LAMK
Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

MOBIILIPELIN KEHITYS UNITY 3D - PELIMOOTTORILLA

LAHDEN AMMATTIKORKEAKOULU
Tekniikan ala
Tieto- ja viestintätekniikka,
Ohjelmistotekniikka (AMK)
Opinnäytetyö
Syksy 2019
Elina Vanamo

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Vanamo, Elina	Opinnäytetyö, AMK	Syksy 2019
	Sivumäärä	
	31	

Työn nimi

Mobiilipelin kehitys Unity 3D -pelimoottorilla

Tutkinto

Tieto- ja viestintätekniikka, ohjelmistotekniikka (AMK)

Tiivistelmä

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa mobiilipeli Android-alustalle käyttäen Unity 3D -kehitysympäristöä. Työssä perehdyttiin ja tutkittiin Unity-kehitysympäristön ominaisuuksia ja siihen, miten sitä käytetään. Toteutettu mobiilipeli on nimettömänä pysyvän startup-yrityksen ensimmäinen tuote.

Työn teoriaosuudessa käydään läpi Unityn toimintaa ja sen työkaluja, joita käytettiin mobiilipelin tekemisessä. Se pitää sisällään Unityn objektit ja komponentit, fysiikkaa ja grafiikkaa sekä yleistä pelisilmukkaa. Tutkimus toteutettiin mobiilipelin näkökulmasta, ja sen prioriteettina on fysiikka- ja grafiikkaominaisuuksien optimointi.

Työn käytännönosuus sisältää katsauksen toteutetun mobiilipelin sisältöön. Siinä käydään läpi mobiilipelin peli-idea, pelin kulkua, grafiikkaominaisuuksia sekä pelin kehittämisen haasteita, testausta sekä tulevaisuudennäkymää.

Työn lopputuloksena on ensimmäinen pelattava versio keväällä 2020 Google Play -kauppaan julkaistavasta mobiilipelistä. Mobiilipeli sisältää kaksi kenttää ja yleisen toimivan pelimekaniikan.

Asiasanat

Unity, Mobiilipelin kehitys, Mobiilipelin optimointi, Blender

Abstract

Author(s)	Type of publication	Published
Vanamo, Elina	Bachelor's thesis	Autumn 2019
	Number of pages	
	31	

Title of publication

Mobile game development with the Unity game engine

Name of Degree

Bachelor of Engineering, Information and Communications Technology

Abstract

The objective of the thesis was to design and implement a mobile game for the Android platform using the Unity 3D development environment. The thesis contains a study of the features of the Unity development environment and how to use it. The mobile game is a debut product of a startup company that stays anonymous.

The theoretical part of the thesis covers Unity's operations and the tools used to make the mobile game. It includes objects and components, physics and graphics, and the general game loop. The research has been carried out from the perspective of a mobile game and its priority is the optimization of physics and graphics.

The practical part of the thesis includes an overview of the content of the implemented mobile game. It presents the game idea, game flow, graphics features, as well as the challenges of the game development, game testing and the future outlook of the game.

The result of the work was the first playable version of the mobile game that will be released on the Google Play Store in the spring of 2020.

Keywords

Unity, mobile game development, mobile game optimization, Blender

SISÄLLYS

1 JOHDANTO

Opinnäytetyö käsittelee Android-mobiilipelin tekemistä Unity 3D-pelimoottorilla. Työn tavoite on tehdä kyseinen mobiilipeli käyttäen Unityä pelimoottorina sekä Blender 3D-mallinnusohjelmaa mallien luomiseen. Työssä myös tutkitaan ja kuvataan Unity-kehitysympäristöä, sen tarjoamia mahdollisuuksia ja teknisiä rajoituksia. Tutkimusongelma on mobiilipelin optimointi mobiililaitteille.

Työssä käsiteltävä mobiilipeli on aloittelevan yhden hengen startup-yrityksen ensimmäinen kehitteillä oleva tuote. Yrityksen toimialaa on alustariippumaton pelinkehitys- sekä muu ohjelmistotekniikka. Mobiilipelien jatkuvasti kasvava suosio innoitti yritystä lähtemään mukaan kehitykseen. Yritys ja mobiilipelin nimi pysyvät nimettömänä työssä. Työssä kuvataan pelin ensimmäistä versiota. Mobiilipelin genre on rajoja rikkova ensimmäisestä persoonasta pelattava avoimen maailman 3D-seikkailupeli. Se sisältää ammuntaa, tasohyppelyä sekä ongelmanratkaisua. Pelin ensimmäinen versio sisältää kaksi eri pelattavaa kenttää ja yleisen pelimekaniikan. Yleinen pelimekaniikka pitää sisällään muun muassa liikkumisen ja ampumisen mekaniikan, vihollishahmojen tekoälyn sekä pelimaailman ja pelaajan välisen vuorovaikutuksen. Peli sisältää myös henkien, ammuksien ja pisteidenlaskennan seurannan.

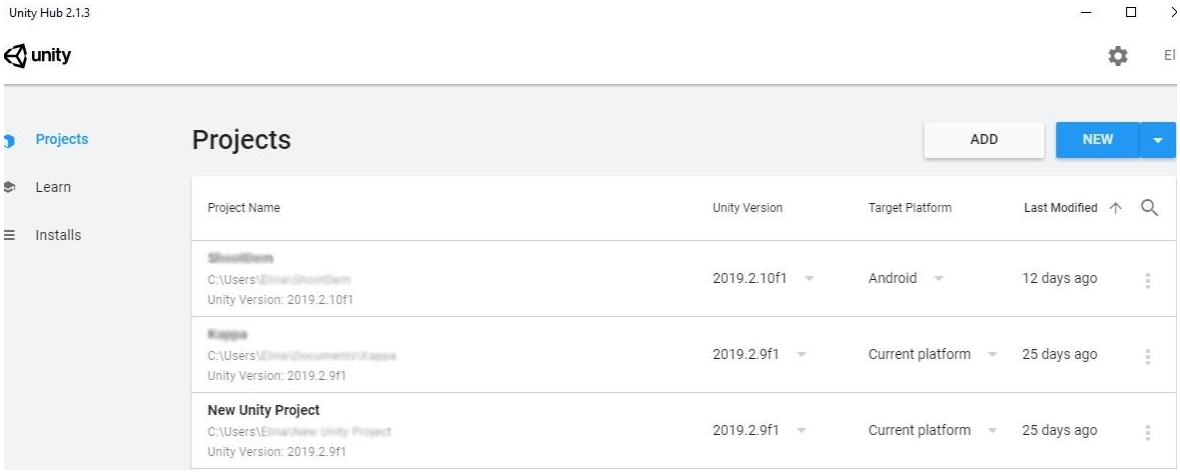
Mobiilipelit ovat jatkuvasti kasvava ja kehittyvä trendi videopeliteollisuudessa (Mobvista 2018). Klassikkoja ovat muun muassa Angry Birds ja Clash of Clans. Mobiililaitteet ovat viimevuosina kehittyneet merkittävästi, mikä on mahdollistanut isompien ja näyttävämpien pelien toteutumisen myös mobiilialustalle. Moni suuri peliyhtiö on lähtenyt trendiin mukaan ja kääntänyt alkuperäisesti tietokoneelle ja konsolille tehdyt pelit myös mobiililaitteille. Näitä ovat muun muassa PlayerUnknown’s Battleground, Fortnite sekä ARK: Survival Evolved.

2 UNITY

Unity on Unity Technologiesin vuonna 2005 julkaisema monialustainen ja laajasti käytetty pelinkehitysympäristö. Se tarjoaa intuitiivisen käyttöliittymän, editorin ja tehokkaan pelimoottorin, joka soveltuu moneen eri tarkoitukseen. Unity on kehitetty luomaan sekä 2D- että 3D-pelejä muun muassa Windowsille, Macille, Linuxille, mobiili- sekä konsolialustoille. Ohjelmointikielinä Unityssä käytetään C#-ohjelmointikieltä. Unity tarjoaa kolmea erilaista versiota ohjelmastaan: Pro (ammattilaisille ja pelistudiolle), Plus (harrastelijoille) sekä Personal (ilmainen versio aloittelijoille). Unityllä on kehitetty monia menestyneitä pelejä, joita ovat muun muassa Trinity, Escape from Tarkov sekä Hollow Knight.

2.1 Unity Hub

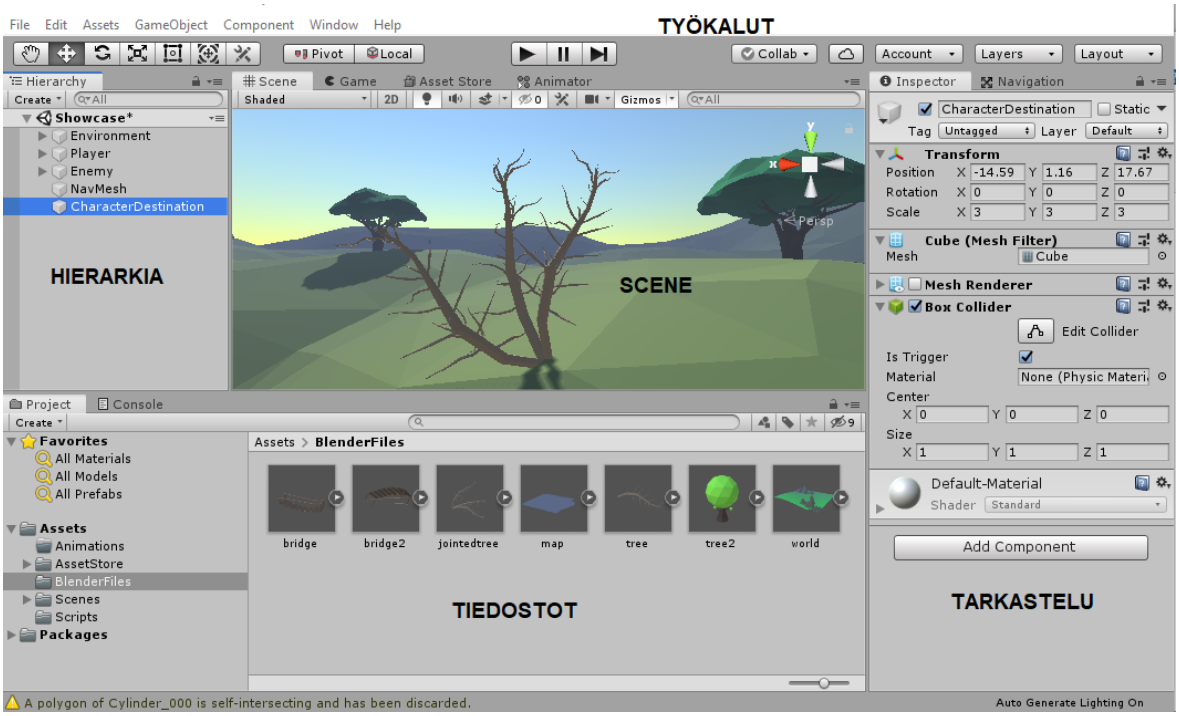
Unity Hub on Unityn uusi ja itsenäinen sovellus Unity-projektien ja versioasennuksien hallintaa varten (kuva 1). Sen avulla voidaan luoda uusia projekteja ja hallita jo olemassa olevia projekteja sekä niiden Unity-versioita sekä hallita eri Unity-versioiden asennuksia. Sovelluksen avulla voidaan myös suoraan määritellä projektin kohdealusta. Hubi tarjoaa myös hallintatyökalut omaa Unity-tiliä sekä editorin lisenssejä varten.



Kuva 1. Unity Hub

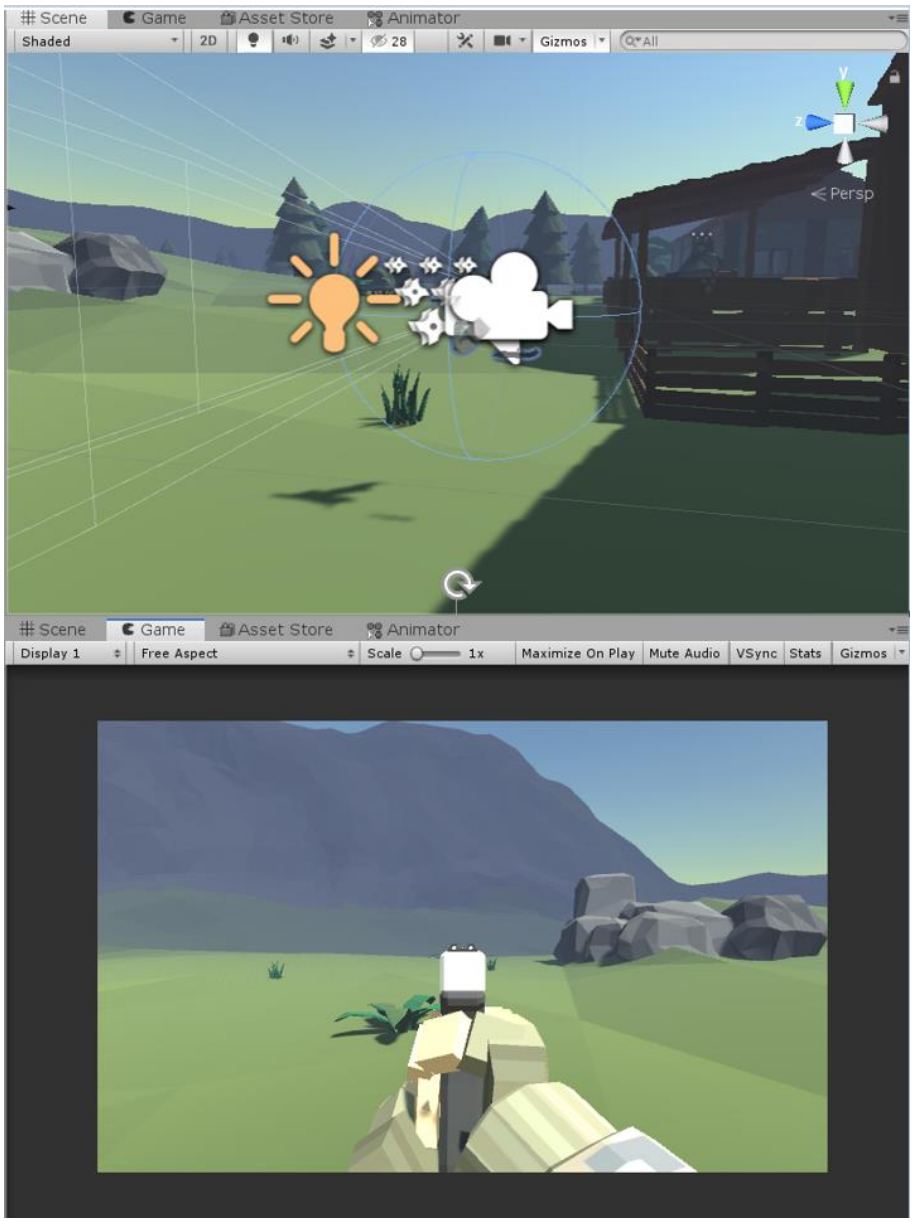
2.2 Editori

Unityn ulkoasu on selkeä ja tyypillinen vastaaville ohjelmistoille. Editori sisältää hierarkianäkymän, projektinäkymän, tarkastelijanäkymän, työkalupalkin ja Scene-näkymän (kuva 2). Eri ikkunoiden paikkoja voi vaihtaa mieleisekseen asetuksista. Projekti-ikkuna sisältää projektin Asset-kirjaston. Assetit ovat mitä tahansa projektissa käytettyjä osia, esimerkiksi skriptit, 3D-mallit sekä animaatiot. Projekti-näkymässä voidaan hallita projektin Asset-tiedostoja.



Kuva 2. Unity editori

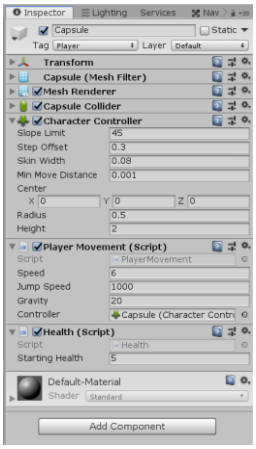
Scene-näkymä on kehittäjälle tarkoitettu näkymä, joka tarjoaa interaktiivisen näkymän siitä, mitä on tehty. Yksi Scene on ikään kuin yksi taso pelissä. Tässä näkymässä kehittäjä voi esimerkiksi lisätä ja valita eri peliobjekteja ja liikutella niitä vapaasti kentässä. Game, eli pelinäkymä sen sijaan on näyte siitä, mitä pelaaja näkee pelin ollessa käynnissä. Seuraavassa kuvassa on esitetty näyte Scene näkymästä ja pelinäkymästä (kuva 3).



Kuva 3. Scene ja pelinäkymä

Hierarkianäkymä on esitys projektin peliobjekteista valitussa kohtauksessa. Se on lista jokaisesta peliobjektista valitussa Scene-näkymässä. Projektinäkymästä voidaan suoraan siirtää objekteja hierarkianäkymään raahaa ja pudota-tyyliin.

Tarkastelunäkymä tarjoaa työkaluja hierarkianäkymästä valitun objektin ominaisuuksien muokkaamiseen (kuva 4). Kyseisessä ikkunassa editoidaan objektin ominaisuuksia, liitetään siihen komponentteja ja muokataan niiden arvoja tarpeen mukaan.

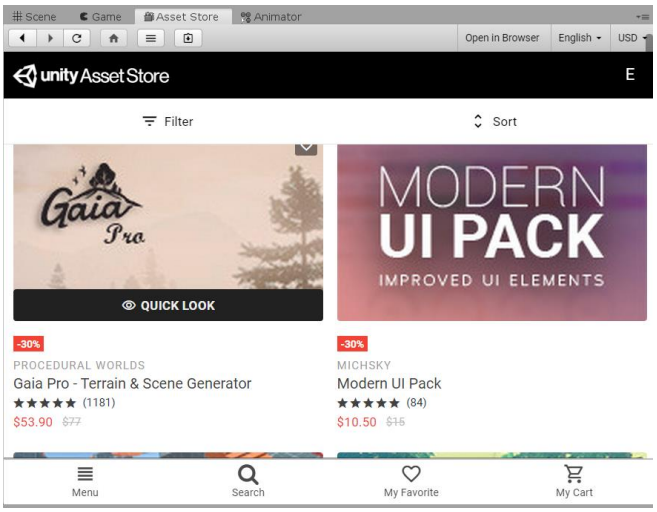


Kuva 4. Tarkastelunäkymä

Työkalupalkki editorin yläkulmassa tarjoaa valikon tärkeisiin ominaisuuksiin, joilla manipuloidaan Scene-näkymää ja sen sisältämiä objekteja. Näitä ovat muun muassa käsityökalu, siirtotyökalu, skaalaustyökalu sekä pelin toiston aloitus, pysäytys ja keskeytys.

2.3 Assetit ja Asset -kauppa

Assetit ovat mitä tahansa Unityn tukemia tiedostoja, joita käytetään projektissa. Näitä ovat muun muassa 3D-mallit, animaatiot, skriptit ja äänitiedostot. Assetteja voi luoda itse tai vaihtoehtoisesti ladata Unityn Asset-kaupasta (kuva 5), joka tarjoaa paljon ilmaisia ja maksullisia assetteja. Kauppaan pääsee käsiksi suoraan Unityn editorista tai vaihtoehtoisesti menemällä Unityn Web-sivuille selaimella.



Kuva 5. Asset-kauppa

2.4 Scene

Unityn Scene-näkymä sisältää pelin ympäristön sekä menun. Yksi Scene on ikään kuin yksi kenttä pelissä. Scenejä lisäämällä saadaan aikaan lisää ratoja tai voidaan vaihtoehtoisesti myös tehdä peliin erilaisia Menu-ikkunoita. Scene sisältää siis kaikki pelin maailmat, objektit ja ylipäänsä näkyvät elementit.

2.5 Peliobjektit ja komponentit

Kaikki peliin luodut asiat ovat nimeltään peliobjekteja. Näitä ovat muun muassa kamera, hahmot, valot ja tavarat. Itsessään peliobjektit eivät tee mitään, ennen kuin niille määritellään ominaisuuksia tarkastelunäkymässä. Peliobjektiin voidaan liittää erilaisia komponentteja ja skriptejä. Objektit sisältävät valmiiksi pakollisen Transform-komponentin, joka määrittää objektin sijainnin, asennon ja skaalautuvuuden näkymässä. Peliobjekti on näin ollen ikään kuin tyhjä säiliö erilaisille komponenteille. Peliobjektiin kiinnitettävät komponentit luovat toiminnallisuuden peliobjektiin. Peliobjekteille voidaan myös määrittää oma malli, tekstuuri, materiaali sekä monenlaisia eri efektejä. Erilaisia komponentteja käsitellään lisää luvuissa Fysiikka ja Mekaniikat sekä Grafiikka.

2.6 Prefab

Prefab tarkoittaa valmiiksi luotua, varastoitua ja konfiguroitua peliobjektia. Tällöin peliobjektille on valmiiksi määritelty komponentit ja ominaisuudet ja se on tallennettu yhdeksi kiinteäksi kokonaisuudeksi. Niiden tarkoitus on tehdä pelinkehityksestä sulavampaa. Sillä voidaan määrittää pelissä usein toistuville peliobjekteille eräänlainen valmisrunko, jota voidaan tarpeen mukaan muokata. Tällöin muutokset astuvat voimaan kaikissa objekteissa, joissa prefab-objektia on käytetty. Esimerkiksi erilaiset NPC-hahmot, toistuvat luonnon elementit ja muut usein toistuvat objektit ovat järkevää tehdä käyttäen prefab-tiedostoja. NPC-hahmolla tarkoitetaan kaikkia ei-pelattavia hahmoja pelissä. Prefab-tiedosto luodaan yksinkertaisesti siirtämällä valmis peliobjekti hierarkianäkymästä projektinäkymään.

2.7 Käyttöliittymätyökalut

Unity tarjoaa kattavan kokoelman valmiita UI eli käyttöliittymäelementtejä käytettäväksi pelin käyttöliittymän rakentamiseen. Seuraavassa kuvassa on esimerkki UI-ikkunasta (kuva 6).

Kuva 6. Käyttöliittymä esimerkki (Unity Manual 2019)

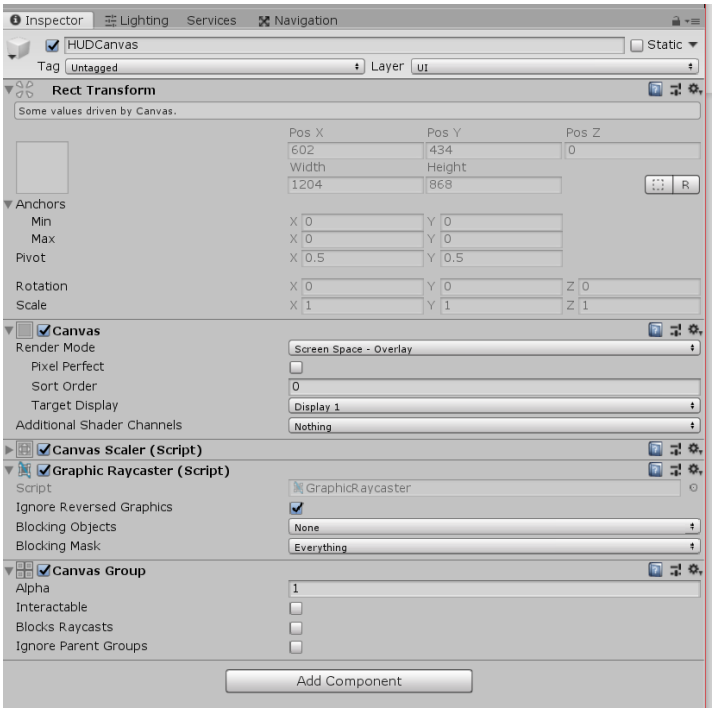
2.7.1 Canvas

Canvas-peliobjekti on tyhjä kanvaasi, johon taiteilija maalaa teoksensa. Se on suorakulmion muotoinen alue, jonka sisälle kaikki UI-elementit sijoitetaan. Kyseistä objektia käyttäen voidaan rakentaa käyttöliittymiä peliin.

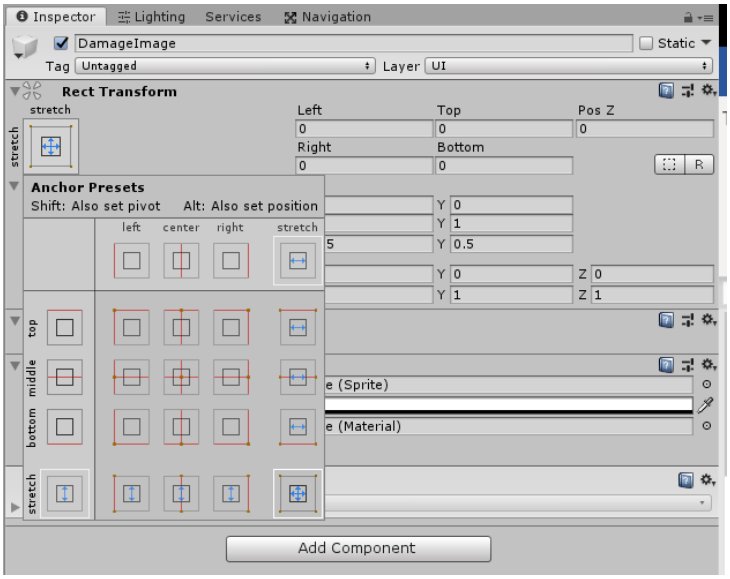
Kaikki Canvas-objektin sisälle lisätyt objektit ovat sen lapsiobjekteja. Canvas-elementti on peliruudun päällä oleva kalvo, jossa näkyy kiinteitä elementtejä, jotka puolestaan ovat näkyvästi esillä pelissä. Tämän avulla voidaan esimerkiksi luoda interaktiivisia näkymiä, jotka avautuvat toimintojen ollessa aktiivisia, kuten pelin aloitus- ja asetusnäkymä ja muut mahdolliset visuaaliset elementit. On huomioitava, että Canvas-elementtien piirto näytölle tapahtuu samassa järjestyksessä kuin ne esiintyvät hierarkiassa, eli ensimmäisenä hierarkiassa esiintyvä elementti piirretään ensimmäisenä.

Objekti sisältää kokoelman erilaisia komponentteja, joiden avulla määritetään sen koko, sijainti, piirtotapa ja käyttäytyminen (kuva 7). Renderöinti- eli piirtovaihtoehtoja on kolmea erityyppistä, jotka ovat päällystys (screen space overlay), kamera (screen space camera) sekä maailmatila (world space). Kokonäytön päällystys on nimensä mukaisesti koko näyttötilan peittävä alue, joka näytön resoluution muuttuessa skaalatutuu automaattisesti laitteelle sopivaksi. Kameratila on samankaltainen kuin näytön peittävä tila, mutta tässä

tapauksessa Canvas-elementti piirretään määritetyn etäisyyden päähän valitusta aktiivisesta kamerasta, jolloin kamera piirtää UI-elementit näytölle. Maailmatilassa Canvas-elementti käyttäytyy kuten muut objektit näkymässä ja näin ollen sen koko ja sijainti voidaan määrittää manuaalisesti käyttäen Rect Transform -työkalua, joka näkyy alla olevassa kuvassa (kuva 8). Kyseistä piirtotapaa käytetään silloin, kun UI-elementtien on tarkoitus näkyä osana pelimaailmaa, esimerkiksi pelaajan henkien tai kerättyjen esineiden näyttämiseen.

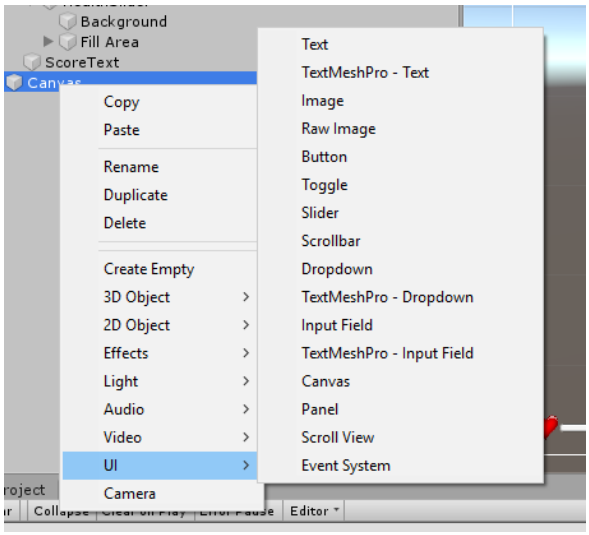


Kuva 7. Canvas-ominaisuudet



Kuva 8. Rect Transform

Canvas-objektiin voidaan kiinnittää erilaisia komponentteja, jotka voivat olla esimerkiksi kuvia, tekstiä, animaatioita, efektejä sekä erilaisia interaktiivisia komponentteja, kuten painikkeita ja säätimiä. Eri vaihtoehtoja kuvattu alla näkyvässä kuvassa (kuva 9).

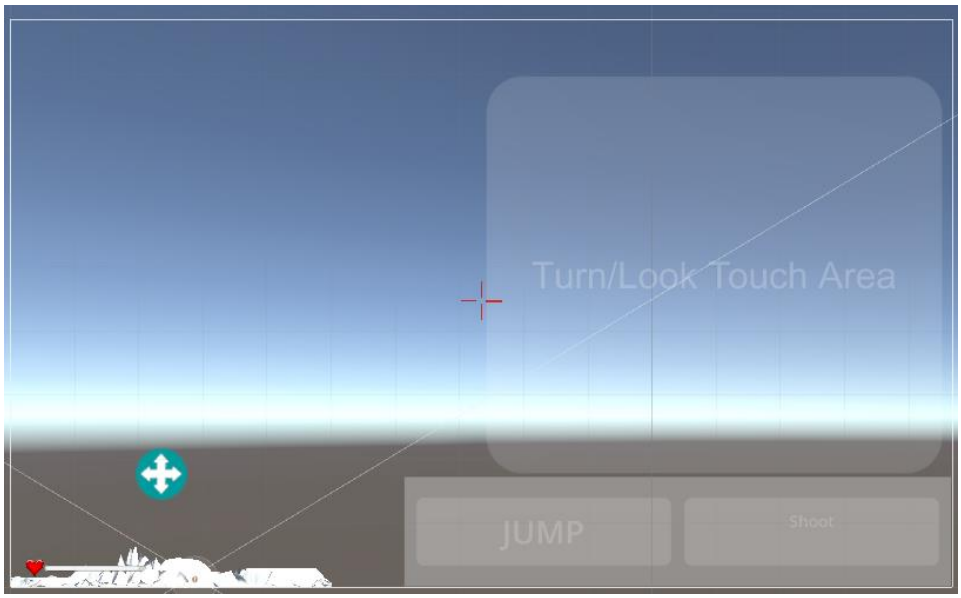


Kuva 9. Canvas-lapsiobjektit

2.7.2 Syöte

Unity tukee tyypillisimpiä syötetapoja, kuten näppäimistöä ja hiirtä, erilaisia peliohjaimia, kosketusnäyttöjä, mobiililaitteiden liiketunnistuskykyä sekä AR ja VR -teknologiaa. Työssä keskitytään mobiilipelin ominaisuuksiin, joten syötetavoista perehdytään vain kosketusnäyttöön perustuvaan syötteeseen.

Mobiililaitteelle pelinsyötteen ohjelmointi on hivenen monimutkaisempaa kuin perinteisille syötelaitteille, mutta Unity tarjoaa itsessään automaattisia tapoja toteuttaa kosketukseen perustuva toimintaa. Unityn Asset-kauppa paikalla on myös kattavasti ilmaisia sekä maksullisia vaihtoehtoja kosketusnäyttösyötteen toteutukseen. Seuraavassa kuvassa on esimerkki runko kosketusnäyttö kontrolleille (kuva 10).

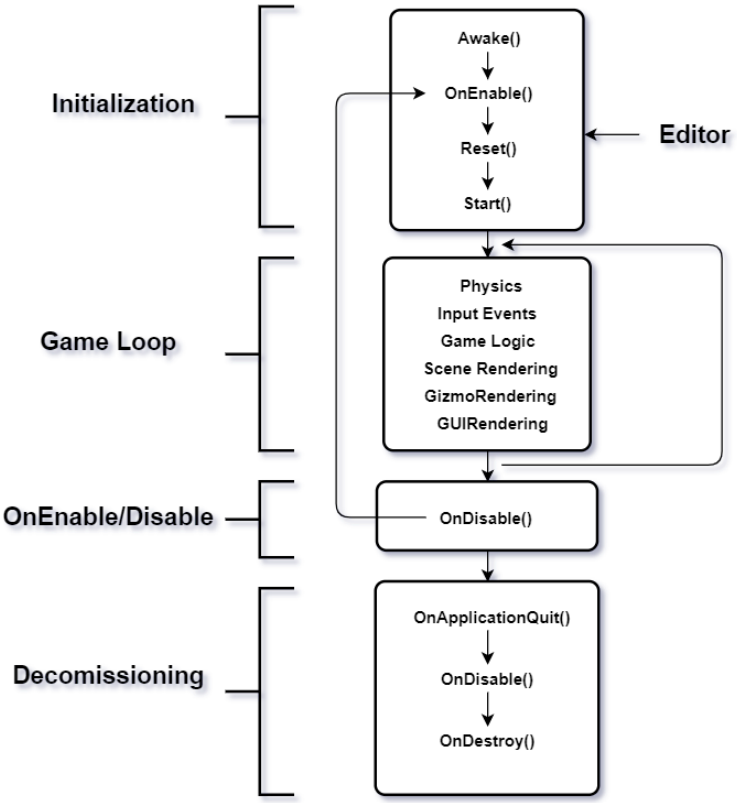


Kuva 10. Kosketusnäyttö kontrollit

Unityssä on sisäänrakennettu syöteluokka, joka tarjoaa pääsyn kosketusnäyttöön ja sen elementteihin, kiihtyvyysanturiin sekä sijaintitietoihin. Kiihtyvyysanturin datan avulla voidaan selvittää lineaariset kiihtyvyysmuutokset xyz-akseleilla kolmiulotteisessa tilassa, joita voidaan hyödyntää tunnistamaan mobiililaitteen kallistuminen suhteessa maahan ja sen muutokset. Käytännössä tämä viittaa siihen, kun pelaaja kallistaa mobiililaitetta ja sovellus tunnistaa kallistuksen ja hyödyntää kyseistä informaatiota esimerkiksi liikkumissa. Sijainti -tietoja puolestaan voidaan hyödyntää esimerkiksi virtuaalitodellisuutta mukauttelevissa peleissä, kuten valtavan suosion saavuttaneessa Pokemon Go:ssa.

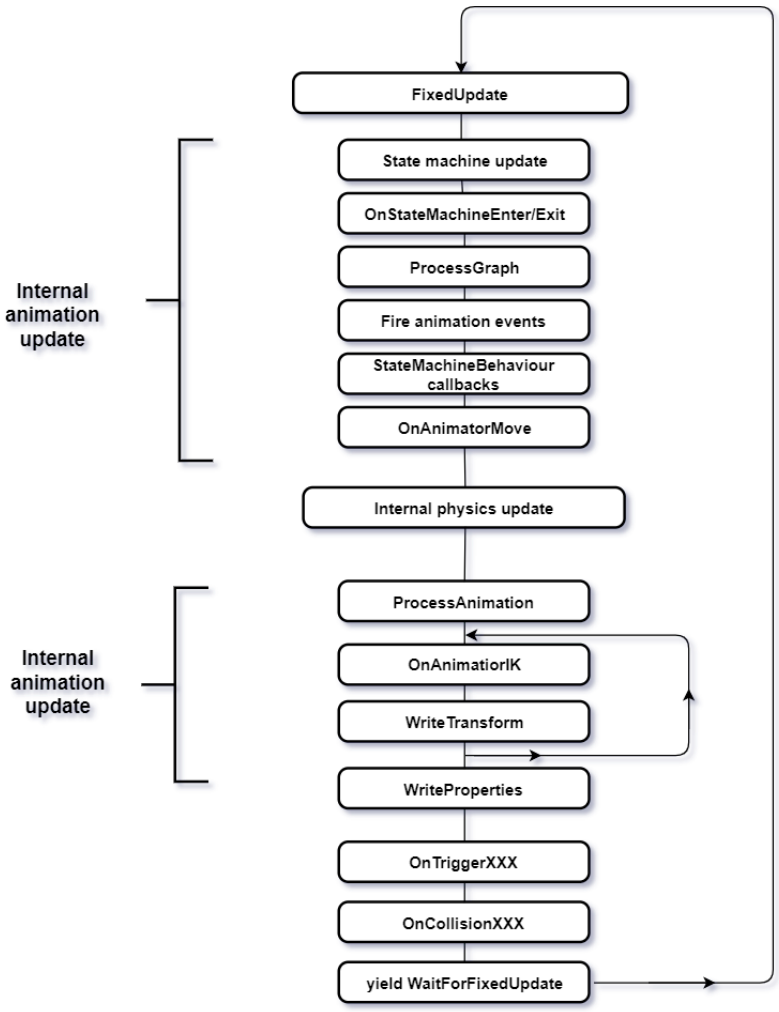
2.8 Pelisilmukka

Seuraavassa kuvassa on esitetty Unityn pelin elinkaari (kuvio 11). Awake()-funktion tarkoitus on initialisoida kaikki skriptin muuttujat ja tilat ennen kuin peli käynnistyy. Sitä kutsutaan ensimmäisenä ja vain kerran skriptin elinkaaressa ja vasta kun kaikki objektit on aktivoitu. OnEnable()-funktio, jota kutsutaan Awake()-kutsun jälkeen, mutta ennen Start()-kutsua, kutsutaan joka kerta, kun objekti on otettu käyttöön ja on aktiivinen. Tämän jälkeen kutsutaan Start()-funktiota, joka suoritetaan myös vain kerran skriptin elinkaaren aikana. Start()-funktio on myös alustusfunktio ja sitä kutsutaan vain, kun skriptin komponentti on aktivoitu.



Kuvio 11. Unity pelin elinkaari

Funktiot, jotka pitävät sisällään pelin logiikan, toiminnot, animaatiot sekä kaiken muun toiminnallisen puolen pelissä, ovat Update()- ja FixedUpdate()-funktiot. Update()-funktio on yleisimmin käytetty funktio pitämään sisällään toiminnot, jotka eivät pohjaudu fysiikkamoottoriin. Näitä ovat esimerkiksi syötteenhallinta, ajastimet, animaatiot ja yleinen pelin logiikka. Funktio ajetaan ennen ruudun renderöintiä. FixedUpdate()-funktiota käytetään, kun toiminnot pohjautuvat fysiikkaan, sillä se synkronoituu fysiikkamoottorin kanssa. Funktiota kutsutaan ennen jokaista sisäistä fysiikkapäivitystä 20 ms:n välein, mikä tarkoittaa, että FixedUpdate()-funktio kutsutaan 50 kertaa sekunnissa. Haluttu fysiikan päivitysväli voidaan asettaa Unityn projektiasetuksissa manuaalisesti.



Kuvio 12. Fysiikan elinkaari

Pelin elinkaaren lopussa suoritetaan funktiot, jotka pysäyttävät kaiken toiminnan. OnApplicationQuit()-funktio kutsutaan kaikille pelissä oleville objekteille ennen kuin peli suljetaan. OnDisable() sekä OnDestroy()-funktiot kutsutaan objektin tuhoamisen seurauksena, kun peli sulkeutuu. On huomioin arvoista, että OnApplicationQuit()-kutsua ei voida tehdä, mikäli Android käyttöjärjestelmä sulkee ohjelman johtuen Androidin omasta elinkaari mallista.

2.9 Fysiikka ja Mekaniikat

Kappaleessa käsitellään fysiikkaan pohjautuvia elementtejä sekä yleisiä pelin mekaniikkoja, jotka ovat oleellisia mobiilipelissä.

2.9.1 Character Controller ja RigidBody

Peliä kehittäessä on tärkeä tehdä valinta, käyttääkö pelaajan hahmo Character Controller-komponenttia vai RigidBody-komponenttia. Kyseinen valinta määrittää, kuinka hahmo liikkuu ja miten liikkuminen on toteutettu. Molemmat tavat tarjoavat käytännössä lähes samat mahdollisuudet, mutta toteutustavat eroavat toisistaan merkittävästi.

Character Controller-komponentti mahdollistaa ympäristössä liikkumisen, joka hyödyntää Collidereita eli niin sanottuja törmäyspintoja. Character Contoller-komponentin mukana objektiin kiinnittyy Capsule Collider-komponentti, joka tarjoaa valmiin törmäysrajapinnan objektille. Character Contoller-komponentti ei reagoi pelimoottorin fysiikkaelementteihin ollenkaan. Se ei siis reagoi voiman suureisiin, joita ovat muun muassa painovoima ja kitka. Näin ollen kehittäjän tulee ohjelmoida fysiikka lähtökohtaisesti itse haluamallaan tavalla. Käytettäessä kyseistä komponenttia saadaan paljon vaikutusmahdollisuuksia siihen, miten hahmo liikkuu maailmassa ja reagoi törmätessään toisiin objekteihin. Tämä mahdollistaa esimerkiksi parkour tyyppisen liikkumisen. Sen mukana tulee hyödyllinen kokoelma valmiita funktioita liikkumisen ohjelmointiin, joita ovat mm. SimpleMove ja Move. SimpleMove on yksinkertainen liikkumisfunktio, joka ottaa parametrikseen vain nopeuden ja liikuttaa objektia sen mukaisesti. Se myös tarjoaa valmiiksi määritetyn painovoiman, joten sitä ei erikseen tarvitse rakentaa. SimpleMove ei kuitenkaan tarjoa mahdollisuutta vaikuttaa y-akselin suuntaiseen nopeuteen. Mikäli halutaan enemmän joustavuutta hahmon liikkumisen toteuttamiseen, voidaan käyttää metodia Move(). Se ottaa huomioon niin sanotun absoluuttisen liikkumisen ja on sen myötä myös riippuvainen ruudunpäivitysnopeudesta eikä sisällä itsessään valmista painovoimaa.

RigidBody-komponentti puolestaan ei nojaudu törmäysrajapintoihin vaan itse pelimoottorin sisäiseen fysiikkaan. Kyseinen komponentti tarjoaa objektille fyysiset ominaisuudet ja sen tuomat rajoitteet automaattisesti. Kyseisiä ominaisuuksia ovat muun muassa painovoima, kitka, ja massa. Näin ollen kappale reagoi realistisesti törmätessään tai ollessaan vuorovaikutuksessa muiden kappaleiden kanssa riippuen siihen kohdistuvista voimista. Kyseisiä voimia voidaan muokata ohjelmallisesti peliin sopivaksi.

2.9.2 Törmäimet

Törmäinkomponentit (Colliders) määrittävät objektille törmäyspinnan sen kohdatessa toisia objekteja. Esimerkiksi kun hahmo seisoo tasaisella pinnalla, on molemmilla objekteilla oltava törmäinkomponentti kiinnitettynä, jotta hahmo ei esimerkiksi tipu seisottavan pinnan läpi. Kyseiset komponentit ovat näkymättömiä ja tarjolla on eri muotoisia valmiita vaihtoehtoja, joista valitaan suunnilleen oikean muotoinen vaihtoehto määrittämään

törmäyspinta tarvittavalle peliobjektille. Esimerkiksi useimmiten ihmismäisen pelaajahahmon törmäinkomponentiksi valitaan Capsule Collider, joka on nimensä mukaisesti kapselin muotoinen (kuva 13).

Kuva 13. Kapselitörmäin

Törmäimen ei tarvitse olla täsmälleen oikeanmuotoinen, vaan lähinnä suuntaa antava riippuen siitä, miten peliobjektin halutaan reagoivan ympäristöönsä. On tyypillistä, että väärässä kohdassa tai vääränkokoinen törmäinkomponentti aiheuttaa erilaisia virheitä hahmon liikkumiseen. Rajapintaa on kuitenkin helppo muokata toimivaksi. On huomioitavaa myös, että törmäinkomponentteja voidaan yhdistellä, mikäli tarvitaan tarkempaa törmäyspintaa hahmolle, esimerkiksi mukailemaan hahmon raajoja. On olemassa myös Mesh Collider-komponentti, jolla voidaan luoda täydellisesti objektin muotoja mukaileva törmäyspinta. Tyypillisimpiä Collider-komponentteja ovat Box Collider, Capsule Collider sekä Sphere Collider. Mitä monimutkaisempi ja tarkempi törmäyspinta objekteille tehdään, sitä enemmän se myös kuluttaa laitteen resursseja. Mobiilipeliä kehittäessä on tärkeää suunnitella, mikä tapa toimia antaa parhaan tuloksen käyttäen mahdollisimman vähän resursseja.

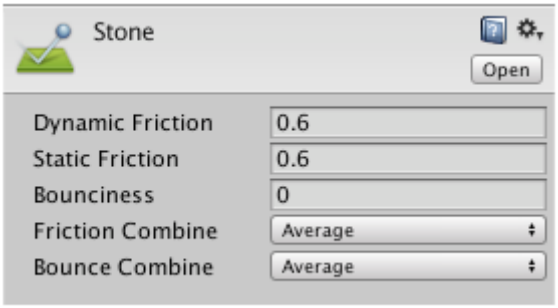
Törmäinkomponentit voidaan vielä jakaa staattisiin ja dynaamisiin. Mikäli komponentti on kiinnitetty staattiseen objektiin ilman RigidBody -komponenttia, kuten maa tai seinät, on komponentti staattinen eikä se tällöin liiku. RigidBody-komponentin omaaviin objekteihin liitetyt törmäyspinnat ovat dynaamisia. Nämä kaksi erityyppistä törmäyspintaa voivat reagoida keskenään, mutta staattinen törmäyspinta ei koskaan liiku siihen törmättäessä.

Dynaaminen törmäyspinta sen sijaan voi reagoida liikkumalla osuessaan staattiseen törmäyspintaan.

Törmäyttimet tarjoavat törmäyspinnan lisäksi mahdollisuuden aktivoida tapahtumia Is Trigger -ominaisuuden avulla. Mikäli ominaisuus on aktivoitu, voidaan siihen luoda yhteys skriptin avulla aktivoimaan haluttu tapahtuma, mikäli kosketus kyseiseen objektiin tapahtuu. Tällainen tapahtuma voi olla esimerkiksi hahmon osuminen maassa olevaan piikkiin, joka aktivoi hahmon henkien määrän vähenemisen.

2.9.3 Fyysinen materiaali

Törmäyspintoja suunnitellessa tulee huomioida erilaiset fyysiset materiaalit (Physics Materials) ja miten ne käyttäytyvät. Pinnan tulee simuloida tilanteeseen sopivaa käyttäytymistä sen reagoidessa muiden objektien kanssa. Esimerkiksi voidaan ottaa tennispallo ja keilapallo. Tennispallo on kevyt ja pomppii korkeammalle ja erilaisella tahdilla, kun taas keilapallo on raskas ja reagoi heittoon ja törmäykseen lattian kanssa eri tavalla. Fyysinen materiaali siis määrittää materiaalin kitkan sekä pomppivuuden objekteissa, joille on määritetty törmäyspinta.

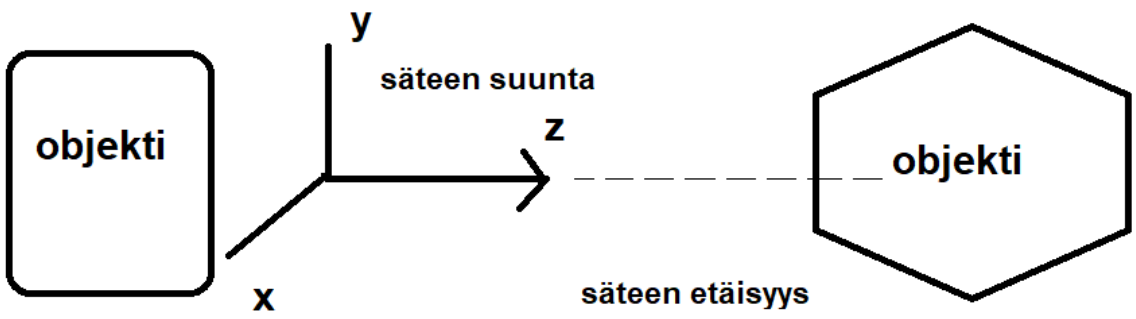


Kuva 14. Fyysinen materiaali esimerkki (Unity Manual 2019m)

2.9.4 Raycasting

Raycasting eli niin sanottu säteen ammunta on nimensä mukaisesti näkymättömän säteen ampumista pisteestä A pisteeseen B, tunnistaen törmäykset sen osuessa objekteihin, joille on määritetty törmäinkomponentti (kuvio 15). Ominaisuus on hyödyllinen tehdessä ammuntapeliä, sillä itse ampuminen voidaan toteuttaa nojautuen kyseiseen mekaniikkaan. Tässä tapauksessa säteen alkupiste asetetaan hivenen kauemmas aseen kärjestä, jotta vältetään aseen ja ammuksen törmäyspintojen törmäys. Tämän jälkeen asetetaan haluttu määränpää piste, mikä on luontevinta asettaa suuntautumaan eteenpäin. Halutut ominaisuudet määritellään koodissa. Työn käsittelemässä mobiilipelissä säteen osuessa vihollishahmoon, aiheuttaa osuma vihollisen henkien vähenemistä. Osuessaan peliobjektiin, se

joko tuhoaa objekteja ympäristössään tai aiheuttaa vain pienen visuaalisen efektin indikoidakseen, että objektiin on osuttu. Mobiilipelissä on myös hyödynnetty sädeammuntaa vihollishahmon liikkumisessa navigaatiokartalla.



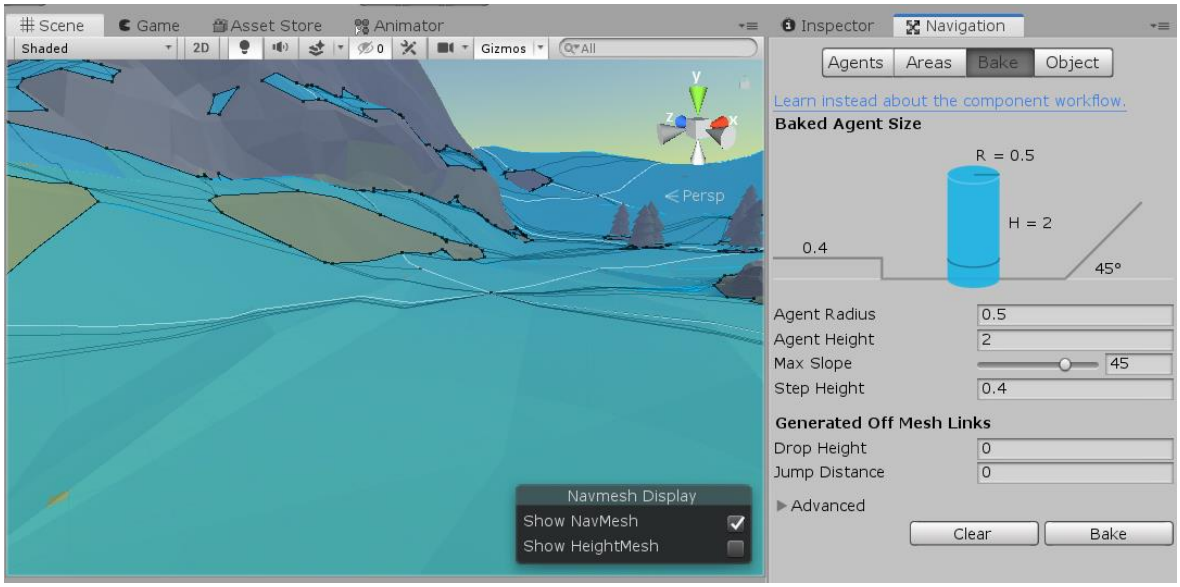
Kuvio 15. Kuvituskuva sädeammunnasta

2.9.5 Tekoäly ja navigaatio

NPC ja vihollishahmoja tehdessä tulee suunnitella, kuinka toteuttaa tekoälyn kaltainen toiminnallisuus ei-ohjattaville hahmoille. Kyseinen mobiilipeli luo vaatimuksia tekoälyominaisuuksille, sillä hahmojen tulee olla vuorovaikutuksessa pelaajan kanssa, liikkua itsenäisesti maailmassa, havaita pelaaja, hyökätä ja perääntyä sekä lähettää signaaleja muille vihollishahmoille kohdatessaan pelaajan.

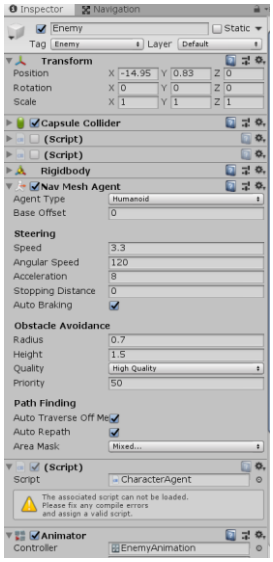
Hahmon liikkuminen on toteutettu käyttämällä Unityn tarjoamaa navigaatio mahdollisuutta, NavMeshiä (navigaatioverkko) (kuva 16). Navigaatioverkko on abstrakti tietorakenne, jota käytetään tekoälyä rakentaessa tarjoten polunetsintä mahdollisuus agentille. Yksinkertaisesti sanottuna navigaatioverkko siis esittää aluetta, jolla agentti voi liikkua. Termi agentti, viittaa autonomiseen kokonaisuuteen, joka toimii ohjaamalla omaa toimintaansa määritettyjen tavoitteiden saavuttamiseksi. Navigointiverkko koostuu kokoelmasta kaksiulotteisia kuperia monikulmiota (two-dimensional convex polygons), jota peittävät alueen, jolla agentit kulkevat. Navigaatioverkon rakentaminen tapahtuu Unityssä automaattisesti. Prosessia kutsutaan NavMesh Bakingiksi, eli navigaatioverkon leipomiseksi. Prosessissa kerätään kaikki pelin staattiset objektit. Näitä ovat esimerkiksi terrain objektit ja renderöinti verkot. Staattisella peliobjektilla tarkoitetaan objektia, joka ei liiku. Näin ollen kaikki halutut pelin kiinteät osat leivotaan navigaatioverkkoon kiinni. Valmiiksi työstetty navigaatioverkko

näkyy sinisenä alueena scenessä. NavMesh-komponenttiin voidaan kiinnittää muita sen tarjoamia komponentteja, mikäli tarvitaan lisää muokkausvaihtoehtoja. Kyseisiä komponentteja ovat mm. NavMesh Surface, NavMesh Modifier, NavMeshModifierVolume sekä NavMeshLink.



Kuva 16. NavMesh-komponentti

NavMesh Agent-komponentti on agentti, joka kulkee navigaatioverkossa määritetyllä tavalla. Kyseinen komponentti lisätään haluttuun hahmoon ja näin ollen siitä tulee agentti. Komponentti on vastuussa polunetsinnästä ja hahmon liikkumisesta, joka toteutetaan pääsääntöisesti skriptien avulla. Toisinaan navigaatiopisteitä (kohde, määränpää) voidaan luoda lisäämällä objekteja alueelle, joita kohti agentti liikkuu määritysten mukaisesti.



Kuva 17. EnemyAgent-komponentti

Mobiilipelissä vihollisagentti (kuva 17) on määritetty liikkumaan sattumanvaraisesti maailmaan leivotulla navigaatioalueella. Näin ollen agentti valitsee sattumanvaraisesti päämäärään navigaatioverkosta ja kulkee navigaatiopisteeseen, kunnes prosessi toistuu taas alusta. Näin ollen alueella on monia vihollisagentteja, jotka liikkuvat sattumanvaraisten navigaatiopisteiden välillä, kunnes pelaaja saapuu tarpeeksi lähelle ja agentit aktivoituvat ja aloittavat hyökkäyksen. Pelissä on myös muita tilanteita, jotka aktivoivat vihollisagentit.

Unityyn on saatavilla beetavaiheessa oleva avoimenlähdekoodin laajennus nimeltään The Unity Machine Learning Agents SDK (ML-Agents). Laajennus mahdollistaa älykkäiden agenttien kouluttamisen. Laajennus tarjoaa mahdollisuuden opettaa agentteja laajemmin, vahvistaa jo opittua, opettaa neuroevoluutiota sekä hyödyntää muita yleisiä koneoppimismenetelmiä Python API -sovelluksen avulla. Lisäosa mahdollistaa pelien ja simulaatioiden toimimisena älykkäiden agenttien kouluttamisessa. Kouluttamisella tarkoitetaan tekoälyn kehitystä, agenttien oppimisen matkimista, neuroevoluutiota.

2.10 Grafiikka

Tässä luvussa käsitellään grafiikkaan liittyviä elementtejä ja komponentteja sekä grafiikan optimoimista mobiilipelissä. Peliä tehdessä mobiililaitteille tulee kehittäjän huomioida mobiililaitteiden laajakirjo ja eroavaisuudet tehokkuudessa. Kyseessä on tehoiltaan tietokoneeseen verrattuna rajoittunut laite, joten optimointi on syytä ottaa huomioon alusta alkaen. Mobiilipelit käyttävät nykyään näyttävämpää grafiikka, sillä mobiililaitteet ovat parantuneet huomattavasti, mutta ne eivät silti kykene samaan kuin tietokoneet ja konsolit. Yksi isoista haasteista mobiililaitteilla on virrankulutuksen hallinta, mikä on iso rajoite upeille grafiikoille.

2.10.1 Grafiikan piirtäminen

Yleisesti ottaen renderöinti tarkoittaa kuvan, videon, animaation tai äänen prosessointimenetelmää, jossa laite muodostaa saamastaan datasta ohjelmallisen kuvauksen. Renderöinti on siis grafiikan piirtämistä näytölle. Renderöinti lisää ns. rautalankamalliin pinnan, tekstuurin ja valaistuksen. (TechTerms, Rendering 2016.) Tässä työssä prosessista käytetään pääsääntöisesti termiä piirto. Piirtotapoja on erilaisia, joita ovat esimerkiksi etukäteen tapahtuva piirto sekä reaaliaikainen piirto. Peleissä yleensä käytetään reaaliaikaista renderöintiä, joka hyödyntää laitteen prosessoria. Peliobjektia näytölle piirrettäessä täytyy pelimoottorin lähettää erikseen piirtopyyntö Grafiikka API: in (kuten OpenGl tai Direct3D), joka puolestaan käyttää huomattavasti resursseja pyynnön käsittelyssä. Mitä enemmän objektissa on käytetty erilaisia elementtejä, kuten eri materiaaleja, lisää se piirtopyynnön kuormittavuutta, sillä näytönohjaimen puolestaan täytyy validoida ja kääntää kyseiset

vaiheet. Unityssä on mahdollista optimoida grafiikkaa käyttämällä Dynaamista tai staattista piirtoa (batching). Dynaaminen piirto soveltuu erityisesti pienille malleille, sillä se muuntaa mallin monikulmioiden kärjet ja ryhmittelee samantyyppiset kärkipisteet yhteen kerralla piirrettäväksi. Staattinen piirto puolestaan yhdistää staattiset eli paikallaan pysyvät objektit yhdeksi isoksi kokonaisuudeksi. Objektien tulee olla merkitty staattisiksi manuaalisesti, jotta staattinen piirto on mahdollista. Peliobjektien piirto voidaan myös optimoida manuaalisesti, mikäli tarve niin vaatii.

Unityssä kamera objekti on vastuussa grafiikan piirtämisen näytölle oletusarvoisesti. Unity tarjoaa monia eri keinoja käyttää kameraa ja hyödyntämällä kameran säätömahdollisuuksia voidaan keventää laitteen prosessorin kuormitusta. Käyttämällä dynaamista resoluutiota kamerassa, voidaan skaalata yksittäisiä piirrettäviä kohteita ja laskea resoluutiota paikoittain ylläpitääkseen tasaista ruudunpäivitysnopeutta. Dynaaminen resoluutio ei kuitenkaan ole tuettu kaikissa Android laitteissa. Käyttämällä Occlusion Culling -ominaisuutta, voidaan puolestaan vähentää turhia objektien piirtokutsuja. Occlusion Culling tarkoittaa menetelmää, jossa kamera piirtää vain kohteet, jotka ovat sille sillä hetkellä näkyvissä jättäen muut kohteet huomioimatta. Edellä mainitussa tilanteessa Unity käyttää virtuaalista kameraa rakentaakseen hierarkian mahdollisesti näkyvistä objekteista ja prosessoi informaation, jota käytetään ajon aikana. Ominaisuutta hyödyntämällä saadaan parannettua pelin suorituskykyä.

Peliobjektille määritelty materiaali, tekstuuri ja varjostimet vaikuttavat piirtotehokkuuteen. Materiaali määrittää, kuinka objektin pinta tulee piirtää, ja varjostin sisältää matemaattiset laskutoimitukset ja algoritmit siitä, kuinka jokainen pikseli objektin tekstuurissa väritetään ja piirretään lopulta näytölle. Värien piirtoon vaikuttaa pelissä vallitseva valaistustyyppi ja objektin materiaali. Tekstuuri itsessään on pelkkä bittikartta-kuva. Tekstuureja voidaan yhdistää yhdeksi isoksi tekstuuriksi, jota materiaalit käyttävät (texture atlas). Yhdellä materiaalilla on yksi ohjelmoitu varjostin, joka puolestaan katsoo, mitä ominaisuuksia materiaalilla on ja toimii sen mukaan. Unity sisältää sisäänrakennetun valmiin varjostimen, jolla on kattavasti säädettäviä ominaisuuksia. Unity tarjoaa myös yksinkertaistettuja mobiilipeleille suunnattuja varjostimia.

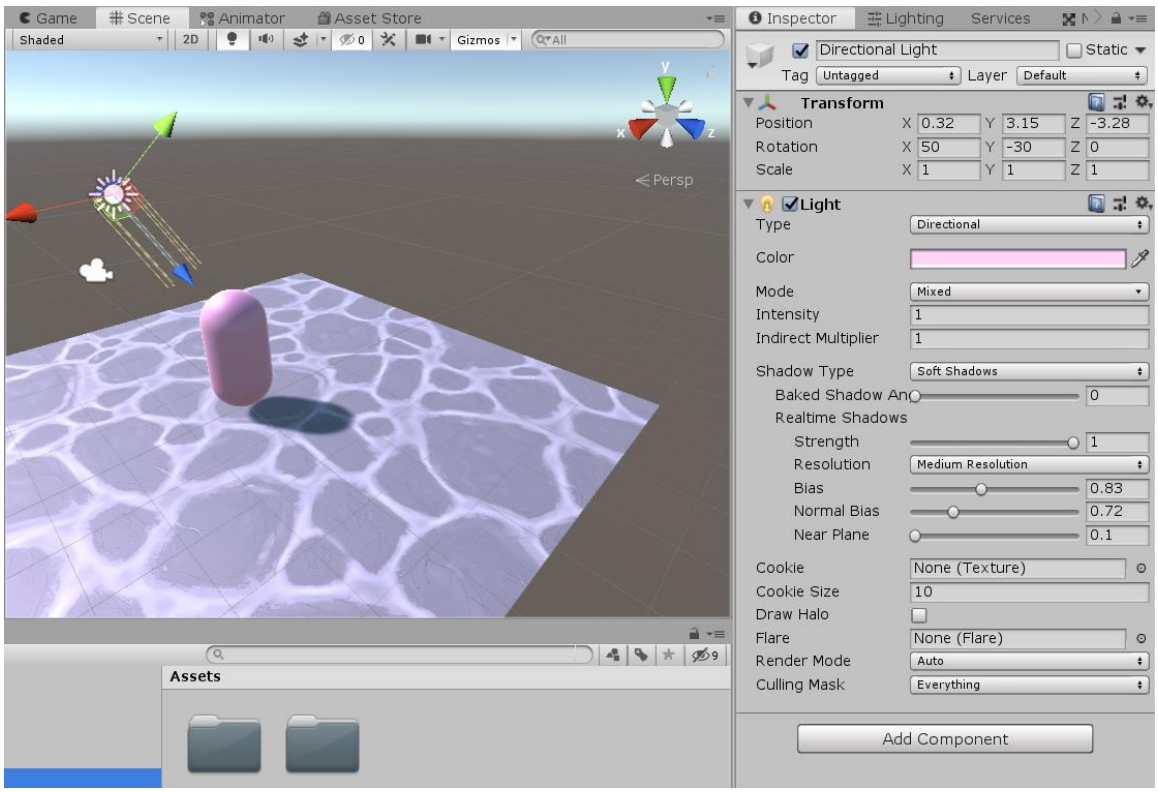
Yksi merkittävimmistä valinnoista pelinteossa on myös sen graafinen ilme. Kuinka monesta monikulmiosta (polygon) 3D -mallit koostuvat? Mitä enemmän monikulmioita malleissa on, sitä enemmän siinä on piirrettävää ja se käy raskaaksi laitteelle. Unityn suosituksen mukaan mobiilipelissä monikulmioiden määrän mallissa tulee olla väliltä 300 – 1500. Vertailun vuoksi suositus tietokonepeleissä on 1500 – 4000 (Unity Documentation.

Modeling characters for optimal performance 2017). Työssä käsiteltävä mobiilipeli sisältää vain 4 – 30 monikulmiota mallia kohden.

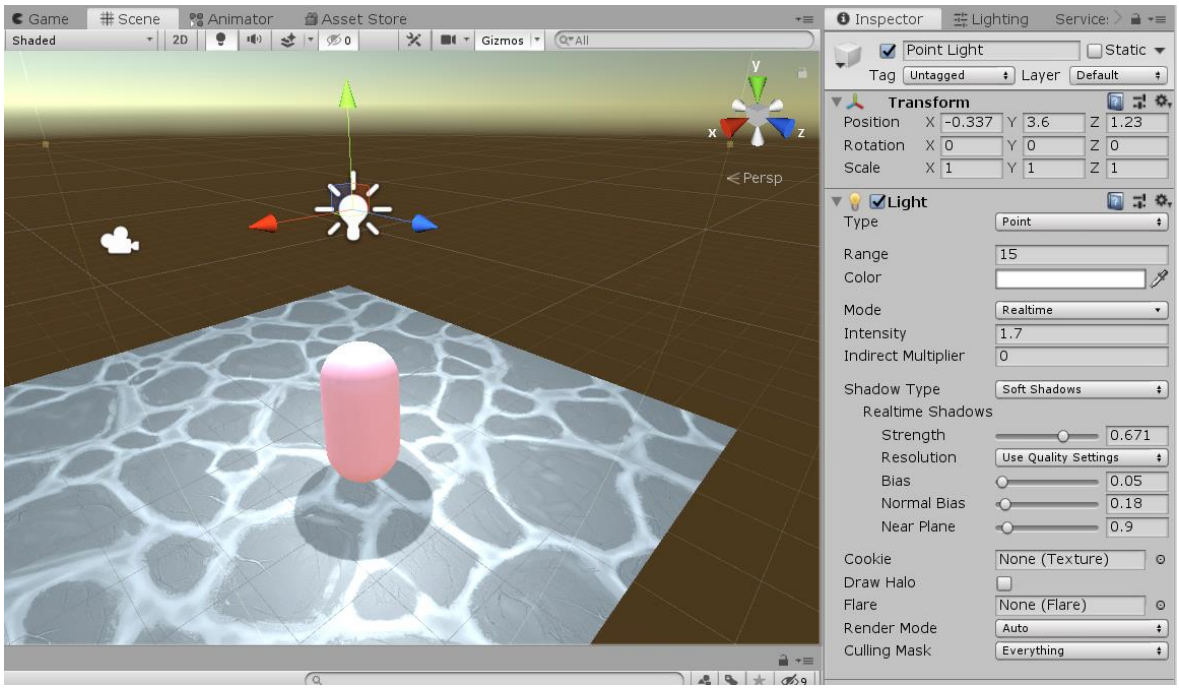
2.10.2 Valaistus

Yleisvalaistus (GL, global illumination) kuvaa tekniikoita ja matemaattisia malleja, joilla simuloidaan valoja ja varjoja pelimaailmassa. Valaistuksen toteuttamiseen on monia eri vaihtoehtoja ja sopivan tekniikan valitseminen mobiilipeliin on tärkeää, sillä se on yksi eniten tehoja kuluttava graafinen elementti. Tyypillisesti valaistuksen vaatimat laskutoimitukset tehdään tästä syystä etukäteen sen sijaan, että laskut tapahtuisivat reaaliaikaisesti pelissä.

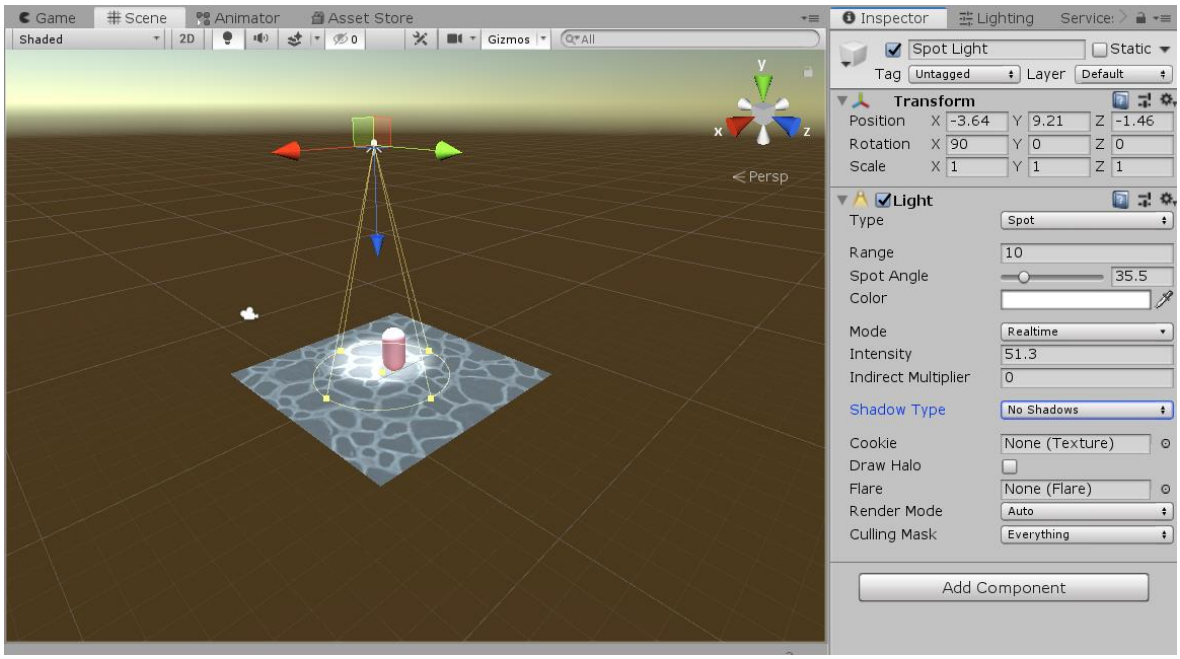
Oletusarvoisesti valaistus Unityssä on reaaliaikaista. Näitä ovat mm. suunta, spotti ja pistevalaistus. Niiden toiminta perustuu siihen, että komponentit tuottavat valoa kohtisuorasti valittuun kohteeseen ja päivittyvät jokaisella ruudulla. Varjoja voidaan muokata suoraan tarkastelijanäkymässä. Huomioitavaa on, että vähiten prosessointia vaativa varjotyyppi on vahvavarjo. Seuraavissa kuvissa esimerkkejä yllämainituista valoista (kuvat 18, 19, 20).



Kuva 18. Kohtisuoravalo



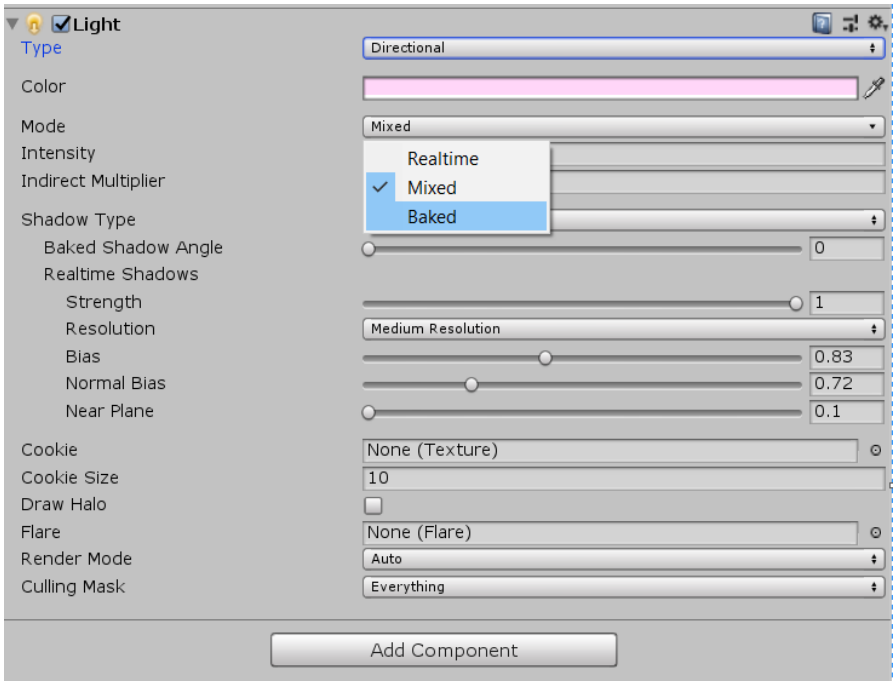
Kuva 19. Pistevalo



Kuva 20. Spottivalo

Mikäli halutaan hieman realistisempaa simulaatiota valaistuksesta, voidaan hyödyntää esilaskettuja valaistusratkaisuja. Näitä ovat niin sanotut leivotut valokartat. Valokartan (lightmap) leipominen tarkoittaa sitä, että valoefekti staattisiin objekteihin esi-lasketaan ja tallennetaan tekstuuriksi, joka peittää kohtauksen. Valokartoissa voidaan käyttää suunta-valoja, jotka osuvat objektiin ja valoja, jotka kimpoavat poispäin objektista johon valo juuri osui. Valon kimpoamista objektista ei kuitenkaan voida toteuttaa käyttämällä pelkkiä

suuntavaloja. Tämän tekniikan rajoite on sen staattisuus, mikä tarkoittaa sitä, että valoja ei voida muuttaa pelin aikaisesti.



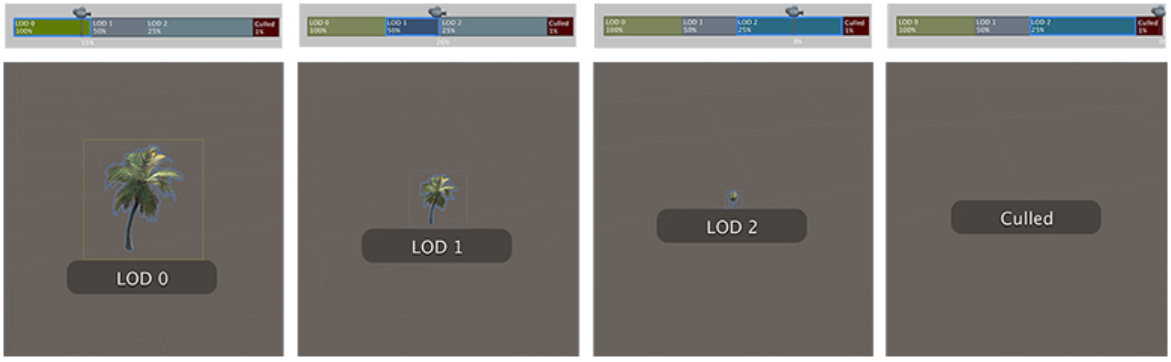
Kuva 21. Valaistuksen säädöt

Mikäli halutaan monimutkainen, reaaliaikainen ja interaktiivinen valaistus maailmaan, voidaan käyttää esilaskettua reaaliaikaista valaistusta (Precomputed Realtime GL Lighting). Kyseinen tekniikka mahdollistaa realistisen valaistuksen, heijastuksen, valon liikkumisen ja kimpoamisen objekteista. Kyseistä tekniikkaa on usein käytetty, kun toteutetaan peliin yö-päivä sykliä.

2.10.3 Level of Detail – LOD

LOD on lyhenne sanoista Level of Detail. Sillä tarkoitetaan sitä, kun pelissä lähellä olevat objektit näkyvät tarkemmin ja yksityiskohtaisemmin kuin kaukana olevat objektit. Eri tasojen yksityiskohtien tarkkuuksia määrittämällä voidaan merkittävästi vaikuttaa siihen, kuinka hyvin peli toimii mobiililaitteella. LOD-tekniikan avulla voidaan vähentää kolmiomuotojen määrää piirrettäessä peliobjekteja kameran ollessa kaukana objektista.

Peliobjekteihin voidaan kiinnittää LOD ryhmä-komponentti, jolla määritetään jokaiselle objektille oma LOD taso. LOD taso määrittää objektien prioriteetin piirtäessä kohtausta, joka toisin sanoen arvottaa objektit suhteessa kameraan määrittäen, millä tasolla yksityiskohdat näkyvät objektissa kameran etäisyyden mukaan. Seuraavassa kuvassa esitellään LOD ryhmää (KUVIO 22). Kuvassa näkyvä Culled taso tarkoittaa objektin jäämistä pois piirtoalueelta.



Kuva 22. LOD esimerkki (Unity Manual 2019)

3 MOBIILIPELIN TOTEUTUS

3.1 Idea

Mobiilipelin idea on toteuttaa Unity-pelimoottoria käyttäen Android-alustalle 3D-tekniikalla toteutettu ensimmäisestä persoonasta pelattava monipuolinen ja genrejä sekoitteleva seikkailupeli. Peli yhdistelee tasohyppelyä, taistelua sekä ongelmanratkaisua.

Pelissä pelaajan tulee selviytyä pelin kentistä taistellen erityyppisiä vihollisia vastaan, ratkaisten ongelmia ja lopulta löytäen ulospääsyn kentästä. Pelin tavoite on kerätä mahdollisimman paljon pisteitä mahdollisimman nopeasti sekä löytää jatkon kannalta tärkeäitä kerättäviä objekteja pelialueilta ratkaistakseen mysteerejä. Pelissä on myös kerättäviä aseita, esineitä sekä pelaajan elinvoimaa nostavia pakkauksia. Pisteitä kerätään ratkaisemalla pulmia, eliminoimalla vihollisia ja saavuttamalla lisäpisteitä riippuen kuinka nopeasti kukin kenttä on ratkaistu. Yksi merkittävä ominaisuus pelissä on sen vaihtuvat fysiikat, jotka vaikuttavat pelaajan liikkumiseen eri kentissä.

Tulevaisuudessa peli saa huomattavan määrän lisäominaisuuksia, kuten moninpelimahdollisuuden ja vaihtoehtoisia tehtäviä pelissä sekä erilaisia pelimuotoja. Pelihahmojen sekä maailman graafinen ilme valmistuu myös vasta toisessa versiossa.

3.2 Pelinkulku

Työssä tehty mobiilipeli on 3D-seikkailupeli, jossa isossa roolissa on taistelu. Pelissä on elementtejä tasohyppelystä sekä ongelmanratkaisusta. Peli alkaa siitä, kun pelaaja herää ja löytää itsensä vieraalta maaperältä. Pelaaja herää haavoittuneena, mikä tarkoittaa, että elinvoimamittarin henkiä indikoiva viiva on puolessavälissä ja pelaajan täytyy löytää syötävää tai lääkepakkauksia maastosta saadakseen lisää elinvoimaa. Elinvoimamittari kertoo pelaajalle, mikä on pelaajan terveydentila. Elinvoiman ollessa 50 – 100%, on elinvoimamittari vihreä. Elinvoiman ollessa alle 50%, muuttuu mittari punaiseksi. Pelaajan ollessa punaisella alueella, ei pelaaja pysty enää juoksemaan yhtä nopeasti kuin mittarin ollessa vihreä. Mitä lähemmäs 0% mittarin elinvoima näyttää, sitä hitaammaksi pelaajan liike muuttuu ja lopulta pelaaja kuolee, eli häviää pelin.

Pelaajalla on valmiina pistooli, jossa on täysi lipas ammuksia, sekä veitsi lähitaistelua varten. Aseiden merkitys pelissä on suuri, sillä pelimaailma on täynnä vihollisia. Uusia aseita ja ammuksia löytyy maastosta sekä erilaisista varustelaatikoista, joiden avaaminen vaatii nokkeluutta. Varustelaatikot voidaan avata ratkaisemalla arvoituksia tai vaihtoehtoisesti löytämällä maastosta avaimia. Pelin ensimmäisessä versiossa on vain yhdentyyppisiä vihollisia, joiden aggressiivisuustaso vaihtelee. Aggressiivisuustasolla tarkoitetaan sitä,

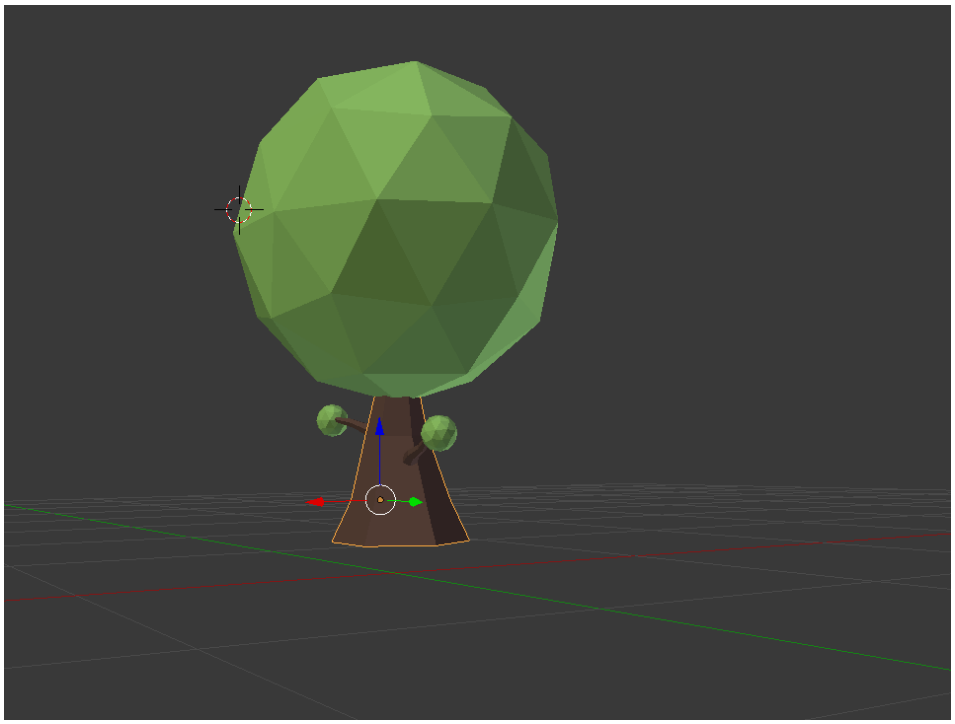
kuinka helposti vihollinen huomaa ja hyökkää pelaajan kimppuun. Se myös määrittelee, mitä vihollinen tekee huomatessaan pelaajan. Toiset viholliset huomaavat pelaajan jo kaukaa ja toiset vain läheltä. Toiset hyökkäävät saman tien, kun taas toiset informoivat muita vihollisia pelaajasta ja keräävät joukkoja etsimään pelaajaa. Vihollisilla on myös vaihteleva elinvoima, jolloin toiset on helpompi eliminoida kuin toiset. Eliminoimalla vihollisia, saa pelaaja kerättyä lisää pisteitä.

Pisteiden keräys on yksi pelin tavoitteista, hengissä pysymisen lisäksi. Pisteitä kerätään eliminoimalla vihollisia, ratkaisemalla varustelaatikoiden arvoituksia sekä päästessä kentistä läpi. Mitä nopeammin pelaaja selviytyy kentästä, sen enemmän pisteitä pelaaja siitä saa. Loppupisteet kentässä näkyvät lopulta pistetaulukossa, josta pelaaja voi arvioida suorituksiaan. Ominaisuus saa enemmän merkitystä pelin myöhemmässä kehitysvaiheessa, jolloin peliin implementoidaan moninpeliominaisuus.

Pelin ainoa haaste ei ole selviytyä hengissä ja eliminoida vihollisia. Lisähaastetta tuo itse pelikenttä, joka toisinaan muuttaa pelin fysiikkaominaisuuksia. Kyseinen ominaisuus on vielä keskeneräinen, mutta se tulee olemaan isossa roolissa pelissä. Esimerkiksi, mikäli pelaaja syö vääränlaisen sienen, muuttuu pelaajan liikkuminen täysin. Toisaalta myös astuminen tietylle määritetylle alueelle kentässä, saattavat painovoiman lait sekä liikkumisen kontrollit muuttua merkittävästi. Toisinaan pelaaja saattaa myös menettää joitakin aistejaan pelissä, kuten näkökyky ja kuulo. Nämä muuttuvat elementit pelissä tuovat haastetta ja yllätyksellisyyttä pelikokemukseen.

3.3 Grafiikka

Tehdyssä pelissä on käytetty niin sanottua Low Poly -tyyliä, jolla viitataan matalaan monikulmioiden määrään objekteissa ja 3D malleissa (kuva 23). Monikulmioiden määrä per objekti on hyvin matala ja näin ollen se ei ole raskas laitteille. Optimoinnin lisäksi tyylin taiteellinen aspekti sopii pelin henkeen.



Kuva 23. Low Poly-tyyli esimerkki

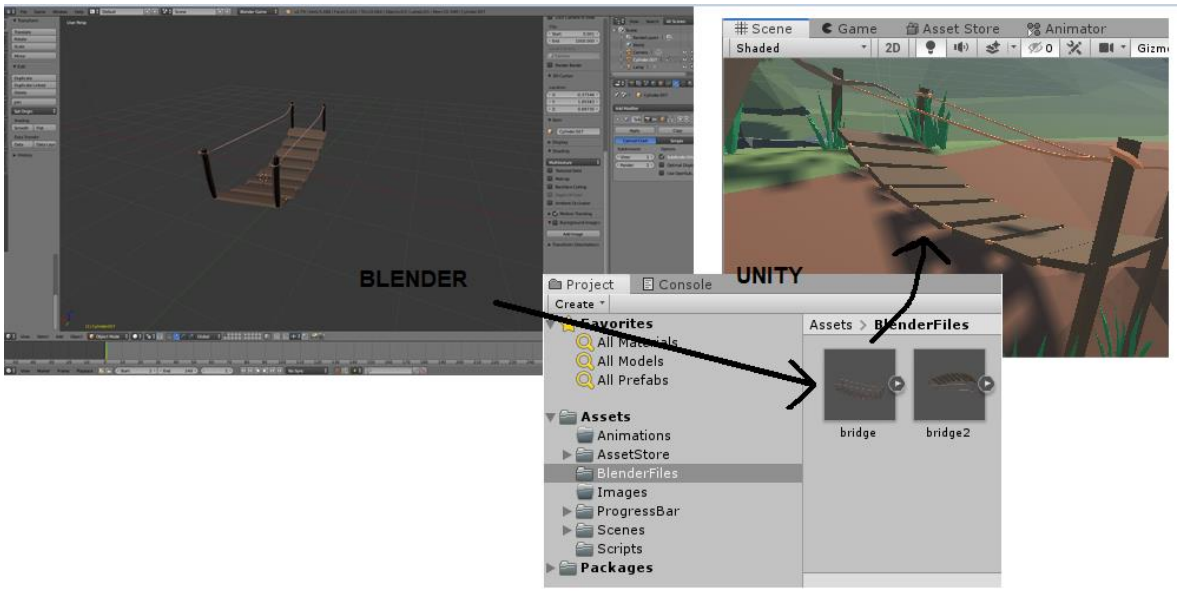
3.3.1 Blender

Blender on ilmainen vuonna 1995 julkaistu 3D-grafiikan mallinnusohjelma. Ohjelma tarjoaa kattavat työkalut 3D-grafiikan mallinnukseen ja animoimiseen. Ohjelma ei kuitenkaan ole aloittelijaystävällinen. Blender perustuu avoimeen lähdekoodiin, mikä tarkoittaa, että sen lähdekoodi on vapaasti saatavilla.

Mobiilipelin 3D-grafiikkaa on tehty Blenderillä, mutta pelin versioissa, jota työ käsittelee, esiintyy vain muutama Blender-malli. Mobiilipelin kehittyessä tämänhetkiset 3D-grafiikat korvautuvat yrityksen omilla malleilla.

Blender toimii suhteellisen saumattomasti yhdessä Unityn kanssa. Unity tukee suoraan Blender tiedostojen tuomista Unityyn. Blender tarjoaa kahta eri tiedostotyyppiä ja tapaa tiedostojen tuontia varten toisiin ohjelmiin. Näitä ovat blend- ja fbx-tiedostot. Mikäli käytetään blend-tiedostoa, voidaan se suoraan tallentaa Unityn Asset kansioon ja se on valmis käytettäväksi. Tällöin Unity kutsuu Blenderin Export skriptiä, joka generoi fbx-tiedoston ja lähettää sen Unityyn. Kyseinen operaatio kuitenkin tapahtuu käyttäjän sitä huomaamatta. Toinen tapa on muuntaa tiedosto jo Blenderissä fbx-tiedostoksi ja sen jälkeen viedä tiedosto Unityyn. Mikäli tiedostot tuodaan Unityyn blend muodossa, tallentuvat muutokset, jotka objektiin tehdään sen jälkeen Blenderissä myös Unityyn. Mikäli käytetään manuaalista fbx-tiedostoksi muuntamista ja viemistä Unityyn, ei jatko muutokset objektissa vaikuta toisiinsa.

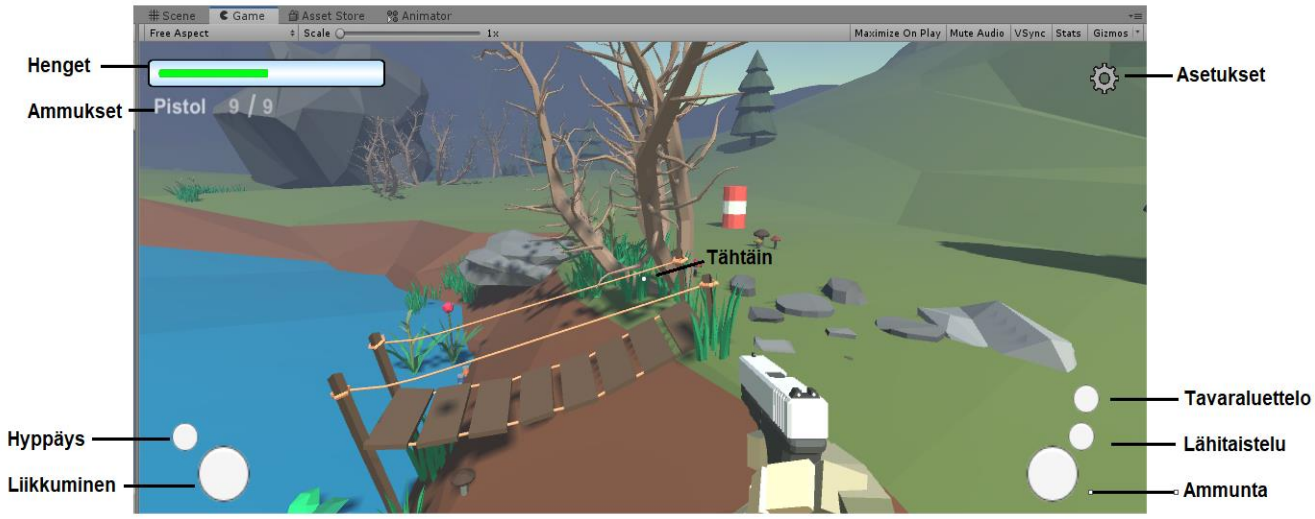
On myös huomioitava, että Unity ja Blender käyttävät eri koordinaatiosysteemejä. Blenderissä z-akseli osoittaa ylöspäin, kun taas Unityssä y-akseli osoittaa ylöspäin. Tästä syystä Blenderistä tuodut objektit näkyvät Unityssä ylösalaisin, mikäli asiaa ei ole huomioitu asettaessa koordinaatiotietoja Blenderissä. Objektin rotaatio ja asento voidaan kuitenkin korjata Unityn puolella. Seuraavassa kuvassa esitetty kuinka Blenderillä luotu 3D-objekti on käytettävissä Unityssä (kuva 24).



Kuva 24. Objekti Blenderistä Unityyn

3.3.2 Käyttöliittymä

Pelin käyttöliittymä on yksinkertainen ja suunniteltu sopimaan kaikille mobiililaitteille (kuva 25). Kuvassa näkyvät painikkeet ja muut elementit tulevat vielä päivittymään lopulliseen muotoonsa. Painikkeet skaalautuvat näytön koon mukaan ja painikkeiden paikkoja voi muuttaa itselle omaan pelityyliin sopivaksi.



Kuva 25. Käyttöliittymä

3.4 Testaus

Pelin testaus on aloitettu jo varhain pelin kehityksen alussa käyttäen Unityn pelinäkymää ja hyödyntämällä debuggeria, joka ilmoittaa virheistä ja ongelmista. Skripteissä toimintaa on testattu alusta-alkaen hyödyntämällä yksikkötestausta, jonka avulla voidaan testata yksittäisiä osia koodista ja toimivatko ne halutulla tavalla käyttäen esimerkiksi yksinkertaisia print-kutsuja. Unity tarjoaa myös oman sisäisen yksikkötestaus työkalun nimeltään Test Runner, mutta kyseistä työkalua ei ole käytetty kyseisessä projektissa toistaiseksi.

Mobiilipeliä on haastava testata pelkässä Unityn pelinäkymässä, sillä mobiilikontrollit eivät pääsääntöisesti toimi oikein, eikä pelin tehokkuudesta saada realistista kuvaa. Tätä varten projektissa hyödynnettiin Unityn maksutonta mobiilisovellusta nimeltään Unity Remote. Unity Remote-sovelluksella pystytään ajamaan peliä käyttämällä Android laitetta, joka on kytketty tietokoneeseen. Peliä voi testata myös käyttäen erilaisia Android-emulaattoreita, mutta tuntuma ei ole sama kuin käyttäisi oikeaa puhelinta eikä sujuva toiminta ole taattua.

Tulevaisuudessa peli kuitenkin siirtyy Google Play-kauppaan, jolloin varsinaiset Alpha ja Beta-testaukset valituilla käyttäjäryhmillä voidaan aloittaa. Toistaiseksi työstettyä mobiilipeliä on testattu vain kolmen eri Android-laitteen ja käyttäjän puolesta.

3.5 Haasteet

Mobiilipelin suunnittelun ja kehityksen haasteita ovat peli-idean suunnittelu ja lopulta sen toteutus. Peliä kehitettäessä peli on saanut uusia ideoita ja suuntaviivoja siitä, mihin suuntaan toinen versio pelistä etenee. Toisinaan jokin suunniteltu elementti pelissä on liian

haastavaa toteuttaa vallitsevien rajoitteiden ja pelintekijän kokemuksen puutteen vuoksi. Näitä ongelmia ovat olleet muun muassa pelin sulava toimiminen mobiililaitteessa tasaisella ruudunpäivitysnopeudella, mobiilikontrollien toimivuus ja intuitiivisuus, grafiikan mallinnus Blender-ohjelmistolla sekä pelin testaus mobiililaitteella käyttäen Unity Remote palvelua. Pelin fysiikkaa rakentaessa valinta pelattavan hahmon tyypistä myös vaihteli. Alkuun luotu hahmo perustui Character Controller-systeemin, mutta vaihtui lopulta RigidBody-komponentiksi. Päätyminen RigidBody-komponenttiin tapahtui siksi, että pelin mekaniikka nojautuu Unityn fysiikkamoottoriin, joten sitä käyttämällä on helpompi implementoida voimansuureita pelaajan ja maailman välillä. Kyseisiä voiman suureita ovat esimerkiksi painovoima, kitka ja kiihtyvyys. Suunnan muutos kuitenkin vaati hahmon skriptin täydellisen muutoksen ja vei aikaa.

4 YHTEENVETO

Työn tavoitteena oli tehdä mobiilipeli Android-alustalle käyttäen Unity-pelimoottoria, sekä tutkia siinä käytettyjä teknologioita ja työkaluja.

Varsinaisen mobiilipelin työstäminen alkoi ideoiden miettimisellä. Graafinen tyylisuuntaus oli ainoa jo tiedetty aspekti työssä ja sen ympärille muodostui itse peli. Pelin idea kuitenkin muuttui matkanvarrella ”battle royale” -tyyppisestä taistelupelistä enemmän seikkailukokonaisuudeksi, joka yhdistelee eri tyyppisiä genrejä keskenään. Haasteita ideointiin ja toteutukseen toivat mobiililaitteen tekniset rajoitukset. Pelin tulee olla tarpeeksi monipuolinen, mutta kuitenkin riittävän yksinkertainen opittavaksi ja toimivaksi mobiililaitteella. Raa’an idean pohjalta alkoi työn toteutus ja Unityn monipuolinen opettelu. Pelin tekeminen aloitettiin tietokoneversiona mikä osoittautui virheeksi, sillä koko liikkumisen kontrolloiminen täytyi tehdä myöhemmin uudestaan mobiilaitteelle sopivaksi. Työssä myös käytettiin monia eri kontrolliratkaisuja, kunnes yksi valikoitui ylitse muiden, sillä se oli ainoa, joka toimi sulavasti projektissa.

Työtä tehdessä tultiin lopulliseen päätökseen mitä tekniikoita Unityssä käytetään saavuttaakseen optimaalinen suoritusteho mobiililaitteilla ja minkälainen tulevaisuus pelillä sen lisäominaisuuksien puolesta on. Mobiilipeli myös loi paljon uusia peli-ideoita tulevaisuuden projekteja varten. Ensimmäinen versio työn mobiilipelistä valmistui, mutta liikesalaisuuksien vuoksi kattavampi kuvaus pelistä on jätetty työstä pois. Unity on tehokas ja kattava kehitysympäristö erityyppisten pelien tekoon, mutta mikäli projektina on isompi peli, voi olla syytä miettiä pelille spesifin pelimoottorin rakentaminen itse, jotta peli voidaan optimoida paremmin. Itse pelinkehitys on ollut palkitsevaa ja mielekästä työtä.

LÄHTEET

Farias, P 2019. How to Import Blender Models into Unity – Your One-Stop Guide [viitattu 26.11.2019]. Saatavissa: https://gamedevacademy.org/how-to-import-blender-models-into-unity-your-one-stop-guide/

Hanshaw, J 2018. Awake vs. Start vs. OnEnable and When to Use Them – Unity [viitattu 11.9.2019]. Saatavissa: https://www.tangledrealitystudios.com/development-tips/awake-vs-start-vs-onenable-and-when-to-use-them/

Mobvista 2018. Why Mobile Gaming Is Now Bigger than Console and PC Gaming Combined, and Still Growing and Always Changing [viitattu 3.10.2019]. Saatavissa: https://www.mobvista.com/en/blog/mobile-gaming-now-bigger-console-pc-gaming-combined-still-growing-always-changing/

Placzek, M 2016. Object Pooling in Unity: GameLoop-Diagram [viitattu 3.10.2019]. Saatavissa: https://koenig-media.raywenderlich.com/uploads/2016/06/GameLoop-Diagram.png

TechTerms 2016. Rendering [viitattu 28.11.2019]. Saatavissa: https://techterms.com/definition/rendering

Unity Documentation 2019. Unity User Manual [viitattu 2.3.2019]. Saatavissa: https://docs.unity3d.com/Manual/index.html

Unity Learn 2019. Introduction to Lighting and Rendering [viitattu 10.9.2019]. Saatavissa: https://learn.unity.com/tutorial/introduction-to-lighting-and-rendering?signup=true#5c7f8528edbc2a002053b52f

Unity Manual 2019a. Working in Unity [viitattu 2.3.2019]. Saatavissa: https://docs.unity3d.com/Manual/UnityOverview.html

Unity Manual 2019b. Graphics [viitattu 5.8.2019]. Saatavissa: https://docs.unity3d.com/Manual/Graphics.html

Unity Manual 2019c. Physics [viitattu 6.8.2019]. Saatavissa: https://docs.unity3d.com/Manual/PhysicsSection.html

Unity Manual 2019d. 3D Physics Reference [viitattu 6.8.2019]. Saatavissa: https://docs.unity3d.com/Manual/Physics3DReference.html

Unity Manual 2019e. User interface (UI) [viitattu 9.10.2019]. Saatavissa: https://docs.unity3d.com/Manual/UIToolkits.html

Unity Manual 2019f. Navigation and Pathfinding [viitattu 15.10.2019]. Saatavissa: https://docs.unity3d.com/Manual/Navigation.html

Unity Manual 2019g. Android [viitattu 1.3.2019]. Saatavissa: https://docs.unity3d.com/Manual/android.html

Unity Manual 2019h. Best practise guides [viitattu 20.10.2019]. Saatavissa: https://docs.unity3d.com/Manual/BestPracticeGuides.html

Unity Manual 2019i. Optimizations [viitattu 21.10.2019]. Saatavissa: https://docs.unity3d.com/Manual/MobileOptimisation.html

Unity Manual 2019j. Practical guide to optimization for mobiles [viitattu 21.10.2019]. Saatavissa: https://docs.unity3d.com/Manual/MobileOptimizationPracticalGuide.html

Unity Manual 2019k. Unity Hub [viitattu 25.11.2019]. Saatavissa: https://docs.unity3d.com/Manual/GettingStartedUnityHub.html

Unity Manual 2019l. Modeling characters for optimal performance [viitattu 6.7.2019]. Saatavissa: https://docs.unity3d.com/Manual/ModelingOptimizedCharacters.html

Unity Manual 2019m. Physics Material [viitattu 3.10.2019]. Saatavissa: https://docs.unity3d.com/Manual/class-PhysicMaterial.html

Unity Manual 2019n. LOD Group [viitattu 5.10.2019]. Saatavissa: https://docs.unity3d.com/Manual/class-LODGroup.html

Unity Technologies 2019. Unity Samples: UI [viitattu 3.10.2019]. Saatavissa: https://assetstore.unity.com/packages/essentials/unity-samples-ui-25468