Mikko Raussi

MOBIILIPELIN KEHITYS JA Jjulkaisu UNREAL ENGINELLÄ

Opinnäytetyö
Tietotekniikan koulutus

2018

Kaakkois-Suomen ammattikorkeakoulu

Tekijä	Tutkinto	Aika
Mikko Raussi	Insinööri (AMK)	Toukokuu 2018

Opinnäytetyön nimi

Mobiilipelin kehitys ja julkaisu Unreal Enginellä

33 sivua

Toimeksiantaja

Kuura Playhouse Oy

Ohjaaja

Lehtori Niina Mässeli

Tiivistelmä

Tämän opinnäytetyön tarkoituksena oli kehittää mobiilipeli Android-laitteille Unreal Enginellä ja julkaista se Googlen Play Kauppaan. Peliä ei ehditty saamaan valmiiksi julkaisua varten. Työssä tutkitaan kuitenkin, mitä pelin julkaisu vaatii Unreal Enginen ja Play Kaupan kannalta.

Pelin tekemiseen valittiin Unreal Engine -pelimoottori, jonka on kehittänyt Epic Games. Yrityksessä käytetään pääasiallisesti Unity3D -pelimoottoria, mutta tämän projektin avulla haluttiin nähdä, miten mobiilipelin kehitys Unreal Enginessä tapahtuu ja voisiko sitä mahdollisesti käyttää muissa yrityksen projekteissa. Ohjelmointi päätettiin toteuttaa Unreal Enginen Blueprint Visual Scripting -järjestelmällä, joka mahdollistaa koodin luomisen visuaalisesti perinteisen kirjoittamisen sijaan.

Tässä työssä käydään läpi, miten uusi projekti luodaan Unreal Enginessä ja mitkä asetukset vaikuttavat mobiilipelin toimivuuteen ja kehitykseen. Työssä selvitetään myös Unreal Enginen käyttämiä termejä sekä kuvaillaan, miten pelin eri ominaisuuksia on toteutettu. Toteutukset käydään läpi vain pääpiirteittäin, koska peli on Kuuran omaisuutta ja se voidaan julkaista myöhempänä ajankohtana.

Peliin ei ehditty toteuttaa kaikkia haluttuja ominaisuuksia. Esimerkiksi valikot ja grafiikat muualta kuin käyttöliittymästä jäivät puuttumaan. Mikromaksuja ei myöskään ehditty toteuttaa. Siinä määrin peli on kuitenkin valmis, että sitä pystyy pelaamaan.

Työ antoi paljon oppia Unreal Enginen käytöstä. Se vaikutti soveltuvan hyvin mobiilipelin kehitykseen. Työn ansiosta jäi myös parempi ymmärrys siitä, mitä pelin julkaisemiseen vaaditaan.

Asiasanat

mobiilipelit, Android, pelikehitys, julkaisu, Google

South-Eastern Finland
University of Applied Sciences

Author (authors)	Degree	Time
Mikko Raussi	Bachelor of Engineering	May 2018

Thesis title

Mobile game development and publishing with Unreal Engine

33 pages

Commissioned by

Kuura Playhouse Oy

Supervisor

Niina Mässeli

Abstract

The purpose of this thesis was to develop a mobile game for Android devices and publish it in Google Play Store. Even though the game was not finished on time for release, this thesis still examines the necessary steps to release a game to Google Play Store with Unreal Engine.

The chosen game engine for this project was Unreal Engine which is developed by Epic Games. Kuura uses primarily Unity3D engine for development, but this thesis study gave a chance to examine Unreal Engine in mobile game development and to see if it could be used in other company projects. Programming was done using Blueprint Visual Scripting which allows the user to create code visually without having to write it.

This thesis describes how a new project is created in Unreal Engine and which settings affect mobile game playability and development. The thesis also explores common terms used by Unreal Engine and how different features of the game were implemented. The implemented features are not necessarily examined thoroughly since the game is property of Kuura and may be released at a later date.

Not of the game features could be implemented on time. For example, menus and graphics from other areas than the user interface were not finished. Microtransactions are also missing from the game. However, the game is still playable.

During the game development much data was gained regarding the use of Unreal Engine. It seemed to work well in mobile game development. This thesis also serves as a source of information as to what publishing a game requires.

Keywords

mobile games, Android, game development, publishing, Google

SISÄLLYS

1 JOHDANTO 6

2 SUUNNITTELU 7

2.1 Unreal Engine 7

2.2 Blueprintien käyttö 7

2.2.1 Actor 11

2.2.2 Pawn 12

2.2.3 Character 12

2.2.4 PlayerController 12

2.2.5 Game Mode 13

2.2.6 ActorComponent 13

2.2.7 SceneComponent 13

2.3 Kuvaus pelistä 14

3 PELIN KEHITYS 15

3.1 Uuden projektin luominen 15

3.2 Androidin pohjustus 16

3.3 Projektin asetukset 16

3.3.1 Use Mouse for Touch 17

3.3.2 Default Viewport Mouse Lock Mode 17

3.3.3 Show Console on Four Finger Tap 18

3.3.4 Orientation 18

3.3.5 Android SDK 19

3.4 Muuttujien tietokanta 19

3.5 Pelihahmojen toteutus 20

3.5.1 Hahmon syntyminen pelikentälle 20

3.5.2 Hahmon liikkuminen 22

3.6 Pelikenttä 22

3.6.1 Triggerbox 23

3.6.2 Syntymispisteet 24
3.6.3 Keskilinja 24
4 JULKAISU 25
4.1 Unreal Enginen säännöt 25
4.2 Julkaisu Androidille 27
4.2.1 Sovelluksen valmistelu 27
4.2.2 APK:n rakennus 28
5 YHTEENVETO 29
LÄHTEET 31
KUVALUETTELO 33

1 JOHDANTO

Kuura Playhouse Oy on suomalainen pelialan yritys, joka sijaitsee Kotkan Karhulassa. Työntekijöitä Kuuralla on kolme. Kuuran toimitusjohtana toimii Marko Haaja. Kuura on keskittänyt aikansa opiskelijoiden taitojen parantamiseen ja pyrkii antamaan opiskelijoille tilaisuuden suorittaa työharjoittelun yrityksessä. Pelien kehityksen lisäksi Kuuralta voi tilata verkkosivuja, muuttaa VHS-nauhoja digitaaliseen muotoon, sekä vuokrata GoPro-kameroita. Kuuralla on myös sauna ja kokoontumistila, joita voidaan vuokrata. (Kuura Playhouse Oy 2018.)

Työn tavoitteena oli kehittää ja julkaista haluttujen vaatimusten mukainen mobiilipeli Googlen Play Kauppaan ja dokumentoida kehityksen sekä julkaisun vaiheet. Peliä ei kuitenkaan saatu kokonaan valmiiksi, joten tässä työssä käsitellään mobiilipelin julkaisua teorian tasolla. Työssä tutkitaan myös, miten Unreal Engine soveltuu mobiilipelin kehitykseen. Kuura valitsi kyseisen pelimoottorin tätä työtä varten nähdäkseen, miten pelin kehitys sillä eroaa yrityksessä pääasiassa käytetyn Unity3D-pelimoottorin sijaan.

Pelin kehitys tapahtui Kuuran tiloissa Karhulassa. Työt aloitettiin tutustumalla Unreal Engineen Internetistä löytyvien opastusvideoiden avulla. Tämän jälkeen luotiin uusi projekti Unreal Enginellä ja aloitettiin kehittämään pelin ominaisuuksia. Pelin koodaamiseen käytettiin Unreal Enginen Blueprint Visual Scripting -järjestelmää. Peliin ehdittiin tuottaa grafiikkaa vain käyttöliittymään, mutta muualta se jäi puuttumaan. Hahmot käyttävät väliaikaisia grafiikoita.

Tässä työssä keskeisiä termejä tulevat olemaan Unreal Enginessä ohjelmointiin liittyvät termit. Blueprint-järjestelmä mahdollistaa visuaalisen ohjelmoinnin Unreal Enginessä. Noodi on visuaalinen esitys jostakin ohjelmallisesta toiminnallisuudesta ja tapahtumat suorittavat noodien sisältämät toiminnallisuudet. Blueprintit, noodit ja tapahtumat tullaan käsittelemään myöhemmissä luvuissa tarkemmin.

2 SUUNNITTELU

Pelin kehityksessä käytetään yleensä erilaisia ohjelmia, joiden avulla peli kasataan toimivaksi kokonaisuudeksi. Tässä projektissa pelin ohjelmointiin käytettiin Unreal Engine -pelimoottoria yrityksessä pääasiassa käytetyn Unity3D:n sijaan. Yritys päätyi käyttämään Unreal Engineä nähdäkseen, kuinka mobiilipelin kehitys sillä eroaa Unity3D:stä. Alakappaleissa kuvataan Unreal Engineä ja sillä tehtyä mobiilipeliä tarkemmin.

2.1 Unreal Engine

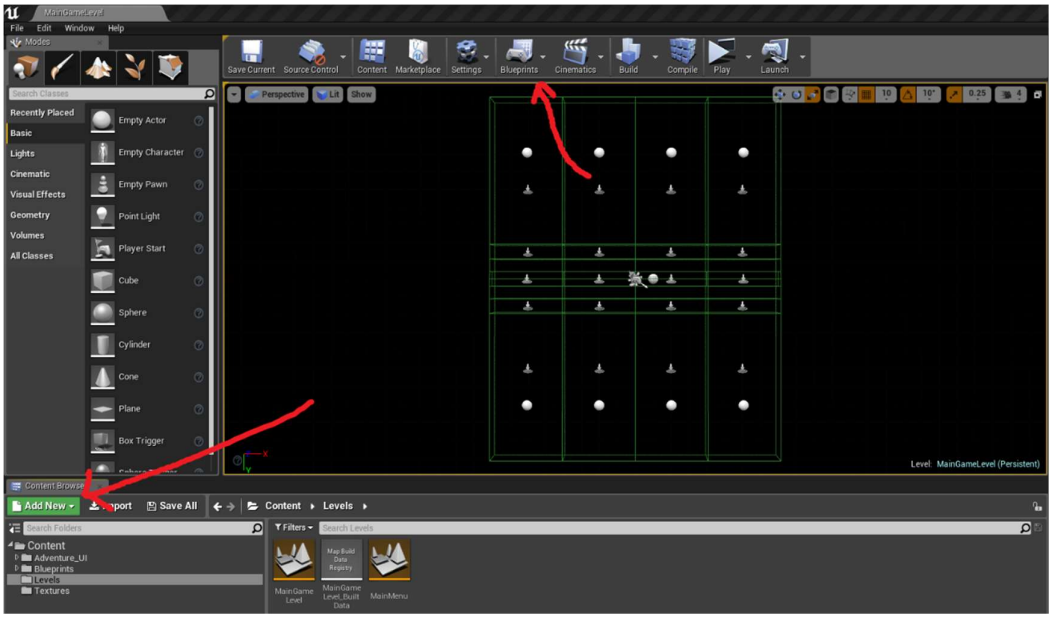
Unreal Engine on kokoelma työkaluja tehty kaikille, jotka työskentelevät reaaliajan teknologian parissa. Ohjelmaa voidaan käyttää yrityssovellusten ja elokuvallisten kokemuksien luomiseen sekä pelin kehitykseen tietokoneelle, konsoleille, ja mobiililaitteille. Unreal Engineä voidaan käyttää myös AR- ja VR-sovellusten kehityksessä. (Unreal Engine Features 2018.)

Tässä projektissa hyödynnettiin Unreal Enginen Blueprint Visual Scripting-järjestelmää. Suunnittelija-ystävällisellä visuaalisella blueprint-ohjelmoinnilla voidaan luoda ja julkaista sisältöä nopeasti kirjoittamatta koodia ollenkaan (Unreal Engine Features 2018). Blueprintien lisäksi Unreal Enginessä voidaan käyttää C++-ohjelmointikieltä pelin ohjelmointiin. Projektissa käytettiin pelkästään blueprintejä henkilökohtaisesta tottumuksesta.

2.2 Blueprintien käyttö

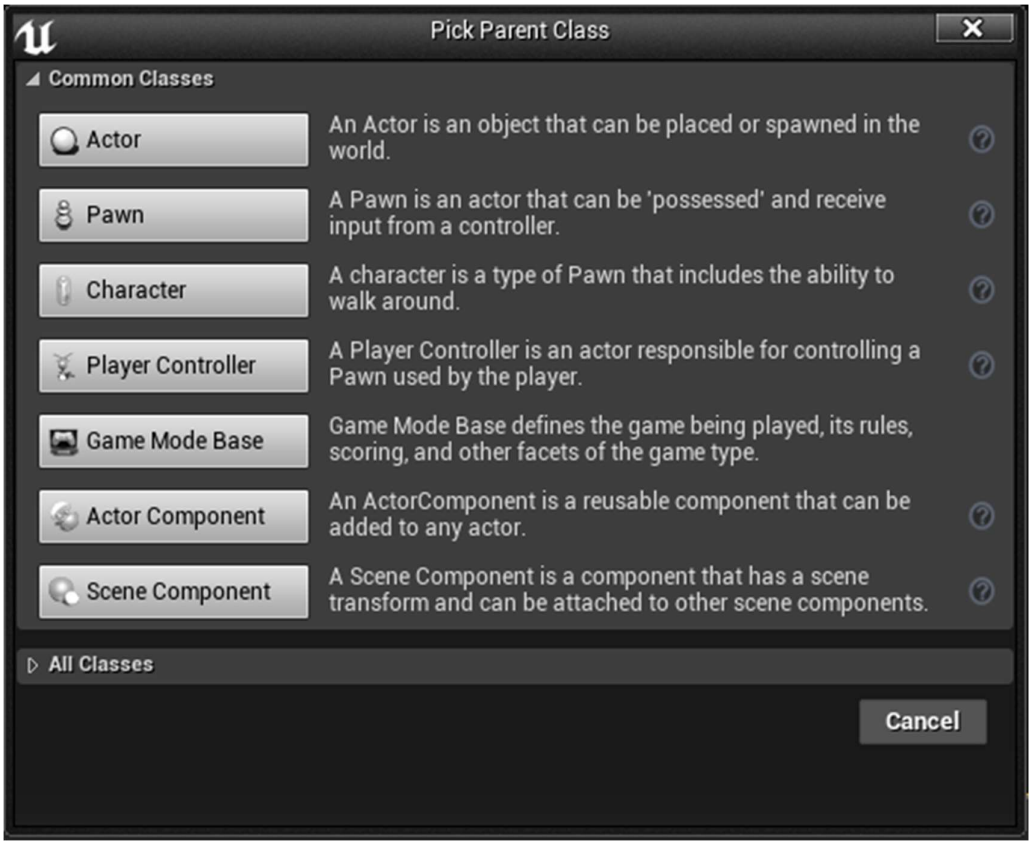
Tässä projektissa kaikki pelin ominaisuudet toteutettiin blueprintien avulla. Esimerkiksi pelihahmot ja kenttä ovat blueprintejä. Seuraavassa kuvataan, kuinka blueprint luodaan ja minkälaisia blueprintejä on mahdollista tehdä Unreal Enginessä.

Uusi blueprint luodaan painamalla “Blueprint”-kuvaketta muokkausohjelman työkalurivissä. Blueprint voidaan myös luoda painamalla “Add New” muokkausohjelman vasemmassa alalaidassa. Alla oleva kuva näyttää, mistä painikkeet löytyvät muokkausohjelmassa.



Kuva 1. Uusi blueprint

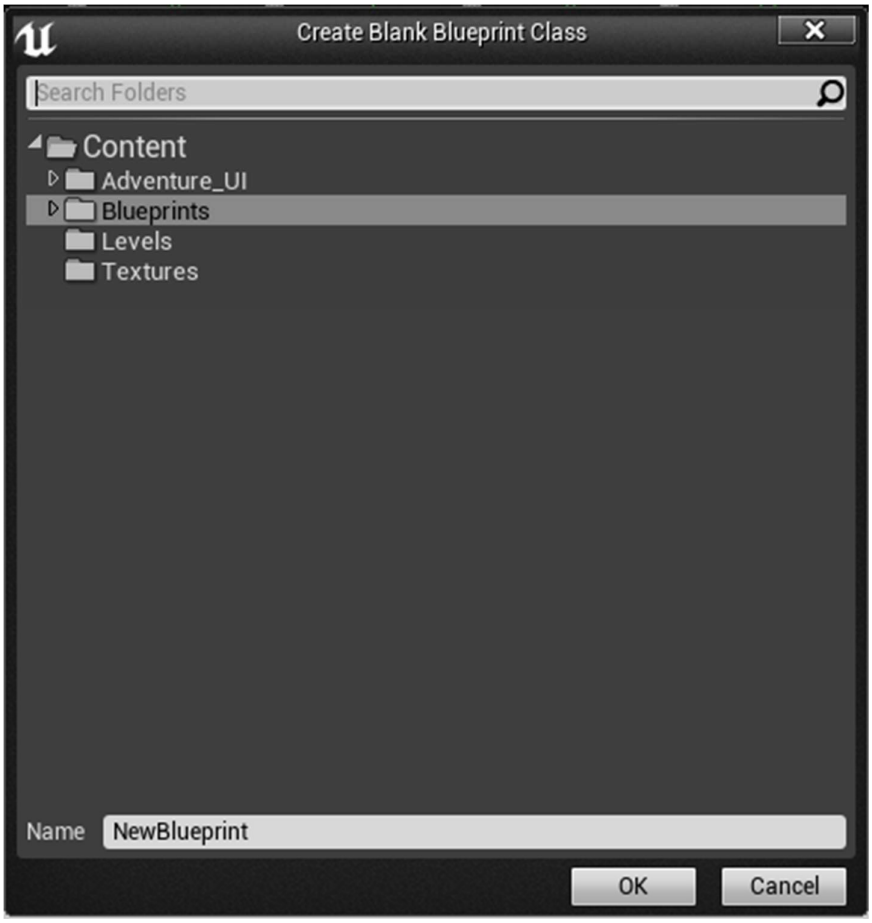
Kuvakkeen painaminen avaa uuden valikon, josta valitaan “New Empty Blueprint Class”. Jos painettiin “Add New”, niin avautuvasta valikosta valitaan “Blueprint Class”. Kumpikin edellisistä painikkeista avaa uuden näkymän, jossa valitaan luokka uudelle blueprintille.



Kuva 2. Blueprintin luokkia

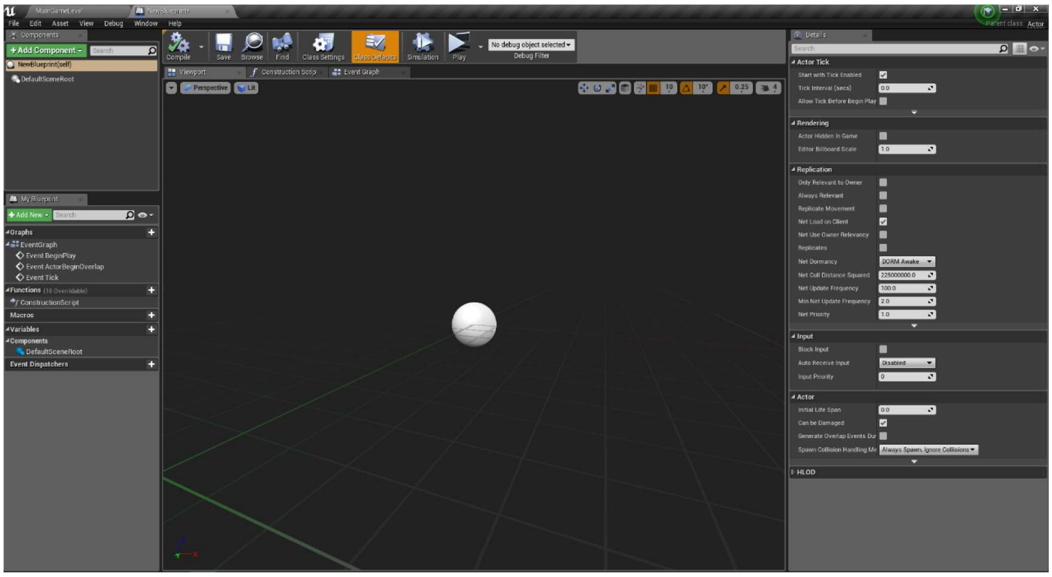
Kuvasta 2 nähdään, että blueprinteillä voi olla erilaisia luokkia. Luokkia kuvaillaan tarkemmin alaluvuissa.

Kun uudelle blueprintille on valittu luokka, avautuu kuvan 3 mukainen ikkuna. Tässä ikkunassa valitaan uudelle blueprintille nimi, sekä tallennussijainti. Projektissa tehtiin blueprinteille oma kansio, mikä auttoi pitämään projektiin liittyvät tiedostot siistissä järjestyksessä.



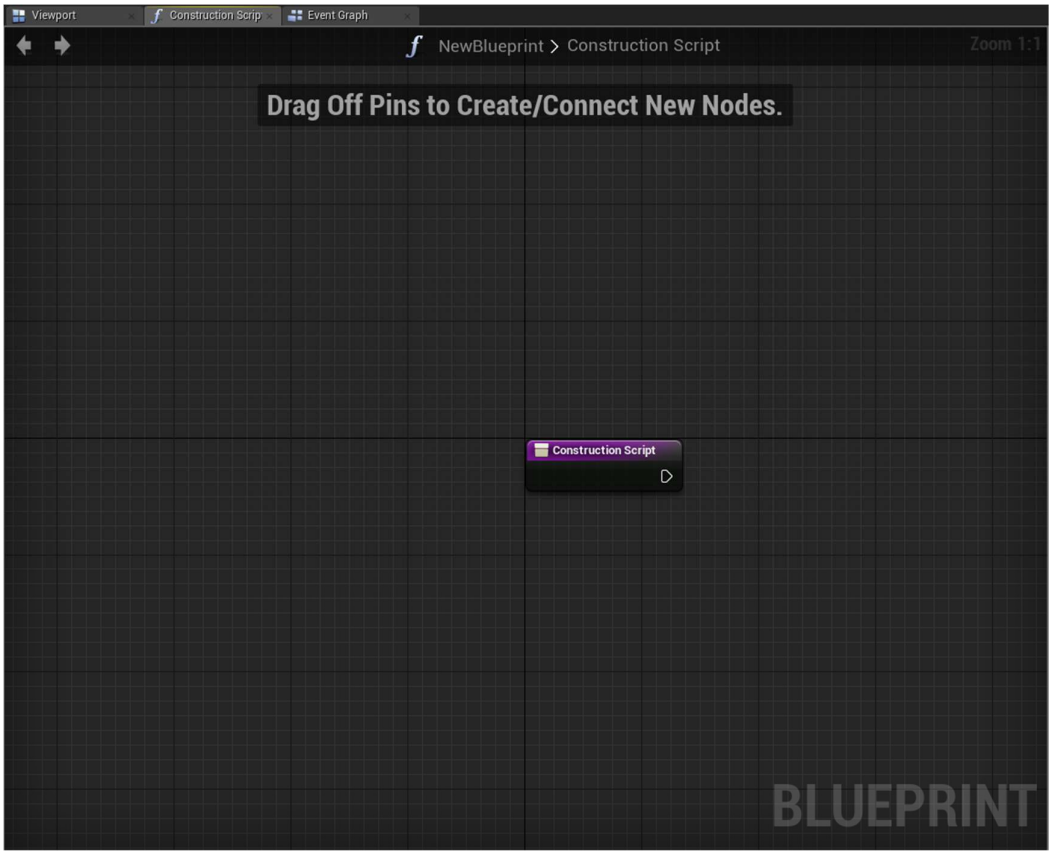
Kuva 3. Uuden blueprintin tallennus

Blueprintin tallentamisen jälkeen avautuu se blueprintin muokkausohjelmassa. Oikeassa laidassa voidaan muokata blueprintin asetuksia valitun luokan mukaan. Vasemmassa laidassa näkyy blueprintin komponentit ja nappula, josta lisätään uusia komponentteja, funktioita tai muuttujia blueprintiin. Keskelle avautuu kolmiulotteinen näkymä blueprintistä.



Kuva 4. Blueprintin oletusnäkymä

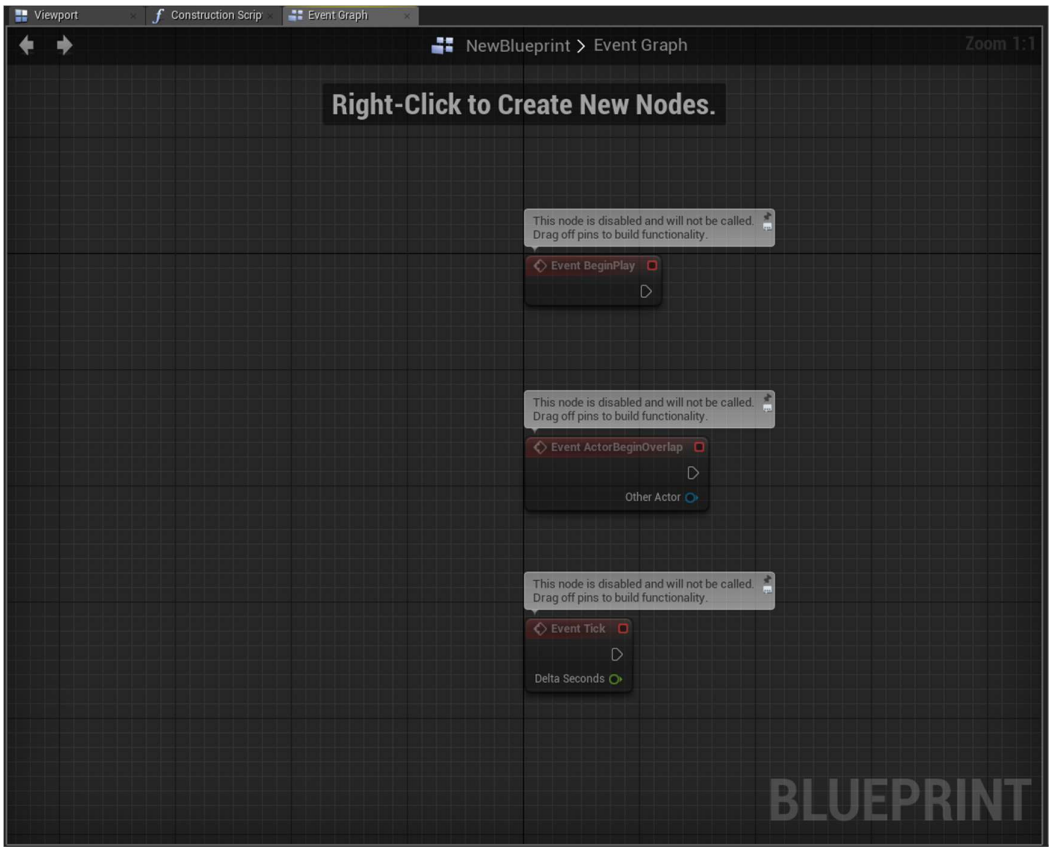
Kuva 4 havainnollistaa, miltä uuden actor-luokan blueprint näyttää. Keskellä olevaa näkymää voi vaihtaa näyttämään rakennuskoodinäkymän (construction script) tai tapahtumakaavionäkymän (event graph).



Kuva 5. Rakennuskoodi

Rakennuskoodissa määritellyt toiminnallisuudet suoritetaan, kun blueprint luodaan peliin. Tässä projektissa ei käytetty rakennuskoodia yhdenkään blueprintin kanssa.

Tapahtumakaavio sisältää blueprintin pelilogiikan, eli sen miten se toimii pelimaailmassa.



Kuva 6. Tapahtumakaavio

Kuvassa 6 on esimerkki uudesta tapahtumakaaviosta. Oletuksena siinä on mukana kolme usein käytettyä tapahtumaa, joista voi alkaa ohjelmoimaan blueprintin toiminnallisuuksia. Tapahtumakaavion käyttöä kuvataan myöhemmin pelin ominaisuuksien toteutuksesta kertovissa kappaleissa.

2.2.1 Actor

Actor on olio (object), joka voidaan asettaa tai luoda maailmaan (Actors 2018). Tässä projektissa actor-tyyppisiä blueprintejä ovat muun muassa pelikentän linjat sekä hahmojen syntymispisteet, joita kuvaillaan tarkemmin myöhemmissä kappaleissa.

2.2.2 Pawn

Pawn on kuin actor, mutta sitä pystyy hallita, ja se pystyy vastaanottamaan syötteitä ohjaimelta. Pawn on fyysinen esitys pelaajasta tai tekoälystä pelimaailmassa. Tämä tarkoittaa sitä, että pawn määrittää, miltä pelaaja tai tekoäly näyttää visuaalisesti ja miten se on vuorovaikutuksessa pelimaailman kanssa törmäysten ja muiden fyysisten vuorovaikutusten kanssa. Joissain peleissä pelaajalla ei välttämättä ole fyysistä muotoa, mutta pawn silti edustaa pelaajan sijaintia, rotaatiota ja muita ominaisuuksia. (Pawn 2018.)

2.2.3 Character

Character-luokka on laajennettu pawnista, lisäämällä luokkaan uusia komponentteja. Character on suunniteltu pystysuorassa olevalle pelaajan hahmolle, joka pystyy kävelemään, juoksemaan, hyppimään ja uimaan maailmassa. Luokka sisältää toteutuksia yksinkertaisista tietoverkko- ja syötemalleja. Toisin kuin pawn, characterin mukana tulee SkeletalMesh -komponentti, joka mahdollistaa edistyneet animaatiot, jotka käyttävät luurankoa. Character sisältää myös capsule-komponentin, mitä käytetään törmäyksen tarkastelussa ja CharacterMovement -komponentin, mikä mahdollistaa hahmon liikkumisen. (Character 2018.)

2.2.4 PlayerController

PlayerController on käyttöliittymä pawnin ja sen hallitseman pelaajan välillä. PlayerControlleria tehdessä tulee miettiä, mitkä toiminnallisuudet PlayerController sisältää ja mitkä toiminnallisuudet taas kuuluvat pawniin. On mahdollista, että pawn sisältää kaikki hahmon syötteet, mutta monimutkaisimmissa tapauksissa olisi suotavampaa, että syötteitä käsittelee PlayerController. Monimutkaisempia tapauksia ovat esimerkiksi useamman pelaajan samanaikainen syötteen käsittely tai mahdollisuus vaihtaa hahmoa pelissä. PlayerController myös säilyy koko pelin ajan, kun taas pawn voi kuolla ja syntyä uudestaan. Tämän johdosta esimerkiksi pistetilanne ja muut tiedot, joita ei haluta muuttuvan, tulisi olla PlayerControllerissa. (PlayerController 2018.)

2.2.5 Game Mode

Game Mode määrittää pelin sääntöjä. Kaikista avoimimmissakin peleissä on sääntöjä ja nämä säännöt muodostavat Game Moden. Yksinkertaisessa tasossa näitä sääntöjä ovat tämän hetkinen pelaajien ja katsojien määrä, sekä pelaajien ja katsojien maksimi määrä. Game Mode voi myös määrittää, miten pelaajat liittyvät peliin. Tähän voi liittyä sääntöjä syntymispisteen valitsemiseen ja muita syntymiseen, tai uudelleensyntymiseen liittyviä sääntöjä. Pelin asettaminen tauolle ja tauon käsittely, sekä siirtyminen kenttien välillä voivat myös kuulua Game Modeen. (Game Mode and Game State 2018.)

2.2.6 ActorComponent

Komponentit (components) ovat erityisen tyyppisiä olioita, jotka ovat tarkoitettu käytettäväksi actoreissa. Niitä käytetään yleensä tilanteissa, missä vaaditaan helposti vaihdettavia osia muuttamaan actorin käyttäytymistä tai funktionaalisuutta tietyissä asioissa. Esimerkiksi autoa ohjatessa se liikkuu eri tavalla kuin ilma-alus tai laiva, mutta niillä on muita yhtenäisyyksiä. Käyttämällä komponenttia ohjaamisen ja liikkumisen käsittelyyn voidaan mikä tahansa kulkuneuvo saada liikkumaan samalla tavalla kuin muut. (Components 2018.)

ActorComponent on perusluokka komponenteille, jotka määrittävät uudelleenkäytettävän käyttäytymisen, mikä voidaan lisätä actoreille. Näitä käytetään esimerkiksi pelaajan hallitsemiseen, actorin äänien toistamiseen ja siihen, miten actor luo valoa tai varjoja maailmaan. Toisin sanoen actorit koostuvat komponenteista. (Components 2018.)

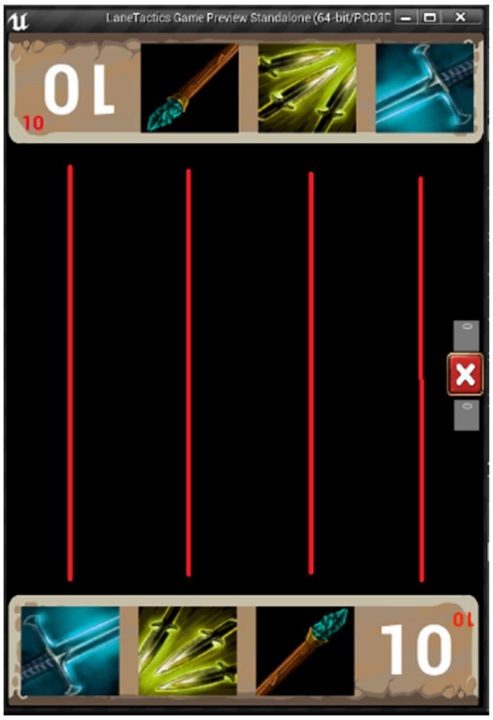
2.2.7 SceneComponent

SceneComponent on jatke ActorComponentista. SceneComponenteilla on sijainti, rotaatio ja skaala. Näiden ominaisuusten ansiosta SceneComponentteja voidaan kiinnittää toisiinsa. (Components 2018.)

2.3 Kuvaus pelistä

Pelistä haluttiin tehdä mobiilipeli Android-laitteille. Peliä pystyy pelaamaan samaan aikaan kaksi ihmistä samalla laitteella. Tavoitteena on voittaa toinen pelaaja kivi, paperi ja sakset -tyyppisessä ottelussa niin monta kertaa, että toisen pelaajan elämäpisteet putoavat nollaan. Elämäpisteiden pudottua nollaan alkaa toinen erä. Pelin voittaa se pelaaja, joka ensimmäisenä voittaa kaksi erää. Kivi, paperi ja sakset ovat pelissä korvattu sotilaalla, varkaalla ja velholla.

Pelissä on neljä linjaa, joista kuhunkin voi asettaa edellä mainitun hahmon. Hahmo liikkuu linjallaan eteenpäin, kunnes se pääsee keskelle linjaa. Hahmo jää paikalleen, jos se on ainut hahmo linjan keskellä. Toisen hahmon kohdatessaan molemmat hahmot poistetaan pelistä. Tämän jälkeen vähennetään elämäpiste siltä pelaajalta, jonka hahmo häviää tappelun.



Kuva 7. Pelinäkymä

Yllä oleva kuva hahmottaa sitä, miltä peli näyttää Unreal Enginessä ja mobiililaitteella. Kuvaan lisättiin punaiset viivat näyttämään, missä linjat sijaitsevat. Valkoiset numerot oikeassa alalaidassa ja vasemmassa ylälaidassa kertovat pelaajien elämäpistetilanteen. Punaiset numerot näyttävät pelaajan oman elämäpistemäärän vastustajalle. Harmaalla taustalla keskellä oikeassa reunassa olevat numerot vastaavat voitettuja eriä. Miekan, tikareiden ja sauvan kuvaa

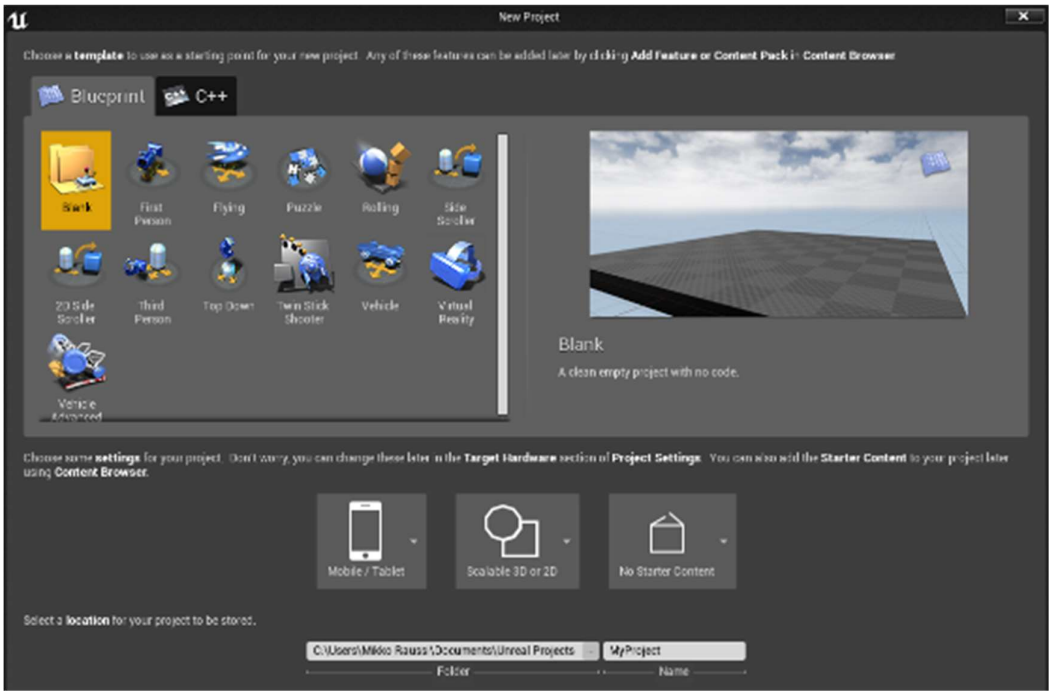
painamalla valitaan kuvaketta vastaava hahmo. Hahmo voidaan sijoittaa kentälle painamalla mitä tahansa linjaa, kun hahmo on valittuna.

3 PELIN KEHITYS

Pelin kehitys alkoi uuden projektin luomisella Unreal Enginessä. Tämän jälkeen aloitettiin ohjelmoimaan pelin eri ominaisuuksia blueprinteillä. Tärkeimpien ominaisuuksien toteuttamista kuvataan tarkemmin alaluvuissa. Ominaisuuksien kuvaukset eivät välttämättä sisällä kokonaista selitystä siitä, miten ne on toteutettu, koska peli saatetaan julkaista myöhempänä ajankohtana.

3.1 Uuden projektin luominen

Uutta projektia luodessa voidaan valita asetuksia projektille. Tässä projektissa valittiin alustaksi tyhjä blueprint-tyyppinen projekti, jolloin projektissa ei ollut luonnin jälkeen yhtään valmista koodia. Grafiikan tasoksi valittiin skaalautuva 3D ja 2D. Kohdelaitteistona projektissa olivat mobiili ja tabletti. Projekti luotiin käyttämättä ylimääräistä aloittelijoille suunnattua sisältöä, koska sille ei ollut tarvetta.



Kuva 8. Uuden projektin asetukset

Kuvasta 8 nähdään, miltä asetukset näyttävät Unreal Enginessä. Kaikkia kuvassa esitettyjä asetuksia voidaan vaihtaa myöhemmin. Lopuksi valitaan kansio, mihin projekti tallennetaan, sekä nimi projektille.

3.2 Androidin pohjustus

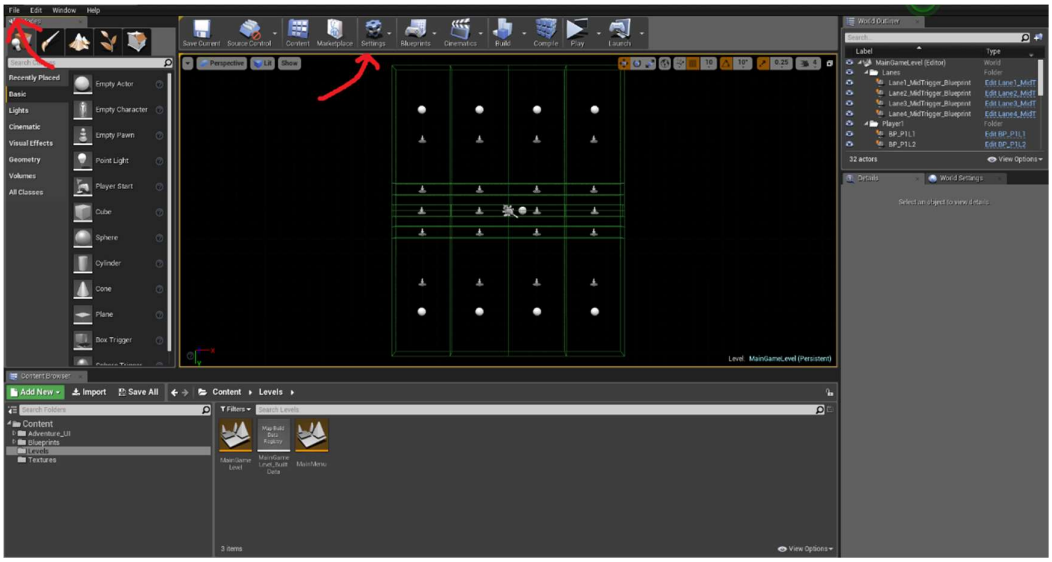
Mobiilipelin kehitys Unreal Enginessä vaatii Android SDK:n asennuksen. Android SDK on kokoelma työkaluja, joita vaaditaan sovelluksen pyörittämiseen Android-laitteilla (Sinicki 2017). Unreal Engine 4 käyttää omaa versiota Android SDK:sta nimeltä CodeWorks for Android 1R6u1, joka on helpointa ottaa käyttöön Unreal Enginessä Android-projektien kehitystä varten (Android Quick Start 2018).

CodeWorks for Android 1R6u1:n asennustiedosto tulee Unreal Enginen mukana, ja se asennettiin tätä projektia varten. Asennus vaatii Visual Studion, jonka tuettuja versioita ovat 2013, 2015 ja 2017. Asennuksen jälkeen Unreal Engine on valmis käytettäväksi Android-projektien kehittämiseen.

Pelin kehityksen ja testaamisen helpottamiseksi pohjustettiin Android-laite toimimaan Unreal Enginen kanssa. Tässä tapauksessa Android-laitteena käytettiin Samsung Galaxy S4 -puhelinta. Ensin puhelin kytkettiin kiinni tietokoneeseen USB-portin kautta. Puhelimen ajuriohjelmisto asentui automaattisesti tietokoneelle. Tämän jälkeen laitettiin päälle sovelluskehittäjien asetukset puhelimessa. Asetuksista aktivoitiin USB-virheenkorjaus, jonka jälkeen puhelin oli valmiina käytettäväksi pelin kehityksen yhteydessä.

3.3 Projektin asetukset

Unreal Enginessä projektia voidaan muokata useiden asetusten avulla. Tässä työssä tärkeimpiä asetuksia olivat mobiililaitteisin liittyvät asetukset, jotka vaikuttavat projektin toimivuuteen mobiililaitteilla. Tämän luvun tarkoituksena on kuvailla kyseisiä asetuksia. Kuva 9 näyttää, mistä projektin asetukset löytyvät.



Kuva 9. Projektin asetukset

Projektin asetuksiin pääsee joko painamalla ”File” vasemmasta ylänurkasta ja valitsemalla ”Project settings” tai painamalla rattaan kuvaa työkalurivissä.

3.3.1 Use Mouse for Touch

Asetuksista laitettiin päälle ”Use Mouse for Touch”, mikä mahdollisti hiiren käytön kosketustapahtumien kanssa.

Kuva 10. Use Mouse for Touch

Kuvan 10 asetus on projektin asetuksissa kohdassa “Engine” ja alavalikossa “Input”. Ohjelmoinnissa kaikki vuorovaikutus laitteen näytön kanssa toimii kosketustapahtumien kautta. Esimerkiksi tässä pelissä hahmon valinta toimii kosketustapahtumilla. Kuvassa esitetyn asetuksen päälle laittamisen jälkeen hiirtä voitiin käyttää kosketustapahtumien kanssa, jolloin peliä pystyttiin pelata ja testata tietokoneella.

3.3.2 Default Viewport Mouse Lock Mode

Kun peli käynnistetään tietokoneella, se avautuu uuteen ikkunaan. Uudessa ikkunassa peli ei oletuksena anna hiiren liikkua peli-ikkunan ulkopuolelle, mikä vaikutti negatiivisesti pelin nopeaan testaukseen. Jotta hiirtä saatiin liikutettua

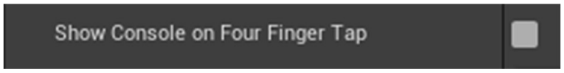
vapaasti, vaihdettiin “Default Viewport Mouse Lock Mode” -asetusta kuvan 11 mukaisesti.

Kuva 11. Default Viewport Mouse Lock Mode

Yllä kuvattu asetus on samassa paikassa, kuin edellisessä kappaleessa mainittu ”Use Mouse for Touch” -asetus. Hiiren vapaalla liikuttamisella mahdollistettiin pelin nopea sulkeminen, sekä peliobjektien tarkastelu muokkausohjelmassa pelaamisen aikana.

3.3.3 Show Console on Four Finger Tap

Kahden pelaajan pelissä voi tulla useita samanaikaisia näytön kosketuksia. Neljä kosketusta saman aikaisesti avaa konsolin, jota ei haluttu näkyvän pelissä. Konsolin avaamista neljällä kosketuksella hallitaan ”Show Console on Four Finger Tap” -asetuksella, joka laitettiin pois päältä.



Kuva 12. Show Console on Four Finger Tap

Ylläolevan kuvan asetus löytyy samalta sivulta kuin aikaisemmin mainittu ”Use Mouse for Touch” -asetus.

3.3.4 Orientation

Mobiililaitteilla sovellusten käyttöliittymän ulkonäkö voi vaihtua puhelimen asennosta riippuen. Tässä projektissa haluttiin pelin näyttöliittymän pysyvän pystysuorassa koko ajan. Sovelluksen orientaatiota hallitseva asetus löytyy kohdasta “Platform” ja alavalikosta “Android”.

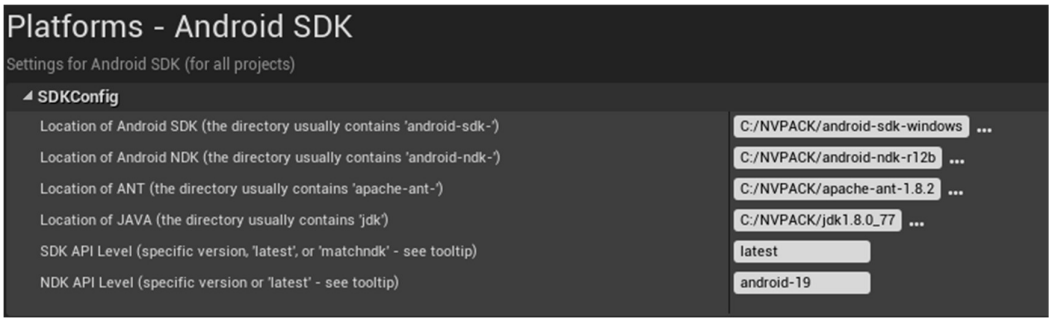
Kuva 13. Orientation

Kuvasta 5 nähdään, että asetuksessa on useampia vaihtoehtoja. Sovelluksen orientaation voi lukita myös vaakasuoraan tai antaa muuttua puhelimen asen-

nosta riippuen. Tähän projektiin valittiin asetukseksi “Portrait”, joka lukitsee sovelluksen orientaation pystysuoraan, jotta peli varmasti näytti siltä, kuin haluttiin.

3.3.5 Android SDK

Luvussa 3.2 kuvattiin, kuinka Android SDK asennetaan käytettäväksi Android-pelien kehityksessä. Pelkkä asennus ei kuitenkaan riitä, vaan asennuksen jälkeen pitää projektien asetuksissa määrittää, mistä Android SDK ja muut vaadittavat komponentit löytyvät tietokoneelta. Alla olevassa esimerkissä komponenttien sijainnit ovat määritelty, kun Android SDK on asennettu asennuspaketin ehdottamaan sijaintiin, eli kansioon C:/NVPACK. Kuvan 14 mukaiset Android SDK:n asetukset löytyvät projektin asetuksista kohdasta “Platform” ja alavalikosta “Android SDK”.



Kuva 14. Android SDK-asetukset

Kuvasta 14 voidaan huomata, että Android SDK:n lisäksi tarvitaan myös muita työkaluja Android projektin toteuttamiseen. Android NDK on kokoelma työkaluja, joka mahdollistaa C- ja C++-koodin käytön Androidin kanssa ja tarjoaa kirjastoja, joiden avulla voidaan hallita laitteen fyysisiä komponentteja, kuten sensoreita ja kosketuksen tunnistinta (Getting Started with the NDK s.a).

3.4 Muuttujien tietokanta

Pelissä on monia eri muuttujia, jotka vaikuttavat pelin toimintaan, kuten pelaajan elämä- ja pistetilanne, hahmojen syntymispisteet sekä keskilinjaa hallitseva pelaaja. Nämä eri muuttujat koottiin niin sanottuun muuttujien tietokantaan. Tietokanta on tyhjä actor -tyyppinen blueprint, joka ei sisällä mitään pelin toiminnallisuuksia, vaan pelkästään muuttujia.

Jotkin blueprintit viittaavat toisen blueprintin muuttujiin. Tietokanta auttaa pitämään ohjelmoinnissa käytetyt viittaukset siistinä, koska kaikki viitatut muuttujat löytyvät samasta sijainnista. Tietokannassa olevat muuttujat eivät myöskään häviä, koska blueprintiä ei tuhota missään vaiheessa, toisin kuin hahmojen blueprint keskilinjalla käydyn tappelun jälkeen. Tämän ansiosta tietokannassa voidaan säilyttää pelaajien elämäpisteisiin ja voittoihin liittyvää tietoa.

3.5 Pelihahmojen toteutus

Pelihahmojen ohjelmointi toteutettiin blueprinteillä. Hahmot ovat pawn-luokan blueprintejä. Pelihahmojen toteutusta kuvataan tarkemmin tämän luvun alakappaleissa.

3.5.1 Hahmon syntyminen pelikentälle

Pelissä vaadittiin, että hahmo syntyy, kun painetaan jotakin pelaajan neljästä linjasta pelihahmo valittuna. Hahmo valitaan painamalla sitä vastaavaa kuvaketta. Tässä luvussa käydään läpi, miten hahmon syntyminen pelikentälle toteutettiin.

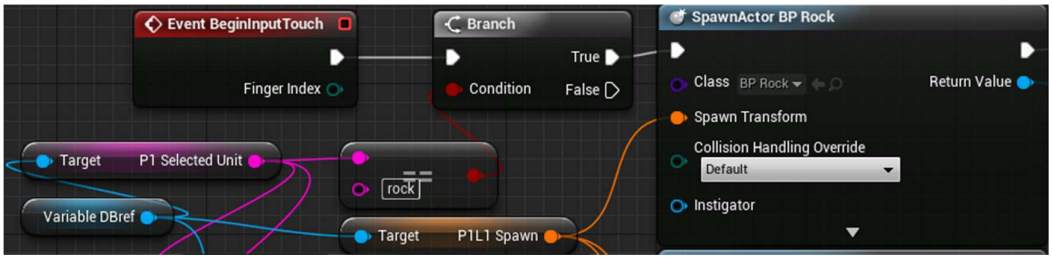
Ensiksi piti saada selville, minkä hahmon kuvaketta painettiin. Tämä toteutettiin OnClicked -tapahtumalla pelin käyttöliittymän blueprintissä. OnClicked aktivoituu aina, kun siinä määriteltyä objektia painetaan. Kuva 15 havainnollistaa, miten pelaaja yhden kiven valinta toimii.

Kuva 15. Valittu hahmo

Kuvan 15 mukaisesti OnClicked-tapahtuma linkitetään Set-noodiin, jonka tarkoituksena on asettaa jokin arvo haluttuun muuttujaan. Set-noodi vaatii muuttujan sijainnin. Tässä tapauksessa pelaajan valitseman yksikön muuttuja sijaitsi muuttujien tietokannassa (Variable DB), joten luotiin tietokannasta viittaus (Variable DBref) ja laitettiin se kohteeksi Set-noodissa. Kuvan mukaisesti

asetetaan pelaaja yhden valitsemaksi hahmoksi ”rock”, eli kivi, koska sen kuvaketta painettiin. Paperin ja saksen valitseminen toimii samalla periaatteella. Pelaaja kahden valittu hahmo tallennetaan omaan muuttujaan, jotta pelaajat pystyvät valitsemaan oman halutun hahmon ja laittamaan sen kentälle haluamalleen linjalle.

Kun hahmo on valittu ja painetaan jotain linjaa, ajetaan funktio, joka luo hahmon pelimaailmaan. Alla oleva kuva havainnollistaa, miten kiven, eli pelissä sotilaan, luominen pelimaailmaan toimii. Funktio toimii käytännössä samalla tavalla varas- ja velhohahmon luomisessa.



Kuva 16. Osa syntymisfunktiosta

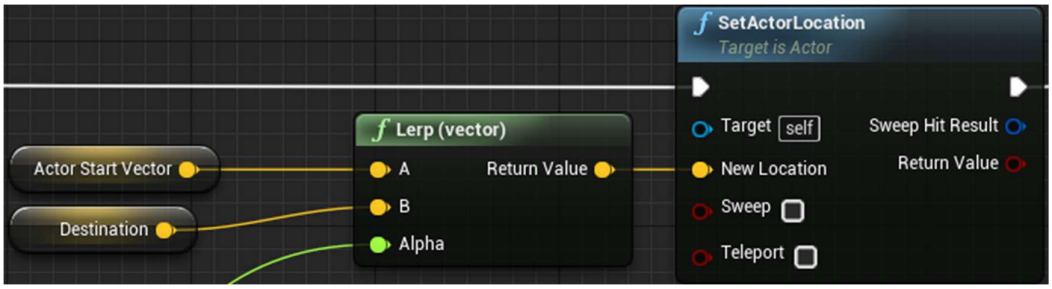
Funktio alkaa BeginInputTouch-tapahtumasta, joka käynnistyy, kun tapahtuman omaavaa blueprintiä aletaan koskettaa. Tämä funktio on kiinni pelaaja yhden ensimmäisen linjan blueprintissä, joten se suoritetaan aina, kun linjaa painetaan. Jokaisella linjan blueprintillä on oma hahmon luomisfunktio, mutta ne toimivat käytännössä samalla tavalla. Seuraavaksi tarkistetaan branch-noodin avulla, minkä yksikön pelaaja on valinnut. Branch tarkistaa sille syötetyn ehdon, ja jos se on tosi tai epätosi, suoritetaan seuraava osa koodista sen mukaisesti. Tässä tapauksessa ehdon ollessa tosi suoritetaan SpawnActor-funktio, joka luo hahmon kentälle.

SpawnActor vaatii luokan (class), jonka mukaan objekti luodaan sekä sijainnin (spawn transform), mihin se luodaan. Kuva 16 havainnollistaa SpawnActor-funktion toimintaa kivihahmon kanssa. Luokaksi on valittu BP_Rock, eli kiven blueprint ja sijainniksi pelaaja yhden ensimmäisen linjan syntymispiste, jonka sijainti on muuttujien tietokannassa. Syntymispisteet käydään läpi luvussa 3.6.2.

3.5.2 Hahmon liikkuminen

Tässä luvussa kuvaillaan hahmon liikkumiseen käytettyjä funktioita. Hahmon liikkuminen toteutettiin hahmon omassa tapahtumakaaviossa. Kaikki pelin hahmot liikkuvat samalla tavalla pelimaailmassa.

Itse liikkuminen saatiin aikaiseksi SetActorLocation-funktiolla. Funktio vaatii uuden sijainnin määrittämisen hahmolle. Kuva 17 hahmottaa pelihahmon liikkumisfunktiota pelissä.

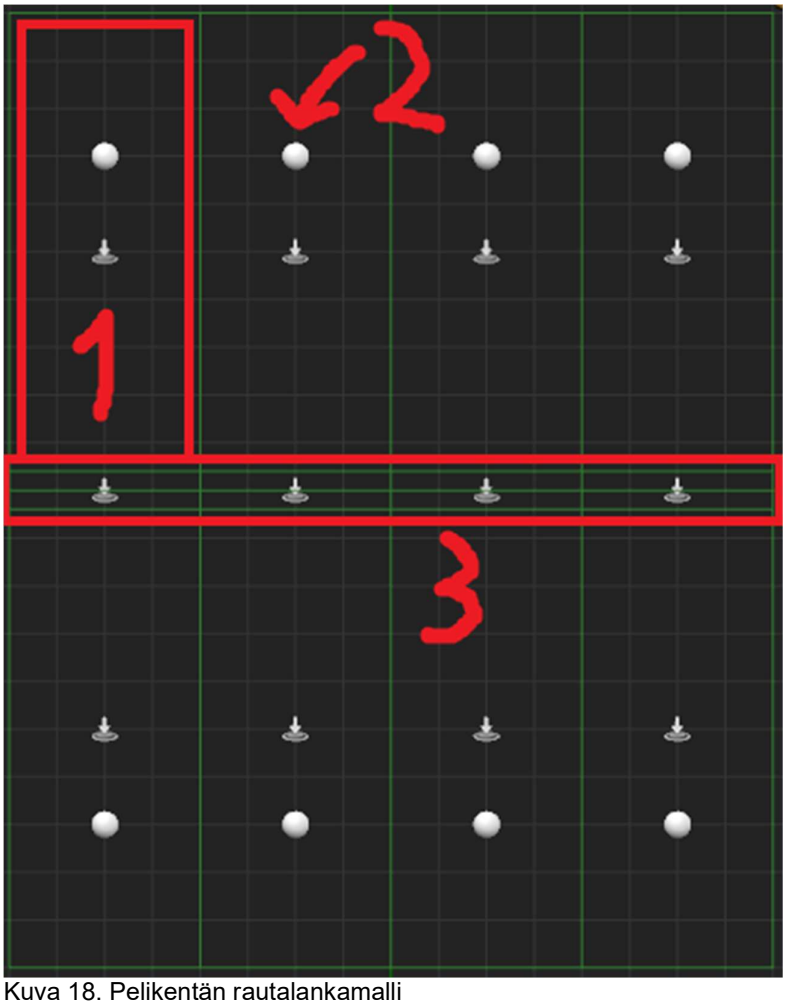


Kuva 17. Pelaajan liikkuminen

Uuden sijainnin määrittämiseen käytettiin Lerp (vector)-funktiota. Funktio vaatii kolme arvoa, A:n, B:n sekä alfan. A ja B ovat pisteitä pelimaailmassa. A:n arvoksi laitettiin hahmon aloitussijainti, eli sijainti mihin se luotiin ja B:n arvoksi määränpää, joka tässä tapauksessa on keskellä kenttää missä hahmojen tappelu tapahtuu. Alfan arvo kuvastaa pistettä A:n ja B:n välillä. Esimerkiksi alfan ollessa 0, palauttaisi funktio A:n sijainnin. Arvolla 0,5 saataisiin sijainti A:n ja B:n keskivälistä. Jotta pelihahmo saatiin liikkumaan A:sta B:hen, annetiin alfalle arvoksi luku, joka nousee jatkuvasti. Luku alkaa nollasta, eli hahmon aloitussijainnista ja luvun maksimiksi laitettiin 1, joten hahmo ei liikkunut päämääränsä ohi.

3.6 Pelikenttä

Pelissä on yksi pelikenttä, mihin koko peli sijoittuu. Kenttä ei muutu missään vaiheessa, vaan se pysyy koko ajan samanlaisena. Hahmot liikkuvat kentällä kohti keskilinjaa, jossa ne taistelevat toisiaan vastaan.



Kuva 18. Pelikentän rautalankamalli

Kuva 18 auttaa hahmottamaan pelikentän toteutusta. Kuvassa on numeroitu pelikentän muodostavat osat. Numero yksi kuvaa yhtä linjaa, jolle pelihahmon voi asettaa. Hahmo luodaan numero kahden mukaiseen syntymispisteeseen linjan päässä, josta se lähtee kulkemaan kohti keskilinjaa, joka on numeroitu numerolla kolme. Pelikentän eri osia kuvataan tarkemmin alakappaleissa.

3.6.1 Triggerbox

Laukaisimet (triggers) ovat actor-luokan olioita, joita käytetään tapahtumien aloittamiseen, kun joku toinen olio vuorovaikuttaa niihin. Toisin sanoen niitä käytetään tapahtumien käynnistämiseen vastauksena johonkin toiseen toimintoon pelikentässä. Kaikki oletuslaukaisimet ovat käytännössä samanlaisia ja ne eroavat vain vaikutusalueen muodossa. (Trigger Actors 2018.)

Tässä projektissa käytettiin laatikon (box) muotoisia laukaisimia pelikentän linjojen toteutukseen, koska ne sopivat malliltaan parhaiten haluttuun

muotoon. Edellisessä luvussa kuvailtiin, että laukaisimet vastaavat eri toimintoihin. Kuvassa 18 yhdellä numeroidun kaltaiset linjat vastaavat, kun pelaaja koskettaa niitä sormella laitteen näytössä. Numerolla kolme merkitty keskilinja vastaa, kun pelihahmot koskettavat sitä.

3.6.2 Syntymispisteet

Pelikentässä on 8 syntymispistettä, joihin pelihahmot voidaan luoda. Nämä pisteet sijaitsevat pelikentän linjojen päässä. Pelihahmot lähtevät liikkeelle syntymispisteistä.

Syntymispisteet ovat pelissä näkymättömiä actor-luokan blueprintejä. Kuva 19 näyttää syntymispisteen tapahtumakaavion sisällön.

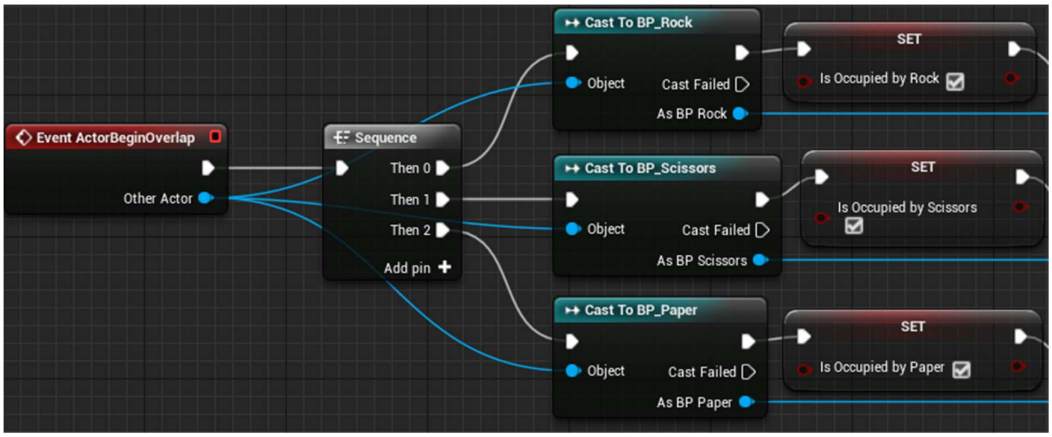
Kuva 19. Syntymispisteen tapahtumakaavio

Syntymispisteiden haluttiin tallentavan oma sijaintinsa muuttujien tietokantaan, mistä kuvan 16 luomisfunktio voi hakea sen pelaajan luomista varten. Tämä toteutettiin käyttämällä BeginPlay-tapahtumaa, joka aktivoituu pelin alkaessa. Tapahtuma linkitettiin Set-funktioon, mikä asettaa arvon muuttujalle. Muuttujan arvoksi haluttiin oma sijainti, mikä saatiin käyttämällä GetActorTransform-funktiota. Tämä arvo sitten tallennettiin muuttujien tietokantaan myöhempää käyttöä varten.

3.6.3 Keskilinja

Kuten kuvasta 18 voidaan huomata, keskilinja sijaitsee pelikentän keskellä. Hahmot liikkuvat linjallaan kohti keskustaa, missä ne joko jäävät odottamaan vastustajaa, tai tappelevat keskilinjalla olevan vastustajan kanssa. Tappelun jälkeen hahmot poistetaan pelistä ja päivitetään pistetilanne tappelun tuloksen mukaan.

Keskilinja on pelihahmojen liikkumisessa käyttämien linjojen kaltaisesti laukaisinlaatikko-tyyppinen (triggerbox) blueprint. Sen tarkoituksena on tarkastella, kumman pelaajan hahmo sitä hallitsee sekä toimia hahmojen taistelukenttänä. Esimerkiksi jos keskilinjaa hallitsee pelaaja yhden kivi ja pelaaja kaksi lähettää paperin sitä vastaan, niin paperin saapuessa keskilinjalle hahmot poistetaan tappelun jälkeen ja vähennetään piste pelaaja yhdeltä, koska paperi voittaa kiven.



Kuva 20. Keskilinjan hallinnan tarkistus

Kuva 20 on osa keskilinjan blueprintin tapahtumakaaviosta. Kuvan mukaisesti funktio alkaa ActorBeginOverlap-tapahtumasta, mikä aktivoituu, kun toinen olio alkaa koskettaa sitä. Tässä tapauksessa tarkistetaan, mikä kolmesta pelihahmosta (kivi, paperi, sakset) osui keskilinjaan, ja laitetaan se keskilinjan omistajaksi. Tämän jälkeen tarkastetaan, kumpi pelaaja omistaa linjaa hallitsevan hahmon. Tätä tietoa tarvitaan, kun hahmot tappelevat toisiaan vastaan, jotta osataan vähentää elämäpiste oikealta pelaajalta.

4 JULKAISU

Peliä ei ehditty julkaisemaan, mutta tässä työssä tutkittiin, minkälaisia vaatimuksia pelin julkaisemiselle on Unreal Enginessä ja Androidissa. Luvun alaluvut kuvaavat tarkemmin julkaisuun liittyviä vaatimuksia ja käytäntöjä.

4.1 Unreal Enginen säännöt

Kun peli on lähes valmis julkaisuun, niin täytetään UE4-projektin julkaisulomake. Lomakkeeseen täytetään tietoja projektista, kuten tuotteen nimi, pelinkehittäjän nimi sekä osoite, mahdollinen julkaisija, kontaktihenkilö jne. Myös

pelin julkaisualusta, arvioitu julkaisupäivä ja pelin hinta täytetään lomakkeeseen. Lomakkeen tarkoituksena on antaa pelinkehittäjän ja tuotteen tietoja Epic Gamesille (Unreal Engine 4 Commercial Game Deployment Guidelines 2018). Lomake on saatavilla Unreal Enginen sivuilta.

Julkaisulomakkeen lisäksi pelille tulee kirjoittaa EULA eli loppukäyttäjän lisenssisopimus, mikä selkeästi irtisanoo kaikki edustukset, vakuudet, ehdot ja vastuut Unreal Engineen liittyen. Kaikki lisenssisopimukset kolmansien osapuolien kanssa tulee käydä läpi, ja tarjota käyttäjille ilmoitus niistä. Pelin lopputeksteihin lisätään seuraava: “[Tuotteen nimi] uses the Unreal® Engine. Unreal® is a trademark or registered trademark of Epic Games, Inc. in the United States of America and elsewhere.” and “Unreal® Engine, Copyright 1998 – xxxx, Epic Games, Inc. All rights reserved.” Hakasuluissa oleva ”Tuotteen nimi” korvataan pelin nimellä, sekä ”xxxx” pelin julkaisuvuodella. (Unreal Engine 4 Commercial Game Deployment Guidelines 2018.)

Jos peli käyttää Unrealiin liittyviä tuotemerkkejä, hankitaan lisenssi Unrealin brändäys opas ja tuotemerkin käyttö -sivustolta ja toimitaan käyttöön liittyvien ohjeiden ja sääntöjen mukaisesti. Ennen julkaisua tulee myös varmistaa, että julkaistava peli noudattaa kaikkia asiaankuuluvia lakeja, lainsäädäntöjä sekä säännöksiä. Tuote ei saa myöskään yhdistää Unreal Enginen lähdekoodia minkään koodin kanssa, jota suojaa ”Copyleft” lisenssisopimus, mikä suoraan tai epäsuoraan vaatisi Unreal Enginen toimivan muiden kuin Unreal Enginen oman lisenssisopimusten mukaisesti. Lopuksi tuote ei saa sisältää Unreal Engine muokkausohjelmaa tai työkaluja, eikä muokkausohjelmia tai työkaluja, jotka pohjautuvat niihin. (Unreal Engine 4 Commercial Game Deployment Guidelines.)

Kun pelistä aletaan keräämään rahaa, tulee pelin bruttotuloja seurata ja maksaa 5 % rojalti summasta ensimmäisen 3000 dollarin jälkeen per peli per neljännesvuosi. Ansiot ilmoitetaan rojaltiraporttilomakkeen avulla. (Frequently Asked Questions (FAQ) 2018.)

4.2 Julkaisu Androidille

Tässä luvussa käydään läpi eri toimenpiteitä, joita julkaisu Android-alustalle vaatii. Toimenpiteiden kuvaus on rajattu projektin kannalta tärkeimpiin toimenpiteisiin.

Ensiksi tulee ymmärtää kehittäjien ohjelmasäännöt, jotka löytyvät Googlen sivuilta. Ohjelmasäännöt ovat suunniteltu pitämään Play Kauppa luotettuna resurssina Androidin käyttäjälle. Sääntöjen rikkomisella on seurauksia, joten säännöt tulee käydä läpi tarkasti. (Launch Checklist s.a.)

Julkaisua varten tulee luoda Google Play -kehittäjätili. Tilin rekisteröinti maksaa 25 dollaria (Play Consolen käyttö 2018). Jos aiotaan myydä sovelluksia, tulee kehittäjätili linkittää maksuprofiiliin. Google Play -kehittäjätili mahdollistaa Play Console -sovelluksen käytön. Play Console antaa tietoja pelistä, kuten käyttäjäpalautteita, tietoja käyttäjien ostotoiminnasta jne. (Google Play -kehittäjätilin linkittäminen maksuprofiiliisi 2018.)

4.2.1 Sovelluksen valmistelu

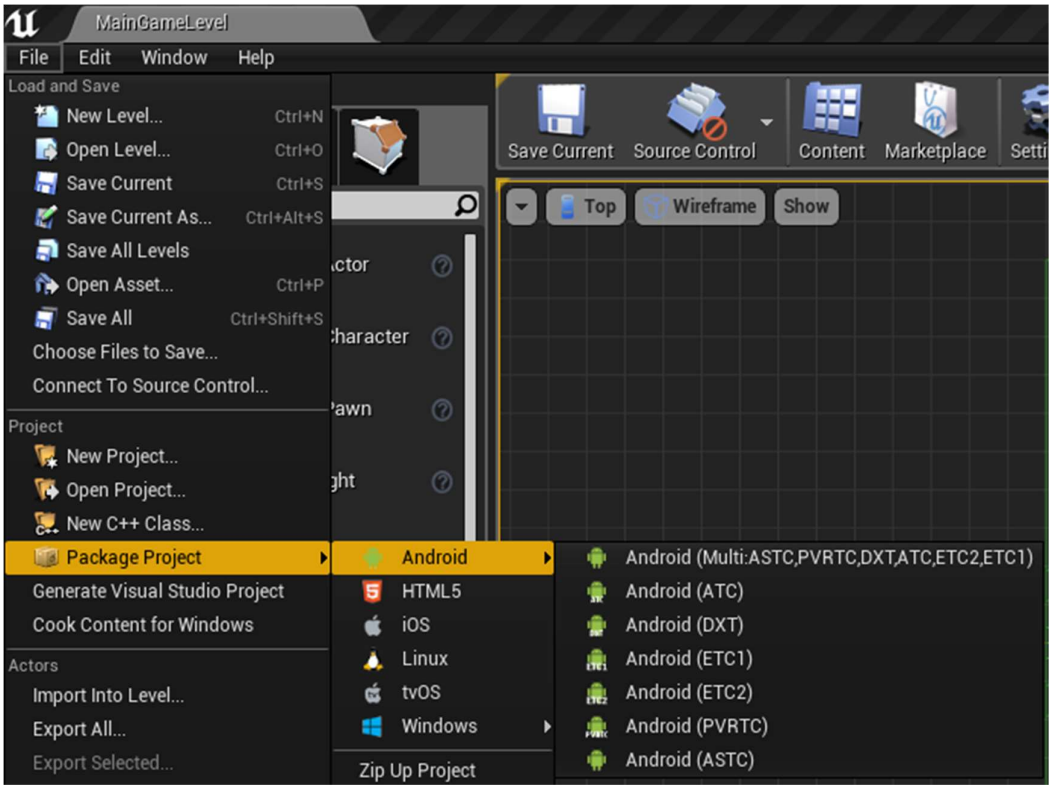
Android järjestelmä vaatii, että jokainen asennettu sovellus on digitaalisesti allekirjoitettu sertifikaatilla, minkä omistaa sovelluksen kehittäjä, eli sertifikaatilla johon kehittäjällä on olemassa yksityinen avain. Android-järjestelmä käyttää sertifikaattia tunnistamaan sovelluksen tekijän. Avaimen voi luoda käyttämällä Android Studio -ohjelmaa. (Prepare for Release s.a.)

Sovelluksella tulee olla ikoni, joka täyttää suositellut ikonin vaatimukset. Sovelluksen ikoni auttaa käyttäjiä tunnistamaan sovelluksen laitteen kotiruudussa. Se näkyy myös muissa sijainneissa, kuten sovellusten hallinnassa ja latauksissa. Lisäksi sovelluksen ikoni näkyy myös julkaisupalveluissa kuten Googlen Play Kaupassa. (Prepare for Release s.a.)

Loppukäyttäjän lisenssisopimuksen (EULA) lisääminen sovellukseen ei ole pakollista. Se voi kuitenkin auttaa suojaamaan immateriaalioikeuksia. Sovellusta varten voidaan myös valmistella mainos- ja markkinointimateriaalia. Jos sovellus aiotaan julkaista Googlen Play Kaupassa, tarvitaan mainosteksti sekä kuvankaappauksia sovelluksesta. (Prepare for Release s.a.)

4.2.2 APK:n rakennus

APK on Androidin käyttämä pakkaus tiedostoformaatti, jolla asennetaan pelejä ja sovelluksia laitteelle (Montegriffi 2018). Kun peli on valmis julkaistavaksi käyttäjille tai testattavaksi laitteella, rakennetaan pelistä APK-paketti. Tässä projektissa paketti rakennettiin Unreal Enginellä.



Kuva 21. APK:n rakennus

APK-paketti rakennetaan kuvan 21 mukaisesti painamalla ensin ”File” muokkausohjelman vasemmasta yläreunasta ja valitsemalla ”Package Project”. Näkyviin tulee lista eri alustoista, jolle pelin voi rakentaa. Listasta valitaan Android, jolloin ilmestyy lista eri tekstuuriformaateista, mitä pelin halutaan käyttävän. Listasta valittiin tätä projektia varten ”Android (Multi:ASTC,PVRTC,DXT,ATC,ETC2,ETC1)”, koska se varmistaa, että peli toimii kaikilla laitteilla.

5 YHTEENVETO

Peliä ei keretty saamaan täysin valmiiksi, joten sitä ei pystytty julkaisemaan, kuten tarkoituksena oli. Tähän vaikutti työn määrän aliarviointi aikatauluun verrattuna. Unreal Engine ei ollut minulle entuudestaan tuttu, joten opiskelin sen käyttöä työnteon ohessa, mikä taas hidasti pelinkehitystä. Projektiin ei myöskään saatu graafikkoa tarpeeksi ajoissa, joten grafiikat jäivät puuttumaan muualta kuin käyttöliittymästä, jota varten käytettiin jo olemassa olevia grafiikoita. Kuuran henkilöstöllä oli myös pelin kehityksen alkuvaiheessa useita koulutuspäiviä, joiden takia pelin kehitys keskeytyi. Kun projektin vaatimukset olivat paremmin tiedossa, pystyttiin peliä myös kehittämään kotona henkilöstön ollessa poissa.

Suurin ongelma pelinkehityksen kannalta oli pelihahmojen liikkumisen toteutus. Oikeiden funktioiden löytämiseen kului paljon aikaa ja siinä välissä oli myös väärinkäsityksiä siitä, miten hahmojen liikkumisen tulisi toimia. Lopuksi hahmot kuitenkin saatiin liikkumaan, vaikka liikkumisen ohjelmallinen toteutus jäi enintäänkin tyydyttäväksi. Projektin aikana ilmeni muita pienempiä ongelmia, jotka ratkesivat selaamalla Unreal Enginen dokumentaatiota ja erilaisia keskustelupalstoja, joissa muut käyttäjät etsivät ratkaisua samoihin ongelmiin.

Opin projektista todella paljon. Aikaisempaa kokemusta Unreal Enginen käytöstä oli vain yhden koulukurssin verran, joten tilaakin oppimiselle oli. Mielenkiinto työn aihetta kohtaan sekä hyvä työilmapiiri vaikuttivat positiivisesti myös haluun oppia uutta. Pelin julkaisuun liittyviä asioita olisi ollut mielenkiintoista kokea käytännössä, ja siitä olisi varmasti oppinut paljon. Otin kuitenkin selvää pelin julkaisuun vaadituista asioista tätä työtä varten ja opin pääpiirteittäin mobiilipelin julkaisuprosessin.

Opintojen myötä on tullut kehiteltyä pieniä pelejä, mutta ei yhtäkään mobiililaitteille. Projektin ansiosta pääsin tutustumaan mobiilipelin kehitykseen, joka on nykyaikana tärkeää peliohjelmoinnissa mobiilipelien suosion takia. En ole myöskään tehnyt aiemmin töitä peliyrityksessä, joten oli mielenkiintoista päästä tutustumaan peliyrityksen arkeen ja oppia hieman, mitä peliyrityksen sisällä tapahtuu. Tulevaisuuden kannalta työstä saatu oppi oli siis hyödyllistä,

koska se voi vaikuttaa työllistymiseen alalla, joka kiinnostaa minua kaikista eniten.

LÄHTEET

Actors. 2018. Epic Games. WWW-dokumentti. Saatavissa: https://docs.unrealengine.com/en-us/Programming/UnrealArchitecture/Actors [viitattu 17.4.2018].

Android Quick Start. 2018. Epic Games. WWW-dokumentti. Saatavissa: https://docs-origin.unrealengine.com/latest/INT/Platforms/Android/GettingStarted/index.html [viitattu 17.4.2018].

Character. 2018. Epic Games. WWW-dokumentti. Saatavissa: https://docs.unrealengine.com/en-US/Gameplay/Framework/Pawn/Character [viitattu 7.4.2018].

Components. 2018. Epic Games. WWW-dokumentti. Saatavissa: https://docs.unrealengine.com/en-us/Programming/UnrealArchitecture/Actors/Components [viitattu 9.4.2018].

Frequently Asked Questions (FAQ). 2018. Epic Games. WWW-dokumentti. Saatavissa: https://www.unrealengine.com/en-US/faq [viitattu 13.4.2018].

Game Mode and Game State. 2018. Epic Games. WWW-dokumentti. Saatavissa: https://docs.unrealengine.com/en-us/Gameplay/Framework/GameMode [viitattu 9.4.2018].

Getting Started with the NDK s.a. Google. WWW-dokumentti. Saatavissa: https://developer.android.com/ndk/guides/index.html [viitattu 3.4.2018].

Google Play -kehittäjätilin linkittäminen maksuprofiiliisi. 2018. Google. WWW-dokumentti. Saatavissa: https://support.google.com/googleplay/android-developer/answer/3092739 [viitattu 17.4.2018].

Kuura Playhouse Oy. 2018. WWW-dokumentti. Saatavissa: https://www.kuuragames.com/ [viitattu 22.2.2018].

Launch Checklist s.a. Google. WWW-dokumentti. Saatavissa: https://developer.android.com/distribute/best-practices/launch/launch-checklist.html [viitattu 17.4.2018].

Montegriffi, N. 2018. What is an APK file and how do you install one? WWW-dokumentti. Saatavissa: https://www.androidpit.com/android-for-beginners-what-is-an-apk-file [viitattu 17.4.2018].

Pawn. 2018. Epic Games. WWW-dokumentti. Saatavissa: https://docs.unrealengine.com/en-us/Gameplay/Framework/Pawn [viitattu 7.4.2018].

PlayerController. 2018. Epic Games. WWW-dokumentti. Saatavissa: https://docs.unrealengine.com/en-US/Gameplay/Framework/Controller/PlayerController [viitattu 9.4.2018].

Play Consolen käyttö. 2018. Google. WWW-dokumentti. Saatavissa: https://support.google.com/googleplay/android-developer/answer/6112435 [viitattu 17.4.2018].

Prepare for Release s.a. Google. WWW-dokumentti. Saatavissa: https://developer.android.com/studio/publish/preparing.html [viitattu 17.4.2018].

Sinicki, A. 2017. Android SDK tutorial for beginners. WWW-dokumentti. Saatavissa: https://www.androidauthority.com/android-sdk-tutorial-beginners-634376/ [viitattu 7.3.2018].

Trigger Actors. 2018. Epic Games. WWW-dokumentti. Saatavissa: https://docs.unrealengine.com/en-us/Engine/Actors/Triggers [viitattu 16.4.2018].

Unreal Engine Features. 2018. Epic Games. WWW-dokumentti. https://www.unrealengine.com/en-US/features [viitattu 2.4.2018].

Unreal Engine 4 Commercial Game Deployment Guidelines. 2018. Epic Games. WWW-dokumentti. Saatavissa: https://www.unrealengine.com/en-US/release [viitattu 13.4.2018].

KUVALUETTELO

Kuva 1. Uusi blueprint.
Kuva 2. Blueprintin luokkia.
Kuva 3. Uuden blueprintin tallennus.
Kuva 4. Blueprintin oletusnäkymä.
Kuva 5. Rakennuskoodi.
Kuva 6. Tapahtumakaavio.
Kuva 7. Pelinäkymä
Kuva 8. Uuden projektin asetukset.
Kuva 9. Projektin asetukset.
Kuva 10. Use Mouse for Touch.
Kuva 11. Default Viewport Mouse Lock Mode.
Kuva 12. Show Console on Four Finger Tap.
Kuva 13. Orientation.
Kuva 14. Android SDK-asetukset.
Kuva 15. Valittu hahmo.
Kuva 16. Osa syntymisfunktiosta.
Kuva 17. Pelaajan liikkuminen.
Kuva 18. Pelikentän rautalankamalli.
Kuva 19. Syntymispisteen tapahtumakaavio.
Kuva 20. Keskilinjan hallinnan tarkistus.
Kuva 21. APK:n rakennus.