Pelimoottorin valinta yritykselle Ghoulish Games

Jere Lämsä

2021 Laurea

Laurea-ammattikorkeakoulu

Pelimoottorin valinta yritykselle Ghoulish Games

Jere Lämsä

Tradenomi, Tietojenkäsittely

Opinnäytetyö

12, 2021

Laurea-ammattikorkeakoulu **Tiivistelmä**

Tietojenkäsittely

Tradenomi (AMK)

Jere Lämsä

Pelimoottorin valinta yritykselle Ghoulish Games

Vuosi 2021 Sivumäärä 48

Opinnäytetyö toteutettiin toimeksiantona opinnäytetyön toteuttajan toiminimelle, Ghoulish Games:lle. Tavoitteena oli löytää yrityksen tarpeisiin sopiva pelimoottori, yrityksen seuraavaa peliprojektia varten.

Opinnäytetyö alkoi ensiksi listaamalla ylös ne tarpeet, joita yrityksellä pelimoottorin suhteen oli. Tarpeista luotiin lista ja myöhemmin taulukko. Seuraavaksi valittiin yhdeksän pelimoottoria alkukarsintoihin. Valinta tehtiin pelimoottoreista kirjoitettujen artikkeleiden perusteella. Näissä artikkeleissa listattiin vuoden 2021 suosituimpia pelimoottoreita. Artikkelit olivat pelimedioiden, kuten esimerkiksi peliaiheisten lehtien, sekä pelimaailmaan kosketuksissa olevien ihmisten tekemiä. Nämä yhdeksän pelimoottoria lisättiin samaan taulukkoon yrityksen tarpeiden kanssa ja sen jälkeen pelimoottoreita verrattiin em. tarpeisiin tutkimalla moottoreiden kotisivuja.

Tällä tavoin yhdeksästä pelimoottorista valikoitui neljä opinnäytetyössä esiteltävää moottoria. Nämä moottorit olivat Unity, Unreal Engine, GameMaker Studio 2 ja Godot. Näistä moottoreista tehtiin lyhyet esittelyt, joissa käytiin läpi millaisia työkaluja moottoreista löytyi pelin rakentamiseen ja miten pelien luominen kyseisellä moottorilla etenisi. Esittelyn tueksi moottoreiden käyttöliittymistä otettiin kuvia toimintojen hahmottamisen helpottamiseksi.

Näiden neljän moottorin esittelyn jälkeen tehtiin pohdinta, jossa päätettiin mihin kahteen moottoriin haluttiin tutustua syvällisemmin. Pohdinnassa mietittiin alkuperäisiä yrityksen tarpeita ja kuinka hyvin moottorit vaikuttivat vastaavan näihin tarpeisiin, nyt kun niiden toimintaa oltiin nähty käytännössä. Tämän myötä kahdeksi vertailtavaksi moottoriksi valikoituivat Unity, sekä Unreal Engine. Molemmilla moottoreilla toteutettiin testipeli, jotta päästäisiin näkemään millaista peliprojektin teko alusta loppuun näillä työkaluilla olisi.

Peliprojektien jälkeen tehtiin analyysi, jonka tuloksena parhaaksi pelimoottoriksi Ghoulish Gamesille valittiin Unity.

Asiasanat: pelimoottori, pelikehitys, vertailu

Laurea University of Applied Sciences **Abstract**

Degree Programme in Business Information Technology

Bachelor's Thesis

Jere Lämsä

Choosing a game engine for the company Ghoulish Games

Year 2021 Pages 48

The present study was commissioned for the thesis creator's own company Ghoulish Games. The objective was to find a game engine that would suit Ghoulish Games' needs. This game engine would be used to create the next Ghoulish Games game project.

The thesis project began with listing the requirements that the company has for a game engine. These were then turned into a chart. In the end, nine game engines were selected for the preliminary qualifications. They were chosen based on articles listing the most commonly used game engines of 2021. The articles were written by game magazines and gaming world professionals. After the selection, the chosen game engines were put into the same chart with the company's requirements. They were compared to the each other by studying each game engine's homepages and the features they had.

With this comparison, four engines were chosen to be shortly presented in the present study. These engines were Unity, Unreal Engine, GameMaker Studio 2 and Godot. In these presentations, the focus was to look at the tools each of the game engines have for game development and to see what steps they would require to create a game. Each engine presentation included pictures to help the reader understand the user interfaces better.

After the four engines were presented, two were chosen for a deeper analysis and comparison. The decision was made based on the original needs of the company and how well each of the engines eventually seemed to respond to these requirements now that their functions were seen in practice. With this the two engines chosen were Unity and Unreal Engine. A test game was created with both to see what creating a game project from start to finish with these engines was like.

After the two test games were created, an analysis was conducted, and Unity was chosen as the best game engine for Ghoulish Games.

Keywords: game engine, game development, comparison

Sisällys

1 Johdanto 6
2 Opinnäytetyön tavoite ja rajaus 7
3 Benchmarking 7
4 Pelimoottorin valinnan kriteerit 8
5 Pelimoottorit 9
5.1 Unity 11
5.2 Unreal Engine 18
5.3 GameMaker Studio 2 23
5.4 Godot 27
5.5 Kahden pelimoottorin valinta vertailuun 32
6 Pelimoottoreiden vertailu 33
6.1 Unity testipeli 34
6.2 Unreal Engine testipeli 40
6.3 Vertailun lopputulos 44
7 Johtopäätelmä 45
Lähteet 47

1 Johdanto

Pelimoottoreiden markkinoilta löytyy nykypäivänä runsaasti kilpailijoita. Pelimarkkinoiden kasvu ja pelien kasvanut kysyntä on luonut paljon mahdollisuuksia uusien pelikehitystyökalujen, eli pelimoottoreiden ilmestymiselle markkinoille. Tämä taas on luonut lisää kilpailua pelimarkkinoille ja nykypäivänä kuka vaan voi aloittaa pelien kehittämisen omalta kotikoneelta.

Tässä opinnäytetyössä tavoitteena oli löytää sopiva pelimoottori opinnäytetyön toteuttajan omalle toiminimelle, Ghoulish Gamesille. Valittua pelimoottoria tullaan käyttämään Ghoulish Gamesin seuraavassa peliprojektissa. Pelimoottoreita löytyi opinnäytetyön toteuttamisen aikana vähän vajaa 200 kappaletta, joten ensiksi lähdettiin pureutumaan tämän määrän karsimiseen. Tämä tapahtui kartoittamalla tämän hetken suosituimpia, markkinoilta löytyviä pelimoottoreita. Kartoittaessa tutkittiin useita pelimaailman artikkeleita, joita olivat luoneet sekä pelien uutissivustot, että pelien kehittäjät. Näiden artikkelien perusteella poimittiin yhdeksän ehdokasta. Näistä yhdeksästä pelimoottorista valikoitiin neljä yrityksen tarpeisiin, alustavalla tutkimuksella, vastannutta moottoria. Tarpeista luotiin taulukko, jossa listattiin mitä vaatimuksia ja toimintoja yrityksellä on moottoriehdokkaaseen liittyen. Tämä taulukko löytyy kappaleesta neljä.

Nämä neljä moottoria esiteltiin, jotta niistä saataisiin yleiskuva, kuinka pelien kehittäminen moottoreilla toimii. Esittelyn ja moottoreihin tutustumisen jälkeen kaksi moottoria otettiin vertailuun keskenään, jotta niihin saataisiin parempi tuntuma ja päästäisiin näkemään, kuinka nämä työkalut toimivat käytännössä. Näillä kahdella moottorilla luotiin yksinkertaiset testipelit, joissa keskityttiin moottorien erinäisten sisäisten työkalujen ymmärrettävyyteen, laatuun ja paljonko aikaa pelin rakentamiseen näitä työkaluja hyödyntäen meni. Pelien eri rakennusvaiheiden esittely on opinnäytetyön kuudennessa kappaleessa.

Opinnäytetyössä löytyy paljon pelimoottorikieltä, joka voi ajoittain kuulostaa sekavalta. Tämän myötä opinnäytetyössä on avattu jokainen termi lyhyesti niiden esiintyessä ensikertaa tekstissä.

Viimeisessä kappaleessa löytyy pohdinta ja johtopäätös siitä, kumpi moottoreista lopulta valikoitui Ghoulish Gamesin pelimoottoriksi seuraavaan peliprojektiin.

2 Opinnäytetyön tavoite ja rajaus

Opinnäytetyön tavoitteena on löytää helposti lähestyttävä, ammattilaiskäyttöön soveltuva pelimoottori opinnäytetyön toteuttajan yritykselle, Ghoulish Gamesille. Pelimoottoriehdokkaita karsitaan asteittain, ensiksi neljään vaihtoehtoon ja siitä kahteen. Näitä kahta verrataan toisiinsa, benchmarkingin ideaa hyödyntäen ja selvitetään, kumpi moottoreista sopii paremmin yritykselle. Valittua pelimoottoria tullaan hyödyntämään yrityksen seuraavassa peliprojektissa. Ghoulish Gamesin seuraava peli tulee sekoittamaan useita eri peligenrejä (ts. pelien tyylilajeja), joten moottorin täytyy olla joustava, eikä suunnattu vain tietyn genren peleille. Tarkemmat kriteerit on listattu kappaleessa neljä. Opinnäytetyön tarkoitus on myös valottaa pelimoottoreiden toiminnallisuuksia ja ominaisuuksia pelisuunnittelusta kiinnostuneille aloittelijoille, sekä muille pienyrityksille, jotka haluavat tutkia tämän hetken suosituimpia pelimoottoreita.

3 Benchmarking

Tutkimusmenetelmänä opinnäytetyössä käytetään benchmarkingia, eli vertailuanalyysiä. Ojasalon, Moilasen ja Ritakosken mukaan benchmarking on menetelmä, jossa vertaillaan kehittämisen kohdetta toiseen kohteeseen. Menetelmän ideana on oppia toisista kohteista parhaita käytäntöjä, kyseenalaistaa oman kehityskohteen tämänhetkiset käytännöt ja näin kehittää omaa kehityskohdettaan eteenpäin vastaamaan parhaaksi nähtyjä käytänteitä. Tätä menetelmää voidaan hyödyntää mm. organisaation toimintatapojen, laadun ja tuottavuuden parantamiseksi. Benchmarking voidaan toteuttaa esimerkiksi vertailemalla eri organisaatioiden mittareita tai hakemalla tietoa organisaation toteuttamista käytännöistä internetistä tai kirjoista. Joskus voidaan suorittaa myös yritysvierailuita. Parhaan hyödyn saavuttamiseksi vertailussa kannattaa asettaa selkeät tavoitteet, eli mihin kehittämiskohteen piirteeseen halutaan keskittyä. (Ojasalo, Moilanen & Ritakoski 2014, 78-79)

Eli käytännössä Benchmarkingissa valitaan ensiksi kohde, jota halutaan lähteä kehittämään ja vertailemaan. Tämän jälkeen valitaan kohteelle vertailukumppani. Vertailukumppaniksi yleensä valitaan tämän hetken ”paras” kohde/toimintatapa. Kun nämä valinnat on tehty, aletaan keräämään tietoa vertailukumppanista ja omasta kehityskohteesta. Tiedonkeruuta seuraa analyysi. Viimeisenä vaiheena suoritetaan itse vertailu. Vertailun tavoitteena on katsoa kummankin kohteen hyviä ja huonoja puolia ja löytää mitä puutteita omassa kehityskohteessa on vertailukumppaniin verrattuna. Vertailun jälkeen saadaan tulos, miten omaa kehityskohdetta voidaan parantaa ja voidaan alkaa ottamaan näitä parannuksia käytäntöön.

Benchmarkingin ideaa hyödyntäen opinnäytetyössä vertaillaan neljää pelimoottoria yrityksen tarpeisiin. Kaksi parhaiten tarpeisiin vastannutta pelimoottoria on otettu tarkempaan tutkimukseen. Näitä moottoreita verrattiin keskenään, tarkemmin vertailussa tutkittiin moottoreista löytyviä, pelikehitykseen tarkoitettuja työkaluja, sekä käyttöliittymän ymmärrettävyyttä ja moottoreiden teknisiä ominaisuuksia. Näillä kahdella moottorilla toteutettiin yksinkertainen testipeli, jotta niistä saadaan vaikutelma, miltä moottoreiden käyttäminen tuntuu käytännössä.

4 Pelimoottorin valinnan kriteerit

Pelimoottoreiden määrän myötä oikean valinta omalle peliprojektille saattaa olla haasteellista. Näin toteaa myös Yann Kronberg, tuotekehitykseen erikoistuneen yrityksen, Zazmic Inc.:in toimitusjohtaja. Ensimmäisenä tehtävänä pelimoottoria valittaessa, on luoda lista tarpeista, joita peliprojekti vaatii ja tämän myötä rajata pelimoottoriehdokkaat muutamasta sadasta kouralliseen. Jokainen yritys ja kehittäjä joutuu tarkastelemaan projektinsa alussa omia tavoitteitaan ja tekemään rajauksen tähän perustuen. Kronberg itse listaa omiksi rajauskriteereikseen sopivuuden (tarkemmin hinta, moottorin tarjoamat työkalut, käytettävä koodauskieli, lähestyttävyys, julkaisumahdollisuudet), pelimoottorin suosion (tarkemmin käyttäjämäärä, yhteisön luomat tukisivustot, markkinointikohteet) ja projektin laajuuden (tarvitaanko moottorilta paljon eri toiminnallisuuksia vai riittääkö pelille yksinkertaisempi kehitysympäristö). (Kronberg 2017)

Tärkeää projektin ja yritysten tarpeiden lisäksi on miettiä itse pelin tarpeita. Tätä painottaa myös Miko Charbonneau, pelistudio prettysmart games:in perustaja. Tehdäänkö peli kaksi- vai kolmiulotteiseksi? Vai kenties sekoittaen molempia elementtejä keskenään? Miten pelin tyyli toteutetaan taiteellisesti, tarvitaanko moottorilta mahdollisesti erikoisefektejä, kuten valaistusta tai partikkeliefektejä? Lisäksi täytyy ottaa huomioon mille alustalle peli luodaan ja tämän myötä millaista ohjainta pelissä tarvitaan ja tukeeko pelimoottori tarvittavia ohjaimia. Charbonneau mainitsee myös yrityksen markkinointisuunnitelman, tullaanko peliä tarjoamaan ilmaiseksi, myydäänkö sitä kiinteällä hinnalla vai tuottaako se rahaa mainosten kautta. (Charbonneau 2013)

Charbonnau nostaa myös esille pelin ydintyylin. Suosituimpiin pelityyleihin saattaa löytyä jo valmiiksi lainattavaa lähdekoodia, mikä säästää aikaa koodin kirjoittamisesta (esimerkiksi tasohyppelypelit). Sen lisäksi jotkin pelimoottorit on rakennettu täysin tietynlaisten pelityyppien ympärille. Hyvänä esimerkkinä tästä toimii pelimoottori RPG Maker, joka on rakennettu japanilaistyylisten roolipelien (JRPG) rakennustyökaluksi. (Charbonnau 2013)

Aiemmalla pelikehityksen kokemuksella on myös rooli moottorin valinnassa, toteaa Răzvan C. Rădulescu, pelikehityksen ammattilainen. Osa markkinoiden alustoista on luotu erittäin helposti lähestyttäviksi, ja ne hyödyntävät niin kutsuttua visuaalista skriptauskieltä. Tämäntyyppiset pelimoottorit vaativat vähemmän teknistä osaamista koodaamisesta, joskin visuaaliset skriptit hyödyntävät samaa logiikkaa mitä koodaamisessa käytetään. Osa alustoista sen sijaan vaatii pidempiaikaista, teknistä kokemusta koodauksesta. Jotkin alustat, kuten Unreal Engine tarjoavat mahdollisuuden molempiin koodauskeinoihin. (Rădulescu 2020)

Ghoulish Games etsii edellä mainittuihin huomioihin perustuen pelimoottoria, joka sisältää seuraavanlaisia ominaisuuksia. Pelimoottorilla täytyy olla mahdollista toteuttaa sekä kaksi-, että kolmiulotteisia pelejä. Tämän lisäksi hyvä piirre mottorissa olisi se, että siinä voidaan tehdä animointia pelin hahmoille moottorin sisällä, moottorin tarjoamilla työkaluilla. Animointityökalu säästää yrityksen tarpeelta hankkia ulkoista työkalua animoimiseen.

Pelimoottorin ei tarvitse olla täysin ilmainen, mutta sen täytyy tarjota tarpeeksi kattava ilmaisversio, joka mahdollistaa pelimoottoriin tutustumisen ja ensimmäisen kunnollisen peliprojektin luomisen. Ghoulish Gamesin seuraavan peliprojektin peligenre tulee olemaan pääosin kaksiulotteinen, avoimen maailman räiskintäpeli, josta löytyy mahdollisuus sekoittaa kolmiulotteisia elementtejä peliin. Tähän perustuen moottorissa täytyy olla mahdollisuus sekoittaa kaksi- ja kolmiulotteisia elementtejä, sekä eri genrejä keskenään (kuten esimerkiksi kaksiulotteiset tasohyppelyt, räiskintäpelit, ajopelit ja avoimen maailman pelit). Pelimoottorissa ei täten saa olla rajoituksia pelin genreihin liittyen. Pelimoottorin täytyy olla myös tarpeeksi suosittu, jotta siihen löytyy kattavasti pelimoottoriyhteisön luomia ohjeistuksia erilaisten pelimoottorin ominaisuuksien käyttöön.

Ghoulish Gamesin seuraava peliprojekti tullaan julkaisemaan tietokoneelle, joten ohjelmiston täytyy tukea Windows -tietokoneille kehittämistä. On myös yrityksen toimintasuunnitelman kannalta positiivista, jos moottori tarjoaa mahdollisuuden laajemmille alustajulkaisuille tulevaisuuden peliprojekteja ajatellen.

Teknisen puolen vaatimuksia ovat, ettei pelimoottori käytä vain omaa kieltään (DSL-kieltä), vaan se tarjoaa mahdollisuuden koodata yleisillä kielillä. Kielillä, kuten esimerkiksi C#:lla, Javalla, Javascriptillä, Pythonilla tai C++:lla. Seuraava peliprojekti ei tule olemaan grafiikoiltaan pitkälle kehitetty, joten moottorin tarjoaman tehokkuuden ja optimoinnin kannalta ei ole erityisiä vaatimuksia.

5 Pelimoottorit

Pelimoottorit ovat yksinkertaisuudessaan raameja, jonka päälle pelejä pystytään rakentamaan. Pelimoottori koostuu monesta eri osasesta. Nämä osaset tarjoavat projekteille

valmiin renderöintityökalun (teknologia, joka piirtää grafiikat peliin), fysiikkamoottorin (luo peliin fysiikan maailman säännöt, kuten painovoiman), äänityökalut, ja mahdollisesti animointityökalun. Moottori mahdollistaa kaikkiin pelien tulevien assettien (kaikki visuaaliset osat, joita peliin laitetaan; hahmot, tasot, äänet, valaistut yms.) liittämisen toisiinsa helposti ja pelin kokoamisen näistä asseteista.

Kuten aiemmin todettu, pelimoottoreita löytyy nykypäivänä lukuisia ja pelien kehittäminen on nykypäivänä suuren moottoritarjonnan myötä helpommin lähestyttävää. Pelimoottorit mahdollistivat sen, ettei pelien tekemisessä tarvitse tehdä jokaiselle pelille omaa moottoria alusta alkaen, säästäen näin aikaa koodaamiselta luovaan työhön ja itse pelin rakentamiseen. Pelimoottoreita löytyy useaan eri tarpeeseen, riippuen pelin tyylistä, kehittäjän osaamisesta ja julkaisualustoista, joille peli halutaan luoda. Markkinoilla olevan tarjonnan myötä pelimoottoria valitessa kannattaa luoda edellä mainittu tarvelista.

Seuraavaksi käydään läpi neljä pelimoottoria, jotka vastasivat alustavalla tutkimuksella edellä mainittuihin Ghoulish Gamesin esittämiin tarpeisiin. Pelimoottorit karsittiin aluksi sadoista ehdokkaista kymmeneen vertailemalla pelitalojen, kuten Developer Housen, pelimaailmaan kosketuksissa olevien ihmisten, kuten Katie Greenen ja Josh Pettyn, sekä peliaiheisten sivustojen kuten The Toolin ja Game Designin tekemiä artikkeleita. Artikkeleissa listattiin tämän hetken suosituimpia pelimoottoreita.

Näitä suosituimpia pelimoottoreita vertailtiin edellä mainittuihin yrityksen tarpeisiin tutkimalla moottoreiden kotisivustoja ja niillä mainostettuja ominaisuuksia. Yrityksen tarpeista muodostettiin alla näkyvä taulukko. Vasemmalla on listattu pelimoottorit ja yläpalkissa yrityksen tarpeista muodostetut osiot. Värit kuvaavat täyttääkö moottori pyydettyä ominaisuutta

Kehitysalusta	Ilmaisversio	2D ja 3D kehitys	Tukisivut	Monialustakehitys**	Ohjelmointikielet***
Game Maker Studio					
Unity					
Unreal Engine					
Stencyl					
Construct					
Cocos2D-x					
GameSalad					
Buildbox					
Godot					

*** Ilmainen meinaten, että käyttöönotto ja testaus ei maksa mitään**

**** Tarkoitetaan sitä, että kehitysalusta mahdollistaa pelien kehittämisen tietokoneelle, puhelimille ja konsoleille**

***** Mahdollisuus ohjelmoida yleisillä ohjelmointikielillä, kuten C#:lla, Javalla, C++:lla.**

Kyllä

Ei

Taulukko 1: Yrityksen tarpeiden ja pelimoottoreiden ominaisuuksien vertailu

Taulukossa tehdyllä vertailulla ehdokkaat karsittiin kymmenestä moottorista neljään, jotka käytiin opinnäytetyössä läpi. Jokaisesta moottorista esiteltiin kaikki perustyökalut, joita käytetään pelin luomiseen. Kaksi vertailuun valittua moottoria käytiin syvällisemmin läpi luomalla niillä molemmilla testipelit. Tämä osio löytyy opinnäytetyön lopusta.

5.1 Unity

Ensimmäisenä käsiteltävänä pelimoottorina on Unity. Unity on Unity Technologiesin kehittelemä pelimoottori, joka on suunniteltu helpottamaan pelien rakennusta ja julkaisua useille eri alustoille. Ensimmäinen versio Unitysta luotiin vuonna 2005. Vuonna 2019 sillä tehtyjä pelejä pelasi yli 120 miljoonaa pelaajaa ja Unitya käyttäviä pelikehittäjiä löytyy yli 190 eri maasta. Unityn oman arvion mukaan sillä tehdyt pelit kattavat n. 50% kaikista markkinoilta löytyvistä PC-, konsoli- ja puhelinpeleistä, tehden siitä yhden käytetyimmistä pelimoottoreista. (Unity 2021)

Riippuen yrityksen koosta, Unity tarjoaa pelimoottorin joko ilmaiseen tai maksulliseen käyttöön. Tässä tutkimuksessa keskitytään ilmaisen version ominaisuuksiin, mutta maksullisten versioiden ominaisuudet on avattu seuraavissa osissa lyhyesti aiheesta kiinnostuneille.

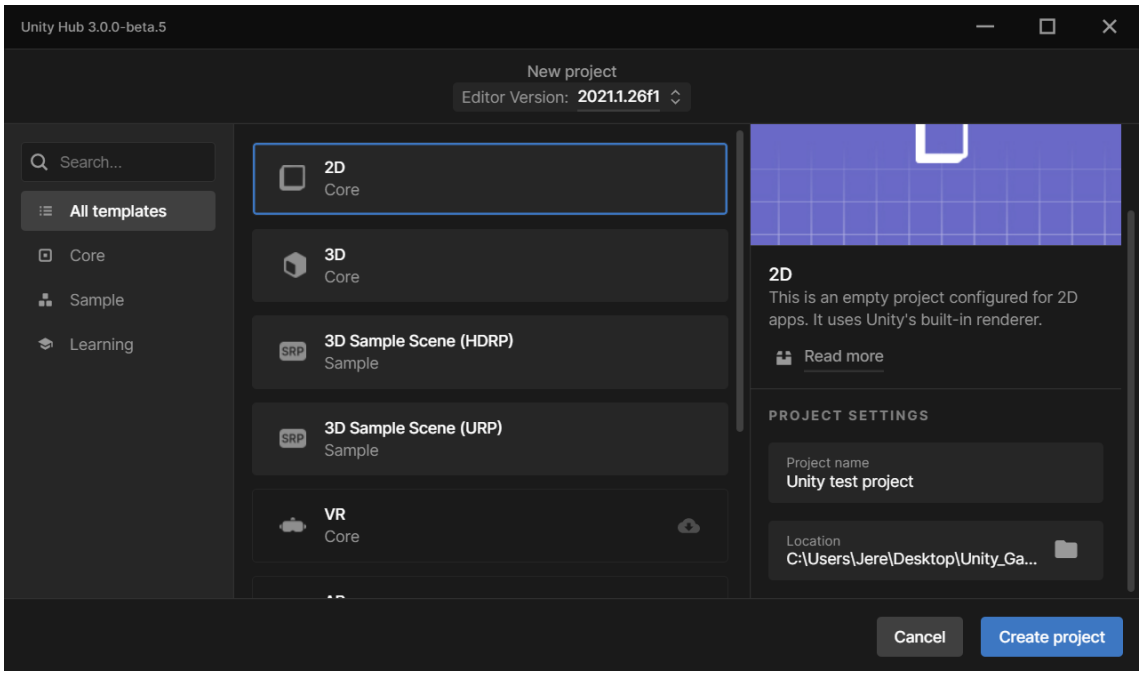
Unitylta löytyy neljä eri pelimoottorituotetta, jotka on suunniteltu käyttäjän/käyttäjien mukaan. Unity Personal on ilmainen ja sitä saavat käyttää yksityishenkilöt, harrastajat ja pienet organisaatiot, joiden liikevaihto tai rahoitus on ollut alle 100 000 $ viimeisen 12 kuukauden aikana. Tästä seuraava taso on halvin maksullinen Unity Plus, joka on tarkoitettu pienille yrityksille ja harrastelijoille. Unity Plus:n hinta on 399 $ vuodessa, tai 40 $ kuukaudessa, ja sitä saavat käyttää organisaatiot, joiden liikevaihto tai rahoitus viimeisen 12kk:n aikana on ollut alle 200 000 $. Jos liikevaihto ylittää tuon rajan, yrityksen täytyy hankkia Unity Pro, jonka hinta on 150 $ kuukaudessa tai 1800 $ vuodessa. Suurille yrityksille, joilla on isot pelikehitystiimit, Unity tarjoaa Unity Enterprisea, jonka hinta on 4000 $ kuukaudessa. (Unity 2021)

Ilmainen Unity Personal tarjoaa Unityn sivustojen mukaan ydinohjelmiston viimeisimmän version, sisältäen kaikki tarvittavat työkalut pelien kehitykselle ja Unityn oppimiselle. Unity Plus sisältää viimeisimmän ohjelmistoversion lisäksi muokattavan splash screenin (ruutu, joka näkyy ennen pelin käynnistymistä. Sisältää yleensä pelin kehittäneen yrityksen ja julkaisijan logot) reaaliaikaiset diagnostiikkatiedot, mikä mahdollistaa pelin kaatumisten ja bugien tarkkailun, tuen peliin lisättäville mainoksille ja pelin sisäisille ostotapahtumille. Näiden lisäksi Unity Plus antaa mahdollisuuden tallentaa projekteja pilveen, jolloin ne ovat saatavilla useilla eri laitteilla. (Unity 2021)

Pro versio lisää Plus -ominaisuuksien päälle vielä pelimoottorin lähdekoodin tarkastelun ja muokkaamisen omien tarpeiden mukaiseksi, sekä teknisen tuen käyttömahdollisuuden (molemmat palvelut tulevat lisämaksua vastaan). Näiden lisäksi Pro:n tilaajat voivat hyödyntää korkealaatuisia assetpaketteja (paketti, jossa on valmiita visuaalisia osia peliin, katso em. määritys assetille), jotka ovat ammattilaisartistien luomia ja kolme lisenssiä Unity Teamsille, Unityn omalle tiimityökalulle, joka hyödyntää pilvitallentamista ja projektin synkronoimista usealle laitteelle. (Unity 2021)

Enterprise -tilauksella Unity antaa edellisten ominaisuuksien lisäksi peliprojektien kehittäjätiimeille teknisen tuen (hintaan sisältyen), etänä käytäviä Unityn järjestämiä opetusluentoja, oppimistukihenkilön ja palvelimen, johon peliprojekti voidaan ladata, jotta sitä voidaan työstää yhtäaikaisesti useamman tiimiläisen toimesta. (Unity 2021)

Unitylla on mahdollista luoda ja julkaista sekä 2D-, että 3D-pelejä puhelimille, tietokoneille, pelikonsoleille ja selaimille. Projektia luotaessa voi valita kumman tyyppisen pelin haluaa tehdä ja Unity tarjoaa projektille tarvittavat työkalut sen mukaisesti. Kuvassa 1 näkyy, miltä projektin luominen näyttää, kun Unity käynnistetään ensimmäistä kertaa.

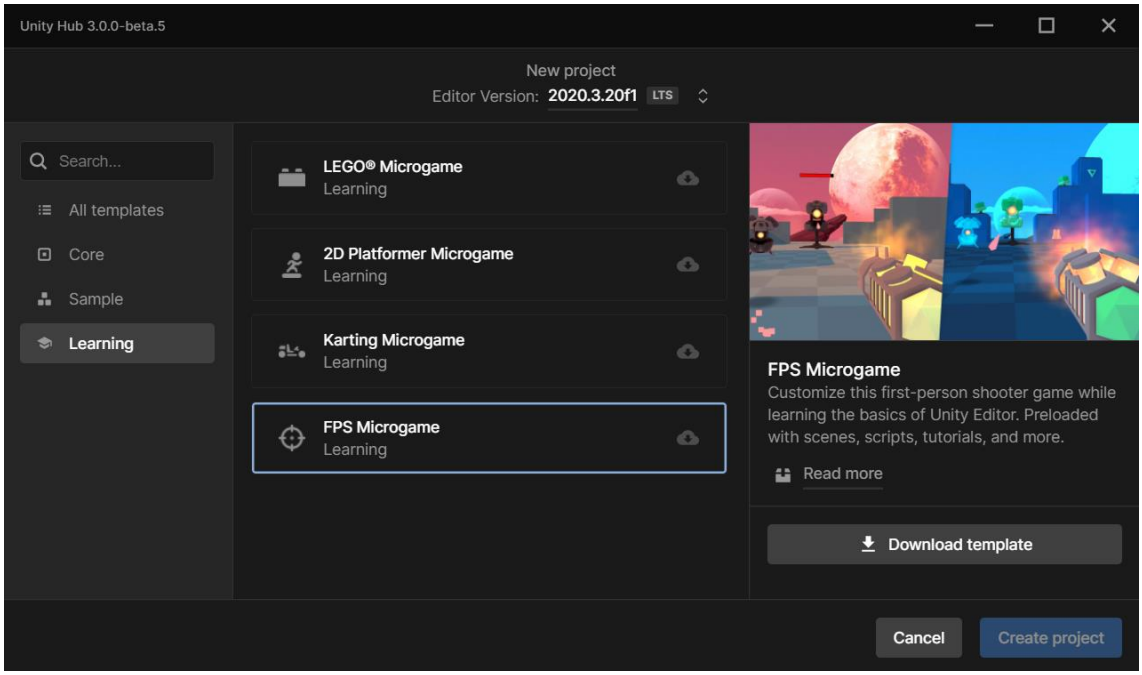


Kuva 1: Unity projektin luonti

Käyttäjä valitsee projektilleen nimen, sekä tallennussijainnin. Tässä vaiheessa tehdään myös valinta, halutaanko peli kehittää kaksi- tai kolmiulotteiseksi, VR laseille, puhelimelle (tässä on myös tarjolla erikseen kaksi- ja kolmiulotteisen pelin luominen) ja mitä Unityn versiota halutaan käyttää. Vaihtoehtoina on myös luoda projekti, joka hyödyntää valmiita Sample

Scenejä. Sample Scene tarjoaa lyhyesti lisätyökaluja esimerkiksi korkealaatuisten kolmiulotteisten ympäristöjen tekemiseen.

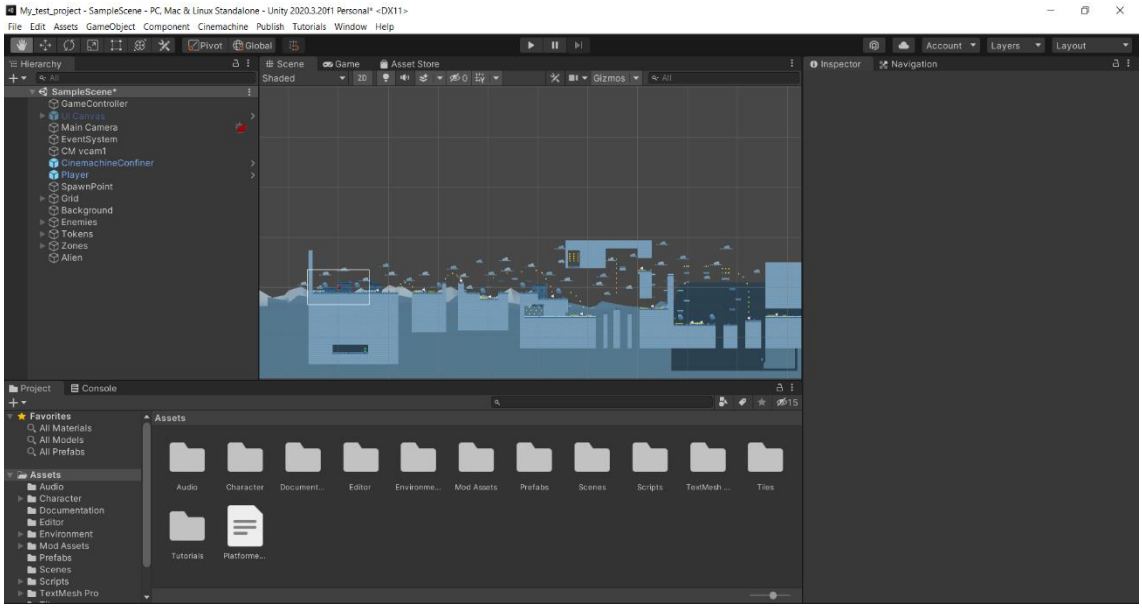
Lisäksi valikosta löytyy ”Opetus” -osio, jos halutaan luoda projekti, jolla on tarkoitus opetella Unityn työkalujen käyttöä. Opetusosio tarjoaa pohjia muutamalle erille pelien alalajille. Opetusprojektit tarjoavat myös vinkkejä ja ohjeita, miten esimerkiksi jokin tietyn tyyppinen ominaisuus lisätään peliin.



Kuva 2: Unity opetusprojektit

Syy miksi projektin luomisvaiheessa tarjotaan erikseen mahdollisuus luoda kaksi- tai kolmiulotteinen projekti on siinä, että kaksi- ja kolmiulotteiset pelit käyttävät erilaisia ”sääntöjä” toimiakseen. Kaksiulotteiset pelit tarvitsevat esimerkiksi erilaisia fysiikan sääntöjä, ne käyttävät kaksiulotteista grafiikkaa, ne piirtyvät ruudulle renderöintityökalulla eri tavalla ja ne animoidaan eri tavalla, kuin kolmiulotteiset pelit. Tämä ei kuitenkaan siitä huolimatta estä sitä, että projektissa voitaisiin sekoittaa molempia ulottuvuuksia keskenään. Unity nimittäin tarjoaa mahdollisuuden vaihtaa kehitysympäristöä kesken projektin, jolloin ulottuvuuksien sekoittaminen on mahdollista.

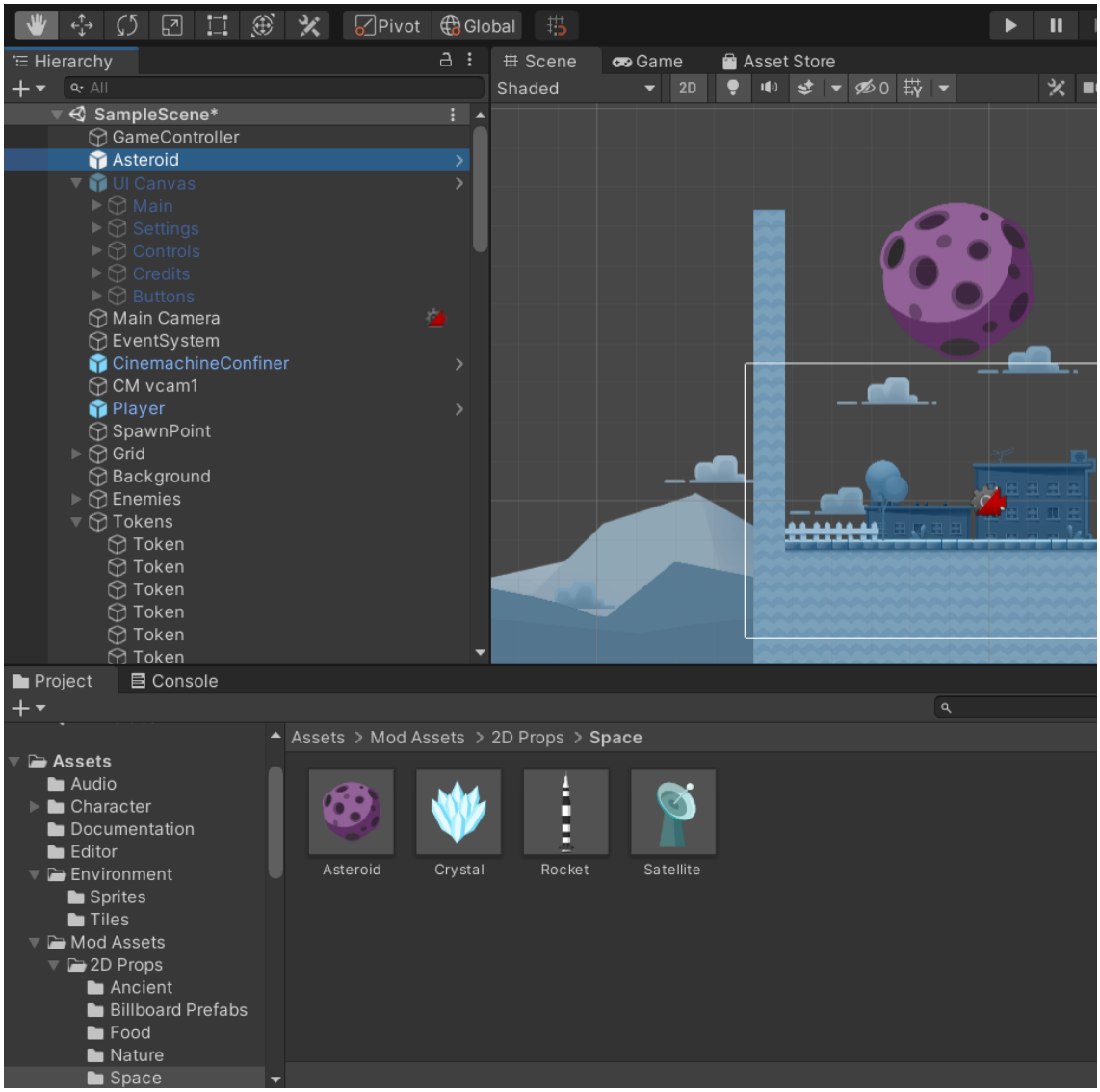
Kun käyttäjä on päättänyt, millainen projekti halutaan rakentaa ja projektille on annettu haluttu nimi, niin projekti voidaan luoda ”Create project” -näppäimellä. Kuvassa 3 nähdään, miltä Unityn aloitusnäkymä näyttää projektin luomisen jälkeen.



Kuva 3: Unity aloitusnäkymä

Unityyn on kuvassa 3 ladattu yksi opetusprojekteista aloitusnäkymän demonstroimiseksi. Keskellä ruutua on alue, johon peli kasataan, näkymää voi suurentaa, pienentää ja liikutella vasemman yläkulman työkalulla (käden kuvake). Keskiruudun yläpuolelta löytyy työkaluja näkymän muuttamiseen kaksi- tai kolmiulotteiseksi, valaistuksen, sekä äänten ja erikoistehosteiden päälle ja pois kytkemiseen. Näiden työkalujen yläpuolelta löytyy toistonappi, jolla itse peliä pääsee pelaamaan ja testaamaan.

Alhaalta löytyy tila kaikille pelissä käytettäville asseteille. Projektin kehittäjä saa itse luoda uusia kansioita tai uudelleennimetä kansioita, miten haluaa. On projektin kannalta suositeltavaa pitää nimet selkeinä, kuten opetusprojektissa näkyy. Kansioiden alta löytyy mm. äänitiedostoja, spriteja (kaksi- tai kolmiulotteisia kuvia) pelaajahahmolle ja vihollisille, sekä animaatiotiedostoja liikkuville spriteille.



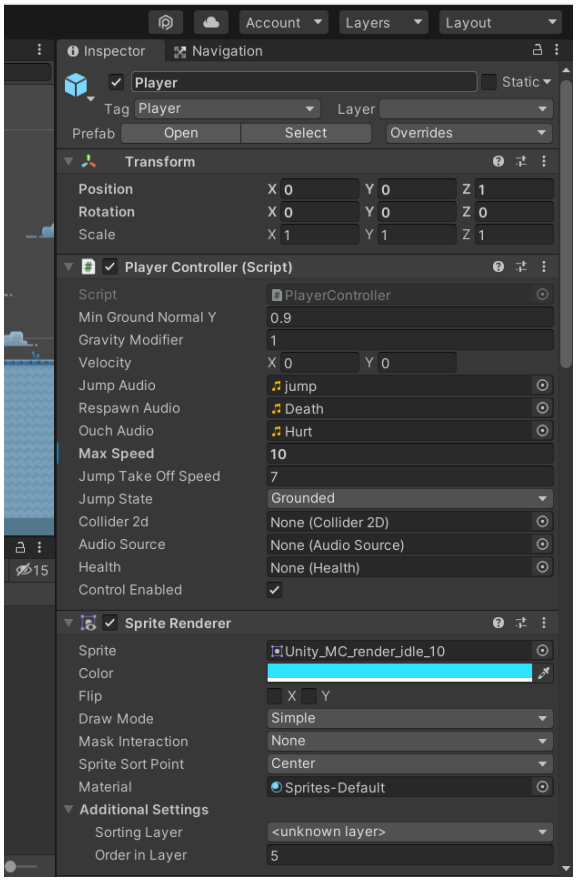
Kuva 4: Spriten lisääminen

Spriten lisääminen peliin käy helposti vetämällä ja pudottamalla se peliruutuun. Esimerkkinä ylemmässä kuvassa vedettiin ja pudotettiin Asteroidi pelin aloitusalueelle. Kun sprite on vedetty kuvaan, sitä pystytään liikuttelemaan vasemmassa yläkulmassa käsityökalun vieressä olevalla työkalulla (eri suuntiin osoittavat nuolet). Liikuttelun lisäksi spriteä voidaan kääntää (pyörivät nuolet) ja sen kokoa voidaan muuttaa suuremmaksi tai pienemmäksi.

Spriten vetäminen ruutuun loi vasemmalla näkyvään ”Hierarchy” -peliobjektilistaan uuden objektin ”Asteroid”. Jokaisella pelissä näkyvällä osalla löytyy peliobjekti. Myös näkymättömillä osilla, kuten esimerkiksi kameralla täytyy olla Unityssa peliobjekti, jossa määritellään säännöt, kuinka objekti toimii. Peliobjekteja pystytään myös luomaan itse klikkaamalla hierarkia-aluetta hiiren oikealla ja valitsemalla avautuvasta listasta millainen

objekti halutaan luoda. Valitaan seuraavaksi ”Player”, eli pelaajan hahmon peliobjekti käsiteltäväksi.

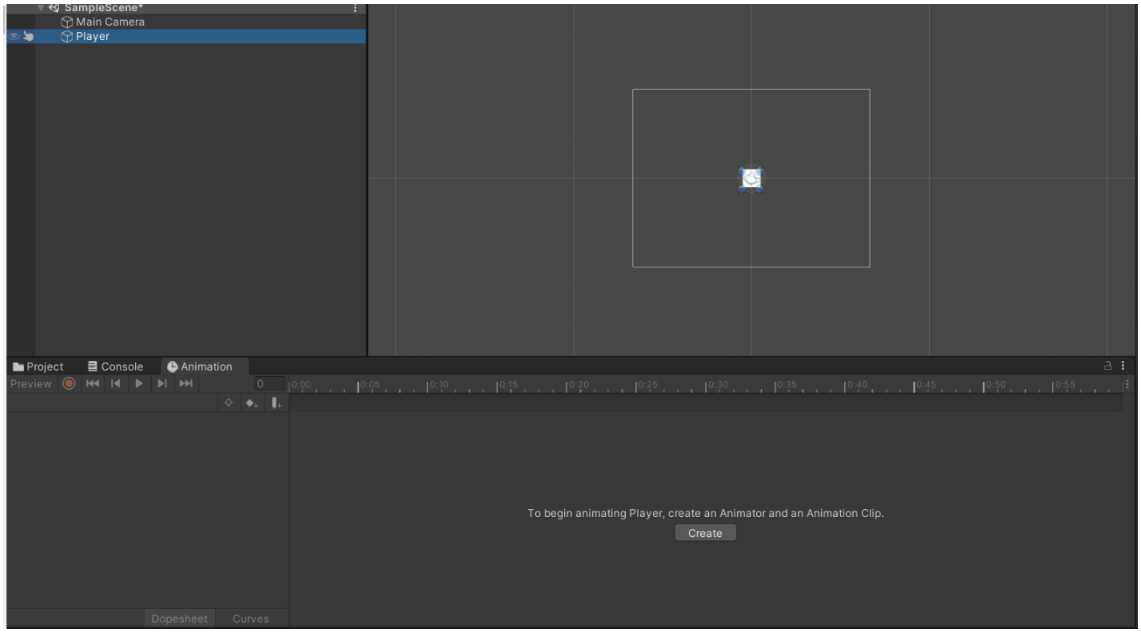
Kun objekti valitaan, niin tämä avaa oikealle puolelle Unitya ”Inspector” -nimisen välilehden. Tällä välilehdellä pystytään määrittelemään säännöt peliobjektille. Jokainen objekti tarvitsee nimen ja määritellyn sijainnin, eli toisin sanoen missä kohdin peliruutua ne ovat, kun peli alkaa. Objektin sääntöjen määritteleminen tapahtuu skripteillä, eli ohjelmistoilla, jossa määritellään koodilla esim. onko objekti pelaajan hallinnassa, kuinka nopeaa se liikkuu, millä näppäimillä sitä ohjataan, mitä käy, jos pelaaja osuu viholliseen. Eli toisin sanoen skripti määrittelee säännöt, miten objekti toimii muiden peliobjektien kanssa. Tämän lisäksi objektille voidaan inspectorissa määrittää fysiikan säännöt, sekä lisätä animointi ja äänet. Komponenttien lisääminen onnistuu inspectorin alaosasta löytyvästä ”Add component” -näppäimestä. Ohessa kuva Unityn inspectori välilehden näkymästä. Kuvassa näkyy esimerkki skriptistä. Skriptin koodia päästään muuttamaan tuplaklikkaamalla ”Script” -kohdassa näkyvää ”PlayerController” -kohtaa.



Kuva 5: Unity Inspector

Unityn aloitusruudussa näkyy myös animointityökalu. Animointi voidaan toteuttaa pääasiassa kahdella eri keinolla. Ensimmäinen keino on luoda nk. sprite sheet, jossa on piirretty

jokaiselle eri liikkeen vaiheelle oma kuva. Nämä kuvat yhdistetään Unityssa ja tämä luo animaation, joka voidaan liittää peliobjektiin ja täten saada objektille eri näköisiä liikkeitä. Toinen keino on käyttää Unitysta löytyvää ”Skeletal animation” -työkalua. Tällä työkalulla pystytään lisäämään hahmolle eräänlainen luuranko. Animointi tapahtuu luurangon osia hiljalleen liikuttamalla ja tallentamalla jokainen eri liikkeen vaihe animointityökaluun. Tällä keinolla animointi voidaan toteuttaa täysin Unityn sisällä. Jotta animaatio voidaan lisätä pelin hahmolle, sille täytyy ensiksi luoda animointikompontentti. Animointikomponentti voidaan lisätä joko ”Inspector” -välilehdellä, tai oheisessa kuvassa näkyvällä ”Animation” -välilehdellä, painamalla ”Create” -nappia (huomiona, jos animaatio -välilehteä ei näy, niin se voidaan lisätä painamalla oikealla näkyvää kolmea pistettä, sen jälkeen ”Add tab” ja ”Animation”).



Animointikomponentin jälkeen voidaan luoda itse animointi joko vetämällä piirretyt liikkeen eri vaiheet animaatio -välilehteen tai luomalla aiemmin mainittu luuranko. Luurankoanimoinnissa painetaan välilehdellä näkyvää punaista nauhoitusnappia, jonka jälkeen luurangon osat liikutetaan haluttuun aloitusasentoon. Tämän jälkeen voidaan siirtyä aikajanalla eteenpäin, siirtää osia uuteen asentoon ja tämä luo animaation hahmolle.

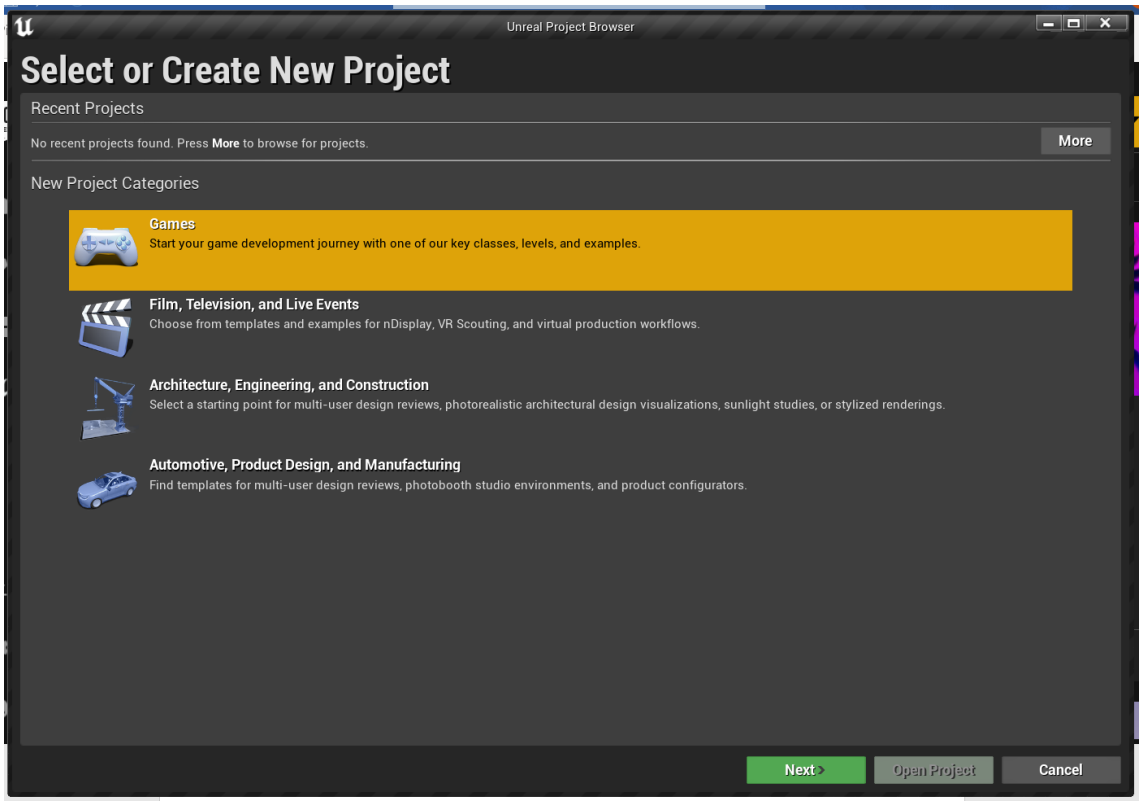
Uuden peliprojektin luominen ja yksinkertaisen pelin rakentamiseen tarvittavat työkalut ovat Unityssa ensivaikutelmassa nopeasti saatavilla jo aloitusnäkymässä. Seuraavaksi käsitellään monien pelitalojen suosima pelimoottori, Unreal Engine.

5.2 Unreal Engine

Unreal Engine on Epic Gamesin omistama ja kehittelemä pelimoottori. Moottoria käytettiin ensimmäistä kertaa vuonna 1998 ”Unreal” -nimisessä räiskintäpelissä ja sen kehitti Tim Sweeney, Epic Gamesin perustaja. Alun perin moottori luotiin tietokoneella pelattavien ensimmäisen persoonan räiskintäpelien kehittämiseen, mutta moottorin kehitysmahdollisuudet ovat sittemmin laajentuneet muille pelialustoille ja pelityypeille. Nykypäivänä Unrealilla on mahdollista luoda pelejä Windowsille, MacOS:lle, Linuxille, Playstation 4:lle ja 5:lle, Xbox Onelle ja X:lle, Nintendo Switchille ja puhelimille. Unreal Engine on tarkoitettu pääasiassa kolmiulotteisten pelien kehittämiseen, mutta siihen pystytään lataamaan lisäosia, jotka mahdollistavat kaksiulotteisten pelien kehittämisen. (Unreal Engine 2021)

Unrealin sivustojen mukaan moottoria ei olla käytetty pelkästään pelien kehittämiseen, vaan sitä on hyödynnetty arkkitehtuurissa, autojen ja muiden kulkuvälineiden suunnittelussa, elokuvissa, tv-sarjoissa, simulaatioissa, sekä AR:ää hyödyntävissä livetapahtumissa. Unreal Enginen tämänhetkinen versio on Unreal Engine 4 (julkaistu 2014) ja pelimoottorissa käytetään ohjelmointikielinä C++:aa ja visuaalista Blueprintiä. Epic Gamesin mukaan Unreal Engine 5 on tulossa markkinoille vuoden 2022 alkupuolella. (Makuch 2021)

Unreal Engine 4 tarjotaan tällä hetkellä muutamaa eri lisenssiä. Pääasialliset lisenssit ovat julkaisija- ja luojalisenssi. Molemmat ovat ilmaisia käyttää, mutta niissä on pieni ero riippuen, käytetäänkö moottoria kaupallisen tuotteen luomiseen. Julkaisijoiden lisenssi on tarkoitettu kaupallisille tuotteille. Tällä lisenssillä käyttäjä allekirjoittaa Unrealia ladatessaan sopimuksen, joka valtuuttaa Epic Gamesin 5% rojalteihin, jos pelimoottorilla luodaan ja julkaistaan kaupallinen tuote, joka elinkaarensa aikana ylittää yli 1 000 000 $ myynneissä. Jos moottorilla halutaan kehittää ei-kaupallisia tuotteita, on sen käyttö täysin ilmaista tässä tapauksessa. Epic Games tarjoaa myös mahdollisuuden neuvotella ja muovata lisenssi yrityksen tarpeita vastaavaksi. Muovatuissa lisensseissä on mahdollista neuvotella yritykselle perustyökalujen lisäksi oikeudet käyttää kaikkia Unrealin oppimateriaalieja, premium tukipalveluita, yksityisiä koulutus-sessioita ja laajentaa Unityn käyttökohteita peleistä muihin aiemmin mainittuihin projekteihin. Tässä tutkimuksessa keskitytään julkaisijalisenssiin sisältyvään Unreal Engineen. (Unreal Engine 2021).



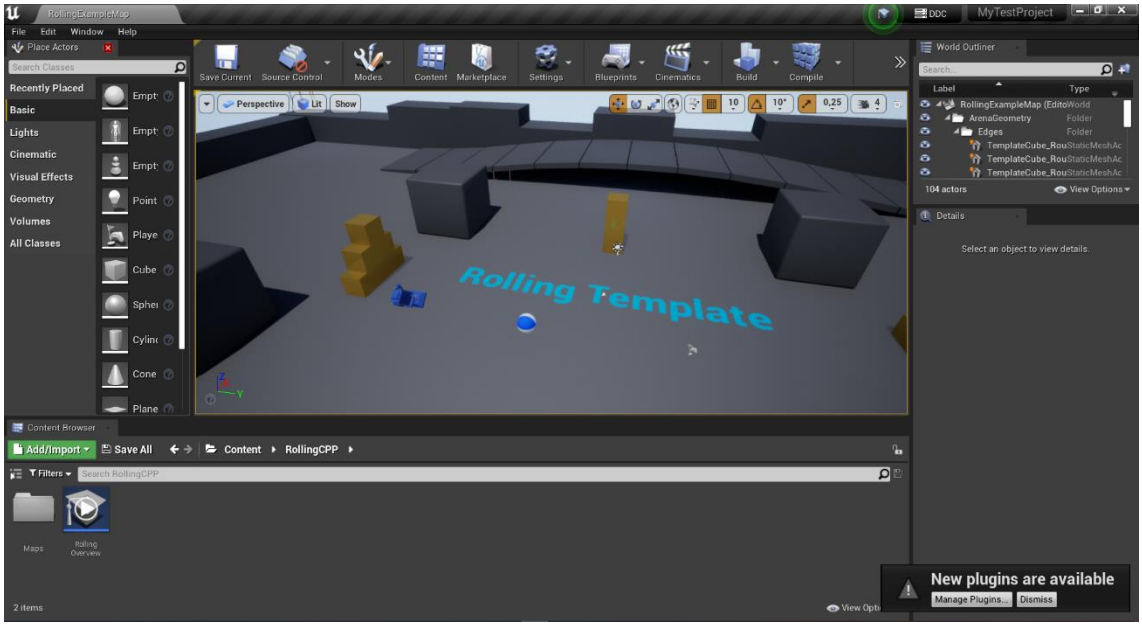
Kuva 6: Unreal projektinluonti

Unreal Engine ladataan Epic Games Launcherin kautta ja tämä vaatii käyttäjätunnuksen Epic Gamesille. Projektia luodessa voidaan päättää eri projektityypeistä, millainen projekti halutaankaan luoda. Vaihtoehtoina on luoda kolmiulotteinen, animoitu projekti, suunnitteluprojekti tai peliprojekti. Tässä tapauksessa valitaan peliprojekti. Tämän jälkeen Unreal tarjoaa seuraavanlaiset valmiit pohjat riippuen siitä minkä tyyppistä peliä halutaan lähtä luomaan. Tämä on tarkoitettu nopeuttamaan pelikehitystä tarjoamalla kehittäjälle valmiit rakennuspalikat, joista voidaan lähtä luomaan omaa peliä. Vaihtoehtoisesti kehittäjä voi halutessaan valita tyhjän alustan, jolloin peliä lähdetään rakentamaan nollapisteestä.

Kuva 7: Unreal projektin alusta

Pohjan valitsemisen jälkeen voidaan päättää, mitä ohjelmointikieltä halutaan käyttää, C++:aa vai Blueprintiä. Blueprint on Unreal Engineen kuuluva visuaalinen ohjelmointikieli. Projektissa voidaan sekoittaa ja käyttää molempia, sekä C++:aa, että Blueprintiä. Visuaalisena ohjelmointikielenä Blueprint vaatii vähemmän teknistä osaamista ohjelmointikielistä ja voi olla helpommin lähestyttävä uusille pelikehittäjille. (Unreal Engine 2021)

Ohjelmointikielen valitsemisen jälkeen voidaan valita, tuleeko projekti olemaan korkealaatuinen grafiikoiltaan, mille alustalle projektia lähdetään kehittämään (tätä voi vaihtaa jälkikäteen projektin asetuksista) ja halutaanko projektiin ladata Unrealin tarjoamia valmiita sisältöpaketteja. Näiden sisältöpakettien mukana tulee mm. tekstuureita, efektejä, ääniä, materiaaleja ja valmiita peliobjekteja (hahmoja, lavasteita yms.). Lopuksi päätetään projektin nimi ja luodaan se. Jos mieltä halutaan muuttaa myöhemmin ja projektin tyyli muuttuu, niin Unrealissa on joustavasti mahdollisuuksia muokata projekti alkuasetuksista erilaiseksi.

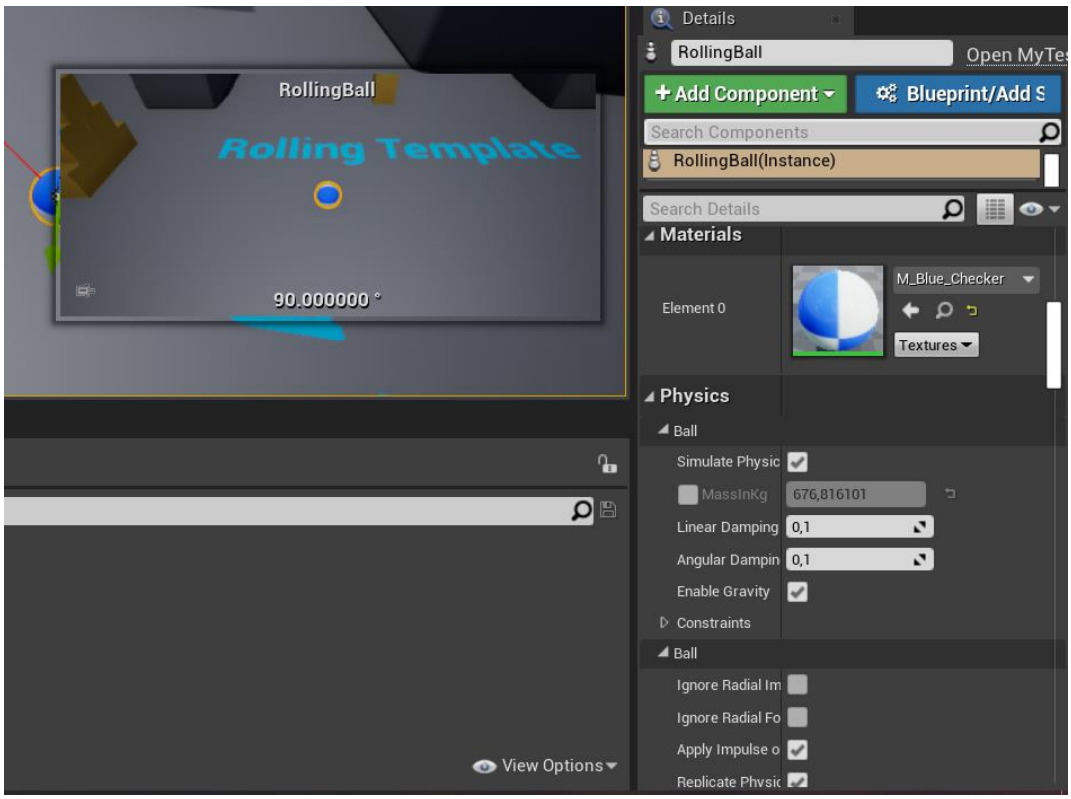


Kuva 8: Unreal perusnäkymä

Projektin luomisen jälkeen avautuu kuvassa näkyvä näkymä, sekä erillinen Visual Studio ikkuna (ohjelmointia varten). Ohjelmiston näkymän rakenteessa on muutamia samankaltaisuuksia aiemmin läpikäydyn Unityn kanssa. Keskellä näkyy itse peliruutu. Peliruutuun voidaan vetää objekteja joko alhaalta löytyvästä asset -osiosta (kehittäjän omat assetit ja projektiin ladatut valmiit assetit), tai vasemmalta ruudussa löytyvästä listasta (Unrealin tarjoamat valmiit assetit). Peliruudulla näkyviä objekteja on mahdollista kopioida ja liikutella kolmiulotteisesti ruudun sisällä. Asset -osioon voidaan tiputtaa kehittäjän omia luomia tiedostoja ja siihen voidaan luoda omia kansioita. Asset -osiosta löytyy myös tämän lisäksi hakutyökalu (jos halutaan hakea esimerkiksi jotain tiettyä assettia nimellä.

Näkymästä löytyvällä Add/Import -napilla voidaan tuoda projektiin uusia assetteja. Tätä kautta voidaan lisätä esimerkiksi uusia hahmoja, erikoisefektejä, materiaaleja, tasoja tai valmiita sisältöpaketteja. Sisältöpaketit sisältävät valmiita assetteja, joilla pelin rakentaminen voidaan aloittaa nopeasti. Sisältöpaketteja voi myös ladata lisää Unreal Enginen kaupasta (Löytyy Epic Games alustalta, Unreal Engine välilehdeltä). Jotkin sisältöpaketit ovat täysin ilmaisia ja joistakin joutuu maksamaan. Tällä työkalulla voidaan myös tuoda projektiin elementtejä muista aiemmin mainituista projektipohjista, joten projektia voidaan muovata kokonaan erityyppiseksi, kuin mitä alkuasetuksissa sille on määritetty. Esimerkiksi jos peli sisältää tasoja, jotka toimivat eri logiikalla (esimerkiksi siirrytään tasohyppelystä tasoon, jossa ajetaan ajoneuvolla), niin tämä työkalu mahdollistaa näiden elementtien yhdistämisen. Unreal mahdollistaa myös assettien ja tasojen tuomisen toisesta Unreal projektista toiseen, assettien siirtotyökalulla.

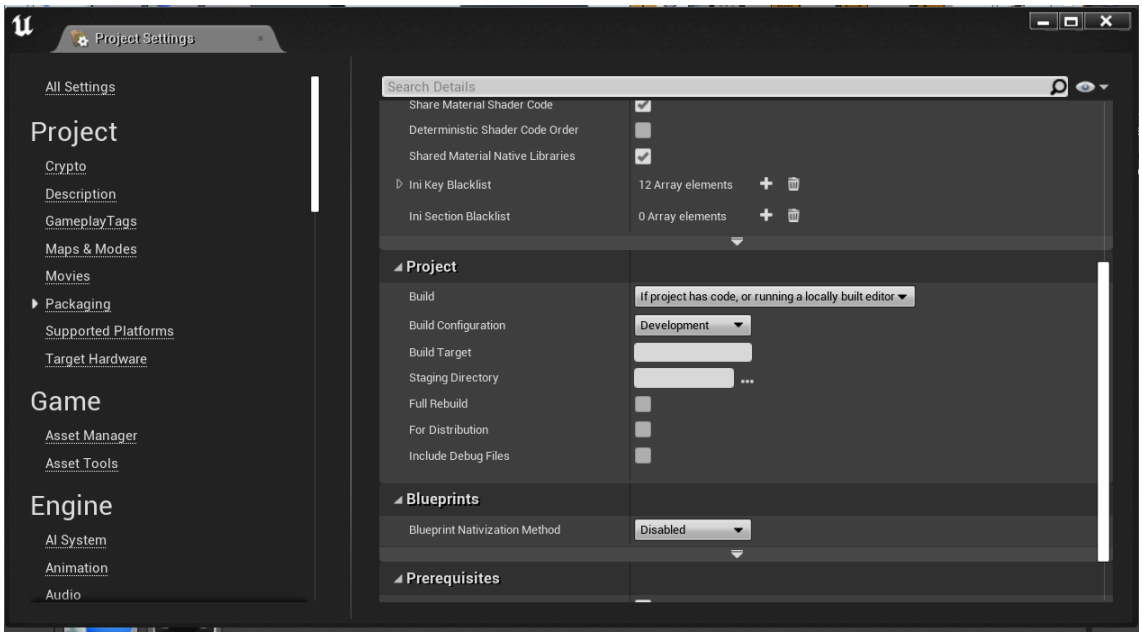
Oikealta näkymästä löytyy lista kaikista osista, mitä pelissä tällä hetkellä on (efektit, kamera, pelaajan hahmo, kerättävät kolikot yms.). Klikkaamalla peliruudussa tai oikealla olevassa listassa jotain pelin osaa, avautuu oikeaan alakulmaan valikko, jossa voidaan päättää mitä ominaisuuksia kyseiselle osalle annetaan (Kuva 9). Tässä voidaan muokata osan fysiikan sääntöjä, mitä materiaalia siinä käytetään, lisätä ohjelmointia (C++ tai Blueprint) ja määritellä miten osa toimii ja käyttäytyy. Peliruudussa näkyviä osia voidaan myös liikutella siihen ilmesytyvillä nuolilla, sekä peliruudun yläpalkissa näkyvillä työkaluilla ja näin uudelleensijoittaa niitä peliin.



Kuva 9: Unreal peliobjektin ominaisuudet

Uuden tason luominen voidaan aloittaa klikkaamalla joko asset -osiossa hiiren oikealla ja tämän jälkeen klikkaamalla ”New level” tai sitten klikkaamalla Unrealin vasemmasta yläkulmasta ”File” ja ”New level”. Tason luominen tarjoaa muutamaa valmista pohjaa tasolle, esimerkiksi valmiin ”lattian”, jonka päälle objektit voidaan lisätä, sekä valmiin valaistuksen ja skyboxin (pelin taustalla näkyvä maisema). Jos kehittäjä niin haluaa, niin hän voi aloittaa myös täysin tyhjästä tasosta. Tason luomisen jälkeen siihen voidaan alkaa lisäämään objekteja, sääntöjä, efektejä, materiaaleja ja ääniä asset -osiosta tai vasemmasta listasta.

Lopuksi, kun on luotu halutut tasot ja peli on valmiina, se täytyy pakata ja viedä Unrealista ulos. Projekti voidaan pakata joko täysin valmiina, tai jos siihen täytyy tehdä vielä jonkinlaista ohjelmointia tulevaisuudessa, niin se voidaan pakata myös ”Kehittäjä” -konfiguraatiossa. Pakkausasetuksia voidaan muuttaa projektin asetuksissa (Kuva 10).



Kuva 10: Unreal - Projektin asetukset

Ennen pakkaamista ja vientiä projekti täytyy rakentaa perusnäkymän ylhäällä näkyvästä ”Build” -napista. Projekti voidaan rakentaa joko osa kerrallaan ”Build” -napin alavalikosta, tai kaikki kerralla pelkästä ”Build” -napista. Osissa rakentamista suositellaan isommille projekteille. Rakentamisen jälkeen projekti pakataan klikkaamalla ”File” ja ”Package project”. Tässä vaiheessa voidaan valita mille alustalle projekti pakataan (PC, puhelin, pelikonsoli). Pakkaamisen jälkeen peli on valmis pelattavaksi ja pelimarkkinoille jaettavaksi.

5.3 GameMaker Studio 2

Game Maker Studio on pitkään markkinoilla ollut, alun perin Mark Overmarsin, kehittelemä pelimoottori. Moottori kehitettiin alun perin nimellä Animo vuonna 1999. Myöhemmin sen kehittäjäksi vaihtui YoYo Games ja moottorin nimestä tuli GameMaker. Uusin versio moottorista julkaistiin vuonna 2017 ja se kulkee nimellä GameMaker Studio 2, lyhennettynä GMS2. (YoYo Games 2021)

Moottorista on tarjolla ilmaisia ja maksullisia versioita. Maksulliset versiot ovat tilauspohjaisia. Ilmaisversio on tarkoitettu pääasiassa GMS2:n opettelemiseen ja se tarjoaa mahdollisuuden jakaa kehitettyjä pelejä YoYo Gamesin omalla pelien jakoalustalla, GXC:llä. Ilmaisversio ei anna kehittäjän julkaista luomuksiaan muilla alustoilla. Ilmaisversio tarjoaa

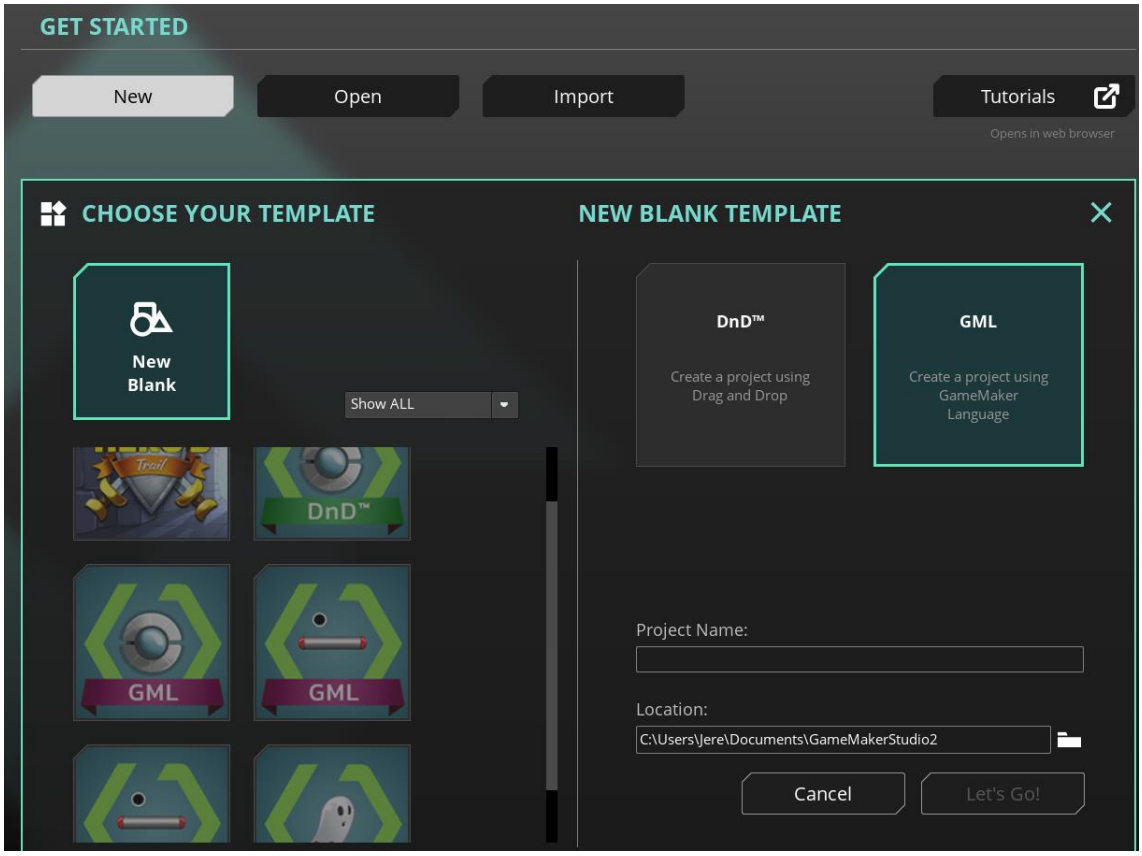
kuitenkin toiminnallisesti kaiken mitä GMS2 sisältää, jotta kehittäjä voi testata työkalua ja halutessaan tilata maksullisen version. (YoYo Games 2021)

Seuraava taso työkalusta on maksullinen ”Indie” -tilaus, joka on tarkoitettu pienille yrityksille ja yksin toimiville kehittäjille. Tämän tilauksen hinta on 8,19 € kuussa tai 84,99 € vuodessa. Erona Indie- ja ilmaistilauksen välillä on mahdollisuus julkaista pelejä GameMakerin oman alustan lisäksi muilla tietokoneen sovellus- tai selainpohjaisilla pelialustoilla, sekä puhelimilla. Tämän lisäksi Indie -tilauksessa tulee mukana tehokkaampi compiler -työkalu (työkalu, joka ajetaan peliprojektin lopuksi, rakentaa pelin yhtenäiseksi jotta se voidaan viedä ulos pelimoottorista.), Yoyo Compiler (”YYC”). Indie-tilaus mahdollistaa myös kolmansien osapuolien tuottamien lisäosien lisäyksen peliprojektiin. (YoYo Games 2021)

Viimeisenä tilausmuotona on ”Enterprise” -tilaus, joka on tarkoitettu keskisuurille ja suurille yrityksille. Aiemmin mainittujen ominaisuuksien lisäksi tässä tilausmuodossa avautuu mahdollisuus kehittää pelejä pelikonsoleille, kuten Playstation 4 & 5:lle, Xbox Onelle (Xbox X kehitysmahdollisuus on tällä hetkellä betatestauksessa, eli viimeisessä vaiheessa ennen ominaisuuden julkaisemista) ja Nintendo Switchille. Lisänä näille, Enterprise -tilaajat saavat pääsyn GMS2:n Helpdeskiin ja vaihtoehtoisiin pelin rakentamiskeinoihin. (YoYo Games 2021)

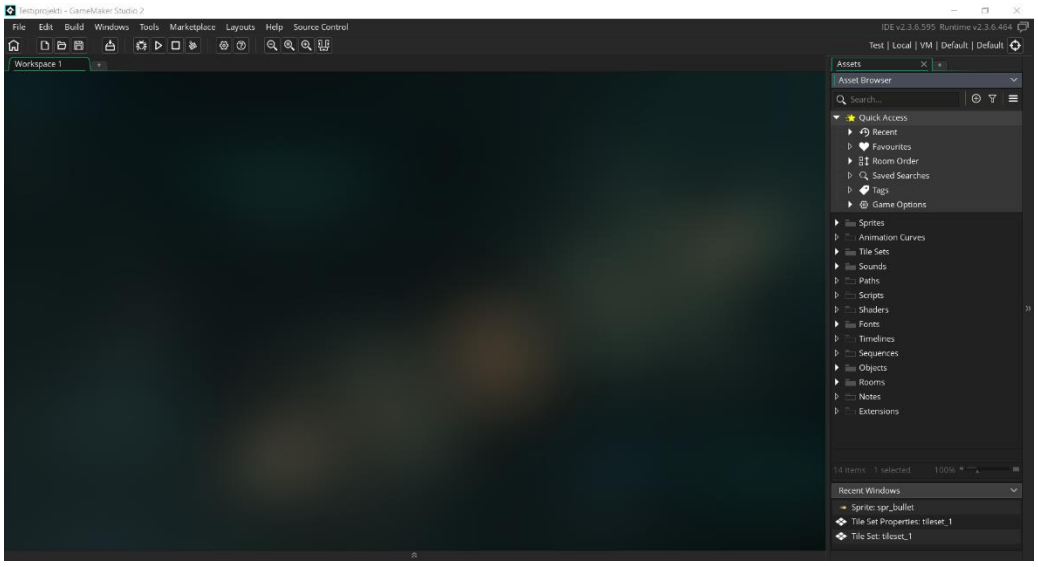
GameMaker Studio 2 käyttää yleisten ohjelmointikielien sijaan täysin moottorille ominaisia ohjelmointikieliä. Nämä kielet ovat moottorin oma GameMaker Language, sekä Drag and Drop. Drag and Drop on aivan täysin visuaalinen, eikä vaadi yhtään käsintehtyä ohjelmointia. Moottorilla on mahdollista luoda kaksi- ja kolmiulotteisia pelejä, mutta sen vahvuudet keskittyvät enemmän kaksiulotteisten pelien ympärille. Kuten Unreal Enginessä, GMS2:en sivuilta löytyy kauppa, josta voi ostaa tai ladata lisäosia ja valmiita materiaaleja pelimoottoriin ja peliprojekteihin. Kaupasta löytyy myös ladattavia opetusprojekteja, jotka helpottavat pelimoottorin käyttöönottoa. (YoYo Games 2021)

Uutta peliprojektia luodessa on mahdollista luoda projekti valmiiseen pelipohjaan, tai vaihtoehtoisesti aloittaa tyhjästä projektista (Kuva 11). Jos valitaan tyhjä projekti, niin projektiin valitaan kumpaa GMS2:n ohjelmointikielistä siinä halutaan käyttää. Valmispohjista löytyy valmiita tasoja, hahmoja ja sääntöjä, joten pelin rakentaminen voidaan aloittaa pienellä etumatkalla tyhjään projektiin verrattuna.



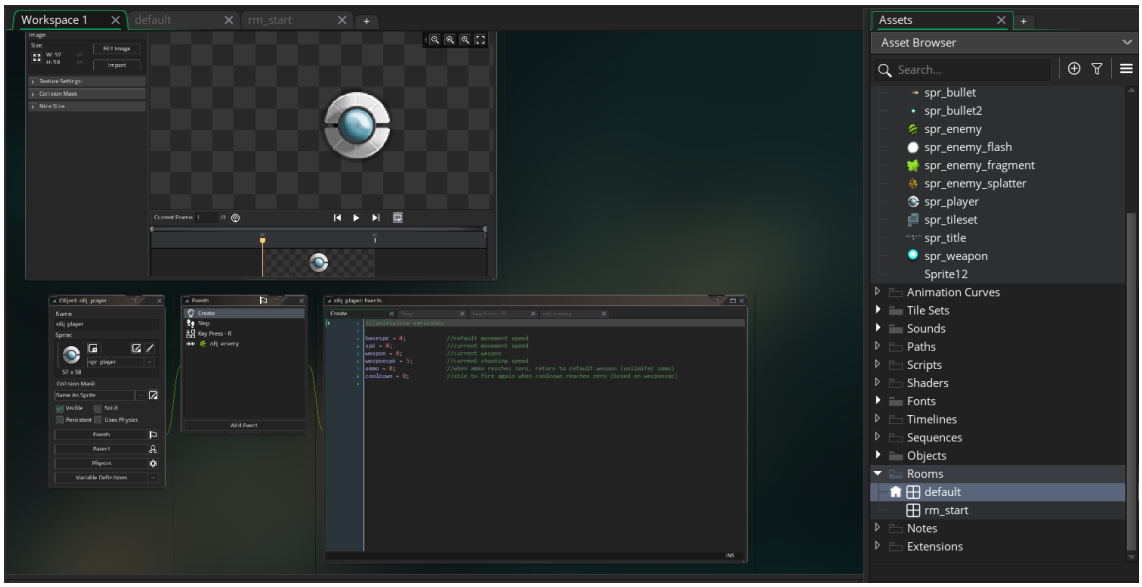
Kuva 11: GMS2 projektin luonti

Kun Game Maker Studio 2 projekti on luotu, avautuu kuvan 12 näkymä. Poiketen aiemmin käsitellyistä pelimoottoreista, aloitusnäkymässä ei näy peliruutua. Ruutu voidaan avata erikseen tuplaklikkaamalla oikealla näkyvän ”Rooms” -kansion alta löytyviä tasoja.



Kuva 12: GMS2 aloitusnäkymä

Oikealla ruudussa on osio asseteille. Jos kehittäjä haluaa tuoda projektiin esimerkiksi toisessa ohjelmassa piirrettyjä pelihahmoja, niin se voidaan tehdä oikean yläkulman plus -näppäintä klikkaamalla. Tämän jälkeen valitaan minkä tyyppinen assetti halutaan luoda, pelihahmojen tapauksessa klikataan ”Sprite”, klikataan ”Import” -näppäintä ja valitaan kehittäjän tietokoneen kansioista hahmot jotka halutaan tuoda. Uniikkina työkaluna GMS2:sta löytyy myös siihen sisäänrakennettu piirtotyökalu sprite -tyyppisille asseteille. Eli jos peli on kaksiulotteinen, kehittäjä pystyy piirtämään suoraan kaikki peliin kuuluvat hahmot pelimoottorin sisällä. Tuplaklikkaamalla oikeasta valikosta spritea, se aukeaa omaan ruutuunsa, jossa voidaan tarkastella esimerkiksi millainen animaatio spritella on (Kuva 13).

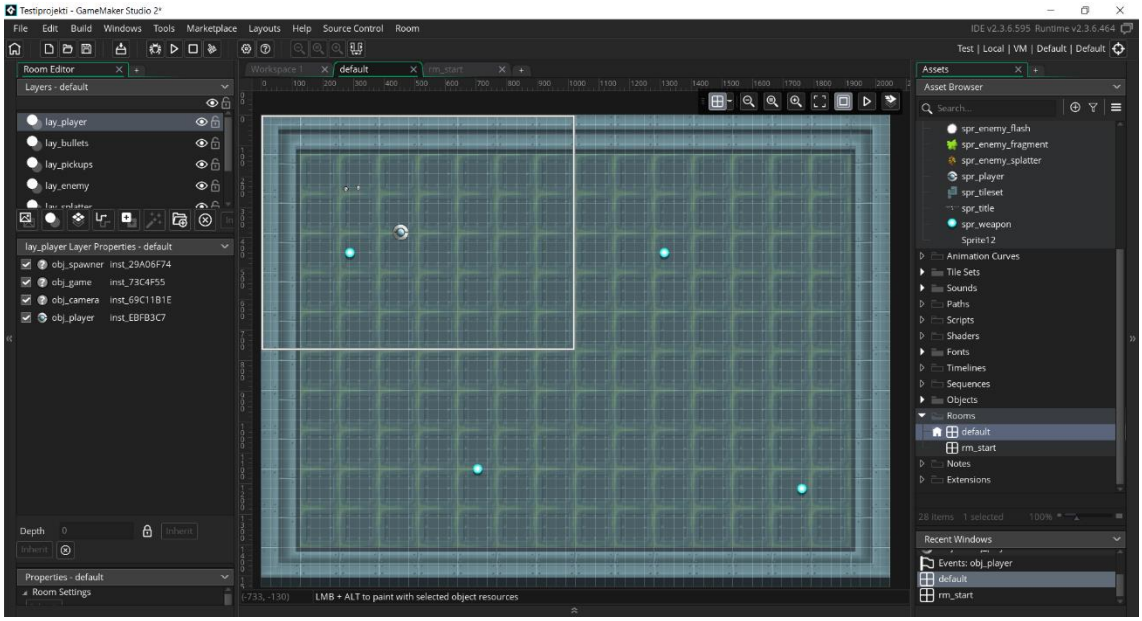


Kuva 13: GMS2 assettien ominaisuudet

Kuvassa 13 nähdään sprite -ikkuna ja sen alapuolella object -ikkuna. Object -ikkunassa määritellään säännöt, miten sprite käyttäytyy peliruudulla. Kuvan esimerkissä on avattu pelaajan hahmo. Jokainen sprite vaatii sille määritellyn säännön (eli GameMaker kielellä objektin) toimiakseen pelissä. Eli sprite on se, miltä assetti näyttää ja objekti on se, miten assetti käyttäytyy. Kuvan 13 esimerkissä on luotu sääntö ”obj_player”, joka on yhdistetty spriteen ”spr_player”. Object -ikkunassa näkyy ”Events” -kohta jonka alta löytyy eri sääntöjä jaotellusti (esim.miten hahmoa liikutellaan). Tuplaklikkaamalla sääntöä aukeaa koodi-ikkuna, josta löytyy säännön määrittelevä koodi. Koska kyseessä on pelaajan hahmo, niin tässä ikkunassa on määritelty, miten hahmoa voidaan liikutella, miten hahmolla ammutaan torpedoja ja mitä käy, jos pelaaja osuu viholliseen.

Kun sprite ja siihen liittyvä sääntö on luotu, niin se voidaan lisätä peliin. Peliruutu voidaan avata tuplaklikkaamalla ”Rooms” -kansion alta löytyviä tasoja. Peliruutuun voidaan lisätä assetteja seuraavasti: Valitaan assetti oikeasta valikosta, painetaan alt -näppäin pohjaan ja

klikataan sitä kohtaa ruudusta johon assetti halutaan lisätä. Kuvassa 14 näkyy esimerkki peliruudusta, johon on jo valmiiksi lisätty erilaisia assetteja.



Kuva 14: GMS2 peliruutu

Jokainen GMS2:ssa luotu peli vaatii siis neljä ydinasiaa, että peli voidaan luoda. Ensimmäisenä luodaan taso luomalla assetti ”Rooms” -kansion alle. Tämän jälkeen luodaan assetti ”Sprites” -kansion alle. Sitten lisätään assetille säännöt luomalla sille objekti ”Objects” -kansion alle. Lopuksi tämä valmis assetti lisätään ”Rooms” -kansion alta avattuun tasoon. Peliä päästään pelaamaan klikkaamalla yläpalkista ”Toista” -nappia.

Kun tasolle on lisätty kaikki halutut hahmot, voidaan luoda uusi taso ”Rooms” -kansion alle. Kun halutut tasot on luotu ja peli on valmis, niin se voidaan tallentaa ja viedä ulos GMS2:sta. Pelin vieminen käy klikkaamalla vasemmasta yläkulmasta ”File” ja tämän jälkeen ”Export Project”. Riippuen siitä mikä versio GMS2:sta on tilattu, tässä vaiheessa tarjoutuu erilaisia vaihtoehtoja. Jos peliä on esimerkiksi tarkoitus pelata Windows -pohjaisilla tietokoneilla, niin ”Export Project” -napin painamisen jälkeen valitaan viemisen tyypiksi zip-tiedosto. Tämä pakkaa pelin, vie sen zippitiedostoon ja tallentaa sen kehittäjän valitsemaan sijaintiin, josta peliä päästään pelaamaan.

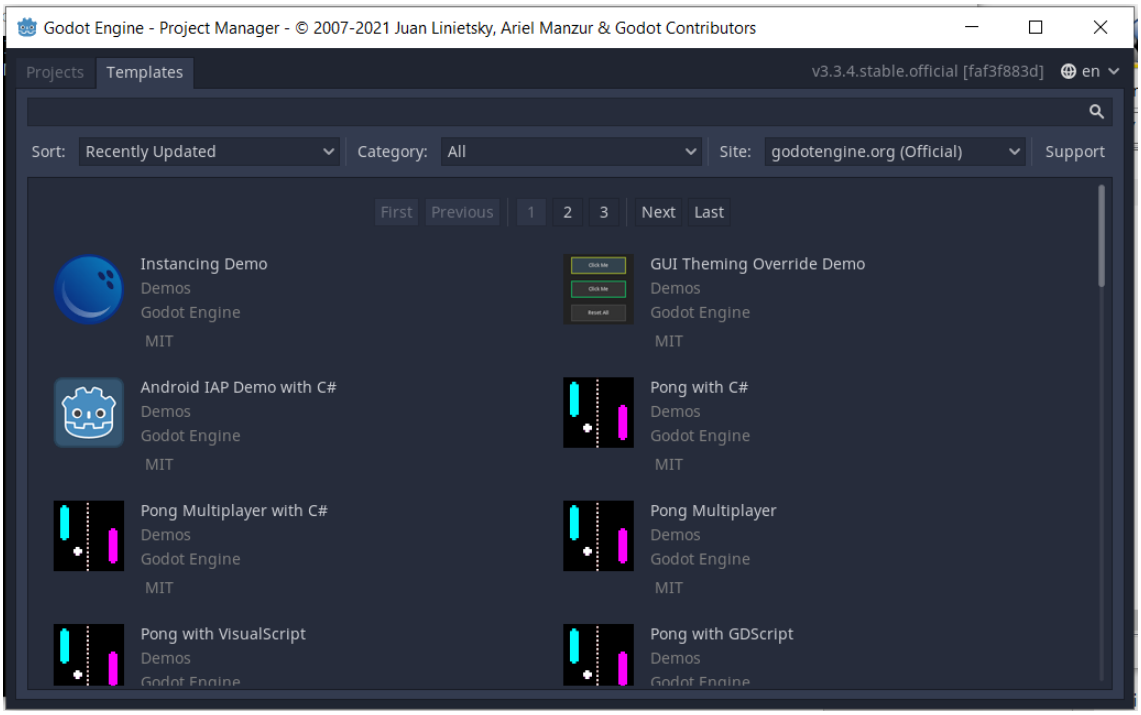
5.4 Godot

Godot on tämän tutkimuksen esiteltävistä pelimoottoreista tuorein julkisille markkinoille tullut pelimoottori. Godotin alkuperäisinä kehittäjinä toimivat Ariel Manzur ja Juan Linietsky ja pelimoottorin ensimmäinen versio kehitettiin vuonna 2014 ja sittemmin se julkaistiin julkisille markkinoille MIT-lisenssin alla. Godot on avoimen lähdekoodin (avoin lähdekoodi

tarkoittaa, että koodi, jolla ohjelmisto rakennettiin, on vapaasti saatavilla internetistä) pelimoottori, joten sitä ei omista yksinoikeudella mikään yritys ja kaikilla Godotin käyttäjillä on mahdollisuus muokata moottoria omiin tarpeisiinsa sopivaksi. (Godot 2021)

Godot on täysin ilmainen ja se tukee sekä kaksi-, että kolmiulotteisten pelien kehittämistä. Virallisella Godotin versiolla pelejä voidaan kehittää tietokoneelle ja puhelimille. Godotilla voidaan myös julkaista pelejä konsoleille kolmansien osapuolien tarjoamilla ohjelmistoilla. Juan Linietsky, toinen Godotin kehittäjistä, kertoi vuonna 2016 peliaiheisessa 80 level -sivuston haastattelussa, että Godotilla on muutamia tuhansia käyttäjiä ympäri maailman, mutta tarkkaa lukua on vaikeaa arvioida. Godotin suosi on kasvanut vuosien aikana ja sitä kehitetään jatkuvasti. Viimeisin vakaa versio moottorista, v3.3.4, julkaistiin 2.10.2021 ja Godot v4.0 on tällä hetkellä kehitteillä. (Linietsky 2016)

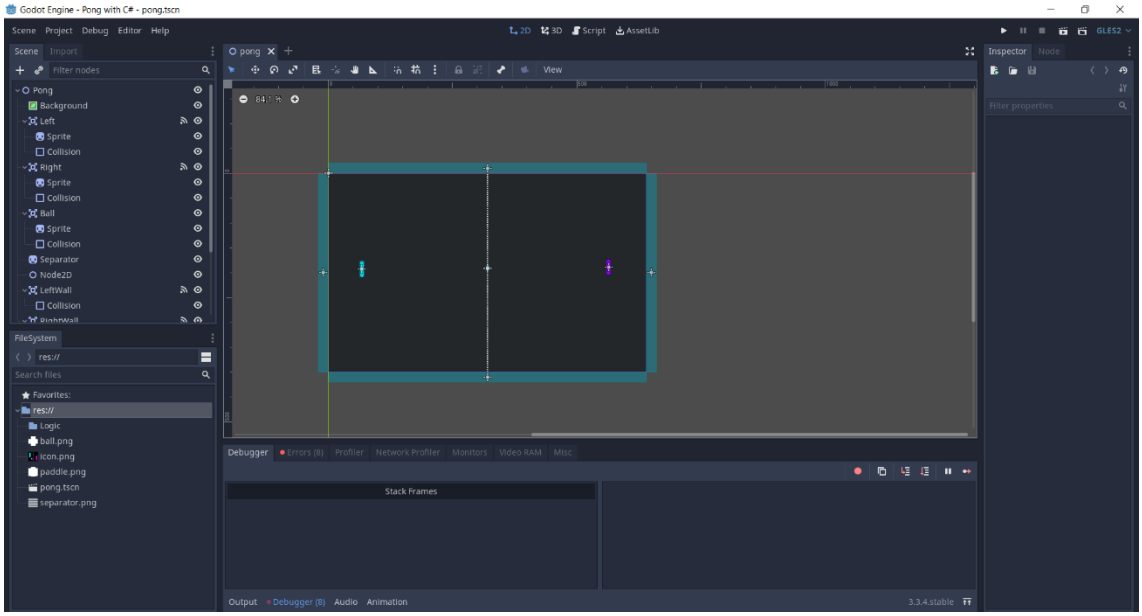
Ohjelmointikielenä virallisessa Godotissa voidaan käyttää Godotin omaa kieltä, GDScriptiä (joka on hyvin samantapainen Python -kielen kanssa), C++:aa ja C#:a. Godotiin on kuitenkin Godot yhteisön toimesta kehitelty myös käyttökieliksi muita kieliä, kuten Rust, Nim, JavaScript, Haskell, Clojure, Swift ja D. Lisäosa voidaan ladata Godot yhteisön Githubista.



Kuva 15: Godot projektipohjia

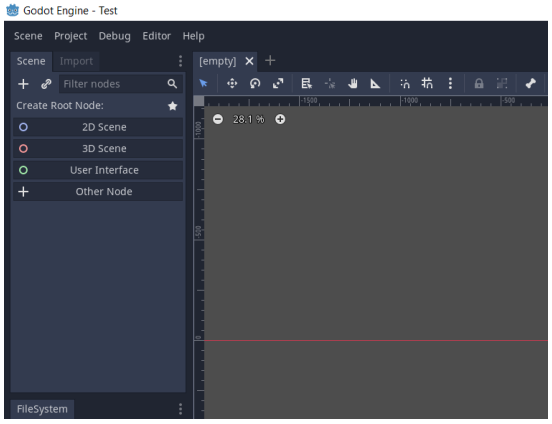
Projektia luotaessa Godotissa, kuten muissakin tämän tutkimuksen pelimoottoreissa, voidaan valita valmiista pohjista. Pohjia voidaan käyttää, jos esimerkiksi halutaan tutkia, miten tietyn tyyppinen peli rakennetaan tai jos halutaan lähteä rakentamaan omaa projektia valmiin

pohjan päälle. Kaikki Godotin tarjoamat pohjat ovat ilmaisia ja kuten kuvassa 15 näkyy, niitä on kattavasti.



Kuva 16: Godot aloitusnäkymä

Kuvassa 16 näkyy aloitusnäkymä, kun projekti avataan Godotissa. Ylhäältä voidaan valita, halutaanko peliä alkaa rakentamaan kaksi- vai kolmiulotteisessa ympäristössä, 2D ja 3D painikkeilla. Ylhäällä löytyy myös ”Script” -nappi, jonne luodaan kaikki peliin liittyvä koodi. Vasemmalla ”Scene” (ts. taso) -välilehden alapuolella ”FileSystem” -kohdassa nähdään mitä kaikkia assetteja tasolla tällä hetkellä on. Klikkaamalla assettia, oikealle ”Inspector” -välilehteen avautuu tarkemmat tiedot assetin ominaisuuksista. Keskellä nähdään peliruutu ja jos sieltä klikataan jotain assettia, niin tämä avaa myös saman ominaisuuslistan oikeaan reunaan. Keskellä alaosasta löytyy työkaluja pelissä ilmenevien ongelmien ja bugien tarkkailuun. Oikeassa yläkulmassa löytyy toistonappi, jolla peliä pääsee pelaamaan.

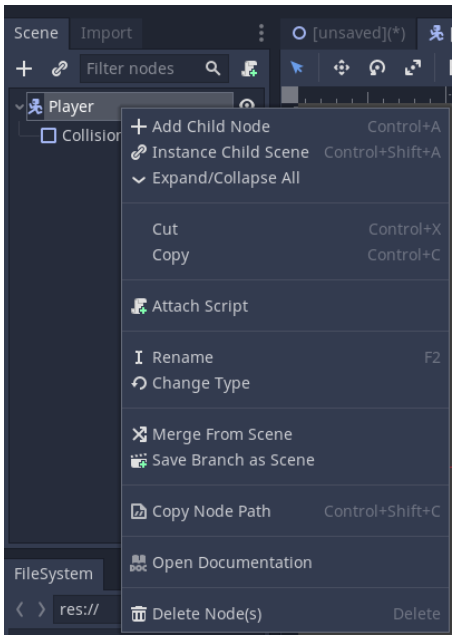


Kuva 17: Tason valinta

Pelin rakentaminen alkaa valitsemalla vasemmalta millainen taso halutaan luoda (Kuva 17). Valintana on kaksi- tai kolmiulotteinen taso ja käyttöliittymä (esim. jos halutaan luoda aloitusvalikko tai paussivalikko peliin). ”Other node” -nappia käytetään, kun halutaan luoda peliin hahmoja, kuten pelaajan hahmo, vihollisia, keräiltäviä esineitä tai esimerkiksi alustoja. Kun taso on luotu ja tallennettu (painamalla CTRL + S-näppäimiä), siihen voidaan alkaa lisäämään hahmoja.

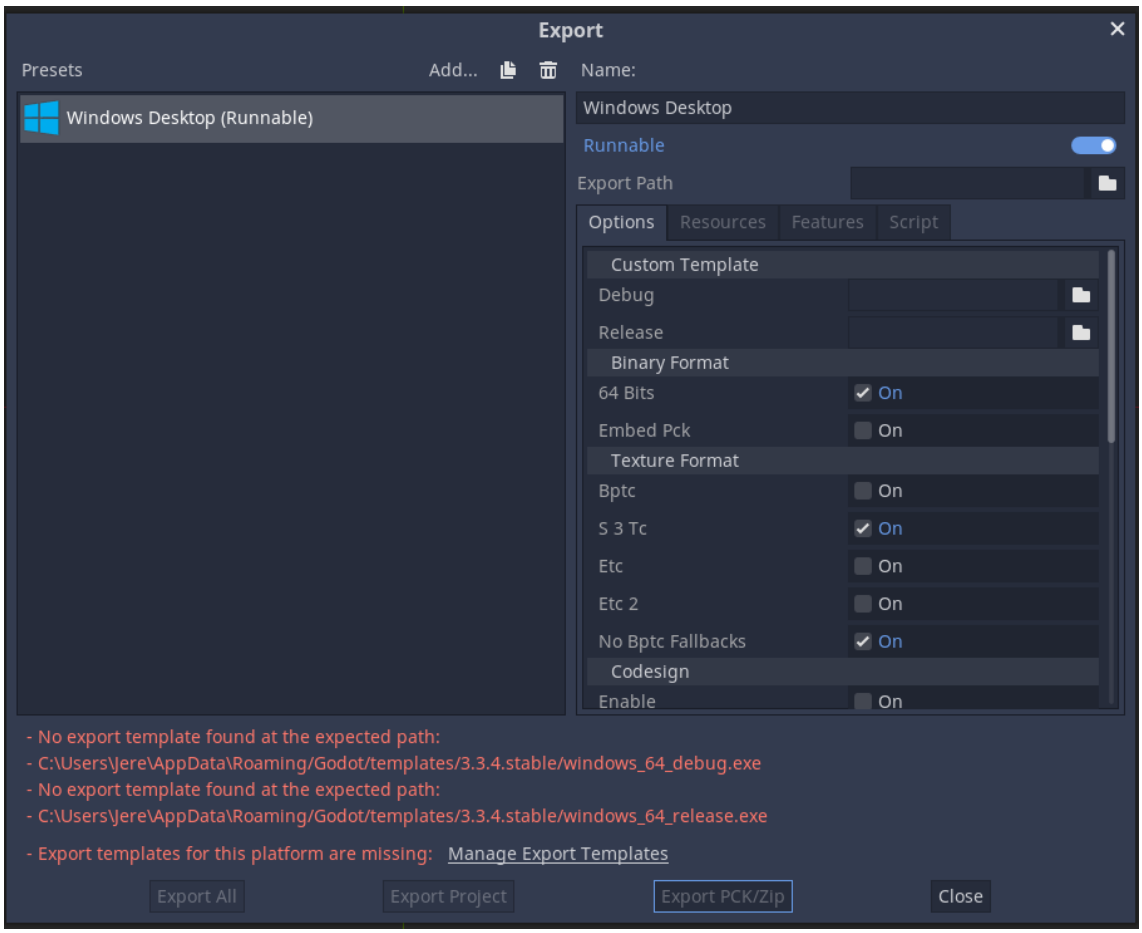
Assettejen lisääminen projektiin tapahtuu avaamalla projektin kansio tietokoneen resurssienhallinnassa ja vetämällä assettikansiot tähän projektikansioon. Projektikansio voidaan avata helposti klikkaamalla ”FileSystem” -välilehdeltä ”res://” -kansiota hiiren oikealla ja klikkaamalla ”Open in File Manager”.

Kun assetit on lisätty ja ne halutaan laittaa peliin, niin sille täytyy ensiksi luoda node. Nodea voidaan pitää kokonaisuutena, joka määrittää millainen assetti on. Noden sisälle määritellään Godotissa kaikki ominaisuudet, joita assetti pitää sisällään. Jos luodaan esimerkiksi node pelaajan hahmolle, niin ensiksi klikataan ”Scene” -näppäintä vasemmasta yläkulmasta ja valitaan ”New Scene”, tämän jälkeen klikataan ”Other node” ja valitaan listasta ”KinematicBody2D”. Tämä nodetyyppi valitaan esimerkissä siitä johtuen, että siihen voidaan lisätä animointia ja fysiikan sääntöjä. Kun tämä on luotu, niin pelaajan hahmo voidaan vetää ruutuun ”FileSystem” -kansiosta. Kaikki pelissä näkyvät hahmot rakennetaan tällä logiikalla. Kuvassa 18 näkyy luotu pelaaja assetti. Ominaisuuksia voidaan lisätä klikkaamalla pääasiallista nodetyyppiä hiiren oikealla ja klikkaamalla ”Add child node”. Hahmon koodi voidaan lisätä oikealta, ”Inspector” -välilehdeltä, ”Script” -kohdan alta.



Kuva 18: Pelaaja node

Kun hahmolle on luotu node, lisätty siihen halutut ominaisuudet ja se on tallennettu, se voidaan lisätä itse tasoon. Tämä tapahtuu siirtymällä tason välilehdelle ja vetämällä node ”FileSystem” -välilehden alta tasoon. Eli kertauksena ensiksi luodaan taso, tämän jälkeen jokaiselle tasoon tulevalle assetille luodaan oma nodensa ja viimeisenä nodet viedään tasoon klikkaamalla, raahaamalla ja pudottamalla.



Kuva 19: Projektin vienti

Kun peli on valmis, se voidaan viedä ulos Godotista helposti klikkaamalla yläpalkista ”Project” ja tämän jälkeen ”Export” (Kuva 19). Godot tarjoaa vakituisena mahdollisuuden viedä peli Windowsille, Macille, Linuxille, Androidille ja iOS:lle. Jokaiseen näistä täytyy ladata ”Export” -työkalu ja tämä tapahtuu klikkaamalla kuvassa 19 näkyvää ”Manage Export Templates” -nappia ja tämän takaa avautuvaa ”Download” -nappia.

5.5 Kahden pelimoottorin valinta vertailuun

Kahden vertailtavan alustan valitseminen neljästä ehdokkaasta ei ollut helppo tehtävä. Loppujen lopuksi kaikki neljä vastasivat enemmän tai vähemmän yrityksen tarpeisiin. Jokainen moottori oli myös joustava ja mahdollistaa peligenrejen sekoittamisen keskenään.

Godot suhteellisen tuoreena pelimoottorina oli positiivinen yllätys joustavuudellaan, sekä sen aktiivisella yhteisöllä. Puhumattakaan siitä, että kaikki moottorissa on ilmaista. Pelimoottoria kehittävä yhteisö vaikutti Githubin, sekä Godotista kertovien YouTube videoiden perusteella hyvin optimistiselta ja pelimoottorilla on arvaten hyvät tulevaisuuden näkymät sen kehittymisen ja suosion kasvun kannalta. Godotin ollessa avoimen lähdekoodin pelimoottori, siihen löytyy paljon lisäosia ja mahdollisuuksia muovata pelimoottorista omalle yritykselle sopiva. Tämä merkkaa kuitenkin sitä, että pelimoottori vaatii paljon opiskelua, että sen käyttömahdollisuudet tulevat kehittäjälle selkeiksi. Pelimoottorina Ghoulish Game kaipaa yhtenäisempää pakettia, kuin mitä Godot tällä hetkellä tarjoaa.

GameMaker Studio 2:sta löytyi hyviä, uniikkeja työkaluja, kuten moottorin sisälle rakennettu piirtotyökalu, joka säästää aikaa ja mahdollistaa hahmojen, sekä pikselitaiteen tekemisen moottorin sisällä. Verrattuna muihin moottoreihin kuitenkin GMS2 käyttöliittymä tuntuu hieman tönköltä. Muut tämän tutkimuksen pelimoottorit noudattavat enemmän tai vähemmän samanlaista logiikkaa käyttöliittymän asettelussa ja pelin rakentamisessa. GMS2 pelin rakentamisen työkalut tuntuvat olevan käyttöliittymässä hajanaisesti. Kehittäjä joutuu avaamaan useita eri ikkunoita ja välilehtiä ja seilaamaan näiden välillä kehityksen aikana, kun taas muut tässä opinnäytetyössä esitellyt moottorit eivät vaadi niin paljon välilehtien tai näkymien vaihtamista. Kaikista näistä moottoreista GameMaker Studio 2 on myös vähiten joustava, kun siirrytään kolmiulotteiseen kehitykseen. Moottorilla on mahdollista tehdä kolmiulotteisia pelejä, mutta tämä on suuremman työn takana muihin moottoreihin verrattuna, koska moottori luotiin alun perin kaksiulotteisille peleille ja sen työkalut eivät ole ”hiottuja” kolmiulotteista kehittämistä varten. Tämän lisäksi GMS2 tarjoaa muihin moottoreihin verrattuna vähemmän toiminnallisuutta kustannuksiin nähden. Kun muissa moottoreissa voidaan ilmaiseksi tehdä pelejä tietokoneelle, GMS2 vaatii maksun, jotta pelejä voidaan julkaista.

Unity on näistä neljästä moottorista tutustumisen jälkeen yksi helpoiten lähestyttävimmistä. Sen etuna ovat suuri käyttäjämäärä ja Unityn yhteisön luomat kattavat ohjeet moottorin käyttöön. Käyttöliittymä oli helposti hahmotettava ja melkeinpä kaikki pelin rakentamiseen tarvittavat työkalut löytyvät samalta sivulta. Mahdollisuudet luoda pelejä kaksi- ja kolmiulotteisesti, sekoittaa näitä elementtejä keskenään, tehdä animointia moottorin sisällä ja julkaista pelejä ilman lisämaksuja heti alusta kaikille alustoille tekevät Unitysta edistyksellisen pelimoottorin. Tärkeänä hyvänä tekijänä on moottorin lisenssimaksujen

suunnittelu. Yrityksen mukana kasvavat lisenssimaksut tarjoavat yritykselle mahdollisuuden kasvaa rauhassa ja Unityn pyytämät maksut pysyvät kohtuullisina.

Unreal Engine yllätti positiivisesti monipuolisuudellaan ja kattavalla työkalupakillaan. Moottorista saa vaikutelman ammattimaisuudesta ja sitä tukee fakta, että moottori on suurien pelitalojen suosiossa. Monipuolisuudestaan huolimatta Unreal Engine onnistuu olemaan helposti lähestyttävä aloittelijoille, koska sen käyttöliittymä on vaivaton ja selkeä. Unreal Enginen valmiit pelipohjat olivat erittäin kattavia ja pelin kehittämisen voi aloittaa hyvin sutjakasti niiden avulla. Moottori on erittäin joustava pelillisten elementtien sekoittamisessa ja kuten Unityssa, mahdollisuus julkaista kaikille alustoille ilman lisämaksuja on todella suuri etu. Epic Gamesin vaatimat rojaltimaksut eivät myöskään ole kohtuuttomia. Unreal on kuitenkin teknisesti raskas pelimoottori, joka syö paljon tietokoneen resursseja. Joten jos kehittäjällä on suunnitelmissa kehittää Unrealilla, tämä täytyy ottaa huomioon ja varmistaa, että kehitysalustana käytettävä tietokone jaksaa pyörittää tätä pelimoottoria.

Näihin huomioihin perustuen vertailun kohteiksi valikoituivat lopulta Unity, sekä Unreal Engine. Molemmat moottorit vaikuttivat ensimmäisellä tutustumisella ammattimaisilta työkaluilta, jotka mahdollistavat todella monipuolisen pelikehittämisen. Molemmista löytyy kattavasti tietoa, niillä voidaan luoda kaksi- ja kolmiulotteisia pelejä, ne tarjoavat ilmaisversion ja tulevaisuuden maksut eivät ole kohtuuttomia, niillä voidaan julkaista pelejä heti alusta helposti useille eri alustoille, sekä niiden käyttöliittymät tuntuvat hyviltä ja helpoilta käyttää. Pelien rakentaminen on näissä kahdessa moottorissa Godotiin ja GMS2:een verrattuna suoraviivaisemman tuntuista ja tämän myötä nopeampaa ensikertalaiselle. Moottoreista ei löytynyt rajoituksia peligenreihin liittyen ja tuleva peliprojekti voidaan toteuttaa kummalla tahansa näistä moottoreista.

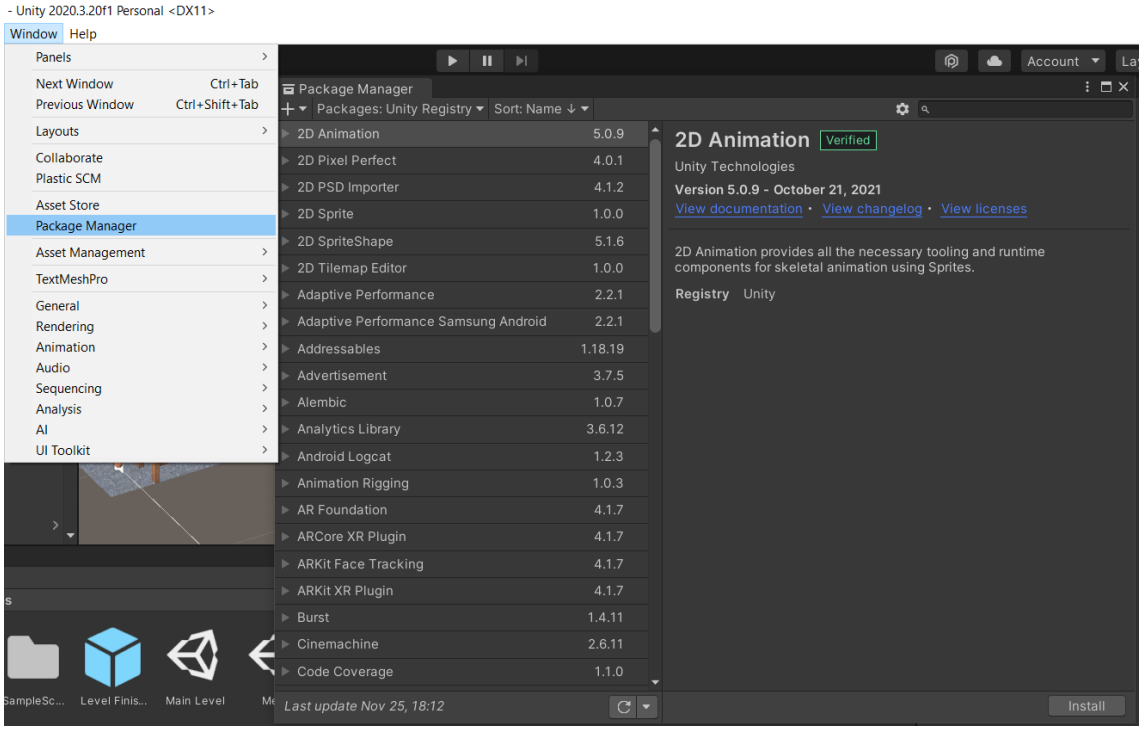
6 Pelimoottoreiden vertailu

Edellä mainituilla, Unitylla ja Unreal Enginellä luotiin testipelit yksinkertaisesta, kolmiulotteisesta pelistä. Pelin tyylilajiksi valikoitui haastava tasohyppely. Molemmat testipelit toteutettiin tasohyppelypelien periaatteella. Tasohyppelypeleistä löytyvistä toiminnallisuuksista luotiin luettelo, jonka avulla selkeytettiin mitä toiminnallisuuksia lähdetään luomaan ja mistä mahdollisesti tarvitsee hakea tietoa. Nämä seuraavat toiminnallisuudet toteutettiin kummassakin pelimoottorissa. Toiminnallisuuksia, joita kolmiulotteiseen tasohyppelypeliin tarvitsee tehdä ovat mm. pelaajan hallitsema hahmo, mahdolliset viholliset tai ansat (ja mitä tapahtuu, kun niihin osuu), keräiltävät esineet, liikkuvat alustat ja haastavat hypyt. Sen lisäksi peliin täytyi luoda käynnistysruutu (ts. menu), pistelaskuri ja voittoruutu, josta peli voidaan lopettaa.

Tässä osiossa käydään läpi, miten testipelin luontiprosessi meni kummallakin moottorilla, mitä haasteita kohdattiin ja paljonko aikaa tiedonhankintaan, sekä eri toiminnallisuuksien kehittämiseen meni.

6.1 Unity testipeli

Unityn testipeli lähti liikkeelle tyhjän projektin luomisesta. Testipeli-ideana oli tehdä kolmiulotteinen tasohyppelypeli, jossa pelaajan hahmo on pyörivä pallo. Tason muoto on pitkulainen suorakulmio ja pelaajan tavoitteena on päästä tämän suorakulmion alusta sen lopussa sijaitsevaan maaliin. Matkalla pelaaja voi kerätä kolikkoja, jotka menevät pelin vasemmassa yläkulmassa näkyvään pistelaskuriin. Ensimmäiseksi täytyi luoda suorakulmion muotoinen alusta, jonka päälle pelin muut objektit kasattiin. Suorakulmion muotoisen alustan luonti kävi kätevästi Unityn valmiiksi tarjoamista kolmiulotteisista muodoista. Ensimmäinen haaste tuli kuitenkin vastaan jo tämän alustan luomisen jälkeen. Suunnitelmana oli tehdä pelaajahahmosta pallon muotoinen, mutta Unity ei tarjoa vakioituna mahdollisuutta luoda kolmiulotteisia palloja. Tästä haasteesta päästiin kuitenkin nopeasti yli, sillä Unityyn pystytään lataamaan lisää työkaluja näppärästi sen sisältä ”Package Manager” -latauskeskuksesta (Kuva 20).



Kuva 20: Unity Package Manager

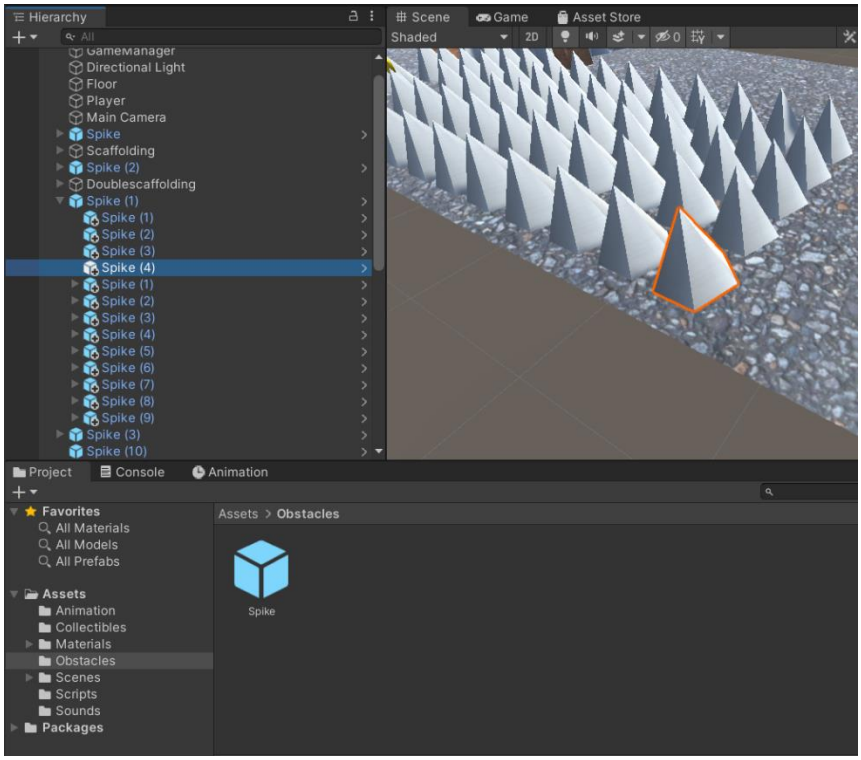
Projektiin ladattiin työkalu ”ProBuilder”, joka mahdollistaa monimutkaisempien kolmiulotteisten muotojen luonnin ja tarjoaa lisää valmiita muotoja, joita voidaan lisätä peliin. ProBuilder -työkalulla luotiin pallo ja seuraavana vaiheena oli ohjelmoida tälle pallolle

säännöt, joilla pelaaja pystyy liikuttelemaan sitä. Ohjelmiston lisääminen pallo -objektiin kävi helposti ”Add component” -napista, Unityn oikeasta reunasta ”Inspector” -välilehdeltä.

Kun pelaajan hahmolle luotiin ohjelmistoa, joka mahdollistaa sen liikuttelun pelissä, kohdattiin muutamia ongelmia. Hahmo liikkui pelissä nykivästi, kun sitä pyöritettiin eteenpäin ja monen hypyn tekeminen peräkkäin ei toiminut kunnolla (hahmo ei reagoinut napin painamiseen muutaman hyppäyksen jälkeen). Nämä olivat enemmän koodissa piileviä ongelmia, kuin itse moottorista johtuvia. Vastaus näihin ongelmiin saatiin nopeasti, tutkimalla internetissä Unitysta kertovia videoita ja Unityn sivuilta löytyviä dokumentteja. Informaation määrä ja saatavuus mahdollisti sen, että koodissa olevat ohjelmointivirheet saatiin suhteellisen vähäisellä työmäärällä korjattua.

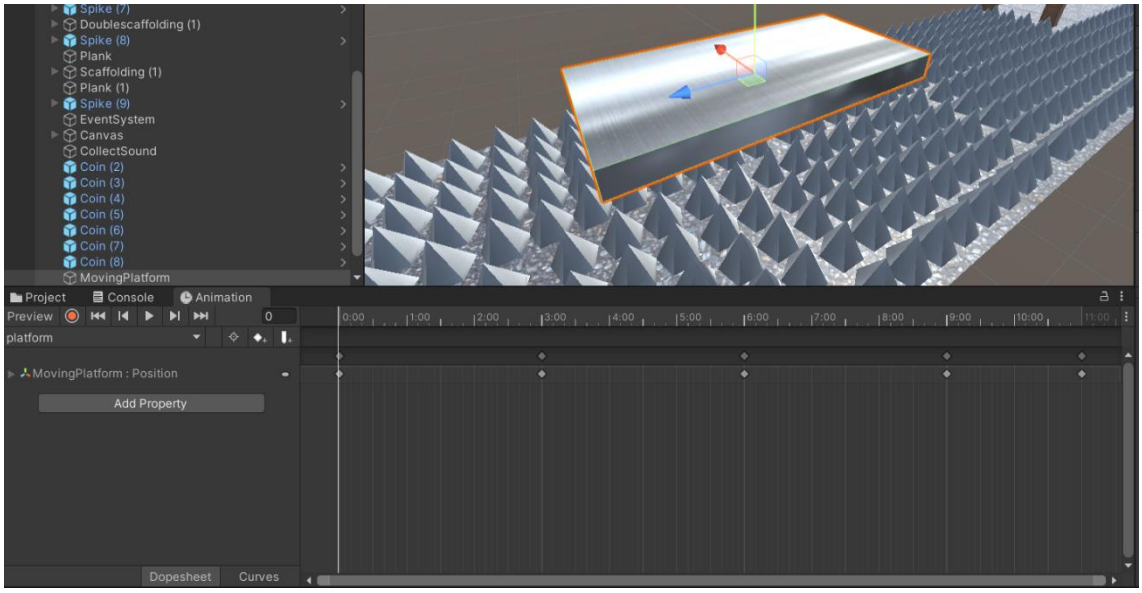
Pelaajan hahmon luomisen jälkeen päästiin luomaan ansoja, eli objekteja, joita koskettamalla pelaaja häviää ja peli alkaa alusta. Tähän väliin mainittakoon, että Unity luo aina jokaiselle uudelle, peliin luodulle kolmiulotteiselle muodolle valmiita fysiikan ominaisuuksia. Esimerkiksi sen, että objektilla on kiinteä pinta, jotta muut objektit eivät voi mennä sen lävitse. Tämä säästää pitkällä juoksulla huomattavasti aikaa kehitystyötä tehdessä. Ansaksi valikoituivat piikit. Projektiin luotiin ensiksi yksi piikki aiemmin ladatulla ProBuilder -työkalulla. Tämä piikki muokattiin pelaajan hahmoon verraten sopivan kokoiseksi oikeasta yläkulmasta löytyvillä skaalaustyökaluilla. Sen jälkeen piikkiin lisättiin koodi, mitä tapahtuu, kun pelaajan hahmo koskettaa piikkiä. Lopuksi piikille lisättiin metallinen tekstuuri, joka oli ladattu netistä ja lisätty projektiin. Tekstuurin lisääminen kävi helposti vain klikkaamalla ja vetämällä tekstuuri piikkiobjektin päälle (drag and drop).

Unitysta löytyi ansan luomisen yhteydessä kätevä ominaisuus. Kun yksi piikki oli täysin valmis, niin se vedettiin peliruudusta Unityn alaosassa sijaitsevaan ”Asset” -osioon. Tämä mahdollisti sen, että ”Asset” -osiosta voitiin nyt vetää loputtomasti valmiita piikkejä peliruutuun. Peliruudussa näkyvät piikit pystyttiin myös ryhmittämään ja näitä piikkiryhmiä voitiin kopioida ctrl + c -painikkeilla ja liittämään ctrl + v -painikkeilla. Näiden työkalujen avulla pelialusta saatiin nopeasti piikkien peittoon ja koodausta vaadittiin loppujen lopuksi vain yhteen niistä. Tämä on erittäin paljon aikaa säästävä toiminto, jota voidaan soveltaa kaikkiin peliin luotaviin objekteihin. Kun peliin luodaan uusia tasoja, niin nämä ”Asset” -osioon vedetyt objektit pysyvät tässä osiossa, joten niitä voidaan kätevästi vetää uusiin tasoihin eikä kaikkea tarvitse aina luoda uudelleen. Kuvassa 21 on assetteihin vedetty piikki, havainnollistamiseksi.



Kuva 21: Unity Asset -osio

Seuraavaksi tuli alustojen luominen, joiden välillä pelaajan täytyisi hyppiä piikkien välttämiseksi. Nämä alustat rakennettiin Unityn valmiilla kolmiulotteisella kuutiomuodolla. Tason loppuosioon luotiin yksi alusta, joka liikkuu edestakaisin ja tällä päästiin testaamaan Unityn animointityökalun käyttämistä. Animointi voitiin aloittaa, kun peliruudussa klikattiin ensiksi alustaa, joka haluttiin liikkuvaksi. Tämän jälkeen siirryttiin Kuvassa 22 näkyvään ”Animation” -välilehteen ja luotiin alustalle animaattori ”Create” -napilla. Jokainen peliruudulla itsekseen liikkuva objekti tarvitsee oman animaattorin, jotta siihen voidaan tehdä animointia.



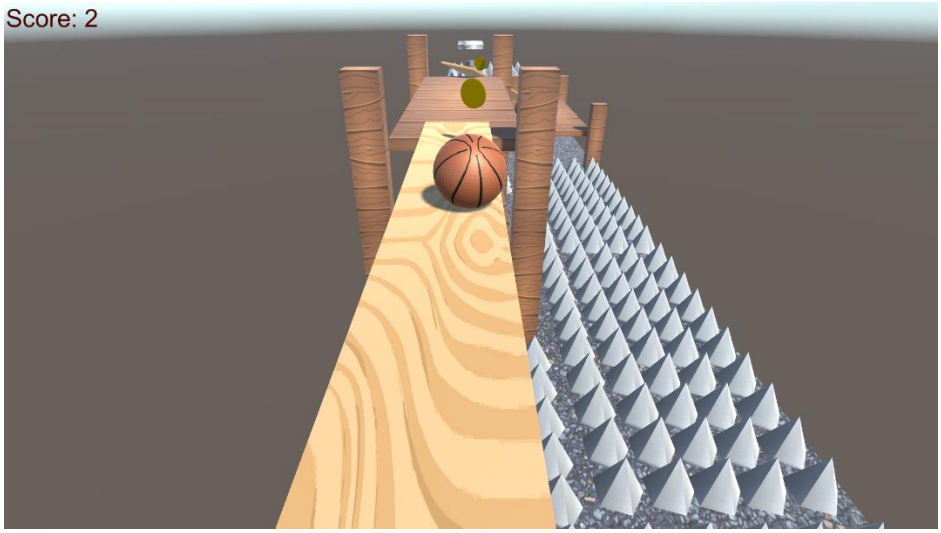
Kuva 22: Unity animointityökalu

Itse alustan animointi kävi helposti painamalla punaista tallennusnäppäintä, jonka jälkeen liikkuva alusta siirrettiin aloituspisteeseen. Tämä loi kuvassa 22 näkyvälle aikajanalle pienen timantin muotoisen väkäsen. Sen jälkeen janalla näkyvä valkoinen viiva siirrettiin seuraavaan kohtaan, alusta liikutettiin eteenpäin seuraavaan sijaintiin ja syntyi toinen väkänen. Kun alusta oli liikutettu alkupisteestä päätepisteeseen ja takaisin aloituspisteeseen, niin animointi oli valmis. Animointia pystyy katselemaan sen rakentamisen aikana painamalla tallennusnäppäimen oikealla puolella olevaa toistonäppäintä. Kokonaisuudessaan animointityökalu oli yksinkertainen, helposti opittava ja kuitenkin pätevä työkalu, joka mahdollisti sen, ettei objektien animoimiseen tarvinnut ladata 3. osapuolen ohjelmia.

Tämän jälkeen luotiin keräiltäviä kolikoita ja pistejärjestelmä. Kolikoiden luontiin käytettiin jo aiemmin läpikäytyjä työkaluja, eikä niiden luomisessa tullut erityisempiä haasteita vastaan. Yhtenä lisänä kolikoihin lisättiin äänitiedosto, jotta päästiin testaamaan äänten lisäämistä projektiin. Äänitiedostolle täytyi luoda näkymätön peliobjekti ja äänitiedosto yhdistää siihen. Tämän jälkeen kyseinen objekti voitiin yhdistää kolikoihin kirjoittamalla niiden ohjelmistoon säännön ääntä varten.

Pistejärjestelmä vaati, että pelin päälle luotiin tyhjä ”kangas”, johon kirjoitettiin pisteet. Haastavinta oli saada pisteteksti oikean kokoiseksi ja sijoitettua oikeaan kohtaan ruutua. Tämän työkalun käyttö tuntui aiempiin verrattuina hieman epäkäytännölliseltä, koska fontin suurentaminen ei esimerkiksi suurentanut automaattisesti tekstille tarkoitettua aluetta kirjainten koon kasvaessa. Työkalussa jouduttiin hienosäätämään paljon ja tarkastelemaan sijoittelua kaksiulotteisessa näkymässä. Pistejärjestelmän ohjelmoimiseen ja sen kommunikointiin pistetekstin kanssa löytyi jälleen internetistä paljon ohjeita, mikä teki

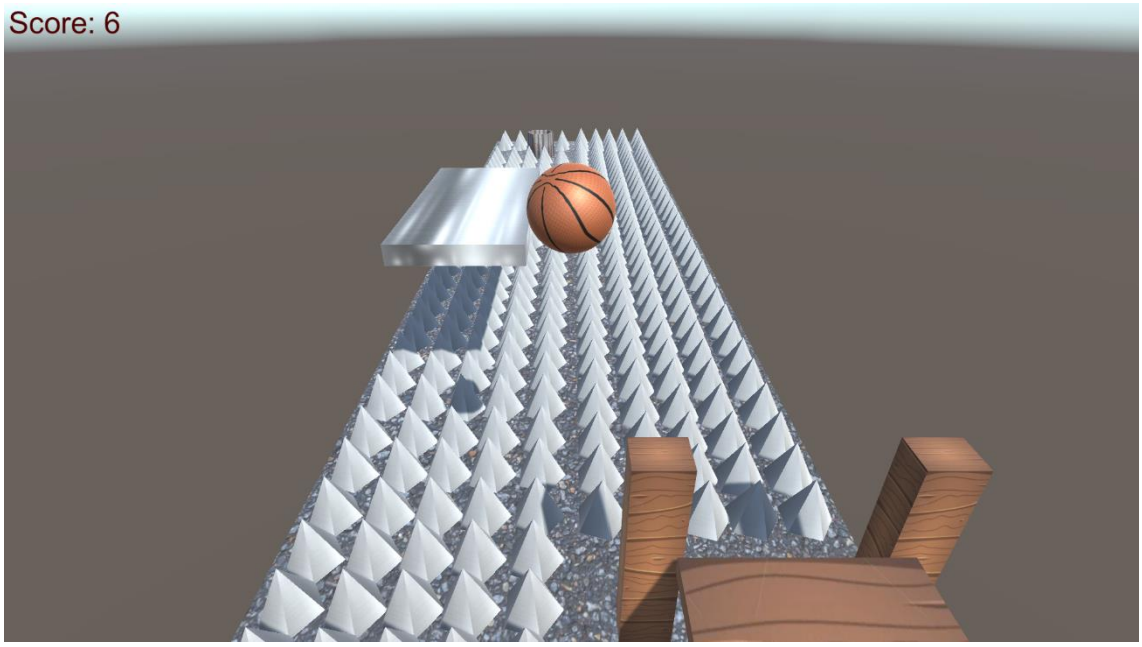
järjestelmän rakentamisesta nopeaa ja helppoa. Pelimoottorin suosion tuomat edut näkyivät selkeästi ohjelmointia ja ohjelmistovirheitä ratkoessa, koska tietoa löytyi niin paljon. Kuvassa 23 on valmiista pelistä otettu kuva, jossa nähdään kolikot ja pistejärjestelmä toiminnassa.



Kuva 23: Otos pelistä

Viimeisinä peliin luotiin taso menu -ruudulle, sekä voittoruudulle. Näille tasoille lisättiin vain kangas samantapaisesti, kuin pistejärjestelmälle, sekä ohjelmistot, jotka kertovat milloin siirrytään millekin tasolle. Tasot järjestettiin lopuksi oikeaan järjestykseen Unityn ”Build” työkalulla ja peli rakennettiin samalla työkalulla valmiiksi. Yhteensä pelin kehittämiseen ja tiedon hankintaan meni aikaa seitsemän tuntia. Alempana muutama kuva valmiista tuotteesta.

Kuva 24: Pelin menuruutu



Kuva 25: Liikkuva alusta

Kuva 26: Pelin lopetusruutu

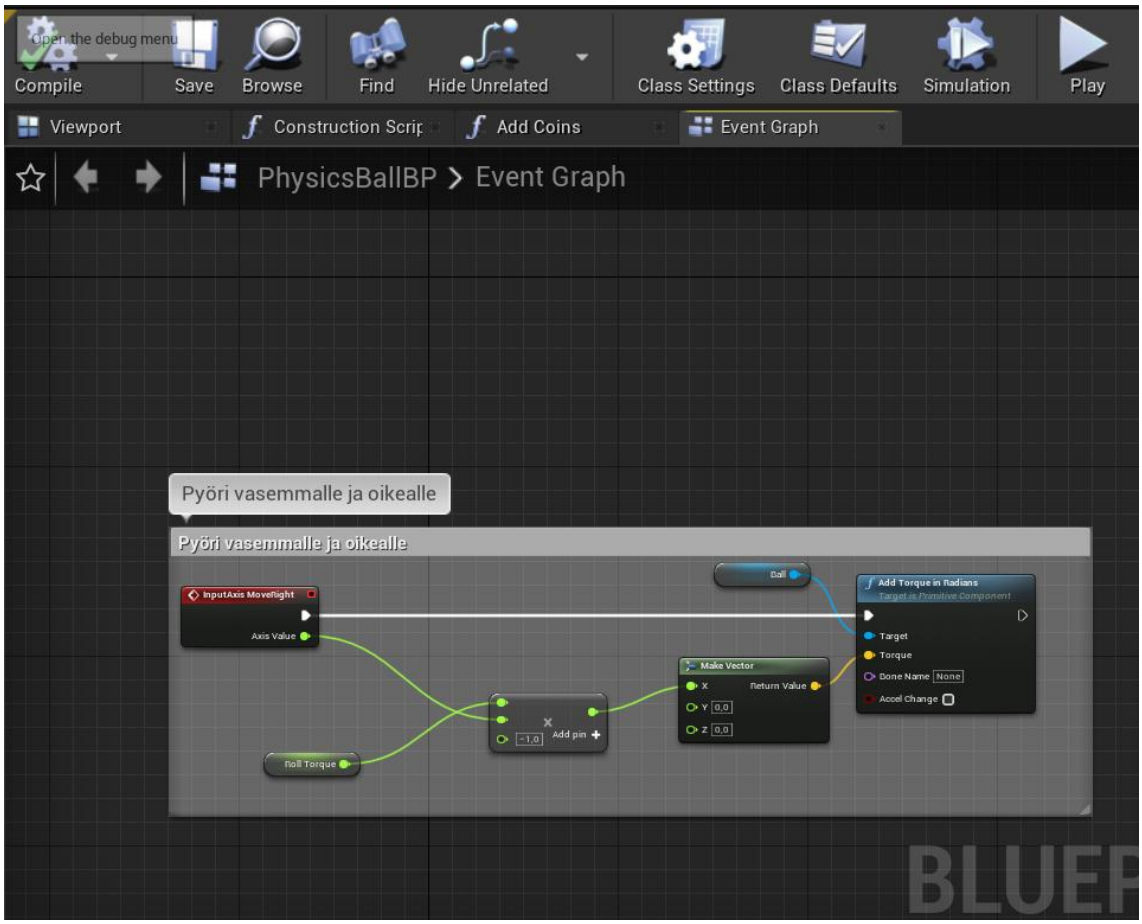
6.2 Unreal Engine testipeli

Samantapaisesti kuin Unityssa, Unreal Enginessa lähdettiin liikkeelle tyhjästä tasosta, johon lisättiin ensimmäiseksi alusta, jonka päälle muut peliobjektit sijoitettiin. Toisin kuin Unityssa, Unreal Engine vaati myös valaistuksen lisäämisen erikseen tasoon, tai muuten kaikki näytti pimeältä. Valaistuksen lisääminen kävi kuitenkin helposti vain vetämällä ja pudottamalla valaistuselementti peliruutuun vasemmasta valikosta. Valaistuksen tehokkuutta pystyttiin helposti säätämään oikealta löytyvällä säätimellä.

Pohjan ja alustan rakentamisen jälkeen luotiin pelaajahahmo. Unreal Engine tarjoaa paljon erilaisia kolmiulotteisia muotoja kehittäjälle, joten moottoriin ei tarvinnut kehityksen aikana ladata lisätyökaluja tai lisäosia uusien muotojen luomiseksi, mikä oli ajan säästämisen kannalta positiivista. Pelaajan hahmoksi rakennettiin pallon muotoinen objekti, kuten Unityssa. Tähän väliin mainittakoon, että projektin alkuasetuksissa voitiin tehdä valinta, tehdäänkö pelin ohjelmointi visuaalisella Blueprint -ohjelmoinnilla, vai perinteisemmällä C++:lla. Projektiin valittiin Blueprint, sillä alustavalla tutkimuksella tämä vaikutti olevan Unreal Engineä käyttävien kehittäjien suosiossa ja tämän myötä tähän ohjelmointitapaan löytyi enemmän vinkkejä ja ohjeita internetistä.

Blueprinttiä käytettäessä Unrealissa avautuu erillinen Blueprint -ikkuna. Sen lisäksi että tässä ikkunassa tehdään itse ohjelmointi, siinä voidaan myös muokata objektin muotoa, lisätä muuttujia, vaihtaa objektin materiaaleja ja valaistusta. Alkuun tämä toiminto vaati opettelua, mutta kun siihen tottui, se osoittautui hyvin käteväksi työkaluksi. Jokaiselle

hieman monimutkaisemmalle (monimutkainen tarkoittaen, että objekti tekee jotain muutakin, kuin on vain paikallaan) objektille luotiin oma Blueprint pelikehityksen aikana. Kuvassa 27 näkyy Blueprint käyttöliittymä, jossa tehtiin ohjainohjelmointia pelaajan hahmolle.

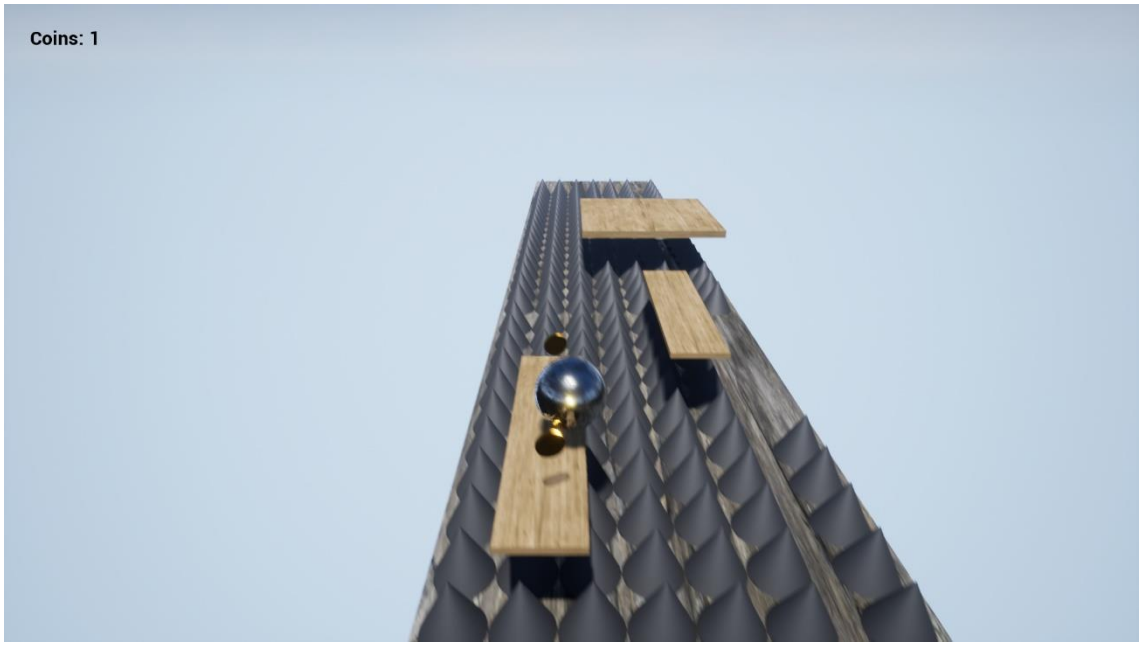


Kuva 27: Blueprint ohjelmointi

Pelaajan hahmon ohjelmoinnin aikana törmättiin muutamaan haasteeseen, sillä hahmon ohjaaminen ei tuntunut vastaanottavan näppäinten painalluksia kunnolla. Erityisesti hyppynapin painamista. Internetistä löytyi kuitenkin nopeasti dokumentaatiota ja hyppiminen saatiin korjattua muutaman muuttujan lisäyksellä. Kun pelaajan hahmo oltiin luotu, seuraavaksi luotiin piikkejä, joita koskettamalla pelaaja häviää pelin ja taso alkaa alusta. Piikkien luomisessa ei kohdattu erityisempiä haasteita ja loogisesti ne toimivat hyvin samalla tavalla, kuin Unityssa.

Peliobjektien asettelu peliin tuntui toimivan hieman paremmin, kuin Unityssa. Tähän vaikuttavana tekijänä toimivat kätevät pikanäppäimet, joilla voitiin vaihtaa nopeasti lennosta esimerkiksi xyz -akselille asettelun ja objektin pyörittämisen välillä. Asettelu tuntui myös hieman tarkemmalta, kuin Unityn puolella.

Seuraavaksi projektiin luotiin keräiltäviä kolikoita ja pistejärjestelmä. Pistejärjestelmä ei aluksi toiminut kunnolla ja näkyi väärässä kohdin ruutua. Pistejärjestelmä on kaksiulotteinen elementti ja tämän asettelu ja yhdistäminen peliruutuun tuntui hieman jäykältä ja vaati enemmän työtä, kuin Unityssa tehty pistelaskuri. Aluksi pistelaskurin Blueprintissä oli jonkinlainen virhe, joka aiheutti sen, ettei peli käynnistynyt ollenkaan. Unreal ei kuitenkaan antanut virhelokia ja Blueprintissa ei näkynyt herjoja, että jokin osa ei toimisi oikein. Pitemmän tutkimisen jälkeen ja Blueprintin uudelleenrakentamisen jälkeen peli alkoi jälleen toimimaan. Virheen syy jäi mysteeriksi. Kolikoita luodessa Unrealista löytyi ominaisuus, joka laittoi kolikot pyörimään itsestään. Tämä säästi mukavasti aikaa animoinnilta. Kuvassa 28 nähdään kolikot ja pistejärjestelmä toiminnassa. Vasemman yläkulman pistelaskuri laskee, montako kolikkoa pelissä on kerätty. Kolikoihin lisättiin Blueprintin puolella äänilaukaisin, joka soittaa Super Mariosta tutun kolikon poimimisäänen, kun kolikko kerätään.



Kuva 28: Kolikot ja laskuri

Testaamisen vuoksi animointityökalua käytiin kuitenkin vilkaisemassa ja kolikoiden animointia testattiin tällä. Animointityökalun käyttöliittymä oli helposti ymmärrettävä, kuten Unityssa, mutta toisin kuin Unityssa, työkalu ei näy suoraan käyttöliittymän aloitussivulla. Sen sijaan animointityökalu löytyi Blueprint ikkunasta.

Viimeisenä komponenttina peliin luotiin menuruutu, sekä voittoruutu. Näille luotiin omat tasot ja ne järjesteltiin projektin asetuksista oikeaan järjestykseen. Ruutujen luomiseen jouduttiin käyttämään enemmän aikaa, kuin Unityssa, koska ne jouduttiin asettamaan ensiksi Blueprintin puolella näkyviksi ja tämän jälkeen niille jouduttiin luomaan omat ohjaimet, jotta hiiri saatiin näkyviin ruutuun. Ei liian vaikeaa, mutta nämä aiheuttivat lisävaiheita ja

enemmän ajankäyttöä yksinkertaiselle toiminnolle. Kuvassa 29 on menuruutu, joka ilmestyy, kun peli käynnistetään ja kuvassa 30 näkyy ruutu, joka ilmestyy, kun peli voitetaan. Kokonaisuudessaan pelin kehittämiseen ja tiedonhakuun Unreal Enginellä kesti 8 tuntia.

Kuva 29: Unreal menuruutu

Kuva 30: Unreal lopetusruutu

6.3 Vertailun lopputulos

Unity ja Unreal Engine olivat molemmat todella hyviä työkaluja pelin rakentamiseen. Projektien toteuttaminen oli hauska kokemus ja kumpaankin moottoriin saatiin projektien aikana hyvä tuntuma. Ei ole ihme, että nämä moottorit ovat suosituimpien pelimoottoreiden joukossa. Niihin rakennetut pelien kehittämistyökalut ovat hyvin pitkälle kehitettyjä ja helposti ymmärrettäviä. Itse moottoreiden sisäisistä, pelin rakentamiseen tarkoitetuista työkaluista ei ole huonoa sanottavaa. Tämä teki paremman moottorin valinnasta huomattavasti haastavampaa, mutta pitkän pohdinnan ja testipelien jälkeen lopputulokseen kuitenkin päästiin.

Unityn käyttöliittymä ansaitsee erityismaininnan sen asettelun ja joustavuuden vuoksi. Kaikki, mitä kehittäjä tarvitsee yksinkertaisen pelin luomiseen, löytyy samalta sivulta. Kun peliä rakennetaan, käyttöliittymä helpottaa hahmottamaan kokonaisuutta ja sitä, miten mikäkin osa vaikuttaa toisen toimintaan. Objektien luominen, ominaisuuksien lisääminen niihin, niiden sijoittaminen pelikartalle ja animoinnin luominen olivat kaikki todella helposti opeteltavia toimintoja. Dokumentaatiota moottorin työkaluista ja Unity yhteisön tekemiä ohjeita moottorin käyttöön löytyi runsaasti internetistä. Kehitystyö sujui tämän myötä sutjakkaasti ja virheet ohjelmistossa saatiin paikattua nopeasti. Ainoa toiminto, jossa oli hieman vikaa itse moottorissa, oli automaattinen valaistus. Jostain syystä automaattinen valaistus meni projektissa ajoittain ”rikki” ja pelin valaistus näytti kummalliselta. Tämä ongelma ei kuitenkaan näkynyt lopputuotteessa sen jälkeen, kun peli oltiin viety ulos moottorista, eli kyseessä oli vain pieni häiriötekijä. Kokonaisuudessaan Unitylla pelikehitys oli hauskaa, helppoa ja nopeaa.

Unreal Engine kuulosti alkuun haastavalta työkalulta, sillä se on isojen pelitalojen käytössä ja sisältää hyvin paljon erilaisia toimintoja, ei pelkästään pelikehitykseen vaan animaatioelokuviin ja kolmiulotteisiin mallinnuksiin. Positiivinen yllätys kuitenkin oli, että pelimoottorin työkalujen käyttäminen oli loppujen lopuksi suhteellisen helppoa ja peruskäytön opetteluun ei mennyt suuria määriä aikaa. Kuten Unityssa, Unreal Enginestä löytyi paljon dokumentaatiota mikä auttoi pelimoottorin opettelussa ja testipelin luomisessa. Kehityksen aikana huomattiin, että monille toiminnoille löytyy monia erilaisia keinoja, miten toiminto voidaan toteuttaa. Tämä johtuu siitä, että Unrealissa on suuri määrä kehitystyökaluja ja ominaisuuksia. Esimerkiksi animointi voidaan toteuttaa muillakin keinoilla, kuin vain animointityökalulla. Objekteihin voidaan lisätä fysiikan ominaisuuksia, jotka saavat objektin liikkumaan, eliminoiden tarpeen erillisen animaation luomiselle. Unrealista löytyi myös esimerkiksi työkalu, jolla voidaan määrittää jokin objekti menemään edestakaisin kahden pisteen välillä ja näin objektista saatiin itsekseen liikkuva eikä animointia vaadittu. Unreal Enginestä huokui kehityksen aikana ammattimaisuus.

Loppujen lopuksi molemmat moottorit ovat todella päteviä, eikä voida sanoa kumpi on toista parempi sataprosenttisesti, sillä molemmilla voidaan luoda hienoja, monipuolisia pelejä. Päätös siitä, kumpi moottori on parempi, riippuu paljolti kehittäjästä/kehittäjistä, kuinka hyvä tietokone kehittäjällä on käytettävissä ja siitä kumman moottorin käyttö tuntuu kehitettävän pelin tyylilajiin sopivammalta vaihtoehdolta.

Unreal Engine tarjoaa pitkällä tähtäimellä enemmän toiminnallisuuksia ja on monipuolisempi työkalu. Kahdeksan työtunnin aikana testipeliä kehitettäessä saatiin vain pintaraapaisu siitä, mihin kaikkeen Unreal -moottorilla pystytään. Testipelien kehittämisen aikana Unity alkoi kuitenkin vaikuttamaan paremmalta vaihtoehdolta muutaman seikan myötä. Unity tuntui henkilökohtaisesti mukavammalta käyttää, pääosin käyttöliittymän helpon ymmärrettävyyden myötä. Unreal Enginessä löytyi kaikki samat toiminnallisuudet, kuin Unityssa, mutta ne tuntuivat olevan useamman klikkauksen takana. Tämä aiheutti sen, että toimintojen etsimiseen jouduttiin käyttämään enemmän aikaa ja moottorin käyttö ei tuntunut yhtä sulavalta, kuin Unityssa. Unityn ymmärrettävyyden myötä, kehitysprojektin aikana ei jouduttu turvautumaan internetistä löytyvään dokumentaatioon niin usein, kuin Unreal Enginellä. Testipelin jälkeen Unityyn jäi hyvä tuntuma ja se, miten työkaluja käytetään, jäi paremmin mieleen kuin Unrealissa.

Kuten aiemmin opinnäytetyössä mainittu, Unreal Engine vaatii kehittäjän tietokoneelta enemmän resursseja. Tämä näkyi myös testipelin aikana, kun jotkin toiminnallisuudet (esimerkiksi peliobjektien koon muuttaminen reaaliajassa) alkoivat toden teolla hidastamaan tietokonetta ja Unreal Engineä. Ongelman voi tietysti ratkaista sillä, että hankkii paremman tietokoneen, mutta jos siihen ei ole mahdollisuutta, niin Unity on Unrealia kevyempi vaihtoehto kehitystyöhön.

Nämä seikat huomioon ottaen, Unity loppupeleissä tuntui henkilökohtaisesti paremmalta vaihtoehdolta. Molemmat moottorit ovat todella hyviä ja päätöksen teko oli vaikeaa, mutta Unrealiin verrattuna Unityn sulavempi käytettävyys ja kevyempi ohjelmisto vetivät Unityn loppujen lopuksi Unrealin edelle. Tämän myötä Unity tulee olemaan käytettävä työkalu Ghoulish Gamesin seuraavassa peliprojektissa.

7 Johtopäätelmä

Opinnäytetyön tavoitteena oli löytää oikea kehitystyökalu opinnäytetyön toteuttajan yritykselle, Ghoulish Gamesille ja tähän tavoitteeseen päästiin. Projektin avulla päästiin kasvattamaan ei ainoastaan omaa tietämystä pelimoottoreista, vaan myös pelikehityksestä yleensä. Pelimoottoreiden ollessa suosittuja ohjelmistoja, niistä löytyi helposti tietoa ja

paljon vinkkejä mihin kannattaa kiinnittää huomiota oikeaa moottoria valittaessa. Tämä helpotti huomattavasti projektin toteuttamista.

Koska nämä ohjelmistot ovat nykypäivänä hyvin pitkälle kehitettyjä, välillä törmättiin haastaviin tilanteisiin moottoreiden karsintaa tehdessä. Oli hetkiä, jolloin täytyi todella pysähtyä miettimään, että mikä pelimoottori on toista parempi ja etenee vertailussa. Rehellisesti sanottuna objektiivisesti toista parempaa moottoria ei ole, vaan kaikki riippuu itse kehittäjästä, hänen tällä hetkellä saatavilla olevista resursseista ja siitä millaista peliä kehitetään. Esimerkiksi jos aiotaan vain kehittää kaksiulotteisia pelejä, niin GameMaker Studio 2 todennäköisesti olisi tähän paras vaihtoehto, vain koska pelimoottori on niin monen vuoden ajan erikoistunut tämän tyyppisten pelien kehittämiseen. Vertailun kohteeksi valitut Unity ja Unreal Engine vaikuttivat kuitenkin kaikista opinnäytetyöstä esitellyistä moottoreista helpoiten lähestyttävimmiltä, eikä niissä ollut rajoituksia pelin tyypin suhteen, vaan ne olivat hyvin joustavia, kokonaisia työkaluja. Tämän takia ne menivät Godotin ja GameMaker Studio 2:en edelle vertailussa.

Projektin toteuttaminen ja testipelien luonti oli hauska kokemus. Lopputulokseen oltiin tyytyväisiä ja valinta, johon päädyttiin, on juuri oikea Ghoulish Gamesille. Opinnäytetyön aikana huomattiin, kuinka hyödyllistä on välillä pysähtyä tutkimaan näitä pelikehityksen työkaluja, koska siinä oppii hyvin paljon moottoreiden lisäksi pelikehityksestä. Tämän myötä Unity ei tule jäämään ainoaksi käytettäväksi pelimoottoriksi Ghoulish Gamesille ja uusiin moottoreihin tullaan tutustumaan tulevaisuuden peliprojekteissa.

Lähteet

Sähköiset

Charbonnau, M. 2013. Choosing the perfect Game Engine. Viitattu 22.10.2021. https://www.gamedeveloper.com/design/choosing-the-perfect-game-engine

Concept Art Empire. Petty, J. 2021. Top 12 Free Game Engines For Beginners & Experts Alike. https://conceptartempire.com/free-game-engines/

Developer House. 2021. TOP 10 BEST GAME ENGINE FOR BEGINNERS IN 2021. https://developerhouse.com/top-10-best-game-engine-for-beginners/

GameAnalytics. Greene, K. 2021. Top 5 Game Engines For Beginners. https://gameanalytics.com/blog/top-5-game-engines-for-beginners/

Game Designing. 2021. The Best Video Game Design Engines. https://www.gamedesigning.org/career/video-game-engines/

GameMaker Studio. 2021. Viitattu 29.10.2021 https://www.yoyogames.com/

Godot. 2021. Viitattu 2.11.2021 https://godotengine.org/

Kronberg, Y. 2017. How to Choose a Game Engine. Viitattu 22.10.2021. https://www.linkedin.com/pulse/how-choose-game-engine-yann-kronberg

Linietsky, J. 2016. Godot 2.0: Talking with the Creator. Viitattu 22.10.2021. https://80.lv/articles/godot2-interview/

Makuch, E. 2021. Unreal Engine 5 - Devs On How The Engine Will Transform Next-Gen Games. Viitattu 27.10.2021. https://www.gamespot.com/articles/unreal-engine-5-devs-on-how-the-engine-will-transform-next-gen-games/1100-6492100/

Ojasalo, K., Moilanen, T. & Ritakoski, J. 2014. Kehittämistyön menetelmät. Uudenlaista osaamista liiketoimintaan. E-kirja. Helsinki: Sanoma Pro, 78-79.

Rădulescu, R. 2020. Choosing the right game engine. Viitattu 22.10.2021. https://www.gdquest.com/tutorial/getting-started/learn-to/choosing-a-game-engine/

The Tool. 2021. The Best 20 iOS and Android Mobile Game Engines + Development Platforms & Tools in 2021. https://thetool.io/2018/mobile-game-development-platforms

Unity. 2021. Viitattu 25.10.2021. https://unity.com/

Unreal Engine. 2021. Viitattu 27.10.2021. https://www.unrealengine.com/en-US/

Taulukot

Taulukko 1: Yrityksen tarpeiden ja pelimoottoreiden ominaisuuksien vertailu 10