytetään omaa GameMaker-ohjelmointikieltä. (2018 Fram.)

GameMaker Studio sai alkunsa vuonna 1999 ja se tunnettiin nimellä Animo. Nykyään uusin versio on todella hyvä työkalu aloitteleville pelinkehittäjille sen helppokäyttöisyyden ansiosta. Se siis sisältää drag and drop -ominaisuuden, eli käyttäjät voivat valita erilaisia objekteja ja muita ominaisuuksia valikoista välittämättä itse varsinaisesta koodaamisesta, joka toki on mahdollista. Myös drag and drop -menetelmää käyttävän on mahdollista nähdä automaattisesti generoitu koodi. Aloittelijaystävällisyyden vuoksi pelimoottoria käytetäänkin opetustarkoituksessa monissa kouluissa. (2017 Cleaver.)

3.4 Unity

Unity-pelimoottori on Unity Technologies -yrityksen omistama. Se on yksi suosituimmista pelimoottoreista markkinoilla. Pelimoottori tehtiin vuonna 2004 alun perin yhtä tiettyä peliä varten, mutta se ei tuottanut tarpeeksi rahaa, joten pelimoottorin kehittäjät alkoivatkin kehittää pelimoottoriaan eteenpäin ja myymään sitä. Pelimoottorista on olemassa maksullisia versioita, mutta ilmainen versio sisältää kaikki tärkeimmät ominaisuudet. Pelimoottoria voi käyttää Windowsilla, Macilla ja Linuxilla. Personal-versio on ilmainen ja sisältää koko pelimoottorin. Ilmaisessa versiossa on 100 000:n dollarin tulorajoitus, eli jos sillä tehty peli tuottaa enemmän, niin pitää siir-

tyä maksulliseen versioon. Unity ei myöskään ota mitään rojalteja saaduista tuloista. (2016 Murphy.) Ilmaisella Personal-versiolla tehdyssä pelissä on myös aloitusruudussa Unityn logo, joka on mahdollista poistaa vain maksullisissa versioissa. Pro-versio maksaa 35 dollaria kuukaudessa. Se nostaa tulorajan 200 000:een ja sisältää muitakin lisäyksiä sekä hyötyjä. Pro-versio maksaa 125 dollaria kuukaudessa, eikä siinä ole tulorajaa ollenkaan. On olemassa myös Enterprise-versio, josta sovitaan erikseen yrityksen kanssa. (2017 Placzek.)

Itse pelimoottori on tehty C++-kielellä, mutta itse peliohjelmointi tapahtuu C#-ohjelmointikielellä (2016 Murphy.) Muita Unityllä käytettäviä ohjelmointikieliä ovat JavaScriptin kaltainen UnityScript ja Boo, mutta Unity on aloittanut näiden alasajon ja vähentämisen, ja siirtynyt enemmän C#:n käyttöön. (2017 Fine.)

Unityllä voi tehdä sekä 3D- että 2D-pelejä ja onkin hyvä työkalu molemmille eri pelityypeille. Unity tukee yli kahtakymmentä eri laitealustaa peleille, muun muassa Windows, Mac, Linux, tunnetut pelikonsolit ja mobiilikäyttöjärjestelmät. Myös VR eli virtuaalitodellisuuspelit ovat tuettuja. Unity mahdollistaa myös moninpelien teon. (2016 Murphy.)

4 DEMO-PELI

Lähtökohtana pelin tekemiselle oli se, että en ole koskaan tehnyt peliä. Siitä siis lähdettiin liikkeelle. Mitään pelimoottoria en ollut käyttänyt ennen, ja sen valinnassa oli hieman miettimistä. Päädyin kuitenkin valitsemaan Unityn sen ilmaisuuden ja aloittelijaystävällisyyden takia. Koska se on erittäin suosittu, oli siitä helppo löytää erilaisia tutoriaaleja ja muita ohjeita.

Koska en ollut ennen peliä tehnyt, niin lähdin suunnittelemaan yksinkertaista 2D-peliä. En ole erityisen taiteellinen enkä ole ennen luonut mitään grafiikkaa, joten pelin tulisi olla yksinkertainen grafiikoiden ja ulkoasun suhteen. Olin epätietoinen kuinka paljon monimutkainen peli vaatii työtä. Myös pelko ajan riittämättömyydestä vaikutti päätökseeni tehdä yksinkertaisempi peli. Ottaen huomioon aloitustasoni olen tyytyväinen lopputulokseen.

4.1 Pelin idea ja tavoitteet

Pelin genreksi valikoitui tasohyppelypeli, joka on yksi toimintapelien alalajeista. Olen aina halunnut tehdä jollain tapaa Super Mario -tyyppisen 2D-tasohyppelypelin, joten sen tyyppistä lähdin suunnittelemaan. Peliä ei ole myöskään sen kummemmin julkaistu erilaisille alustoille vaan sitä on testattu ja pelattu vain Windowsilla.

Pelissä on tarkoitus liikkua pelihahmolla 2D-maailmassa, varoa erilaisia vaaroja ja päästä jokaisen kentän lopussa olevaan maaliin. Maalista pääsee seuraavaan kenttään tai pelin loppuun, jos kyseessä on viimeinen kenttä. Pelin grafiikat ovat yksinkertaisia ja pelissä pelattava hahmo onkin yksinkertainen neliö, jolla on silmät ja suu. Pelissä on myöskin yksi vihollinen, joka myöskin on neliö, joskin vähän eri kokoinen ja värinen. Pelaajan on joko väistettävä tai hypättävä tämän vihollisen päälle välttääkseen kuoleman. Pelissä on myös piikkejä, joita pelaajan on vältettävä. Pelaaja ei saa tippua kentän ulkopuolelle ja jokaisessa kentässä on aikarajoitus. Ajan loppuun kuluminen aiheuttaa kentän alkamisen alusta.

4.2 Pelin toiminnallisuus

Pelissä liikutaan nuolinäppäimillä ja siinä voi liikkua sivuille sekä hypätä ja tietysti tippua alas. Pelaajan on mahdollista tehdä myös toinen hyppy ilmassa. Eli jos pelaaja on hypännyt kerran ensin maasta, on hänellä vielä yksi mahdollinen hyppy käytettävänään ilmassa. Tätä onkin hyödynnetty kenttäsuunnittelussa, sillä kentissä on kohtia, joissa pelaajan täytyy tehdä hyppy ilmassa päästäkseen eteenpäin. Pelissä voi siis kuolla tai käytännössä kenttä alkaa alusta, kun pelaaja osuu vihollista sen sivulle, osuu piikkeihin, tippuu kentän rajojen ulkopuolelle tai kun aika loppuu. Vihollinen liikkuu eteenpäin niin pitkään kunnes taso sen alapuolella loppuu, minkä jälkeen se kääntyy ympäri ja jatkaa taas matkaansa. Pelin kamera on kiinnitetty pelihahmoon ja täten mahdollistaa sen, että pelaaja näkee aina missä hahmo liikkuu.

Pelin kentissä käytettävät tasot, joiden päällä pelaaja liikkuu, on tehty yksinkertaisista tiilistä, joita on laitettu vierekkäin ja päällekkäin. Nämä muodostavat perusrungon jokaiselle kentälle ja käytännössä määrittävät pelaajalle suunnan. Pelissä on myös tippuvia tasoja, jotka tippuvat tietyn ajan kuluttua pelaajan kosketettua niitä. Nämä tasot palautuvat kuitenkin omille paikoilleen jonkin ajan kuluttua niiden tippumisesta.

Peliin olisi voinut lisätä myös ääniä, mutta en kokenut sitä niin tärkeäksi. Siihen olisi voinut lisätä esimerkiksi valmiita ilmaisia äänitehosteita. Kuitenkin halusin tehdä kaiken itse ja koska minulla ei ole taitoja tai osaamista tehdä niitä itse, niin peli jäi äänettömäksi.

4.3 Käytetyt työkalut

Valitsin pelin tekemiseen Unity-pelimoottorin sen helppokäyttöisyyden ja siihen löytyvien erilaisten tutoriaalien perusteella. En ollut toki ennen käyttänyt sitä, joten siinä oli aluksi hieman opettelemista. Unityllä uutta projektia aloittaessa voi heti valita tekeekö 3D- vai 2D-pelin. Projektiin olisi voinut myös ladata valmiita assetteja, joilla voi saada esimerkiksi valmiit grafiikat, mallit ja pelihahmot, scriptit ja äänet. Tarkoituksena oli kuitenkin tehdä kaikki itse, joten en käyttänyt mitään niistä. Unityllä peliä

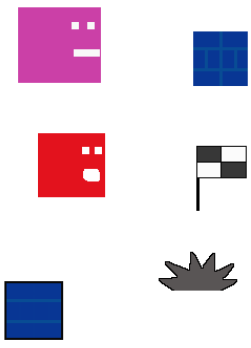
tehtäessä pelin kentät, valikot ja kaikki muutkin eri tilat tehdään ja tallennetaan omaan sceneensä, joiden välillä voidaan hyppiä eri tavoilla, esimerkiksi seuraavaan kenttään siirtyminen. Scenejä voi myös kopioida, mikä helpottaa muun muassa uusien kenttien tekemistä.

Unityssä kaikki asiat pelissä tehdään erilaisina objekteina ja näille voidaan lisätä erilaisia komponentteja, grafiikoita, ääniä ja scriptejä. Omassa pelissä käytin pelihahmoihin varsinkin komponentteja Box collider 2D ja Rigid body 2D, joista ensimmäisen avulla voidaan havaita erilaisten objektien väliset törmäykset ja kosketukset, kun taas jälkimmäinen lisää muun muassa painovoiman vaikutuksen objektiin, minkä avulla voidaan myös esimerkiksi muokata objektin massaa.

Grafiikan tekemiseen käytin Gimp-kuvanmuokkausohjelmaa. Se on melko suosittu ja ilmainen ohjelma. Gimp tarpeeksi hyvä varsinkin yksinkertainen grafiikan luomiseen. Tein sillä kaikille objekteille tarvittavat grafiikat. Tärkeää on vain se, että tiedostoissa ei ole taustaa ollenkaan.

Ohjelmointiin käytin Visual Studiota. Se asentui siinä samalla kun asensin Unityn. Olisin todennäköisesti käyttänyt muutenkin sitä, sillä se olen käyttänyt sitä aikaisemminkin. Ohjelmointikielenä tietysti C#.

4.4 Käytetyt grafiikat

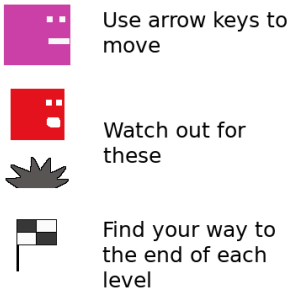


Kuva 2. Pelissä käytettävien objektien grafiikat

Grafiikoita tarvittiin pelissä käytettäviin erilaisiin objekteihin. Pelattava hahmo vaati luomisensa, vaikka siitä yksinkertainen oli tarkoitus tullakin. myös pelissä oleva vihollinen, piikit, käytettävät tasot ja jokaisen kentän maali tarvitsivat omat grafiikkansa. Ne oli kuitenkin mahdollista laittaa kaikki samaan tiedostoon (Kuva 2), sillä Unityssä ne on mahdollista erottaa toisistaan. Pinkki neliöhahmo toimi pelihahmona ja punainen neliö vihollisena.

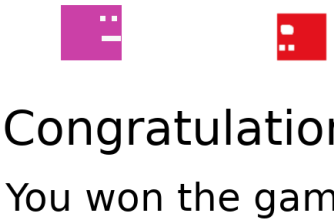
Myöskin pelissä käytettävät valikot ja näppäimet vaativat grafiikan luomisen. Pelin päävalikkoon halusin näppäimet eri toiminnoille (Kuva 3).

Kuva 3. Pelissä käytetyt näppäimet



Kuva 4. Ohje pelaajille

Päävalikossa pelaajan on myös mahdollista nähdä eräänlaiset ohjeet peliin, joten tein myös siitä oman kuvansa (Kuva 4). Myöskin pelin loppuun päästyään tulee oma loppuscenensä, joka myös vaati oman kuvan tekemisen (Kuva 5).



Kuva 5. Pelin lopussa oleva kuva

4.5 Pelin scenet ja kentät

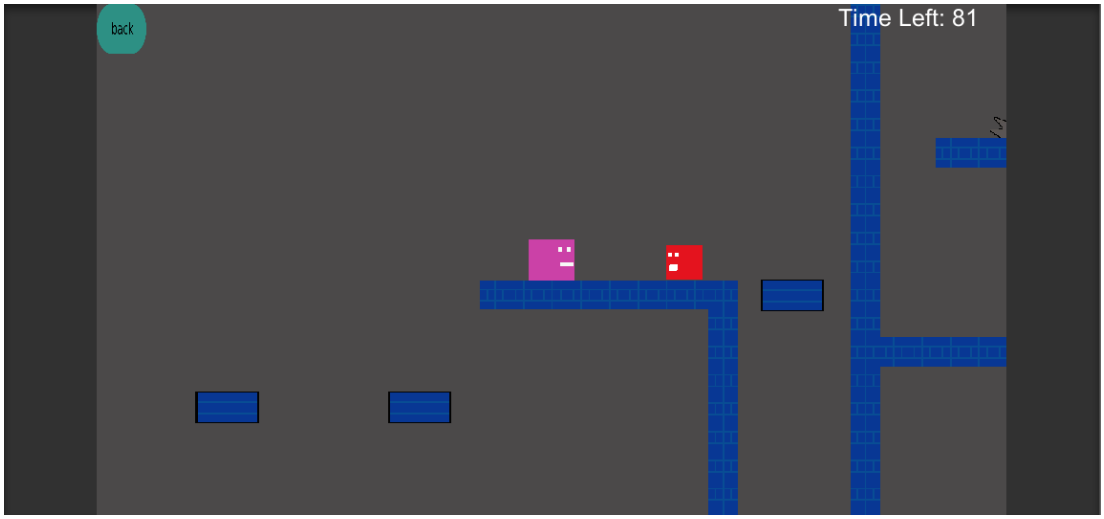
Peli sisältää yhteensä viisi eri sceneä, joista yksi on päävalikko, kolme on kenttiä ja viimeinen on loppuruutu, josta pääsee takaisin päävalikkoon. Tein peliin kolme kenttää, joista jokainen on suhteellisen pitkä. Kenttiä voi helposti tehdä lisää tulevaisuudessa, jos tätä projektia haluaa jatkaa.

Päävalikko on erittäin yksinkertainen. Siinä on vain neljä näppäintä ja yksinkertainen valkoinen tausta. Play-näppäimestä päästään ensimmäiseen kenttään. About-näppäintä painamalla avautuu ohjekuva (Kuva 4) pelaajalle ja muiden näppäinten tilalle tulee back-näppäin, josta päästään takaisin päävalikon (Kuva 6) alkuperäiseen tilaan. Quit-näppäin sulkee pelin.

Kuva 6. Päävalikko

Kenttäsceneissä on ruudun yläreunassa pieni käyttöliittymä. Siinä näytetään jäljellä oleva aika ja siinä on näppäin, josta pääsee takaisin päävalikkoon.

Ensimmäisestä kentästä (Kuva 7) pyrin tekemään suhteellisen helpon, mutta kuitenkin niin, että siinä on jotain haastetta. Siinä on muutamia vihollisia ja piikkejä. Siinä on muun muassa kohta, jossa pelaajan täytyy tippua putoavan tason mukana jonkin aikaa päästäkseen eteenpäin.

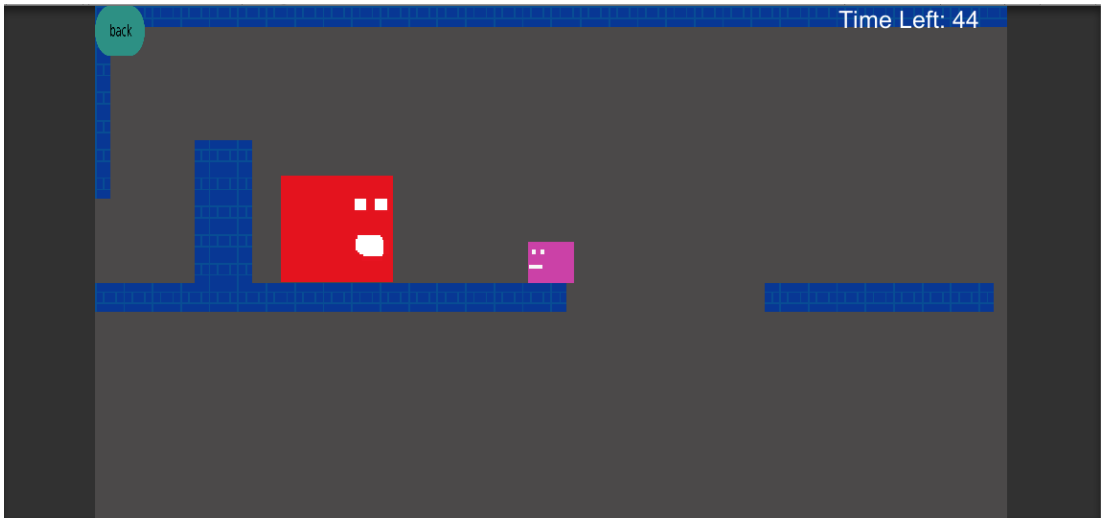


Kuva 7. Pelikuvaa ensimmäisestä kentästä

Toinen kenttä on suhteellisen yksinkertainen ja suoraviivainen kenttä. Siinä on alussa yksi keskivaikea hyppykohta. Kentän lopussa olevien piikkien välistä hyppiminen ja väistäminen saattaa olla haasteellista, ja niihin osuminen tarkoittaa koko kentän alusta alkamista.

Kolmannesta ja viimeisestä kentästä halusin tehdä jonkin verran erikoisemman ja hauskemman. Siinä on kaksi erilaista harhaanjohtavaa ja yllättävää kohtaa. Heti alussa pelaaja todennäköisesti lähtee kohti umpikujaa, sillä aluksi se näyttää selvästi ainoalta vaihtoehdolta, mutta sitä se ei kuitenkaan ole. Toisessa kohdassa tasoina käytettävän tiilen kohdalla onkin vain kuva tiilistä ja pelaaja hyvin todennäköisesti tipahtaa siitä alas ensimmäisellä kerralla, mutta ei kuitenkaan kuolemaan vaan hän joutuu menemään takaisin samaa reittiä. Kentässä on myös yksi vaikea tippuvien tasojen kohta, jossa pelaaja joutuu hyppimään useiden tippuvien tasojen päälle sekä käyttämään kaksoishyppyjä. Tässä kohdassa kuluukin helposti aikaa ja aika saattaakin lop-

pua kesken, vaikka pelaaja pääsisi tämän kohdan lopulta läpi. Kentän lopussa on vielä eräänlainen loppuvastus (kuva 8), joka tosin ei ole kuin iso versio normaalista vihollisesta.



Kuva 8. Pelikuvaa kolmannesta kentästä

Viimeinen scene eli loppuruutu sisältää vain onnitteluviestin (kuva 5), sekä back-näppäimen, josta pääseekin takaisin päävalikkoon. Kaikki scenet ovat siis eräänlaisessa loopissa toistensa kanssa, joten ei tule tilannetta missä scene yhtäkkiä loppuisi, eikä siitä ei pääsisi enää eteenpäin seuraavaan.

4.6 Scriptit

Peli ja siinä käytettävät objektit vaativat myös omia koodejaan toimiakseen halutulla tavalla. Pelihahmo ja vihollinen muun muassa täytyi saada liikkumaan ja reagoimaan halutulla tavalla toisiinsa. UnityEngine-nimiavaruuden avulla pystyi hyödyntämään valmiita luokkia ja metodeja näihin tarkoituksiin, mutta myös omia metodeja piti tehdä.

4.6.1 PlayerController

Tässä luokassa on metodit pelihahmon liikuttamiseen sivuille, hahmon kääntäminen aina menosuuntaan, hyppimiseen sekä hahmon ja vihollisen välinen kosketus.

PlayerMove-metodissa (Kuva 9) tallennetaan ensin pelaajan syöte moveInput muuttujaan, minkä jälkeen voidaan pelihahmon liikettä muuttaa haluttuun suuntaan halutulla nopeudella. Lisäksi hahmoa käännetään menosuuntaan. Tämä tarkistetaan if-lauseilla. Itse kääntämisen hoitaa Flip-metodi, joka käytännössä kääntää hahmon toisin päin ja muuttaa facingRight-muuttujan bool-arvoa.

```
void PlayerMove()
{
    //player moving
    moveInput = Input.GetAxis("Horizontal");
    rb.velocity = new Vector2(moveInput * speed, rb.velocity.y);

    if (facingRight == false && moveInput > 0)
    {
        Flip();
    }
    else if (facingRight == true && moveInput < 0)
    {
        Flip();
    }
}
```

Kuva 9. PlayerMove-metodi

Pelihahmon kaksoishyppyä varten tarvitaan hahmolle ensin objekti, jonka avulla voidaan tarkistaa onko hahmo maassa. Tämä tapahtuu valmiin UnityEnginestä saadun Physics2D.OverlapCircle-metodin avulla. PlayerJump-metodissa (Kuva 10) tarkastellaan, että onko hahmo maassa vai ei. Jos se on, niin sille voidaan antaa lisähyppyjä. Jos pelaaja painaa ylänuolta, niin hahmo hyppää. Jos hahmo on ilmassa ja sillä on vielä lisähyppy jäljellä, niin myös siinä tapauksessa hahmo hyppää vielä toisen kerran. Tämän jälkeen lisähyppyjä kuitenkin vähennetään. Tässä tapauksessa kun on vain yksi lisähyppy, on pelaajan on mahdollista hypätä uudelleen vasta hahmon tultua takaisin maahan.

```
void PlayerJump()
{
    //Player jumping
    if (Grounded == true)
    {
        extraJump = extraJumpValue;
    }
    if (Input.GetKeyDown(KeyCode.UpArrow) && extraJump > 0)
    {
        rb.velocity = Vector2.up * jumpForce;
        extraJump--;
    }
    else if (Input.GetKeyDown(KeyCode.UpArrow) && extraJump == 0 && Grounded == true)
    {
        rb.velocity = Vector2.up * jumpForce;
    }
}
```

Kuva 10. PlayerJump-metodi

Hahmon ja vihollisen väliset tapahtumat hoidetaan valmiilla OnCollisionEnter2D-metodilla (Kuva 11), joka suoritetaan kun hahmo koskettaa toista hahmoa, jolla on myös oma Rigidbody, eli tässä tapauksessa vihollista. Ensin luodaan enemy-olio, jolle annetaan collisionista saatu arvo. Jos tämä olio ei ole tyhjä eli törmäys on tapahtunut, niin tarkistetaan seuraavaksi, onko törmäys tapahtunut vihollisen yläpuolelta ja jos on, niin vihollinen tuhotaan. Muussa tapauksessa pelaaja on osunut viholliseen toisesta suunnasta, ja suoritetaan Hurt-metodi, joka käytännössä lataa kentän uudestaan eli pelaaja kuolee.

```
void OnCollisionEnter2D(Collision2D collision)
{
    enemy enemy = collision.collider.GetComponent<enemy>();
    if(enemy != null)
    {
        foreach(ContactPoint2D point in collision.contacts)
        {
            if (point.normal.y >= 0.9)
            {
                enemy.Hurt();
            }
            else
            {
                Hurt();
            }
        }

    }
}
```

Kuva 11. OnCollisionEnter2D-metodi.

4.6.2 Enemy

Enemy-luokkaa tarvitaan vihollisen liikuttamiseen. Myös vihollinen tarvitsee oman objektinsa tunnistaakseen maan. Tällä kertaa maata skannataan Physics2D.Raycast-metodin avulla. Tarkoituksena on saada vihollinen liikkumaan edestakaisin ja kääntymään aina kun maa loppuu. toisin sanoen aina kun skannaus ei tunnista enää maata, niin vihollinen käännetään liikkumaan toiseen suuntaan. Tämä toteutetaan if-ehtolauseilla. Näin saadaan edestakaisin liikkuva vihollinen aikaiseksi. Luokka sisältää myös Hurt-metodin, jota käytetään vihollisen tuhoamiseen luokassa PlayerController.

4.6.3 FallingPlatform

Tippuvat tasot ovat hieman erilaisia normaaleihin tasoihin verrattuna. Ne tipahtavat tietyn ajan kuluessa pelaajan koskettaessa niitä ja tippumisen jälkeen ne vielä palaavat omille paikoilleen tietyn ajan kuluttua. Tämän jälkeen pelaaja voi yrittää uudelleen.

Luokassa (Kuva 12) ensin tason nykyinen sijainti tallennetaan Vector2-tyyppiseen muuttujaan startPosition. Tason Rigidbody on myös asetettu kinematic-tilaan, joten se pysyy paikoillaan. OnCollisionEnter2D-metodissa tarkastetaan, onko pelaaja osunut tasoon ja jos on, niin suoritetaan Fall-metodi. Fall-metodi asettaa tason kinematic-tilan false-tilaan, jolloin taso tippuu. Tämän jälkeen suoritetaan IEnumerator ReturnPlat, joka asettaa kinematic-tilan taas trueksi ja palauttaa tason alkuperäiselle paikalleen startPosition-muuttujan avulla.

```
void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.collider.CompareTag("Player"))
    {
        Invoke("Fall", fallDelay);
        StartCoroutine(ReturnPlat());
    }
}

void Fall()
{
    Rb.isKinematic = false;
}

IEnumerator ReturnPlat()
{
    yield return new WaitForSeconds(returnTime);
    Rb.isKinematic = true;
    transform.position = startPosition;
    Rb.velocity = Vector2.zero;
}
```

Kuva 12. FallingPlatform

4.6.4 Muut luokat

PlayerDeath-luokkaa käytetään kentän uudelleen aloittamiseen, kun pelaaja osuu piikkeihin tai tippuu tarpeeksi alas kentän ulkopuolelle. Ulkopuolelle joutuminen tarkistetaan ehtolauseella, eli jos pelaaja joutuu rajan alapuolelle, niin suoritetaan Death-metodi, joka taas lataa kentän uudelleen. Piikkeihin osuminen tarkastetaan OnCollisionEnter2D-metodilla ja osuman sattuessa suoritetaan myös Death-metodi. Finish-luokka on sidottu maalilippuobjektiin ja siinä vain käytännössä OnColli-

sionEnter2D-metodilla tarkastetaan pelaajan osuminen siihen, minkä tapahtuessa ladataan seuraava scene, joka on joko seuraava kenttä tai loppuruutu.

TimeLeft-luokkaa käytetään jäljellä olevan ajan näyttämiseen ja kentän uudelleen lataamiseen, jos aika pääsee loppumaan. Sen avulla jäljellä oleva aika saadaan reaaliaikaisesti näkyville ruudun yläpuolella olevaan käyttöliittymäelementtiin.

MainMenu-luokka on tehty näppäimiä varten. Siinä on kolme metodia, joita voidaan liittää eri näppäimille ja ne suoritetaan niitä painamalla. Metodeja ovat seuraavat:
- StartScene, joka lataa seuraavan scenen. Käytetään play-näppäimessä
- QuitGame, joka sulkee pelin. Käytetään quit-näppäimessä
- Menu, joka lataa päävalikon. Käytetään back-näppäimessä

4.7 Projektin jatkaminen

Peliä voisi työstää eteenpäin vielä vaikka kuinka paljon. Olen jo jonkin verran miettinyt erilaisia ominaisuuksia ja objekteja mitä peliin voisi vielä myöhemmin lisätä. Hahmolle voisi lisätä esimerkiksi wall jump -hyppimisen, jolla voisi hyppiä seinistä, mikä lisäisi paljon uusia mahdollisuuksia kenttien suunnitteluun. Kenttiä voisi tehdä enemmän. Peliin voisi lisätä kerättäviä esineitä, joista voisi vaikka saada pisteitä, jotka näkyisivät jossain. Ehkä hahmo voisi myös jostain esineestä esimerkiksi muuttaa muotoa ja siten tulla kestävämmäksi vihollisia vastaan. Vihollisia voisi myös tehdä enemmän ja erilaisia esimerkiksi lentäviä tai vaikka hyppiviä. Hahmoille voisi muutenkin lisätä jonkinlaisia animaatioita.

Haluaisin toteuttaa pelin Androidilla pelattavaksi, mikä vaatisi kosketusnäyttöohjauksen luomista. En ole suunnittelemassa sen julkaisua yleisölle vaan pidän sen vain omana projektinani.

Kyseessä oli ensimmäinen pelini, joten olen tyytyväinen lopputulokseen ja siihen, että sain tehtyä pelistä toimivan ja sopivan haasteellisen.

LÄHTEET

Anupama, S. 2018. How to create a production plan concept – Important Phases to consider. Viitattu. 13.11.2018. Saatavissa: https://www.ommzi.com/create-production-plan-concept-important-phases-consider/

Bates, B. 2004. Game Design, second edition. Boston: Course Technology. Viitattu 11.6.2018. Saatavissa: https://samk.finna.fi/Record/nelli26_samk.1000000000032451

Bycer, J. 2018. Why UI Design is a Challenge for Video Games. Viitattu. 13.11.2018. Saatavissa: https://medium.com/@GWBycer/why-ui-design-is-a-challenge-for-video-games-67b2fc7ecd51

Cleaver, S. 2017. GameMaker Studio 2: Changing the GameMaker. Viitattu 27.6.2018. Saatavissa: https://www.mcvuk.com/development/gamemaker-studio-2-changing-the-gamemaker

Clinton, K. 2014. Agile Game Development. Viitattu. 26.9.2018. Saatavissa: http://blog.agilegamedevelopment.com/2014/09/why-agile-game-development.html

Crump, T. 2018. Top 10 Graphic Design Software for Game Devs. Viitattu. 13.11.2018. Saatavissa: https://www.buildbox.com/top-10-graphic-design-software/

Enger, M. 2013. Game Engines: How do they work? Viitattu 18.6.2018. Saatavissa: https://www.giantbomb.com/profile/michaelenger/blog/game-engines-how-do-they-work/101529/

Fine, R. 2017. UnityScript's long ride off into the sunset. Viitattu. 28.6.2018. Saatavissa: https://blogs.unity3d.com/2017/08/11/unityscripts-long-ride-off-into-the-sunset/

Fram, L. 2018. Best Game Engines for Indie Game Development. Viitattu 26.6.2018. Saatavissa: https://blog.g2crowd.com/blog/game-engine/best-game-engines-indie-game-development/

Gamedesigning.org. 2018. Beginner's Guide to Game Mechanics. Viitattu 13.11.2018. Saatavissa: https://www.gamedesigning.org/learn/basic-game-mechanics/

Gamedesigning.org. 2018. The Ultimate Guide to Character Design. Viitattu 15.6.2018. Saatavissa: https://www.gamedesigning.org/learn/character-design/

Khurramsamad. 2016. Game development process. Viitattu. 26.9.2018. Saatavissa: https://blogs.geniteam.com/2016/02/01/game-development-process/

Lecky-Thompson, G. 2008. Video Game Design Revealed. Boston: Course Technology. Viitattu 11.6.2018. Saatavissa: https://samk.finna.fi/Record/nelli26_samk.1000000000713094

Lozada, D. 2018. The Future of Video Game Platforms. Viitattu. 2.10.2018. Saatavissa: https://www.keengamer.com/article/18340_the-future-of-video-game-platforms

Mirillis Team. 2017. Complete List of Game Genres. Viitattu 11.6.2018. Saatavissa: https://mirillis.com/blog/en/complete-list-of-game-genres/

Mitchell, B. L. 2012. Game Design Essentials. Indianapolis: John Wiley & Sons Icn. Viitattu 11.6.2018. Saatavissa: https://samk.finna.fi/Record/nelli26_samk.2670000000166876

Murphy, K. 2016. Unity Game Engine Review. Viitattu 28.6.2018. Saatavissa: https://www.gamesparks.com/blog/unity-game-engine-review/

Newgenapps.com. 2018. Unreal Engine Review: Pros, Cons, and Suitability. Viitattu 26.6.2018. Saatavissa: https://www.newgenapps.com/blog/unreal-engine-review-pros-cons-suitability

Pc dreams. 2016. Different Types of Video Game Platforms Popular Today. Viitattu. 3.10.2018. Saatavissa: https://pcdreams.com.sg/different-types-of-video-game-platforms-popular-today/

Placzek, M. 2017. Introduction to Unity: Getting Started – Part 1/2. Viitattu. 28.6.2018. Saatavissa: https://www.raywenderlich.com/147687/introduction-unity-getting-started-part-12

Richter, F. 2018. The Most Important Gaming Platforms in 2018. Viitattu. 2.10.2018. Saatavissa: https://www.statista.com/chart/4527/game-developers-platform-preferences/

Smiley, B. 2012. What is a game engine? Viitattu 18.6.2018. Saatavissa: http://www.deluge.co/?q=what-is-a-game-engine

Tefler, A. 2018. Why Games Fail: Pre-Production. Viitattu. 26.9.2018. Saatavissa: https://mobilefreetoplay.com/why-games-fail-pre-production/

Unity3d.com. 2018. Game engines—how do they work. Viitattu 14.11.2018. Saatavissa: https://unity3d.com/what-is-a-game-engine

Wawro, A. 2018. Appreciating the magic (and power) of hidden game mechanics. Viitattu. 13.11.2018. Saatavissa: https://www.gamasutra.com/view/news/315652/Appreciating_the_magic_and_power_of_hidden_game_mechanics.php

Williams, G. 2015. Understand game platform types. Viitattu. 3.10.2018. Saatavissa: https://georgiawilliams5.wordpress.com/2015/01/12/understand-game-platform-types/

Wilson, J. 2018. Level design: Tricks of the trade. Viitattu. 13.11.2018. Saatavissa: https://www.gamasutra.com/blogs/JonathonWilson/20180720/322564/Level_design_Tricks_of_the_trade.php