OULUN AMMATTIKORKEAKOULU

Arttu Heikkinen

Mobiilipeliin Perustuvan PC-Pelin Kehitys

Mobiilipeliin Perustuvan PC-Pelin Kehitys

Arttu Heikkinen
Opinnäytetyö
Kevät 2023
Tietojenkäsittely
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittely

Tekijä: Arttu Heikkinen
Opinnäytetyön nimi: Mobiilipeliin Perustuvan PC-Pelin Kehitys
Työn ohjaaja: Matti Viitala
Työn valmistumislukukausi ja -vuosi: Kevätlukukausi 2023
Sivumäärä: 35 + 5 liitettä

Opinnäytetyön tavoitteena oli luoda yksinkertaisen mobiilipelin pohjalta monimutkaisempi PC-pelidemo sekä tutustua PC-pelien kehitykseen. Peli toteutettiin Unity pelimoottorilla, C#-ohjelmointikielellä ja Visual Studio Codella.

Työssä tutustutaan Unity-pelimoottoriin sekä sen kilpailijoihin. Pelissä käytetyt grafiikat ja äänet hankittiin Unity Asset Storesta. Projektissa versiohallintaan käytettiin GitHubia.

Raportissa käsittelen PC- ja mobiilipelien eroja ja eri alustojen hyviä ja huonoja puolia. Kerron eri pelimoottoreista ja työkaluista, joita tarvitaan pelikehityksessä. Lisäksi käyn läpi eri vaiheet, jotka kuuluvat pelikehitykseen. Kerron pelin suunnittelusta, kehityksestä sekä jatkokehityksestä.

Asiasanat: pelikehitys, pelisuunnittelu, peliohjelmointi, ohjelmointi

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems

Author: Arttu Heikkinen
Title of thesis: Development of a PC Game Based on a Mobile Game
Supervisor(s): Matti Viitala
Term and year when the thesis was submitted: Spring 2023
Number of pages: 35 + 5 appendices

The goal of this thesis was to develop a demo of a PC game based on a casual mobile game and to learn more about the development of PC games. The game was developed using Unity game engine, C#, and Visual Studio Code.

In the report we get familiar with Unity game engine and its competitors. All the graphics, textures and sounds used in the game are from Unity Asset Store. For version control I used GitHub.

In the report I go through the differences between PC and mobile games and talk about what is great about each platform. I will tell you about the different tools and engines needed for game development. I also talk about different stages that are part of game development from planning all the way to further development.

Keywords: game development, game design, game programming, programming

SISÄLLYS

1 JOHDANTO 6
2 PELIALUSTOJEN VERTAILUA 7
2.1 Suosituimmat alustat 7
2.2 Pelien siirtäminen uusille alustoille 8
3 PELIMOOTTORIT 10
3.1 Suosituimmat pelimoottorit 10
3.2 Unity 10
3.2.1 Unityn editorin osat 11
3.2.2 Unityn käyttö ja ohjelmointi 12
3.2.3 Unity Asset Store 13
4 TYÖKALUT 14
4.1 Visual Studio Code 14
4.2 GitHub 14
5 SUUNNITTELU 15
5.1 Purrrfect 15
5.2 PC-versio 16
6 PELIKEHITYS 18
6.1 Kehityksen aloitus 18
6.2 Maailman luonti 19
6.3 Pelihahmo 22
6.4 Eläinten generointi maailmaan 23
6.5 Eläinten tekoäly 24
6.6 Eläinten silittäminen 25
6.7 Animaatio 25
6.8 Silitys 28
6.9 Eläinten generoiminen pelaajan saarelle 29
7 JATKOKEHITYS 30
8 POHDINTA 31
LÄHTEET 32
LIITTEET 35

1 JOHDANTO

Idea opinnäytetyöhön kehittyi halustani oppia lisää pelikehityksestä ja ohjelmoinnista yleensä. Tavoitteenani opinnäytteen tekemisen aikana oli tutustua PC-pelien kehitykseen aiemman mobiilipelikehityskokemuksen pohjalta ja kehittyä C#-ohjelmoinnissa.

Raportissa tutustutaan PC- ja mobiilipeleihin, pelikehityksessä käytettyihin työkaluihin, sekä eri vaiheisiin, joita kuuluu pelin tekemiseen. Käyn raportissa läpi pelikehityksen eri vaiheet suunnittelusta aina jatkokehitykseen saakka.

Ennen opinnäytetyön aloittamista olin harjoitellut Unityn käyttämistä itsenäisesti, ja olin mukana Oulu Game Labsin koulutusohjelmassa, jonka aikana teimme ryhmässä Purrrfect-mobiilipelin. Opinnäytetyön aiheen keksin Purrrfectista, ja työssä tekemäni peli pohjautuu siihen. Purrrfectissa keskityin lähinnä ohjelmointiin ja hieman käyttöliittymään liittyviin asioihin. Edellisessä peliprojektissani mukana oli paljon muitakin henkilöitä, joten tämä projekti on ensimmäinen, jossa teen kaiken itsenäisesti. Raportin kirjoittamisen aikana hyödynsin paljon YouTube-videoita ja keskustelufoorumeita haastavien tilanteiden ja ongelmien ratkaisemisessa.

2 PELIALUSTOJEN VERTAILUA

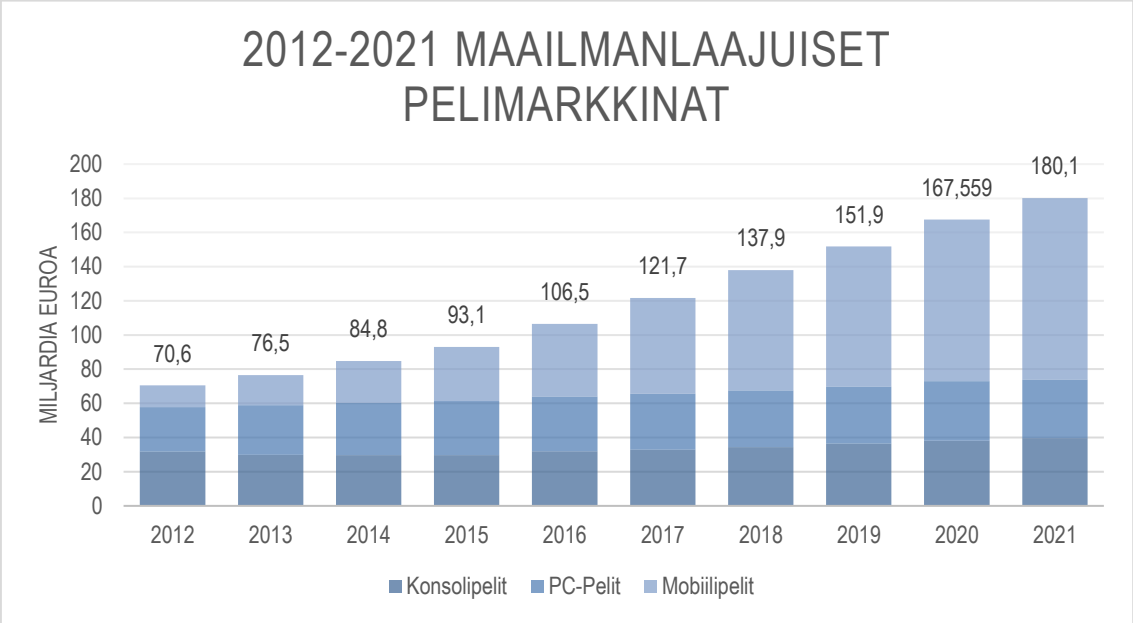
Nykyään jokaiselle alustalle on tarjolla toinen toistaan kiehtovampia videopelejä, ja pelaajat voivat valita usein vapaasti, millä laitteella he haluavat pelata suosikkipeliään. Mobiililaitteilla voi pelata yksinkertaisia ja koukuttavia pelejä missä ja milloin tahansa, kun taas PC- ja konsolipelit tarjoavat pelaajille syvällisempää viihdettä, joka voi lumota pelaajat pelimaailmaan tuntikausiksi. Mobiilipelejä pelataan monesti ajanvietteenä, joten ne ovat usein yksinkertaisia ja helposti saatavilla. Mobiilipelejä voi ladata maksutta, mutta suurin osa mobiilipeleistä sisältää sovelluksen sisäisiä ostoksia. (starloopstudios.com 2022.)

Tehokkaat tietokoneet ja konsolit mahdollistavat laadukkaamman näköisiä pelejä, joissa voi olla todella suuria ja mukaansatempaavia maailmoja. Lisäksi PC- ja konsolipelien immersio paranee, kun niitä pelataan kuulokkeiden ja suurten näyttöjen kanssa, kun taas mobiilipelejä pelatessa käytössä on yleensä vain pieni näyttö.

PC- ja konsolipeleihin voidaan kehittää monipuolisemmat ja helpommat kontrollit kuin mobiilipeleihin, koska käytössä on muitakin resursseja kuin pelkkä kosketusnäyttö. Toisaalta mobiilipeleihin voidaan implementoida pelimekaniikkoja, jotka eivät olisi mahdollisia PC:llä tai konsolilla. Mobiilipeleissä voidaan hyödyntää esimerkiksi pelaajan sijaintia ja puhelimen kameraa. Lisäksi kosketusnäytölle voidaan luoda kontrolleja, jotka eivät välttämättä olisi mahdollisia PC:llä tai konsolilla. Esimerkiksi Pokémon GO -pelissä hyödynnetään mobiililaitteen komponentteja, joiden käyttö ei välttämättä onnistuisi muilla laitteilla. PC- ja konsolipelit on siis tarkoitettu pelaajalle, joka etsii mukaansatempaavia ja monipuolisia pelejä, kun taas mobiilipelit ovat satunnaiselle pelaajalle, joka haluaa pelata pelejä ajankuluna.

2.1 Suosituimmat alustat

Mobiilipelien suosio on ollut hurjassa nousussa viimeisten vuosien ajan. Mobiilipelit olivat aluksi todella pieni osa maailmanlaajuisia pelimarkkinoita, mutta tällä hetkellä noin puolet kaikista pelialan tuloista menee mobiilipeleille (giffgaff.com 2018). Vuonna 2012 vain 18 % pelialan tuloista kuului mobiilipeleille, mutta jo vuonna 2016 se oli johtava pelialusta (kuvio 1).



KUVIO 1. 2012–2021 maailmanlaajuiset pelimarkkinat (giffgaff.com 2018, muokattu)

Mobiilipelien suosioon vaikuttavat kehittyneet mobiililaitteet, jotka jaksavat pyörittää jatkuvasti raskaampaa sisältöä. Kehittyneiden laitteiden ja suurten käyttäjämäärien takia tunnetut pelifirmat ovat alkaneet julkaisemaan enemmän mobiilipelejä. Sony, Electronic Arts, Microsoft, Nintendo ja Activision ovat hyviä esimerkkejä firmoista, jotka ovat levittäneet pelejään myös mobiililaitteille (builtin.com 2023). Lisäksi yhä useammalla on mahdollisuus käyttää mobiilipuhelimia, joten mobiilipelit saavat jatkuvasti uusia käyttäjiä. Mobiilipelien hankkiminen on paljon edullisempaa kuin PC- tai konsolipelien, joten pelaajilla ei ole niin suurta kynnystä mobiilipelien asentamiseen. (medium.com 2022.)

2.2 Pelien siirtäminen uusille alustoille

Monet PC- ja konsolipelifirmat ovat hiljattain alkaneet siirtymään mobiilialustoille saavuttaen täten edelleen suuremman yleisön. (medium.com 2022.) Mobiilipuhelimille on nykyään tarjolla vanhempia PC- ja konsolipelejä, joista on tehty mobiililaitteille sopivat versiot. Nopeasti kehittyvien laitteistojen ansiosta myös moderneja PC- ja konsolipelejä on saatavilla mobiililaitteille koko ajan enemmän. PlayerUnknown's Battlegrounds ja Call of Duty ovat hyviä esimerkkejä suuren budjetin peleistä, jotka ovat nyt saatavilla mobiililaitteille. (medium.com 2021.)

Pelikehittäjillä on muutamia eri taktiikoita siirryttäessä PC:ltä mobiilimarkkinoille. Esimerkiksi Minecraftin PC-versio on lähes identtinen mobiiliversioon (lifewire.com 2021). Minecraftin tekijät ovat siis julkaisseet melkein saman pelin usealle eri alustalle. Tämä on toimiva ratkaisu etenkin vanhempien pelien kohdalla, mutta raskaampien pelien kanssa tulee toimia toisten.

Tietokoneelle alun perin suunniteltu Fallout-pelisarja sai ensimmäisen mobiililla toimivan pelin vuonna 2015, kun Bethesda Softworks julkaisi Fallout Shelterin (fallout.fandom.com 2022). Peli sijoittuu tuttuun Fallout-maailmaan, mutta se on suunniteltu toimimaan mobiililaitteilla, joten peli on sitä edeltäviä pelejä kevyempi ja paljon yksinkertaisempi. Fallout Shelterin pelimoottorina toimii Unity (youtube.com 2015).

Mobiilille julkaistaan jatkuvasti uusia pelejä, mutta toisaalta moni pelikehittäjä haluaa luoda pelejä sekä PC:lle että konsolille, koska silloin pelit voivat olla monipuolisempia. Mobiilipelit saattavat olla tällä hetkellä suosituimpia, mutta se ei tarkoita sitä, etteikö PC- ja konsolipelien teko olisi kannattavaa. Mobiilipelejä julkaistaan todella paljon, joten menestyksekkään mobiilipelin luonti on vaikeaa. Pelkästään vuonna 2021 Google Play Kauppaan julkaistuun yli 200 000 mobiilipeliä (statista.com 2022).

Alto’s Adventure on esimerkki pelistä, josta oli aluksi ainoastaan mobiiliversio, mutta nykyään se on saatavilla lähes kaikilla alustoilla. Alto’s Adventure on itsenäisen Snowman-pelistudion kehittämä ja julkaisema päättymätön juoksupeli. Pelin alkuperäinen julkaisu oli helmikuussa 2015 iPhonelle. (ign.com 2015.) Peli oli yhtiön ensimmäinen menestynyt julkaisu, ja suuren suosion vuoksi pelistä tuli Android-, Windows- ja Mac versiot. Yhtiö on julkaissut Alto’s Adventuren jälkeen useita pelejä lukuisille alustoille, kuten PC:lle, konsoleille ja mobiilille (builtbysnowman.com 2023).

Rovio Entertainment on suomalainen yhtiö, joka tuli tunnetuksi Angry Birds -pelin avulla 2010-luvulle tultaessa. Rovio Entertainment on hyvä esimerkki yhtiöstä, jonka suuri suosio pohjautuu mobiilipeliin. Rovio sai alkunsa vuonna 2003, mutta se tuli tunnetuksi vuonna 2009, jolloin yhtiö julkaisi ensimmäisen Angry Birds -pelin. Rovio teki vuonna 2009 lähinnä mobiilipelejä, mutta suuren kysynnän vuoksi yhtiö on laajentanut tarjontaansa myös muille alustoille. Yhtiön kehittämistä mobiilipeleistä on tullut PC-versioita, ja Angry Birds -hahmojen pohjalta on tehty tv-sarjoja ja elokuvia. (rovio.com 2023.)

3 PELIMOOTTORIT

Pelimoottorit ovat ohjelmia, joiden käyttö keskittyy pääsääntöisesti pelien kehitykseen. Pelimoottoreiden päätoimintoihin kuuluvat 2D- tai 3D-pelimaailmojen ja hahmojen muokkaus, fysiikkamoottori, animaatio, ääni, sekä tekoälyn toiminnot. (fullscale.io 2021)

3.1 Suosituimmat pelimoottorit

Pelimoottoreita on nykyään tarjolla lukuisia, mutta suosituimmat vaihtoehdot ovat ehdottomasti Unity ja Unreal Engine. Molemmat pelimoottorit ovat suhteellisen samankaltaisia, joten jommankumman valinta omaan käyttöön saattaa olla haastavaa. Unity tarjoaa suuremman määrän mahdollisia alustoja, joille pelejä voidaan luoda, ja sitä pidetään parempana vaihtoehtona 2D- sekä mobiilipelien luomiseen. Unreal Engine on taas kehittyneempi pelien visuaalisessa puolessa, joten se on hyvä valinta etenkin realististen 3D-pelien luomiseen. Unreal Engine on suositumpi vaihtoehto isojen pelien luomiseen, kun taas Unity on suositumpi indie-pelikehittäjien keskuudessa. (gamedevacademy.org 2022) Loppujen lopuksi molemmilla pelimoottoreilla voidaan luoda laadukkaita 2D-, 3D- ja VR-pelejä, ja oikean pelimoottorin valinta on monesti käyttäjälähtöinen valinta.

Moni pelistudio käyttää pelien tekemiseen heidän omia pelimoottoreitaan. Esimerkiksi Nintendo käyttää omaa pelimoottoria pelien tekemiseen. Nintendo Switch tukee Unreal Engineä ja Unityä, jotta pelikehittäjät voivat julkaista omia pelejään. (perforce.com 2020)

3.2 Unity

Valitsin raportissa käytettäväksi pelimoottoriksi Unityn, koska käytimme sitä mobiilipelissä, josta sain idean tähän opinnäytetyöhön. Unity on myös minulle kaikista tutuin pelimoottori. Unity on vuonna 2005 julkaistu pelimoottori, jonka avulla voidaan kehittää sisältöä lukuisille alustoille. Unity on hyvä työkalu sekä suurille pelifirmoille että opiskelijoille, jotka ovat kiinnostuneet ohjelmoinnista. Unityssä ohjelmointi tapahtuu C#-kielellä. (educationecosystem.com 2022.)

Vuonna 2021 Unityllä luotuja sovelluksia ladattiin joka kuukausi yli viisi miljardia kertaa ja 70 % tuhannesta suosituimmasta mobiilipelistä oli Unityllä rakennettuja. Unityllä tehdyt luomukset toimivat yli kahdellakymmenellä alustalla. (unity.com 2023c.)

Unityä ei käytetä yksinomaan pelimoottorina, sillä sen monipuolisilla työkaluilla voidaan luoda laajalti eri medioita. Unityn sekä muiden pelimoottoreiden käyttö on yleistynyt eri aloilla, kuten arkkitehtuurissa ja elokuvateollisuudessa. Unityn avulla voidaan luoda näyttäviä animaatioita, ja sitä käytetään jopa elokuvien kohtausten esirenderöinnissä. Unity on suunniteltu suurten datamäärien käsittelyyn, joten sitä voidaan käyttää arkkitehtuurimallien luomiseen. Unityn avulla voidaan helposti simuloida todentuntoisia tilanteita rakennusmalleissa ennen rakennustöiden alkamista. (transforminteractive.com 2022.)

3.2.1 Unityn editorin osat

Unityn editorin käyttöön tottuu nopeasti, koska siinä olevat osat ovat yksinkertaisia. Unityn editori muistuttaa tavanomaista kuvankäsittelyohjelmaa (Liite 1). Unity-editori voidaan jakaa neljään eri osaan, jotka ovat Hierarchy, Scene, Inspector ja Project. Hierarkianäkymässä (Hierarchy) voi hallita kaikkia pelissä olevia objekteja ja sen avulla lisätään pelimaailmaan uusia asioita. Kun hierarkianäkymässä valitaan peliobjekti, avautuu Inspector-näkymä. Inspector-näkymässä voi muokata peliobjektien paikkaa ja lisätä niihin komponentteja, kuten scriptejä. Inspector-näkymän avulla voidaan myös säätää peliobjektien asetuksia.

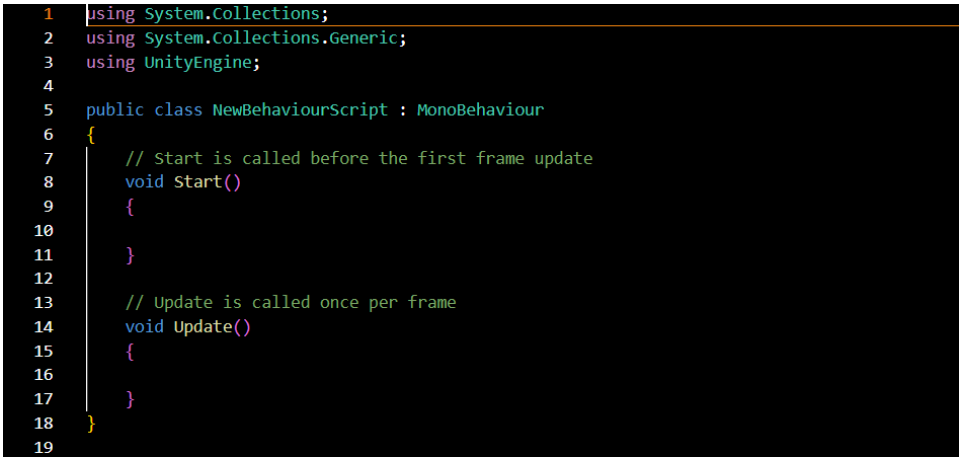
Scene-näkymän avulla voi tutkia ja muokata pelimaailmaa. Scene-näkymän avulla voidaan myös käynnistää valittuna oleva scene ja tutkia peliä sen ollessa päällä. Opinnäytetössä luomani peli sijoittuu kokonaan yhteen sceneen, mutta monipuolisemmissa peleissä voi olla useita scenejä. Esimerkiksi pelin kenttä voi olla yksittäinen scene.

Viimeinen Unity-editorin osa on Project-osio. Sen avulla voi tarkastella kaikkia projektissa olevia tiedostoja. Project-osion kautta voi lisätä hierarkianäkymään peliobjekteja. Osiosta löytyy myös tekstikonsoli, jota käytetään ongelmien ilmestyessä niiden paikantamiseen.

3.2.2 Unityn käyttö ja ohjelmointi

Ensimmäinen vaihe Unityn käyttöönotossa on Unity Hubin asennus. Unity Hubin kautta voidaan hallita Unityn editorin versioita, sekä luoda ja hallita projekteja. Unity Hubista löytyy erilaisia oppimisreittejä, joiden avulla voi helposti oppia Unityn käyttöä. (unity.com 2023b)

Toiminnallisuuksien luominen objekteille vaatii ohjelmointia C#-ohjelmointikielellä (kuvio 2). Unityssä ohjelmointi suoritetaan vapaavalintaisella koodieditorilla ja koodit lisätään projektiin skriptitiedostoina. Unityn käyttäjien ei tarvitse itse luoda pohjaa reaaliajassa piirrettävällä interaktiiviselle 3D-sisällölle. Unityssä tarvittava ohjelmointi keskittyy pitkälti erilaisten pelitoimintojen luomiseen. (unity.com 2023a)



KUVIO 2. Tavanomainen C#-tiedosto Visual Studio Codessa

C#-ohjelmoinnissa käytetään erilaisia luokkia, jotka helpottavat ohjelmointia. Luokkien käyttö määritellään heti C#-tiedoston alussa. Luokkien avulla ohjelmointi nopeutuu, koska toiminnallisuudet välittyvät luokkien välisen perinnän ansiosta. Tämä tarkoittaa sitä, että kirjoitettavaa on vähemmän.

MonoBehaviour on perusluokka, josta saadaan kaikki Unityn komentosarjat. MonoBehaviour luo Start- ja Update-funktiot automaattisesti, kun C#-tiedosto luodaan. Funktioita kutsutaan pelin aikana eri ajanjaksoina. Start-funktiota kutsutaan aina, kun peli käynnistetään tai kun pelimaailma latautuu uudestaan. Update-funktiota kutsutaan jokaisen framen eli kuvaruudun aikana. MonoBehaviour sisältää myös muita elinkaarifunktioita Start- ja Update-funktioiden lisäksi, kuten Awake- ja FixedUpdate-funktiot.

3.2.3 Unity Asset Store

Assetit eli aputiedostot ovat Unityllä tehtyjen pelien rakennuspalikoita, joita voidaan hankkia Unity Asset Storesta tai luomalla niitä eri sovelluksilla. Unityn aputiedostot voivat olla kuvia, äänitiedostoja, 3D-malleja, tai mitä tahansa muita tiedostoja, joita Unity tukee.

Unity Asset Store on Unityn ylläpitämä kauppapaikka, josta voidaan hankkia ilmaisia ja maksullisia aputiedostoja, kuten tekstuureja, skriptejä, animaatioita, ääniä ja malleja. Unity Asset Storen tuotteet ovat joko Unityn tai yhteisön tekemiä. Unity Asset Storen tuotteilla voi helposti vähentää tarvittavaa työmäärää projekteissa. (unity.com 2022)

4 TYÖKALUT

Pelikehityksessä tarvitaan paljon muitakin pelikehityksentyökaluja pelkän pelimoottorin lisäksi. Esimerkiksi Unity ei sisällä äänen, kuvan tai 3D mallien käsittelyohjelmaa. Pelit vaativat paljon grafiikoita ja ääniefektejä, joiden luontiin tarvitaan erilliset ohjelmat. Hyödynsin opinnäytetyössä paljon netistä hankittuja grafiikoita ja ääniä, koska minulla ei ollut riittävästi aikaa tai resursseja luoda kaikkia tarvittavia asioita itse.

4.1 Visual Studio Code

Unityn skriptien kirjoittamiseen käytetään kehittäjän vapaavalintaista koodieditoria. Käytän Unity-projekteissa Microsoftin kehittämää Visual Studio Code-editoria, koska olen tottunut sen käyttöön.

Visual Studio Codessa on sisäänrakennettu debugger, joka tekee työskentelystä nopeaa ja vaivatonta. Visual Studio Code tukee kaikkia C#-ominaisuuksia, ja VS Code Marketplacesta voi asentaa erilllisen C#-lisäosan, joka helpottaa työskentelyä. Unity tukee Visual Studio Coden käyttöä editorina. (visualstudio.com 2022)

4.2 GitHub

Käytän tässä projektissa Git-versionhallintaan GitHubin työpöytäsovellusta GitHub Desktopia sekä heidän verkkosivujaan. GitHub on suosituin versiohallintatapa ohjelmistokehityksessä. GitHubia voi käyttää esimerkiksi pilvitallennukseen, koodin jakamiseen ja verkostoitumiseen (hubspot.com 2022). Projektin aikana tallensin muutokset tasaisin väliajoin GitHubiin, jotta pääsin koodeihin käsiksi muillakin laitteilla.

GitHubin käyttö on kannattavaa, koska sen avulla voidaan tallentaa ja seurata projekteja kaikilla laitteilla. Lisäksi GitHub mahdollistaa yhteistyön projekteissa, kun sen avulla jaetaan projektin edistymisiä muille. GitHubin käyttö on kannattavaa myös sooloprojekteissa, koska sen avulla voi seurata omaa edistymistä ja paikantaa tiedostomuutoksia.

5 SUUNNITTELU

5.1 Purrrfect

Luomani pelidemon innoittajana toimii Purrrfect-mobiilipeli, jonka kehityksessä olin mukana vuonna 2022. Pelidemon tekoa helpotti se, että loin pelin valmiin pelin pohjalta. Tämän takia minun ei tarvinnut käyttää paljoa aikaa pelin suunnitteluun, vaan pystyin keskittymään sen kehitykseen.

Purrrfect on yksinkertainen 2D-eläinten silityspeli, joka julkaistiin Google Play Kauppaan vuoden 2022 lopussa. Purrrfect luotiin Oulu Game Labsin pelikehityskurssilla pienessä ryhmässä. Pelin innoittajina toimivat esimerkiksi Pokémon Go, sekä Tamagotchit. Purrrfectin grafiikat ja pelimekaniikat ovat todella yksinkertaiset, ja pelistä tekee houkuttelevan sen söpöt grafiikat ja rauhoittava gameplay.

Pelin idea on löytää kartalta eläimiä ja silittää niitä, kunnes eläimistä tulee tyytyväisiä, minkä jälkeen niitä voi tutkia pelin galleriassa tai aloitusnäytöllä. Pelissä joutuu käyttämään ainutlaatuisia silitystekniikoita eri eläinten kohdalla ja eläimistä on saatavilla eri harvinaisuuksia. (play.google.com 2022)

Purrrfectissa on sydänmittari, joka kertoo, kuinka paljon eläintä pitää vielä silittää, ja energiamittari, joka kertoo, kauanko pelaaja voi vielä jatkaa silittämistä. (kuvio 3.) Pelaajan energia palautuu ajan kuluessa tai mainoksia katsomalla.

KUVIO 3. Purrrfectin pelinäkymä (play.google.com 2022)

5.2 PC-versio

Halusin haastaa itseäni luomalla Purrrfectista PC-version, jonka pääidea on sama mutta jonka pelimekaniikat, maailma ja grafiikat olisivat PC:lle sopivammat. Suunnitelmani oli kehittää lähes kaikki mobiilipelissä toistuvat mekaniikat myös PC-versioon. Purrrfectin pelimekaniikat ovat todella yksinkertaisia, ja halusin luoda 3D-maailman, jossa pelaaja voi liikkua vapaasti, etsiä uusia eläimiä ja tutkia paikkoja. Pelin alustana toimii PC, joten hahmon liikuttamiseen käytetään näppäimistöä ja kameran ohjaukseen hiirtä.

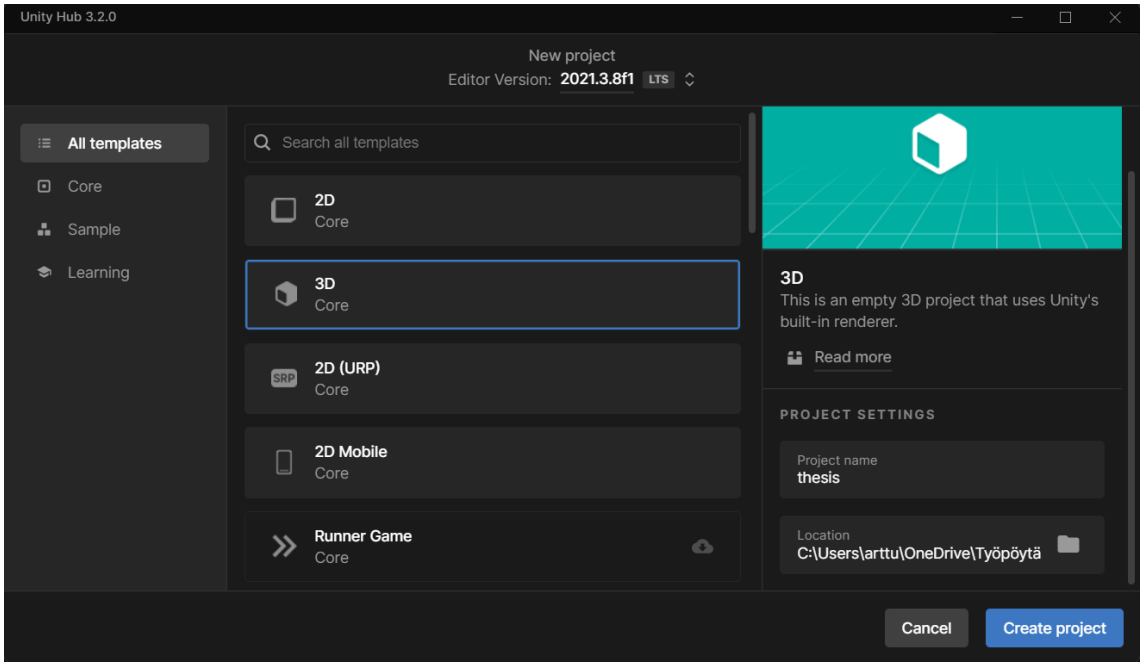
Päätin työn alkuvaiheessa, että pelin maailma sijoittuisi saarelle, koska alkuperäisen Purrrfectin pelimaailma sijoittui myös saarelle. Ajattelin luoda peliin pelaajalle oman saaren, jonne pelaaja voi kerätä eläimiä. Purrrfectin mobiiliversiossa eläimet siirtyivät pelaajan taloon, mutta ajattelin, että PC-versiossa oma saari silitetyille eläimille olisi parempi vaihtoehto.

Ajattelin luoda pelidemon grafiikoista hieman sarjakuvamaiset ja päätin hyödyntää pelissä low poly -tyylisiä grafiikoita. Tämä tarkoittaa sitä, että grafiikoissa on pieni määrä polygoneja, tehden grafiikoista kulmikkaita ja yksinkertaisia. Low poly -grafiikoiden luonti on suhteellisen yksinkertaista, joten ne sopivat projektiini. Lisäksi Unity Asset Storessa on paljon ilmaisia Low Poly -aputiedostoja, joiden hankkiminen onnistuu vaivattomasti.

6 PELIKEHITYS

6.1 Kehityksen aloitus

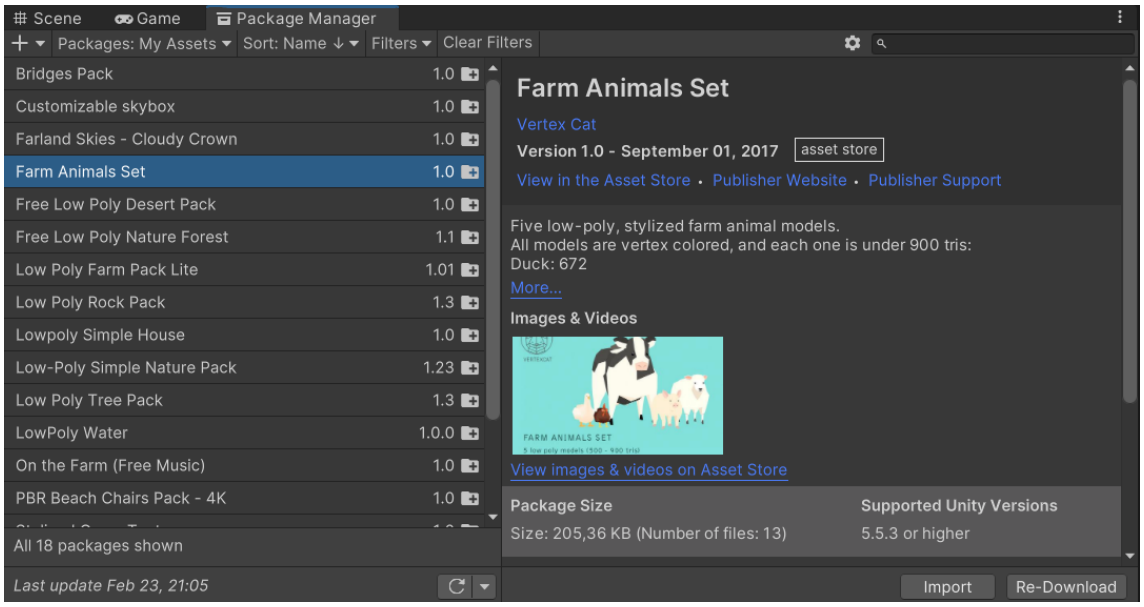
Aloitin pelinkehityksen luomalla uuden projektin Unity Hubissa, ja valitsin projektin pohjaksi 3D:n (kuvio 4). Projektin luomisen jälkeen Unity lataa kaikki tarvittavat tiedostot ja avaa editorin, jossa pelinkehitys tapahtuu. 3D-templaten valitseminen lisää projektiin valmiiksi muutamia tekstuureja ja muokkaa projektin asetukset 3D-ympäristöön sopiviksi. 2D-pelien kehitys onnistuu myös 3D-ympäristössä, ja se saattaa olla jopa hyvä vaihtoehto, koska 3D-ympäristö on monessa asiassa vapaampi. 3D-ympäristössä voidaan käyttää 3D-aputiedostoja ja 3d-näkymää, vaikka peli olisi 2D. Oikean ympäristön valitseminen nopeuttaa suuresti projektin etenemistä etenkin projektin alkuvaiheessa.



KUVIO 4. Unity Hub

6.2 Maailman luonti

Aloitin pelidemon kehityksen luomalla pohjan pelin maailmalle, koska sen jälkeen muita asioita on helpompi työstää. Latasin aluksi muutamia aputiedostoja, jotka olin jo aiemmin valinnut peliä varten. Tiedostot voidaan lisätä Unityyn verkossa Unity Asset Storessa, minkä jälkeen ne löytyvät Unityn Package Managerista (kuvio 5). Käytin maailman luontiin pelkästään Unityn sisäisiä työkaluja tai Unity Asset Storesta hankkimiani aputiedostoja.

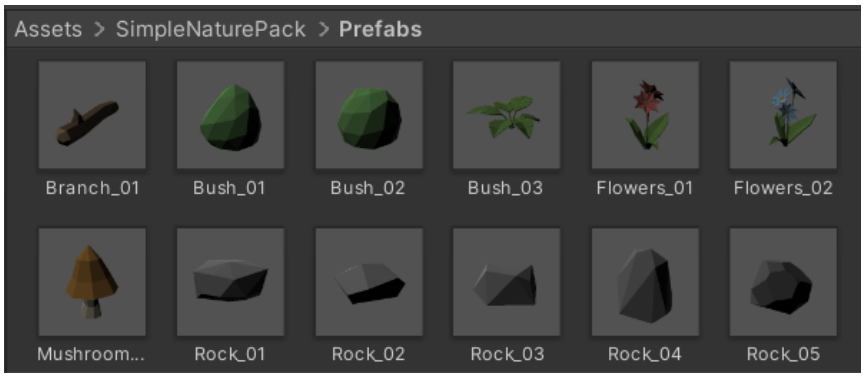


KUVIO 5. Unity Package Manager

Aloitin pelimaailman rakentamisen luomalla pelin maaston. Maaston luomiseen käytin terrain-objektia, jota muokkasin maalaustyökalulla. Unityn terraintyökalun avulla voi helposti luoda yksityiskohtaisia ja realistisia maastoja nopeasti ja helposti. Jätin pelin maailman aika pieneksi, koska minulla ei ollut tarpeeksi aikaa suuren maailman luomiseen. Saatuani maastosta sopivan muotoisen ja kokoisen aloin lisäämään peliin aputiedostoja. Aputiedostojen lisääminen teki maailmasta paljon eloisamman ja miellyttävämmän näköisen (kuvio 6).

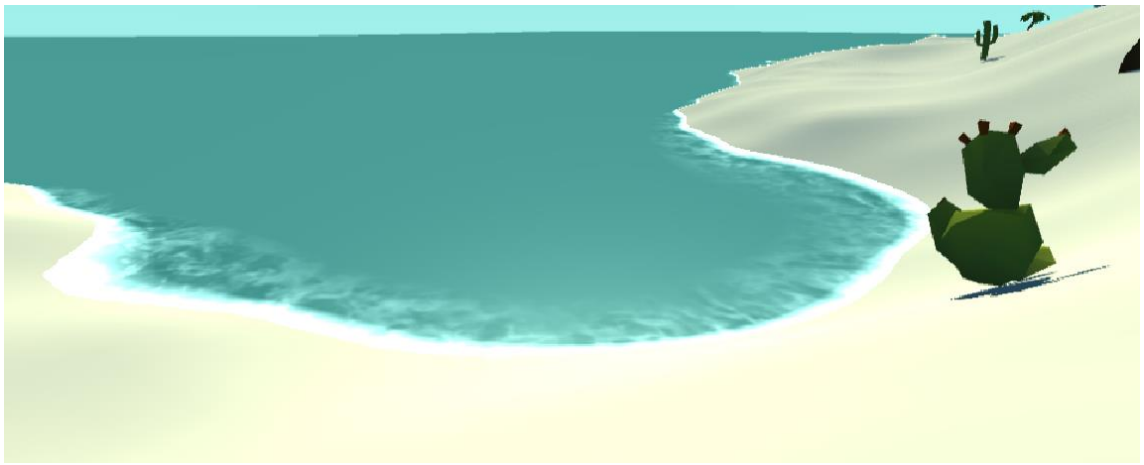
KUVIO 6. Pelimaalimassa olevan terrainin päälle lisättyjä aputiedostoja Unityn Game-näkymässä

Lisäsin maailmaan Unity Asset Storesta hankkimaani kasvistoa, kuten puita, ruohoa, pensaita ja kaktuksia. Lisäsin myös kiviä ja pieniä vuoria, tehden maailmasta mielenkiintoisemman (kuvio 7). Käytin maailman luonnissa vain muutamia eri aputiedostoja, ja lisäsin maailmaan lopulta aika vähän sisältöä, koska halusin pitää pelin tyylin yksinkertaisena.



KUVIO 7. Valintanäkymä, josta valitaan sisältöä pelimaailmaan visuaalisuuden parantamiseksi.

Seuraavaksi lisäsin pelimaailman ympärille vettä (kuvio 8). Käytin veden lisäämiseen LowPoly Water - aputiedostoa, jonka skriptiä ja asetuksia muokkasin hieman, jotta voin luoda vedestä peliin sopivamman.



KUVIO 8. Saaren reuna pelimaailmassa

Viimeistelin pelimaailman lisäämällä Unity Asset Storesta hankkimani taivaan skyboxiin ja musiikkia peliin. Taivaan lisääminen saa pelimaailman vaikuttamaan suuremmalta ja eloisammalta kuin se oikeasti on. Skybox on kuutio, joka näkyy pelissä kaikkien muiden grafiikoiden takana. Skyboxin jokaista sivua varten tarvitaan oma tekstuuri. Käytin pelimaailman muokkaamiseen aikaa yhteensä yhden viikonlopun, minkä jälkeen olin melko tyytyväinen maailmaan (kuvio 9).

KUVIO 9. Pelimaailma Unityn Game-näkymässä

6.3 Pelihahmo

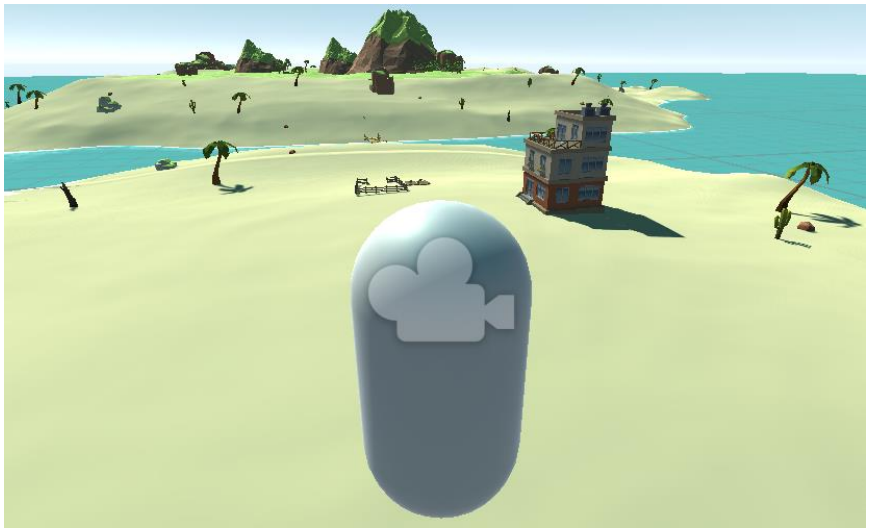
Seuraavaksi aloin kehittämään ominaisuuksia pelaajaan liikkumista varten. Pelihahmon liikkumista varten on olemassa paljon ilmaisia paketteja, joiden avulla hahmon liikkumisen voi kehittää vaivatta. Halusin kuitenkin kehittää hahmon liikkumisen alusta alkaen itse, koska minua kiinnosti, miten se tapahtuu, ja halusin muokata pelihahmon liikkumisen tarpeitteni mukaan.

Pelaajan liikkuvuus on oleellinen osa pelien kehitystä, joten aiheesta löytyy paljon eri tutoriaaleja netistä. Aloitin luomalla editorissa uuden kansion nimeltä Scripts kaikkia kooditiedostoja varten. Loin kansioon Controller.cs-tiedoston, joka sisältää kaikki koodit pelaajan liikkumista varten.

Aloitin koodaamisen lisäämällä kaikki muuttujat tiedoston alkuun. Määrittelin kävely- ja juoksunopeuden, hypyn voimakkuuden ja painovoiman. Lisäsin myös arvoja kameran liikkumista varten. Sen jälkeen lisäsin Start-funktion sisälle toiminnot characterControllerin noutamiseen ja hiiren osoittimen piilottamiseen. CharacterControllerin avulla voi helposti kehittää törmäysten rajoittamia liikkeitä.

Sitten siirryin Update-funktion muokkaamiseen. Ohjelmoin Update-funktioon hahmon liikkumisen, hyppimisen ja kameran liikuttamisen. (liite 2.)

Tämän jälkeen lisäsin pelimaailmaan kapselin muotoisen olion, johon lisäsin pelin kameran ja Controller.cs-tiedoston. Kapseli toimii pelissä pelattavana hahmona ja törmäyspintana, koska en halunnut pelihahmon kulkevan pelien objektien läpi. Pelaaja ei tule näkemään kapselia, koska peli rajoittuu ensimmäiseen persoonaan eli pelihahmon näkökulmaan. Tässä vaiheessa pelihahmo on valmis, ja pystyy liikkumaan maailmassa. (kuvio 10.)



KUVIO 10. Liikutettava pelihahmo Unityn Scene-näkymässä

6.4 Eläinten generointi maailmaan

Eläinten generoimista varten loin uuden C#-tiedoston nimeltä AnimalSpawn.cs. En halua, että eläimiä ilmestyy saaren ulkopuolelle, joten ensimmäisenä etsin pelimaailmasta oikeat koordinaatit, joille haluan luoda eläimiä. Koordinaatit löysin liikuttelemalla satunnaista oliota pelimaailmassa ja seuraamalla sen sijaintia inspectorissa.

Koodin alussa määrittelin kaikki muuttujat. Kyseisessä skriptissä tarvitsen omat muuttujat eläimille, x- ja z-koordinaateille, sekä yhden muuttujan eläinten laskemista varten. Koordinaattien avulla määrittelen, mihin eläimet generoituvat, ja niiden avulla voi seurata eläinten sijaintia. Muuttujien jälkeen lisäsin while-silmukan, jossa laitoin eläinten enimmäismääräksi 10, koska en halunnut generoida maailmaan loputtomasti eläimiä. Tämän jälkeen määrittelin oikeat koordinaatit ja lisäsin tarvittavat tiedot Instantiate-luokkaan. Instantiate mahdollistaa olioiden luomisen pelimaailmaan.

Tämän jälkeen C#-tiedosto on valmis (liite 3). Tässä vaiheessa eläimet ilmestyvät kartalle, mutta ne eivät vielä liiku ympäriinsä, eivätkä ne ole vuorovaikutteisia.

6.5 Eläinten tekoäly

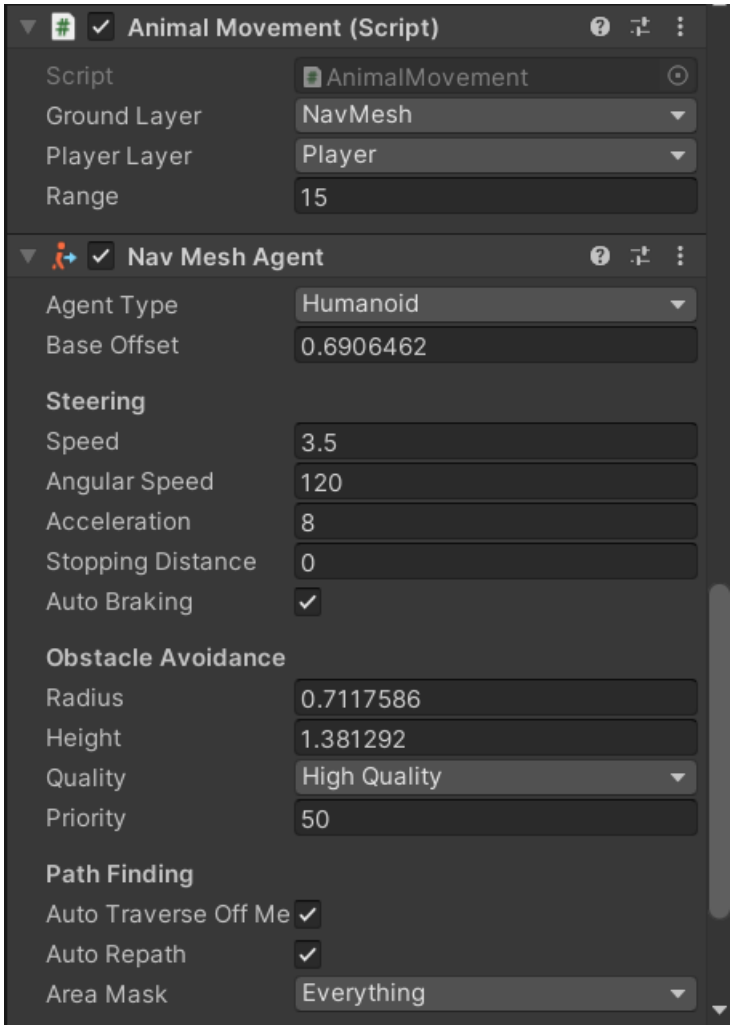
Eläinten liikkumista varten eläimiin tulee lisätä komponentti nimeltä NavMeshAgent. NavMeshagent mahdollistaa objektien kehittyneemmän liikumisen pelimaailmassa, ja sen avulla voidaan säätää hahmojen liikkumiseen liittyviä asetuksia, kuten nopeutta. NavMeshAgentin avulla voi helposti hallita tekoälyn kykyjä kulkea erilaisissa maastoissa ja esteiden lähettyvillä. Tekoälyn liikkumisen kehittäminen on siis mahdollista ilman NavMeshAgentia, mutta se olisi vähemmän älykästä.

Seuraavaksi pelimaailmasta pitää tehdä sopiva eläinten liikkumista varten. Lisäsin Inspectorissa maastoon komponentin NavMeshSurface. Kyseinen komponentti mahdollistaa eläinten liikkumisen pelimaailmassa. Komponentin asetuksista voi määrittää useita eri asetuksia maailmaan liittyen.

Saatuani NavMeshSurfacen ja NavMeshAgentin asetukset kuntoon aloin ohjelmoimaan scriptiä tekoälyä varten. Haluan, että pelin eläimet liikkuvat pelimaailmassa satunnaisesti. Lisäksi haluan, että pelaajan ollessa eläinten lähellä niiden tulisi lähestyä pelaajaa. Kun eläimet tulevat pelaajaa kohti, pelaajan on helpompaa toimia eläinten kanssa.

Kooditiedoston alussa importoin käyttämäni nimiavaruudet ja määrittelin kaikki tarvittavat muuttujat ja asetukset. Tiedoston riveillä 50–61 on SearchForDest-funktio, jonka tehtävänä on etsiä satunnainen koordinaatti, johon eläin matkustaa. Funktio takaa sen, että eläimet liikkuvat satunnaisesti kartalla eivätkä vain tiettyjä reittejä pitkin. Kyseistä funktiota käytetään sen yläpuolella olevassa Patrol-funktiossa. Patrol-funktio on voimassa niin kauan, että pelaaja on tarpeeksi lähellä eläimiä, jolloin Chase-funktio käynnistyy. Chase-funktion aikana eläimet liikkuvat pelaajan koordinaatteja kohti, kunnes pelaaja on taas liian kaukana eläimistä. Chase- ja Patrol-funktiot ovat Update-funktion sisällä, joka tarkistaa kumpaa funktiota tulee käyttää jokaisen animaation aikana piirrettävän kuvaruudun vaihduttua (liite 4).

Lopuksi lisäsin valmiin skriptikomponentin jokaiselle eläimelle. Tämän jälkeen tekoäly on valmis (kuvio 11).



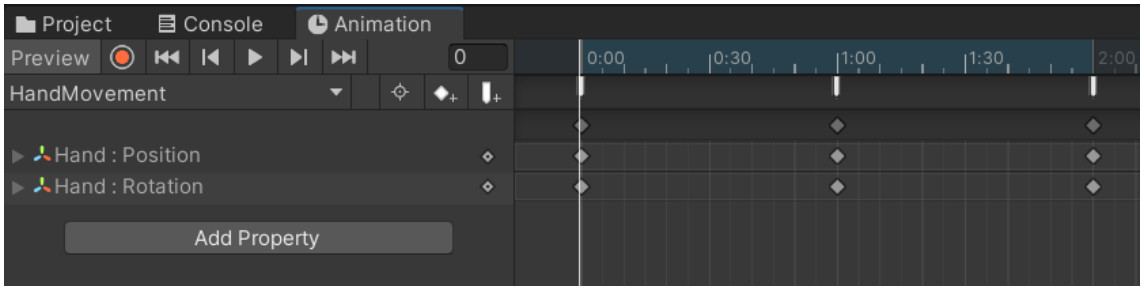
KUVIO 11. Valmiin eläimen asetukset Unityn Inspectorissa

6.6 Eläinten silittäminen

6.7 Animaatio

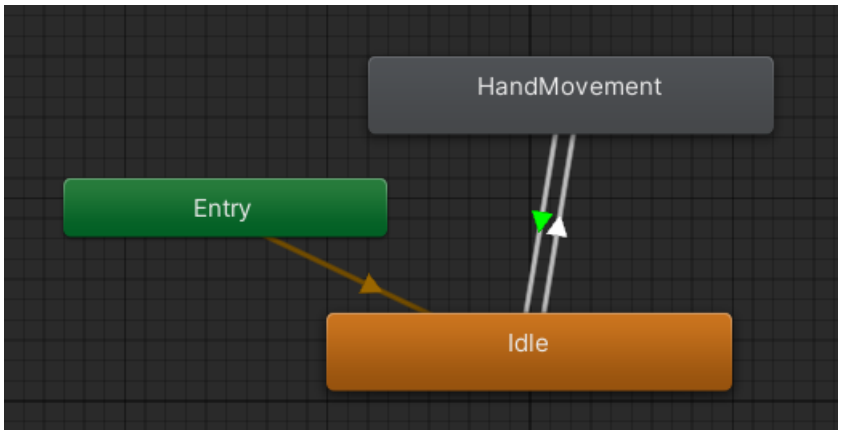
Lisäsin eläinten silittämistä varten pelihahmolle käden. Kättä esittävän 3D-mallin hankin Unity Asset Storesta ja lisäsin sen hierarkianäkymässä pelihahmolle. Käsi on hierarkianäkymässä Player-objektin lapsiobjekti, jolloin se liikkuu Player-objektin mukana. Tämän jälkeen muokkasin käden koordinaatit oikeiksi Inspectorissa. Koordinaatteja piti muokata, koska halusin käden näkyvän oikein pelinäkymässä. Tämän jälkeen lisäsin kädelle tekstuurit ja aloin valmistelemaan käden animaatiota. Käsi liikkuu aina pelaajan painaessa silitysnappia.

Projektinäkymässä lisäsin Unityyn muutamia animaatiokomponentteja, joita tarvitaan animoimisessa. Loin unityn animator-komponenttia käyttäen HandMovement-objektin, joka sisältää kaikki käden animaatiot. Muokkasin ohjainta Animation-näkymässä (kuvio 12). Animation-näkymässä voidaan määritellä objektin liikkeet jokaiselle animaation aikana piirrettävälle kuvaruudulle animaation käynnistymisen jälkeen.



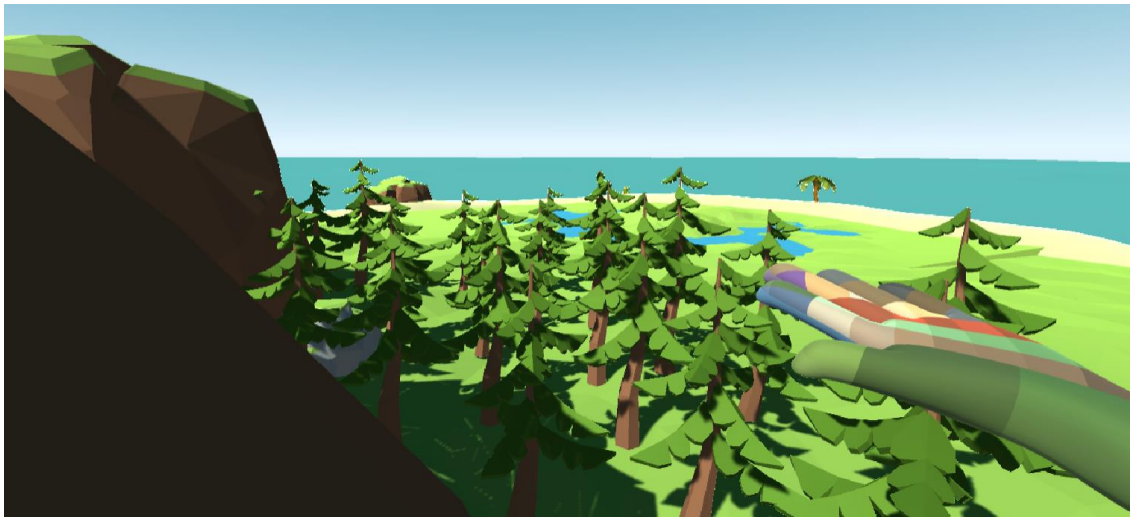
KUVIO 12. Unityn Animation-näkymä

Saatuani animaation valmiiksi Animation-näkymässä, siirryin Unityn animaattorinäkymään. Animaattorissa säädetään animoitujen peliobjektien logiikoita. Siellä voi säätää animaatioiden kestoja, järjestyksiä ja siirtymiä. Animaattorissa animaatioiden välisiä siirtymiä kuvataan nuolilla ja animaatioita suorakulmioina (kuvio 13).



KUVIO 13. Animaation logiikat animator-näkymässä

Saatuani animaation logiikat toimimaan lisäsin pelihahmon kädelle yksinkertaisen kooditiedoston (liite 5). Tiedoston alussa Start-funktiossa haetaan viiteyhteys tarvittavaan peliobjektiin eli Animator-komponenttiin. Tämän jälkeen Update-funktio tarkistaa jokaisen animaation aikana piirrettävän kuvaruudun kohdalla, onko pelaaja painanut silitysnäppäintä. Jos on, peli käynnistää animaation. Kooditiedoston lisäämisen jälkeen käsi on valmis (kuvio 14).



KUVIO 14. Pelihahmon käsi pelinäkymässä

6.8 Silitys

Päätin luoda PC-version silityksestä yksinkertaisemman, koska se sopii paremmin pelin kontrolleihin. Mobiiliversiossa käytetty silitystekniikka ei olisi toiminut PC:llä yhtä hyvin, joten tyydyin yksinkertaisempaan silitykseen, jossa silityspainikkeen painaminen käynnistää animaation. Lisäksi työtä tehdessä ei ollut tarpeeksi aikaa tai resursseja monimutkaisemman silitystaktiikan kehittämiseen.

Eläinten silitystä varten loin uuden kooditiedoston AnimalPetting.cs. Tiedoston alussa määrittelin, montako yksittäistä silitystä eläinten silittämiseen tarvitaan. Päätin että kolme silitystä on hyvä määrä, koska jokainen animoitu silitys kestää muutaman sekunnin.

Tämän jälkeen loin yksinkertaisen Stroke-funktion, joka tarkistaa, montako kertaa eläintä pitää vielä silittää. Kun eläintä on silitetty kolme kertaa, on silitys suoritettu loppuun ja funktio poistaa eläimen saarelta. Eläinten poistaminen saarelta mukailee alkuperäistä peliä, koska siinäkin eläimet siirtyvät suoraan pelaajan kotiin. Tässä pelikehityksen vaiheessa eläimet vain katosivat pelistä, joten seuraavaksi tuli kehittää ominaisuus, joka generoi eläimet pelaajan saarelle. (Kuvio 15.)

```
6   public class AnimalPetting : MonoBehaviour
7   {
8       [SerializeField] float strokesLeft, strokes = 3f;
9
10      private void Start()
11      {
12          strokesLeft = strokes;
13      }
14
15      public void Stroke(float SingleStroke)
16      {
17          strokesLeft -= SingleStroke;
18
19          if(strokesLeft <= 0)
20          {
21              Destroy(gameObject)
22          }
23      }
24  }
```

KUVIO 15. AnimalPetting.cs Visual Studio Codessa

6.9 Eläinten generoiminen pelaajan saarelle

Lisäsin aiemmin luomaani AnimalPetting.cs-tiedostoon kohdan, joka silityksen päätyttyä generoi eläimet pelaajan saarelle. Käytin eläinten generoimiseen pitkälti AnimalSpawn.cs-tiedoston koodeja. Muutin koodissa olevat koordinaatit pelaajan saarelle, ja vaihdoin generoituvien eläinten määrän yhdeksi. Tämän jälkeen eläimet generoituvat pelaajan saarelle silityksen päätyttyä (kuvio 16). Lisäsin pelaajan saaren ympärille aidan, jota eläimet eivät pysty ylittämään. Täten minun ei tarvinnut muokata eläimissä olevaa AnimalMovement.cs-tiedostoa erialaiseksi eläimille, jotka ovat silitettyjä.

KUVIO 16. Silitettyjä eläimiä pelaajan saarella

7 JATKOKEHITYS

Työn loppuvaiheella pelin pääidea toimii hyvin, mutta pelissä on vielä runsaasti kehitettävää ja se on vielä hiomaton. Pelin kehityksessä ideana oli oppia uutta eikä valmistaa täydellistä peliä, joten en todennäköisesti jatka tämän pelin kehitystä raportin jälkeen.

Alkuperäisessä Purrrfectissa jokaisesta eläimestä oli monta eri variaatiota, joista osa oli harvinaisempia. Jos jatkaisin pelin kehittämistä, lisäisin peliin uusia eläimiä ja jokaisesta eläimestä erivärisiä versioita. Raportissa käyttämäni eläimet olivat ainoastaan yhdestä aputiedostosta, joten eläimiä on vielä liian vähän. Lisäksi nykyisillä eläimillä ei ole lainkaan animaatioita, joten ne näyttävät melko elottomilta. Lisäksi jokaiselle eläimelle tulisi kehittää omanlaiset liikkumistavat. Peliin tulisi lisätä myös uusia ääniä.

Purrrfectin mobiiliversiossa jokaisella eläimellä on oma erityinen tapa, miten niitä tulisi silittää. Tämän raportin aikana ehdin kuitenkin kehittämään vain yhden silitystavan, joten jatkokehityksessä niitä pitäisi luoda lisää. Suunnittelin, että jos jatkaisin pelin kehitystä, pelaaja pystyisi keräämään uusia esineitä pelimaailmasta, joita pelaaja tarvitsisi haastavampien eläinten silitykseen. Pelaaja pystyisi keräämään erilaisia hoitovälineitä sekä ruokia.

8 POHDINTA

PC-pelin kehittäminen mobiilipelin pohjalta on mielestäni hyvä tapa oppia pelikehitystä ja haastaa omaa osaamista. Pelin luonti valmiin pelin pohjalta on hyvä tapa harjoitella, koska silloin voi keskittyä itse kehitykseen eikä tarvitse käyttää niin paljoa aikaa suunnittelemiseen.

Opin raporttia tehdessä paljon Unityn käytöstä ja C#-ohjelmoinnista mutta myös eri pelialustoista ja -teollisuudesta. Raporttia tehdessä pohdin paljon asioita, joita tulee ottaa huomioon, kun kehitetään pelejä eri alustoille. Pelimekaniikat tulee toteuttaa eri tavoilla jakelualustan rajoja ajatellen. Mobiili- ja PC-pelien grafiikat eroavat todella paljon toisistaan ja pelimekaniikat, kuten eläinten silitys tulee toteuttaa eri tavalla.

Opin projektia ja raporttia tehdessä paljon. Opinnäytetyön tekeminen oli kaiken kaikkiaan mielekäs kokemus. Tämä oli ensimmäinen näin laaja peliprojekti, jota työstin itsenäisesti, ja opin todella paljon uutta.

Pelikehitys vaatii paljon työtä sekä suunnittelua, ja etenkin yksin pelien kehitys on todella hidas prosessi. Olen kuitenkin tyytyväinen siihen, kuinka paljon sain kehitettyä peliä. Peli jäi melko yksinkertaiseksi, mutta ajattelin jo raportin alussa, että pelin kehittämisen idea on uuden oppiminen eikä niinkään valmiin pelin luominen. Sain mielestäni kaikki tavoitteeni tehtyä ja olen tyytyväinen lopputulokseen.

LÄHTEET

Built In 2023. 29 Mobile Game Companies to Know. Hakupäivä 6.4.2023. https://builtin.com/media-gaming/mobile-game-companies.

Education Ecosystem 2022. UNITY HISTORY. Hakupäivä 15.1.2023. https://educationecosystem.com/guides/game-development/unity/history.

Fallout Wiki 2022. Fallout Shelter. Hakupäivä 20.2.2023. https://fallout.fandom.com/wiki/Fallout_Shelter.

Full Scale 2021. WHAT IS A GAME ENGINE. Hakupäivä 4.1.2023. https://fullscale.io/blog/what-is-game-engine/.

GameDev Academy 2022. Unity vs. Unreal – Choosing a game Engine. Hakupäivä 4.1.2023. https://gamedevacademy.org/unity-vs-unreal/.

giffgaff 2018. Mobile gaming overtakes PC and console for the first time. Hakupäivä 18.1.2023. https://www.giffgaff.com/blog/mobile-gaming-overtakes-pc-and-console-for-the-first-time/.

Google Play 2022. Purrrfect: Pet & Collect. Hakupäivä 19.2.2023 https://play.google.com/store/apps/details?id=com.NCD_Studios.Purrrfect.

HubSpot 2022. What Is GitHub? (And What Is It Used For?) Hakupäivä 16.1.2023. https://blog.hubspot.com/website/what-is-github-used-for.

IGN 2015. Alto's Adventure Wiki Guide. Hakupäivä 7.2.2023. https://www.ign.com/wikis/altos-adventure/.

Lifewire 2021. Pros and Cons of Minecraft: Pocket Edition. Hakupäivä 20.2.2023 https://www.lifewire.com/positives-and-negatives-of-minecraft-pocket-edition-3863634.

Medium 2021. Mobile Game Genre on the Rise: AAA Mobile Games. Hakupäivä 6.2.2023. https://appnava.medium.com/mobile-game-genre-on-the-rise-aaa-mobile-games-d72a5904e90e.

Medium 2022. Best practices for bringing PC and console games to mobile. Hakupäivä 5.2.2023. https://medium.com/googleplaydev/best-practices-for-bringing-pc-and-console-games-to-mobile-863cedb9fbc6.

Perforce Software 2020. What Are the Best Game Engines? Hakupäivä 4.1.2023. https://www.perforce.com/blog/vcs/most-popular-game-engines.

Rovio 2023. This is Rovio Hakupäivä 20.3.2023. https://www.rovio.com/about/

Snowman 2023. About. Hakupäivä 7.2.2023. https://www.builtbysnowman.com/about.html.

Starloop Studios 2020. Mobile Games Vs. PC Vs. Console Games: What Market is the Best Bet?. Hakupäivä 5.2.2023. https://starloopstudios.com/mobile-games-vs-pc-vs-console-games-what-market-is-the-best-bet/.

Statista 2022. Number of available gaming apps in the Google Play Store from 1st quarter 2015 to 3rd quarter 2022. Hakupäivä 21.3.2023. https://www.statista.com/statistics/780229/number-of-available-gaming-apps-in-the-google-play-store-quarter/.

Transform Interactive 2022. The 5 Best Uses for Unity (Besides Game Development). Hakupäivä 16.1.2023 https://transforminteractive.com/5-best-uses-for-unity/.

Unity 2015. Unity Europe 2015 Highlight Reel. Hakupäivä 24.6.2015. https://www.youtube.com/watch?v=c3GpKjTaXw4&ab_channel=Unity.

Unity 2023a. PROGRAMMING IN UNITY. Hakupäivä 13.1.2023. https://unity.com/solutions/programming#program-gameplay-c.

Unity 2023b. UNLOCK YOUR CREATIVITY. Hakupäivä 13.1.2023. https://unity.com/download.

Unity 2023c. WELCOME TO UNITY. Hakupäivä 13.1.2023. https://unity.com/our-company.

Unity Support 2022. What is Unity Asset Store and how do I purchase Assets? Hakupäivä 3.2.2023. https://support.unity.com/hc/en-us/articles/210142503-What-is-the-Unity-Asset-Store-and-how-do-I-purchase-Assets-.

Visual Studio Code 2022. Unity Development with VS Code. Hakupäivä 4.2.2023. https://code.visualstudio.com/docs/other/unity.

LIITTEET

Unityn Editorin osat

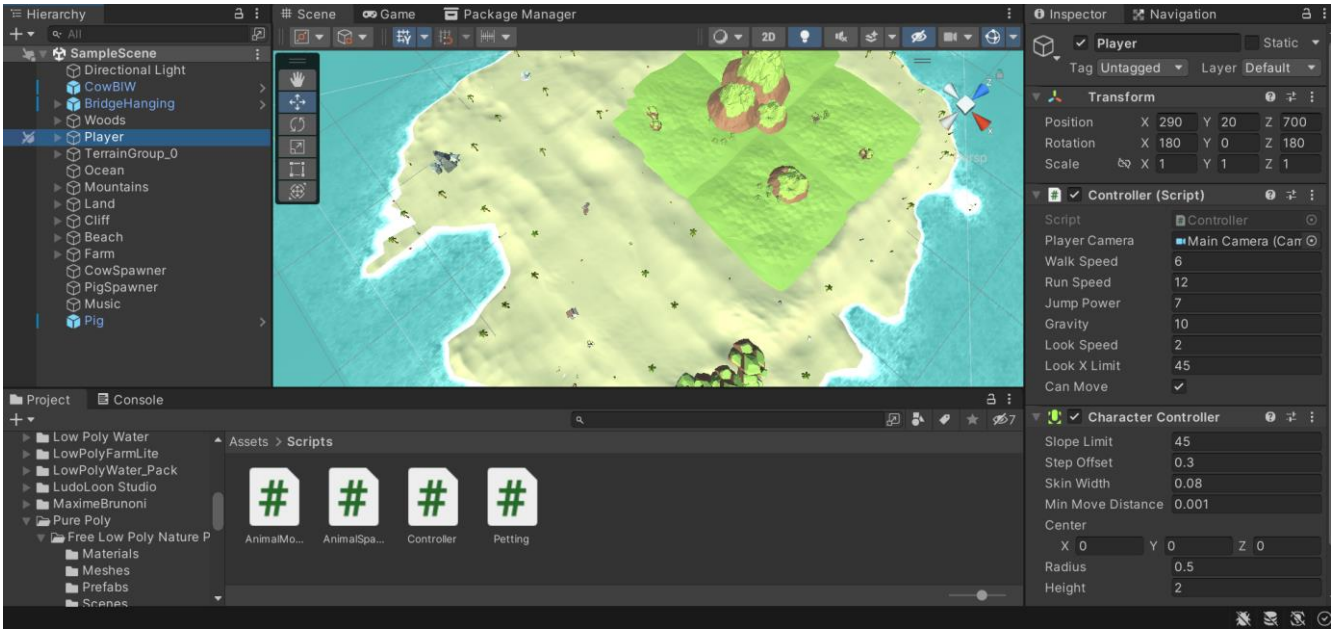
Controller.cs

AnimalSpawn.cs

AnimalMovement.cs

HandMovement.cs

Liite 1. Unityn Editorin osat



Liite 2. Controller.cs

```
33      void Update()
34      {
35
36          #region Handles Movment
37          Vector3 forward = transform.TransformDirection(Vector3.forward);
38          Vector3 right = transform.TransformDirection(Vector3.right);
39
40          bool isRunning = Input.GetKey(KeyCode.LeftShift);
41          float curSpeedX = canMove ? (isRunning ? runSpeed : walkSpeed) * Input.GetAxis("Vertical") : 0;
42          float curSpeedY = canMove ? (isRunning ? runSpeed : walkSpeed) * Input.GetAxis("Horizontal") : 0;
43          float movementDirectionY = moveDirection.y;
44          moveDirection = (forward * curSpeedX) + (right * curSpeedY);
45
46          #endregion
47
48          #region Handles Jumping
49          if (Input.GetButton("Jump") && canMove && characterController.isGrounded)
50          {
51              moveDirection.y = jumpPower;
52          }
53          else
54          {
55              moveDirection.y = movementDirectionY;
56          }
57
58          if (!characterController.isGrounded)
59          {
60              moveDirection.y -= gravity * Time.deltaTime;
61          }
62
63          #endregion
64
65          #region Handles Rotation
66          characterController.Move(moveDirection * Time.deltaTime);
67
68          if (canMove)
69          {
70              rotationX += -Input.GetAxis("Mouse Y") * lookSpeed;
71              rotationX = Mathf.Clamp(rotationX, -lookXLimit, lookXLimit);
72              playerCamera.transform.localRotation = Quaternion.Euler(rotationX, 0, 0);
73              transform.rotation *= Quaternion.Euler(0, Input.GetAxis("Mouse X") * lookSpeed, 0);
74          }
75
76          #endregion
```

Liite 3. AnimalSpawn.cs

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

public class AnimalSpawn : MonoBehaviour {

    public GameObject Animal;
    public int xPos;
    public int zPos;
    public int AnimalCount;

    void Start () {
        StartCoroutine(AnimalDrop());
    }

    IEnumerator AnimalDrop()
    {
        while (AnimalCount < 10)
        {
            zPos = Random.Range(295, 440);
            xPos = Random.Range(210, 370);
            Instantiate(Animal, new Vector3(xPos, 17, zPos), Quaternion.identity);
            yield return new WaitForSeconds(0.1f);
            AnimalCount += 1;
        }
    }
}
```

Liite 4. AnimalMovement.cs

```
6   public class AnimalMovement : MonoBehaviour
7   {
8       GameObject player;
9
10      NavMeshAgent agent;
11
12      [SerializeField] LayerMask groundLayer, playerLayer;
13
14      // Patrol
15      Vector3 desPoint;
16      bool walkPointSet;
17      [SerializeField] float range;
18
19      // State change
20      [SerializeField] float sightRange;
21      bool playerInSight;
22
23      void Start()
24      {
25          agent = GetComponent<NavMeshAgent>();
26          player = GameObject.Find("Player");
27      }
28
29      // Update is called once per frame
30      void Update()
31      {
32          playerInSight = Physics.CheckSphere(transform.position, sightRange, playerLayer);
33
34          Patrol();
35          Chase();
36      }
37
38      void Chase()
39      {
40          agent.SetDestination(player.transform.position);
41      }
42
43      void Patrol()
44      {
45          if (!walkPointSet) SearchForDest();
46          if (walkPointSet) agent.SetDestination(desPoint);
47          if (Vector3.Distance(transform.position, desPoint) < 10) walkPointSet = false;
48      }
49
50      void SearchForDest()
51      {
52          float z = Random.Range(-range, range);
53          float x = Random.Range(-range, range);
54
55          desPoint = new Vector3(transform.position.x + x, transform.position.y, transform.position.z + z);
56
57          if (Physics.Raycast(desPoint, Vector3.down, groundLayer))
58          {
59              walkPointSet = true;
60          }
61      }
62  }
```

Liite 5. HandMovement.cs

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

public class HandMovement : MonoBehaviour
{
    Animator Petting;

    void Start()
    {
        Petting = GetComponent<Animator>();
    }

    void Update()
    {
        if(Input.GetMouseButtonDown(0))
        {
            Petting.SetTrigger("Petting");
        }

    }
}
```