HYVÄT KÄYTÄNTEET PIENIMUOTOISESSA PELINKEHITYKSESSÄ

Esimerkki 3D-pelistä Unity-pelimoottorin kanssa

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tieto- ja viestintätekniikka, insinööri (AMK)

Syksy, 2022

Jesse Kankaisto

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tieto- ja viestintätekniikan koulutus Tiivistelmä
Tekijä Jesse Kankaisto Vuosi 2022
Työn nimi Hyvät käytänteet pienimuotoisessa pelinkehityksessä
Ohjaaja Petri Kuittinen

Opinnäytetyön tavoitteena on antaa laaja kokonaiskuva projektityöskentelystä pelikehityksen näkökulmasta keskittyen Unity-pelimoottoriin. Käsitellään tekijöitä, jotka mahdollistavat sujuvan työskentelyn saavuttaen lopputuloksena julkaisukelpoisen pelin aikataulun puitteissa. Tavoitteena on tarttua soveltavassa peliprojektissa kohdattuihin haasteisiin ja käsitellä hyviä käytäntöjä, miten ryhmätyöskentely ja projektihallinta saadaan säilytettyä peliprojektin aikana. Opinnäytetyön tarkoituksena on luoda ymmärrystä pelikehityksen projektityöskentelystä, jättämällä opinnäytetyön ulkopuolelle pelimekaniikan ja itse pelin rakentamisen, joihin on saatavilla tietoa rajattomasti.

Ensimmäisessä osiossa käsitellään itse Unity-pelimoottoria ja muutamia tärkeitä ominaisuuksia, jotka ovat oleellisia projektityöskentelyn osalta. Elementtien hankintaa peliin ja näihin vaikuttavista tekijänoikeudellisista tekijöistä.

Toisessa osiossa keskitytään yleisesti projektihallintaan, keskittyen pelikehittäjän näkökulmaan. Käydään läpi työskentelyyn, aikatauluun, dokumentointiin ja ryhmätyöskentelyyn liittyviä tekijöitä ja tapoja hallinnoida näitä. Lisäksi käydään läpi yleisesti käytettyjä työkaluja, jotka mahdollistavat hyvän ja tehokkaan työskentely-ympäristön ja yhteydenpidon projektiryhmän kanssa.

Viimeisessä osiossa keskitytään ohjelmointiin ja Unityssa tapahtuvaan hallintaan ja osien rakenteisiin ja nimeämisiin. Tarkastellaan tapoja varmuuskopioida ja mitkä soveltuvat Unityn kanssa. Lisäksi käydään läpi tärkeimmät tekijät optimoinnin kannalta.

Opinnäytetyönä syntyi tiivis kattava kokonaisuus hyvistä käytänteistä Unityn peliprojektin parissa. Saavutettiin laaja ymmärrys projektiryhmätyöskentelystä, Unityn ominaisuuksista ja optimoinnista. Soveltavan peliprojektin kautta opittiin Unitysta ja sen ominaisuuksista, jonka lopputuloksena syntyi kokonainen peli, monipuolisesti toteutettuna oppimisen maksimoimiseksi.

AVAINSANAT
Unity, projektityöskentely, pelimoottori, peliohjelmointi, projektihallinta

Sivut 44 sivua ja liitteitä 0 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Information and Communication Technology
Author Jesse Kankaisto
Subject Good practices in small-scale game development
Supervisor Petri Kuittinen

Abstract
Year 2022

The aim of the thesis is to provide a broad overview on the project work from the viewpoint of game development, with focus on the Unity game engine. Factors that enable a smooth workflow, arriving at a game ready for release within the schedule, are processed. The goal is to tackle challenges encountered in an applied game project, and to utilize good practices to maintain group work and project management within it. The meaning of the thesis is to create understanding on the project work in the game development, leaving out of the thesis the game mechanics and the production of the game, which are described by a large amount of information elsewhere.

In the first section, the Unity game engine is described as such, along with a few important properties that are relevant for project work, component contracting, and copyright matters related to them.

The second section focuses on project management from the viewpoint of the game developer. Matters associated with workflow, schedule, documentation and group work, as well as ways to manage them, are described. In addition, generally used tools that enable a good and efficient work environment and communication with the project group, are described.

The last section focuses on programming and management within Unity, and the structures and naming of elements. The compatibility of backup tools with Unity is presented. In addition, the most important factors for optimization are processed.

The thesis was created a concise and comprehensive general view on good practices with the Unity game project. A broad understanding on project group workflow, properties of Unity, and optimization, was reached. Via an applied game project, Unity and its properties were learnt, and the final goal was a whole game, versatilely completed to maximize learning.

Keywords Unity, project work, game engine, game programming, project management
Pages 44 pages and appendices 0 pages

Sisällys

SANASTO 1
1 Johdanto 3
2 Pelituotannon elementit 4
2.1 Unity-pelimoottori 5
2.1.1 Unity Hub 6
2.1.2 Unityn ominaisuus - tagit 6
2.1.3 Unityn ominaisuus - prefab 8
2.2 Internetistä ladattavat valmiit elementit ja niiden tekijänoikeudet 9
2.2.1 Elementtejä tarjoavat Internet-sivustot 9
2.2.2 Unity Asset Store 10
2.2.3 Yleisesti pelikehitykseen vaikuttavat tekijänoikeudet ja termistö 12
2.2.4 Creative commons lisensointi 14
3 Projektihallinta 15
3.1 Projektihallinnan perusteet 15
3.2 Projektin pelisäännöt 17
3.3 Kommunikointi projektiryhmässä 18
3.4 Projektikansio 19
3.5 Projektihallinnantyökalut 21
3.6 Action List - toimenpidelista 22
3.7 Projektin aikataulutus ja laatutason ylläpito 23
3.8 Priorisointi 25
3.9 Työnjako ja peliprojektin aloitus 25
4 Projektin organisointi ja hallinta Unityssa 26
4.1 Versionhallinta 27
4.1.1 Versionhallinta sovelluksia 27
4.1.2 Versionhallinta paikallisesti 28
4.2 Resurssienhallinta ja perusteet Unity-ympäristössä 29
4.2.1 Unityn näkymät ja peruskäsitteet 29
4.2.2 Nimeäminen ohjelmoinnissa 30
4.2.3 Hierarkia ja elementtien nimeäminen Unityn scenessä 30

4.2.4 Unityn resurssienhallinta näkymä 32
4.3 Uudelleen käyttäminen 33
5 Pelin suorituskyvyn ja tallennustilan hallinta Unityssa 35
5.1 Suorituskyky ja sen tehostaminen Unityssa 36
5.2 Välimuistin ja laskentatehon käyttäminen Unityssa 37
5.3 Polygonien vaikutus suorituskykyyn 39
5.4 Object pooling 41
6 Yhteenveto 42
Lähteet 45

SANASTO

3D-malli	Kolmiulotteinen elementti
Asset	Valmispaketti, joka voi sisältää grafiikkaa, pelimekaniikkaa tai muita elementtejä, joita voidaan käyttää suoraan Unity-pelimoottorissa
Collider	Tarkoittaa muotoa pelinkehityksessä, joka mahdollistaa fyysisen törmäyksen elementtien välillä
Cross-platform	Alustariippumaton sovellus tai ohjelmisto, joka ei ole sidottuna tiettyyn käyttöjärjestelmään tai laitteistoon
C Sharp	Olio-ohjelmointikieli, jota käytetään esimerkiksi Unity-pelimoottorissa
Debuggaus	Ohjelmoinnissa käytettävä termi, jolla tarkoitetaan yleisesti ohjelmoinnissa syntyneiden virheiden selvittämistä ja niiden korjaamista
Elementti	Pelinkehityksessä käytettävä termi, joka kuvastaa kaikkia pelissä olevia asioita esimerkiksi objekteja, ääniä, kuvia ja skriptejä
Koonti	Englannin kielen sanasta build, joka tarkoittaa ohjelmistoprojektin kokoamista yhteen sovellukseksi, joka ei ole riippuvainen kehitysympäristöstä esimerkiksi Unitysta
Pelimoottori	Yleisnimitys ohjelmistolle, joka auttaa pelinkehityksessä tuomalla ominaisuuksia valmiina. Esimerkiksi fysiikkaelementtejä ja työkaluja pelin luomiseen

Polygoni	Tietokonegrafiikassa käytettävä termi samassa tasossa olevista kolmesta pisteestä muodostavasta kolmiosta
Prefab	Unityn sisäinen ominaisuus, jonka avulla voidaan luoda elementeistä paketteja, joita voidaan kopioida ja jonka muokkaus synkronoituu kaikkiin tehtyihin kopioihin pelissä
Renderöinti	Englannin kielen sanasta render tuleva suomenkielinen termi, jolla tarkoitetaan digitaalisen datan kääntämistä näytölle kelpaavaan muotoon
Skripti	Ohjelmointikoodia sisältävä tiedosto
Tag	Unityn sisäinen ominaisuus, jonka avulla voidaan kutsua yhtä tai useampaa elementtiä pelissä
Unity	Pelimoottori
Unity Asset store	Unityn oma kauppa, josta ladataan elementtipaketteja
VR	Virtual Reality eli virtuaalitodellisuus

1 Johdanto

Opinnäytetyö käsittelee pienimuotoisten peliprojektien käytäntöjä, ylläpitämistä ja työkaluja, jotka mahdollistavat tehokkaan työskentelyn useimpien pelimoottoreiden kanssa, keskittyen Unity 3D:n ympärille. Tavoitteena on antaa ymmärrystä monipuolisesti useilta eri osa-alueilta, jotka jokaisen pelinkehittäjän tulisi hallita aloittaessaan peliprojektia. Näitä ovat muun muassa projektinhallinta, sen aikatauluttaminen ja asioiden priorisointi. Peleihin voi laittaa lähes rajattomasti asioita, jotka eivät välttämättä ole ollenkaan tarpeellisia itse pelin ja pelaamisen kannalta.

Opinnäytetyössä käydään läpi yleisesti projektiryhmään vaikuttavia tekijöitä ja tapoja projektin toteuttamiseen. Ryhmätyöskentelyssä on tärkeää toteuttaa asiat hallitusti etenkin projektin alussa, koska tällöin pelille muodostuu pohjarakenne ja toimintatavat, joita noudattamalla säästetään huomattavasti aikaa. Lisäksi käsitellään yleisesti pelin elementteihin liittyvistä tekijänoikeuksista sekä mistä näitä valmiita elementtejä saa hankittua, koska aina ei kannata keksiä pyörää uudestaan. Unityllä on mahdollista rakentaa pelejä ilman, että osaa 3D-mallintaa, mutta tämä asettaa rajoituksia toteutuksen osalta.

Toimivaa pelin kokonaisuutta silmällä pitäen käydään läpi myös pelioptimoinnin peruspilarit. Tavoitteena ei ole käydä optimoinnin kaikkia osa-alueita läpi, vaan keskittyä isoihin tekijöihin. Nämä kannattaa huomioida peliä rakentaessa, jotta toimivuus olisi vähintään kohtuullinen. Nämä kaikki osa-alueet jäävät monesti pelin konkreettisen luomisen ja eri tekniikoiden varjoon, joka saattaa johtaa pahimmassa tapauksessa siihen, että projekti ei koskaan valmistu. Jos projektista katoaa täysin hallinta, sen eteenpäin vieminen muuttuu työläämmäksi kuin projektin alusta aloittaminen. Vaihtoehtoisesti peliprojekti muuttuu kelvottomaksi julkaisua silmällä pitäen, koska se rikkoo tekijänoikeuksia tai jää toimivuudeltaan ala-arvoiselle tasolle.

Tämän opinnäytetyön ulkopuolelle jätettiin Unityn ja muiden pelimoottoreiden tekniikoiden käyttö ja ohjelmointi, joita tarvitaan pelin luomisessa. Perustekniikoihin löytyy internetistä tuhansia oppaita kaikkiin eri tarpeisiin. Tämän työn tarkoituksena on vastata tärkeisiin

peliprojektin hallintaan ja yleisesti projektityöskentelyyn liittyviin kysymyksiin, jotka jäävät pelin ideoimisen, rakentamisen, ohjelmoinnin ja julkaisemisen ulkopuolelle.

Opinnäytetyön pohjana käytetään pelituotanto-kurssilla toteutettua Teddy Bear Adventure -peliä, jossa seikkaillaan kolmannessa persoonassa nallella keräten esineitä ja päihittämällä yksinkertaisia vihollisia. Peliprojektissa tavoiteltiin Unityn oppimista, jonka lopputuloksena on toimiva peli, josta löytyy mahdollisimman paljon seikkailupeleistä tuttuja ominaisuuksia. Unityn opiskeluun ja toteuttamiseen oli varattu vain kahdeksan viikkoa. Projekti toteutui ajallaan, mutta ei kivuttomasti. Oma työnkuvani oli monipuolinen, vastasin projektista ja dokumentoinnin hallinnasta, jonka ongelmista ja haasteista syntyi ajatus opinnäytetyölle, jossa käsitellään muita osa-alueita kuin pelkkää pelin rakentamista ja sen tekniikoita.

Tämä opinnäytetyö käsittelee asioita, jotka mahdollistavat projektin kokonaisvaltaisen onnistumisen ja hallinnan. Lisäksi tavoitteena on vastata kysymyksiin, joihin ei löydy suoria vastauksia Unity-verkkokursseista ja kirjallisuudesta, koska niiden sisältö keskittyy pääsääntöisesti erilaisten pelien luomiseen ja tarvittaviin tekniikoihin. Opinnäytetyön päämääränä on antaa kultainen keskitie projektin hallintaan oman näkemykseni, standardikäytäntöjen, muiden mielipiteiden ja teorioiden kautta.

2 Pelituotannon elementit

Tämä luku käsittelee lyhyesti suosittuja ja helposti lähestyttäviä pelimoottoreita, niiden vahvuuksia ja hallintaa helpottavia ominaisuuksia Unityssä. Lisäksi tarkastellaan mistä saa hankittua valmiita elementtejä peliin ilmaiseksi tai pientä maksua vastaan. Elementillä tarkoitetaan yleisesti kaikkia komponentteja, mitä peli voi sisältää. Näitä ovat esimerkiksi 3D-objektit, musiikit, ääniefektit, taustakuvat ja ohjelmointikoodit. Lisäksi tarkastellaan näihin vaikuttavia tekijänoikeudellisia tekijöitä, jotka tulee ottaa erityisesti huomioon, jos peli on tarkoitus julkaista yleisölle.

2.1 Unity-pelimoottori

Unity on cross-platform eli järjestelmäriippumaton pelimoottori, joka julkaistiin vuonna 2005. Pelimoottori on erittäin suosittu tietokone-, mobiili- ja konsolialustoilla. Tuettuja alustoja löytyy tällä hetkellä noin 20 erilaista. Myös VR eli virtuaalitodellisuuteen perustuva pelien luominen Unitylla on kasvanut voimakkaasti viime vuosien aikana. Pelimoottorin saa käyttöön ilmaiseksi ja suurimman osan toiminnoista. Jos tuotot nousevat 100 000 dollariin vuodessa, Unity muuttuu maksulliseksi. (”Unity (game engine)”, 2022)

Unity-pelimoottorissa on useita hyviä ja muutamia huonoja puolia. Pelimoottori laajentuu ja kehittyy jatkuvasti ja siitä on saatavilla useita eri versioita. Yleisesti kannattavaa on aloittaa uusimmalla vakaalla versiolla. Unity on helposti lähestyttävä ja sopii aloittelijoista ammattilaisiin. Tämän pelimoottorin kanssa pääsee alkuun nopeasti, vaikka ei ymmärtäisi kuin vähän saatavilla olevasta tekniikasta ja pelimoottoreista. Tämän pelimoottorin pyörittäminen vaatii vain hieman C# -kielen ymmärtämistä. Unityn sisäänrakennetut ominaisuudet ja valmiit assetit eli valmiit muiden luomat elementit mahdollistavat pelien rakentamisen palapelin tavoin. Pelinkehittäjän ei tarvitse paljon pohtia näytönohjaimen ja prosessorin osuuksia pelin optimoinnin kannalta, vaan käyttäjä voi keskittyä pääsääntöisesti pelin konkreettiseen luomiseen ja sisällön rakentamiseen. (Tuliper, 2014)

Unity tarjoaa tekniikoita, joilla optimoinnin ja pelin perustoiminnot kuten liikkuminen, valikot ja tallentaminen saadaan helposti toteutettua. Myös pelin testaaminen nousee esille helppoudessaan, koska voit testata heti, miten luomasi asia toimii ja näkyy pelissä. Testitilaan ei voi turvautua täysin, koska esimerkiksi valot ja varjot voivat olla erilaiset testitilassa kuin kootussa versiossa. Lisäksi jotkin lisäosat ja koodit toimivat moitteettomasti testitilassa, mutta ne saattavat estää tai rikkoa pelin koontivaiheessa. Säännöllinen testaaminen koottuna on tärkeää, jolloin debuggaus eli ongelmien selvittely on helpompaa. Muita huonoja puolia on suljettu lähdekoodi, jolloin kaikkia ongelmia ei ole mahdollista korjata itse. (Tuliper, 2014)

Graafisilta ominaisuuksiltaan Unity häviää esimerkiksi toiselle suurelle pelimoottorille Unrean Enginelle. Unreal Engine on tarkoitettu lähinnä suuriin peliprojekteihin ja moni

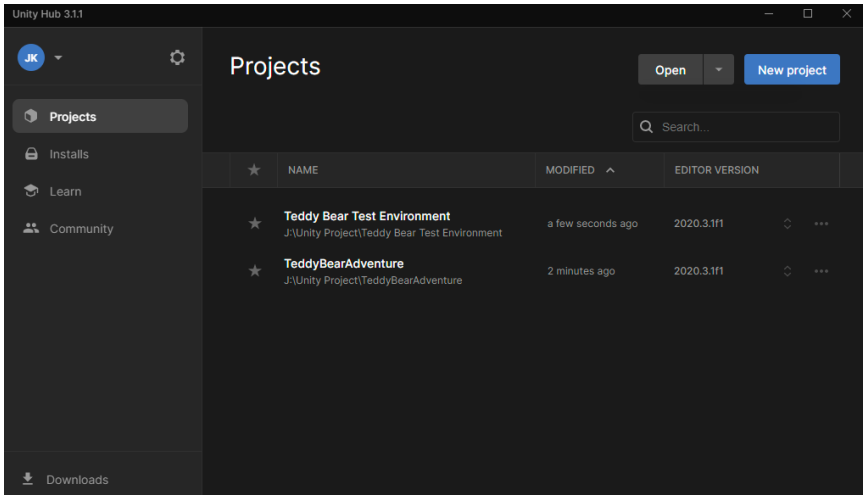
pelialan yritys onkin siirtynyt siihen Unityn jälkeen. Unity on hyvä pelikehityksen alkutaipaleella, josta on helppo siirtyä haastavampiin sekä monipuolisempiin pelimoottoreihin.

Seuraavaksi käydään pintapuolisesti muutamia tärkeitä Unityn toimintoja läpi, joiden vaikutus on suuri projektihallinnan osalta ja ovat avainasemassa lähes kaikissa Unitylla toteutettavissa peliprojekteissa.

2.1.1 Unity Hub

Unity Hub on tehokas osa Unitya, jonka avulla voi hallita projekteja, käyttää eri Unityn versioita, varmuuskopioida, opiskella Unitya harjoitusprojekteilla ja työstää projekteja yhdessä muiden kanssa. Harjoitteluprojektit antavat tietoa ja taitoa erilaisiin peleihin ja tilanteisiin. Unity Hub on selkeä ja helppo käyttää, josta näet kaikki projektit ja pääset niihin käsiksi missä vaan nopeasti, jos ne on tallennettu Unityn pilvipalveluun (Kuva 1). (Unity, 2021)

Kuva 1. Unity Hub 3.1.1 -version projektinäkymä.



2.1.2 Unityn ominaisuus - tagit

Unity-sovelluksesta löytyy hallintaa ja työskentelyä helpottavia tageja, joita voidaan käyttää erittäin monipuolisesti ja ne auttavat pitämään projektin selkeänä. Tageja voidaan kohdistaa

yhteen tai useampaan pelielementtiin samanaikaisesti. Tagien avulla on helppo tunnistaa ja käskyttää eri peliobjekteja. Voit esimerkiksi siirtyä helposti kameranäkymän kamerasta toiseen, jos olet määrittänyt jokaiselle kameralle oman tagin.

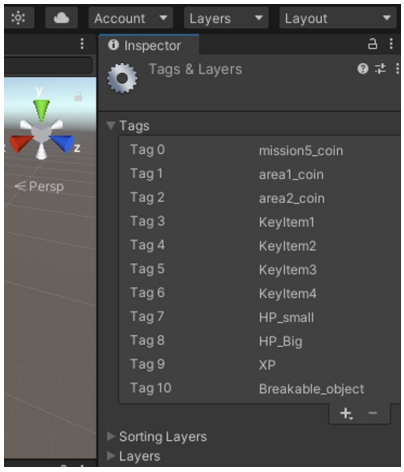
Otetaan esimerkki tagien hyödyntämisestä. Pelissä pitää kerätä 50 kolikkoa tietyltä alueelta ja joita on alueella 100 kappaletta. Kun haluttu määrä on kerätty, voidaan yhdellä käskyllä tuhota loput kolikot alueelta (Kuva 2). Tageja voidaan hyödyntää myös kerättävien asioiden esimerkiksi ammuksien poimimiseen. Ammuksen tagin avulla ne voidaan rekisteröidä tiettyyn muuttujan arvoon, käyttäen muuten samaa koodia tai funktiota. Tagien käyttäminen säästää roimasti aikaa ja vähentää tarvittavan koodin määrää. (Docs Unity3D, 2022a)

Kuva 2. Esimerkki 5 riviä pitkästä koodista, joka tuhoaa loput kolikot, kun 50 on kerätty.

```
void mission5_collect_coins (){
if (mission5_coins >= 50){
GameObject[] mission5_coins = GameObject.FindGameObjectsWithTag("mission5_coin");
foreach(GameObject mission5_coin in mission5_coins)
GameObject.Destroy(mission5_coin);
}}
```

Tagit löytyvät peliobjektien Inspector-kohdasta, jossa voidaan valita tagi objektille. Tag-valikossa sisällä voidaan muokata ja luoda tageja (Kuva 3). Elementillä voi olla yksi tagi kerrallaan ja tämän lisäksi Unityn sisäänrakennettuja tageja, jotka eivät tosin näy normaalissa tag-valikossa.

Kuva 3. Kuvakaappaus Tags & Layers -valikosta, johon on luotu 11 tagia.



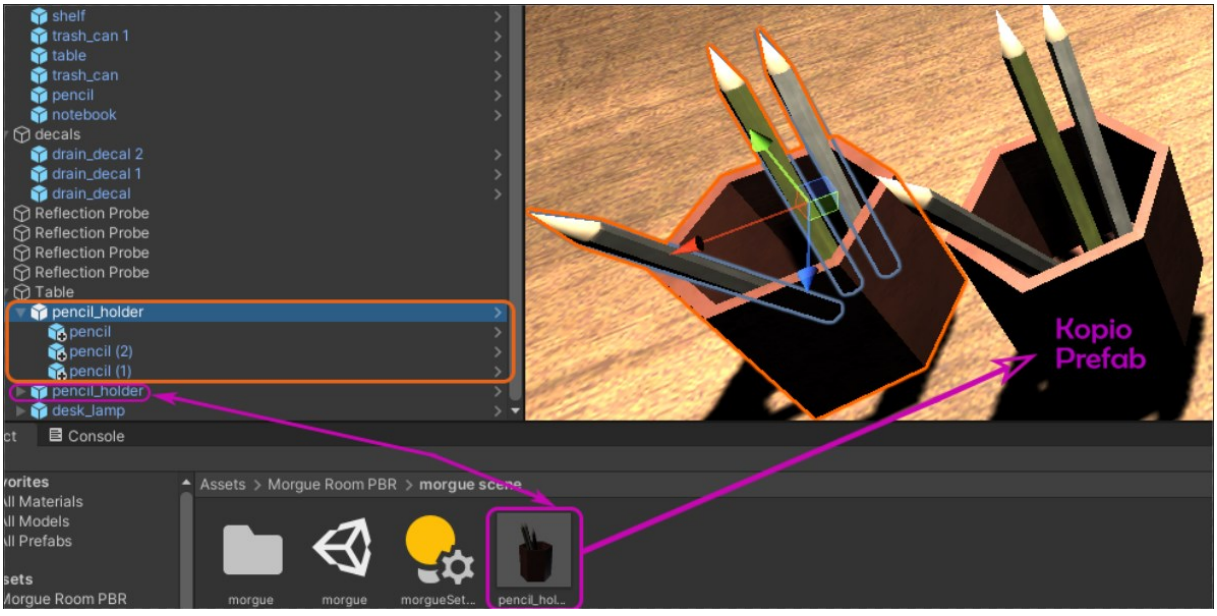
2.1.3 Unityn ominaisuus - prefab

Unityn yksi tärkeimmistä kokonaisuuksista on prefab eli elementeistä luotu paketti. Se nopeuttaa sekä helpottaa työskentelyä ja lisäksi tuo selkeyttä projektiin. Tämä on myös kätevä tapa siirtää elementtejä ja elementtikokonaisuuksia projektista toiseen.

Kaikki elementit, jotka toistuvat pelissä useamman kerran tai sisältävät paljon alaobjekteja on kannattavaa luoda niistä prefab -paketti. Prefab -paketin muokkaaminen synkronoituu kaikkiin esiintymiin projektissa automaattisesti. Tämä nopeuttaa huomattavasti työskentelyä, lisäksi näitä on helppo uusiokäyttää ja nämä saavat samat ominaisuudet sekä toiminnot kuin alkuperäinen ja prefab -paketin muokkaaminen vaikuttaa kaikkiin kopioihin. Tarvittaessa aikaisemmin luodusta prefabista voi luoda erillisen kopion, jonka muutokset eivät synkronoidu prefabien kanssa. (Docs Unity3D, 2022b)

Esimerkkinä kuvassa (Kuva 4) on kuppi, jossa on kolme kynää. Kaikki ovat omia objektejaan. Kynät on sijoitettu pencil_holder -elementin alaobjekteiksi. Näistä on tämän jälkeen luotu prefab, joka on otettu Unityn resurssienhallinnasta alkuperäisen prefabin oikealle puolelle.

Kuva 4. Esimerkki Prefabin uudelleen käyttämisestä.



2.2 Internetistä ladattavat valmiit elementit ja niiden tekijänoikeudet

Pelikehityksessä on järkevää priorisoida ja hyödyntää internetistä saatavia elementtejä. Pelinkehitystä varten löytyy käsittämättömän paljon sisältöä ilmaiseksi, mutta tähän tulee suhtautua varauksella, koska kuka tahansa voi lisätä sinne oman teoksensa oli se sitten laadukas tai todella huono. Arvioisin, että yli puolet internetistä löytyvistä objekteista, jotka on kohdistettu pelimaailmaa varten ovat rikkinäisiä tai todella huonosti optimoituja. Yksi huonosti optimoitu 3D-objekti voi kuluttaa tietokoneen tehoja enemmän kuin kaikki muut pelissä olevat objektit yhteensä. Käydään seuraavaksi läpi muutamia sivustoja ja esimerkkejä valmiina saatavista elementeistä.

2.2.1 Elementtejä tarjoavat Internet-sivustot

Internetistä löytyy tuhansia sivuja, jotka tarjoavat ilmaiseksi käyttöön kuvia, musiikkia, ääniefektejä ja 3D-objekteja. 3D-objekteja tarjoavia sivustoja on paljon, mutta suurin osa teoksista on maksullisia, tiedot ovat puutteellisia tai elementit soveltuvat vain henkilökohtaiseen käyttöön, joten näiden elementtien soveltuvuus kaupalliseen tarkoitukseen kannattaa varmistaa huolella.

Internetissä on mahdollista törmätä tietämättään materiaaliin, jota jaetaan ilman tekijän lupaa ilmaiseksi tai maksua vastaan. Tällaiset tilanteet saattavat johtaa ongelmiin, jos olet erehtynyt käyttämään näitä elementtejä pelissä ja julkaisutilanteessa, esimerkiksi Steamin kauppapaikassa, pelin julkaiseminen evätään tekijänoikeusrikkomuksien takia. Jos rikkomukset ovat lieviä, näistä ei yleensä aiheudu ongelmia, mutta pelinkehittäjän harteille jää useasti selvittäminen, miten peli rikkoo tekijänoikeuksia, koska siihen harvoin annetaan tarkempaa selvitystä. Tämän kaltaiset ongelmat ovat ajaneet pelialanyrityksiä konkurssiin, koska isoissa peliprojekteissa selvitys on haastavaa, ellei peräti mahdotonta.

Luotettavuutta lisää, jos sivustolle pitää tunnistautua, jotta materiaalia voi ladata sivustolle jaettavaksi. Jos tekijä tulee vahvasti esille omalla nimellään ja omistaa mahdollisesti oman verkkosivuston, tekijä on suurella todennäköisyydellä teoksien luoja ja näin ollen omistaa teoksien tekijänoikeudet.

Seuraavassa taulukossa (Taulukko 1) on eri elementtejä ilmaiseksi tarjoavia sivustoja, jotka voi luokitella luotettaviksi ja soveltuvat kaupalliseen käyttöön.

Taulukko 1. Pelikehitykseen elementtejä tarjoavia verkkosivustoja.

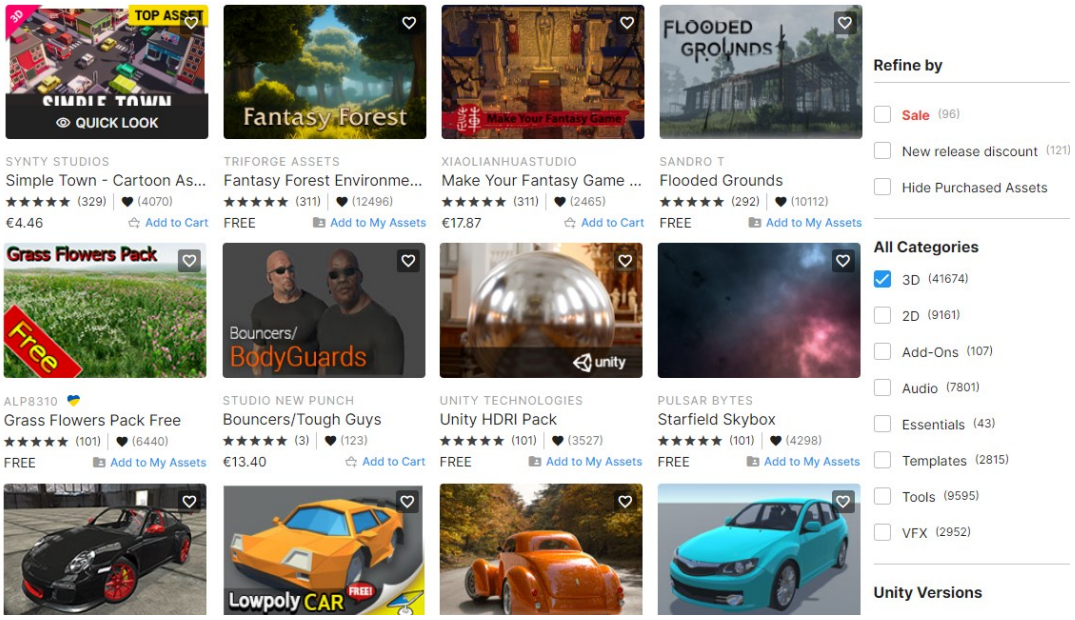
Sivuston nimi	**Tarjottava sisältö**	**Tekijänoikeus**
Free SFX	Ääniefektejä, taustamusiikkia	Tekijä mainittava projektissa
99sounds	Ääniefektejä	Vapaa käyttö
Zapsplat	Ääniefektejä, taustamusiikkia	Tekijä mainittava projektissa tai maksullinen jäsenyys
Orange Free Sounds	Ääniefektejä, taustamusiikkia, loop -musiikkia	Teoskohtainen, ilmaisesta rajoitettuun käyttöoikeuteen
Sounds Crate	Ääniefektejä, musiikkia, kuvia, 3D-objekteja	Ilmaista ja maksullista sisältöä, vaatii rekisteröitymisen
Pixabay	Ääniefektejä, musiikkia, kuvia	Vapaa käyttö
Cgtrader	3D-objekteja	Vapaa käyttö, rajoitettua, maksullista sisältöä
Creazilla	3D-objekteja, vektori kuvia	Vapaa käyttö

2.2.2 Unity Asset Store

Unity-pelimoottoria varten on olemassa Unity Asset Store -kauppapaikka, josta voi ostaa ja ladata ilmaiseksi Unityyn kokonaisia maailmoja, hahmoja, ääniä, visuaalisia efektejä, työkaluja ja skriptejä. (Kuva 5). Kaikki sivuston elementit ovat vapaasti käytettäviä kaupalliseen tarkoitukseen ja maksullisen sisällön käyttöoikeudet vaihtelevat ikuisesta

käyttöoikeudesta kertaluontoiseen käyttöön eli elementtejä on mahdollista ostaa huomattavasti edullisemmin yksittäistä peliprojektia varten.

Kuva 5. Unity Asset Storen tarjonnasta esimerkkejä.



Sivusto tarjoaa kymmeniä tuhansia paketteja, jotka sisältävät muutamasta elementistä tuhansiin elementteihin. Näitä on luvallista muokata omiin käyttötarkoituksiin, mutta muokattujen elementtien jakaminen ja myyminen eteenpäin on kiellettyä. Koska käyttöoikeuslupa tulee Unityn välityksellä, tämä on turvallisin paikka hankkia elementtejä peliin. Käyttäjät on vahvasti suojattu mahdollisilta tekijänoikeusrikkomuksilta, kunhan noudattaa elementeissä mahdollisesti mainittuja käyttörajoituksia.

Unity Assets Storesta lunastetut paketit tulevat suoraan Unity-sovellukseen näkyville paketinhallintavalikon alle, josta ne voidaan ladata kokonaisena pakettina tai osittain projektiin. Lunastettu sisältö rekisteröityy käyttäjätilille, jossa ne pysyvät tallessa uudelleen käyttöä varten. Suurin osa ladattavasta sisällöstä on selkeästi ja ammattitaitoisesti jaoteltu järjestelmällisesti, jolloin sisällön muokkaaminen ja käyttäminen on erittäin sujuvaa.

2.2.3 Yleisesti pelikehitykseen vaikuttavat tekijänoikeudet ja termistö

Pelikehitykseen liittyy paljon erilaisia tekijänoikeudellisia tekijöitä, jotka perustuvat tekijänoikeuslakiin. Nämä tulee ottaa huomioon tarkasti, jos peli on tarkoitus julkaista yleisölle edes ilmaiseksi. Tekijänoikeudelliset tekijät ovat useasti hyvin yksityiskohtaisia ja ne myös muuttuvat koko ajan, mutta niihin on olemassa yleinen toimintaperiaate.

Käytän tämän kappaleen lähteiden lisäksi apuna kahta haastattelua, jotka käytiin lakimiehen ja tekijänoikeudellisen neuvojan kanssa. Tarkempia henkilötietoja ja tarkennuksia haastattelusta ei ole lupaa käyttää julkisesti.

Peleihin ei ole tekijänoikeuslaissa erillisiä säännöksiä. Pelien nimet, tekniikat ja ideat eivät ole tekijänoikeuden suojan piirissä, pois lukien rekisteröidyt tavaramerkit, kuten esimerkiksi The Witcher ja Monopoly nimet. Muita tavaramerkillä suojattuja asioita voi olla esimerkiksi fiktiivisten hahmojen ulkonäkö, hahmojen nimet, sloganit, logot ja graafiset teokset.

Peleissä olevat elementit ovat eri tekijänoikeuslakien piirissä. Yleisesti Suomessa musiikin käyttöoikeudet tulee hakea Teostolta tai Gramexilta ja nämä osastot vaikuttavat tekijäsuojattuun musiikkiin aina, vaikka musiikki olisi itse tehtyä. Musiikki on aina tekijänoikeuden piirissä, vaikka se soisi vain sekunnin ajan. Musiikissa tekijänoikeudellinen suoja on 70 vuotta tekijän kuolinvuodesta. Esimerkiksi hyvin vanhan musiikin käyttäminen, kuten Bach ja Beethoven ovat vapaassa käytössä, tekijänoikeuslaki koskee kuitenkin uusia sovituksia. Teoston ja Gramexin määritykset eivät kuitenkaan vaikuta itse pelikehityksessä ja pelissä käytettyyn musiikkiin. Peleissä käytettävään musiikkiin riittää lupa teoksen tekijältä tai luvan haltijalta. Jos musiikkia esiintyy esimerkiksi pelin trailerissa, sen julkaisemiseen tarvitaan pääsääntöisesti erillinen lupa julkaisupaikasta ja musiikista riippuen. Jotkin julkaisupaikat kattavat tekijänoikeudet julkaista vapaasti käytettävissä olevaa musiikkia.

Äänitehosteisiin, joita ei lasketa musiikiksi pätee tekijänsuojaus, joka on kiinni tekijästä ja miten hän on teoksensa käyttöoikeudet määrittänyt. Sama tekijänoikeuslaki koskee kuvia, videoita, 3D-objekteja ja muita elementtejä. Toisin sanoen, musiikki on ainoa, johon voi tarvita erikseen lupa tekijän luvan lisäksi, jotta sitä voidaan käyttää ja esittää julkisesti

Suomessa. Muissa tapauksissa teoksen tekijä omistaa oikeudet ja määrittelee sen käyttöoikeudet.

Pelinkehittäjillä tai yrityksellä on aina tekijänoikeus luotuun peliin eli teokseen. Tämän teoksen voi luovuttaa eteenpäin, kunhan huomioi, että teoksessa voi olla muiden tekijänsuojaamaa materiaalia mukana, joiden uudelleen käyttäminen toisessa teoksessa, kuten pelin jatko-osassa saattaa olla kiellettyä.

Seuraavaksi käsitellään yleisesti tekijänoikeusmäärityksiä ja -termistöä, joihin vaikuttava taho on teoksen tekijä tai teoksia tarjoavat sivustot. Tärkeää on huomioida, että sivustot eivät käytä standardisanastoa, joten eri sivustot saattavat käsitellä termejä hieman eri tavoin, jolloin niiden merkitys voi muuttua. Valtaosa elementtejä tarjoavista sivustoista määrittelevät, että ladatut teokset on suojattava kopioinnilta. Eli projektin avoin lähdekoodi tai muutoin aukinainen rakenne voi rikkoa näitä määrityksiä. Unity ja useat muut pelimoottorit suojaavat automaattisesti sisällön pelin kokoamisen yhteydessä, jolloin pelinkehittäjän ei tarvitse huolehtia asiasta.

Maksullisissa teoksissa tulee kiinnittää huomiota siihen, millaisen käyttölisenssin lunastat, saako teosta muokata vai pitääkö se säilyttää sellaisenaan. Joissakin tilanteissa esimerkiksi interaktiivisuuden lisääminen tai sen poistaminen voi olla lisenssin vastaista. Onko käyttömäärä rajoitettu esimerkiksi yhtä peliä varten. Esimerkiksi Unity Asset Storesta ostettaessa lisenssi voi koskea elinikäistä käyttöoikeutta tai rajoittua yhteen Unitylla toteuttavaan peliprojektiin. Maksulliset teokset sisältävät aina jonkinlaisen kaupallisen käyttöoikeuden, mutta jos maksu kohdistuu sivustolle, eikä suoraan yksittäiseen teokseen tai kokoelmaan, lisenssien tyyppi saattaa vaihdella sivusto- ja teoskohtaisesti.

Public domain -merkintä tarkoittaa, että teos on vapaassa käytössä tai se on vapautettu vapaaseen käyttöön. Näiden teoksien osalta tekijä on luopunut oikeuksista ja ne sisältävät niin vähän rajoitteita, kuin se on lainpuitteissa mahdollista.

Royalty-Free (RF) eli rojaltivapaa-lisenssi, jonka käytöstä ei tarvitse maksaa. Tämä lisenssi sisältää useampaa eri tyyppiä. Pelikehityksen yhteydessä käytetyt termit ovat yleisesti RF Personal use- ja RF commercial use -lisenssit. RF personal use antaa luvan käyttää melko

vapaasti, mutta sitä ei saa käyttää peliin, jota on tarkoitus myydä tai jakaa ilmaiseksi suurelle yleisölle. Tämä lisenssi antaa luvan pienimuotoiseen käyttöön esimerkiksi opiskeluun ja työnäytteisiin. RF commercial use antaa luvan käyttää sitä vapaasti, ainoastaan suoranainen teoksen myyminen eteenpäin on kiellettyä, mutta peliä voi myydä vapaasti missä teos on mukana. Ilmaiseksi saatavilla olevat teokset sisältävät useasti vaatimuksia, esimerkiksi kyseisen teoksen tekijä tulee mainita käytön yhteydessä, kuten lopputeksteissä tai teosta ei saa muokata, vaan se pitää säilyttää sellaisenaan.

Pääsääntöisesti on mahdollista pyytää tekijältä lisenssitodistus myös ilmaiseksi saatavilla olevista teoksista. Tämä on suositeltavaa etenkin musiikin kohdalla. Erittäin suositeltavaa on pitää kirjaa käytetyistä teoksista, kuka on tekijä tai mistä ne on hankittu, jos hankinta on tehty muualta kuin Unity Asset Storesta. Tästä voi olla suuri apu mahdollisissa tekijänoikeusongelmissa.

Tekijänoikeudellisesta näkökulmasta selkein vaihtoehto on hankkia teoksia suoraan tekijältä, esimerkiksi hänen sivujensa kautta tai sivustoilta, jotka tarjoavat sisällölle selkeästi määritykset, miten teoksia saa käyttää. Sivustoja on paljon, josta voi ladata pelikehitykseen vapaasti sisältöä ja muokata niitä omien tarpeiden mukaisesti. Lähtökohtaisesti kannattaa tutustua aluksi sivuston lisensseihin, ja mitä ne oikeasti tarkoittavat käytännössä, etenkin jos tarkoituksena on hyödyntää ilmaiseksi jaossa olevia teoksia.

2.2.4 Creative commons lisensointi

Creative commons tarjoaa vaihtoehtoisia tapoja lisensointiin, normaalin kaikki oikeudet pidätetään käytännön tilalle. Creative Commons tarjoaa selkokielisen lisensoinnin kolmikerroksisella tekniikalla. Yksi lisenssi sisältää kolme versiota oikeudellisen lakitekstin, selkokielisen version ja koneluettavan version hakukoneita ja muita varten.

Seuraavassa taulukossa (Taulukko 2) on listattuna Creative Commonsin käyttämät lisenssit ja niiden tarkoitukset. Kaikki lisenssit vaativat tekijän nimeämisen teoksessa.

Taulukko 2. Creative commons lisenssi tyypit

Lisenssi tyyppi	Selkokielinen versio	Selitys
CC BY	Nimeä tekijä	Vapaa käyttö, kunhan nimeää alkuperäisen tekijän teoksessaan
CC BY-SA	Nimeä tekijä – Jaa samoin	Vapaa käyttö, teokseen perustuvilla on sama lisenssi ja sallii johdannaisten teosten kaupallisen käytön
CC BY-ND	Nimeä tekijä – Ei muutoksia	Vapaa käyttö, teoksen muutos kiellettyä
CC BY-NC	Nimeä tekijä – Ei kaupallinen	Kaupallinen käyttö kielletty
CC BY-NC-SA	Nimeä tekijä – Ei kaupallinen – Jaa samoin	Ei kaupallinen, sama lisensointi pätee johdannaisissa teoksissa
CC BY-NC-ND	Nimeä tekijä – Ei kaupallinen – Ei muutoksia	Rajoitetuin versio, ei kaupalliseen käyttöön, eikä sitä saa muokata

3 Projektihallinta

Tässä luvussa käsitellään projektihallintaan liittyviä asioita silmällä pitäen ryhmätyöskentelyä. Vaikka peliprojekti toteutettaisiin yksin ilman aikarajoitteita, tämä luku sisältää paljon tietoa, miten projekti pidetään järkevästi dokumentoituna ja hallinnassa, jolloin asiat eivät jää ainoastaan oman muistin varaan.

3.1 Projektihallinnan perusteet

Projektinhallinta on yksinkertaisimmillaan erittäin helppoa, listataan tehtävät, toteutetaan listatut asiat ja lopuksi tarkistetaan, onko ne tehty oikein. Tällä menetelmällä pääsee erittäin pitkälle, mutta koska muuttujia on paljon, kannattaa tehdä syvällisempää suunnittelua.

Lehtimäen (2006, s. 2) mukaan, projektisuunnitelmissa on satoja eri osa-alueita, mutta keskittymällä päätekijöihin pääsee erittäin hyvään vauhtiin. Neljä tärkeintä asiaa:

1. Mitkä ovat projektin tehtävät.
2. Kuka tehtävän tekee.
3. Milloin tehtävä tehdään.
4. Mitä tehtävistä syntyy.

1. Mitkä ovat projektin tehtävät eli mitä tarvitaan, jotta kyseinen projekti saadaan toteutettua. Pelituotannossa avainasemassa on pelin ideointi eli pohjasuunnitelma ja jonkun asteinen juoni tai tapahtumaketju, mihin pelissä suunnataan. Tästä kannattaa luoda dokumentti tai ajatuskartta, johon voi yhdessä ideoida asioita, mitä peli voisi sisältää. Niistä valitaan toteuttamisen arvoiset asiat ja priorisoidaan niiden tärkeys- sekä toteutusjärjestys.

2. Kuka tekee tehtävän, tämä on tärkeää suunnitella, kuka tekee ja mitä tehdään milloinkin, koska tämä vaikuttaa aikataulutukseen ja siihen, mitä muut voivat tehdä samanaikaisesti. Jos tärkeä osa myöhästyy, se saattaa pysäyttää projektin etenemisen. Mukavuuden ja tehokkuuden parantamiseksi on hyvä huomioida ryhmän osaaminen ja kiinnostuksen kohteet, jotta työt voidaan jakaa järkevästi ryhmän kesken. Hyvä vaihtoehto on parityöskentely, jolloin tiimissä yhdistyy kokenut ja kokematon, jos se on aikataulun puitteissa mahdollista. Tällöin kumpikin voi oppia uutta, kokenut saa uusia näkemyksiä sekä kokematon oppii ja saa tukea. (Lehtimäki, 2006, ss. 16-17)

Kuten opinnäytetyön soveltavassa projektissa, aikataulu on tiukka ja oppimishalu on kaikilla kova. Sovelsimme yksilö- ja parityöskentelyä tilanteen mukaan ja jaoimme vastuualueet. Itse toimin soveltavassa projektissa muun muassa projektipäällikkönä. Varmistin, että asiat valmistuvat ajallaan ja paketti pysyy kasassa sekä tarvittaessa muutin ja uudelleen priorisoin tehtäviä tilanteen mukaan. Vaikka työskennellään ryhmänä, jollekin jäsenelle muodostuu nimetön projektin vetovastuu. Jonkun täytyy pitää huolta, että pelisäännöissä ja aikataulussa pysytään. Vaikka projekti jaettaisiin osiin ja kaikilla olisi omat tehtävät mitä suoritetaan, jossain kohtaa tulee vastaan vaihe, kun projekti pitää kasata yhteen. Tätä silmällä pitäen on hyvä seurata tilannetta ajoissa, jotta vältytään ylimääräisiltä ongelmilta.

3. Milloin tehtävät tehdään, yksi epäonnistuminen voi pysäyttää koko projektin. Mentäessä tuntemattomaan on erittäin hankala luoda tarkkaa aikataulua etukäteen. Muuttujia tulee

paljon itse projektissa ja niin myös ihmisissä. Ihmiset voivat esimerkiksi sairastua, jolloin suunnitelmat ja aikataulut muuttuvat. (Lehtimäki, 2006, ss. 20-22)

Tästä syystä on kannattavaa jättää aikatauluun pelivaraa, jos asia toteutuu aikaisemmin, voidaan se aika käyttää seuraavan osion työstämiseen tai vaihtoehtoisesti ominaisuuksiin, jotka on alun perin jätetty priorisoinnissa vähemmän tärkeisiin ominaisuuksiin.

4. Mitä tehtävistä syntyy, tällä tarkoitetaan sitä, että jokaisen projektin osa-alueen on tarkoitus tuottaa jotakin. Suunnitteluvaiheessa tulee keskittyä siihen, mihin lopputulokseen projekti tähtää. Erityisen tärkeää on tehdä kaikille osapuolille selväksi, mitä kohti on tarkoitus suunnata, jolloin kokonaiskuva pysyy projektin jäsenille selkeänä. (Lehtimäki, 2006, s. 22)

3.2 Projektin pelisäännöt

Projektin alussa on kannattavaa luoda standardimenetelmät, joita noudatetaan projektin aikana. Jos asioista ei ole sovittu etukäteen, kaikki toimii oman mielensä mukaisesti, törmätään todennäköisesti tilanteeseen, että projektin osat eivät välttämättä kytkeydy toisiinsa kuten on toivottu. Näitä ongelmia voi tulla esimerkiksi mittasuhteissa ja ohjelmoinnissa. Toimintamallin luominen on tärkeää myös kansioiden ja tiedostojen hierarkiaan sekä miten projektin dokumentaatioita toteutetaan. Aluksi pääprioriteetti on sopia tapa, jolla versionhallintaa toteutetaan projektissa. (Lehtimäki, 2006, ss. 94–99, s. 122)

Ohjelmoinnin osalta oman koodin tulee noudattaa sovittua koodaustyyliä, joka voi olla yleisesti käytetty standardi tai muu sovittu käytäntö, joka määrittelee miten luokkia, funktioita, muuttujia ja niin edelleen käytetään projektin sisällä. Etenkin kun puhutaan Unitysta, jossa samaa koodia voidaan ja tullaan käyttämään lukuisissa eri tilanteissa, joten sen tulisi noudattaa samaa logiikkaa kuin muukin koodi. Pienissä peliprojekteissa poikkeavaisuuksista ei muodostu niin suurta ongelmaa, mutta pelin luonteesta riippuen sillä saattaa olla suurikin vaikutus tehokkuuden ja lopputuloksen kannalta.

Nämä osa-alueet voivat tuntua turhalta ylimääräiseltä työltä, mutta aikaa säästyy paljon, kun kaikki pidetään hallinnassa projektin aikana. Useasti törmää tilanteisiin, jossa projektinhallinta on aloitettu hyvin, mutta pikkuhiljaa asioista lipsutaan. Kun projektia tarkastellaan myöhemmin, voidaan huomata massiivinen ero, kun säännöistä on päästetty irti. Pienenkin asian selvittäminen ja hahmottaminen kestää kohtuuttoman pitkään. Yksi tärkeä syy pitää projektihallinta järjestyksessä on aika, jota säästetään projektin selvityksissä, muutoksissa ja lopulta projektin julkaisun, loppuraportin tai luovutuksen yhteydessä.

3.3 Kommunikointi projektiryhmässä

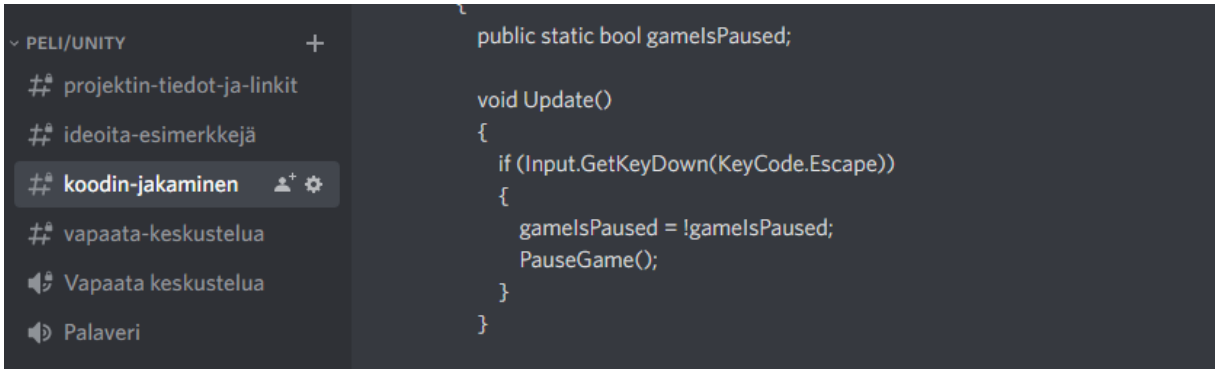
Kommunikointi ja tiedon jakaminen ryhmän sisällä on avainasemassa, se helpottaa yksilön toimintaa, kun kokonaiskuva on selvillä. Huonolla kommunikoinnilla voi olla negatiivisia vaikutuksia projektin etenemiseen. Useampi henkilö voi työskennellä tietämättään saman osion parissa kuin toinen, jolloin aikaa kuluu turhaan tai voi syntyä ongelmia esimerkiksi tiedostojen synkronoinnin osalta. Kunhan muistaa, että kaikkea informaatiota ei tarvitse jakaa. Hyvänä nyrkkisääntönä voi pitää, että haluaisitko itse tietää asiasta, jos olisit toisen henkilön asemassa, ellei tarkempia pelisääntöjä ole viestinnän osalta sovittu. (Lehtimäki, 2006, ss. 56-58)

Suositeltavaa on käyttää useampaa viestintämenetelmää samanaikaisesti. Yksi alusta, jossa ideoidaan ja voidaan keskustella vapaasti projektiin liittyvästä, säilyttäen kuitenkin viestintä hallittuna ja asiallisena. Tämän tueksi voidaan lisäksi käyttää pikaviestimiä, johon on mahdollista luoda ryhmä projektin jäsenille, esimerkiksi Whatsapp tai Telegram, jotka toimivat sekä puhelimella että tietokoneella. Tärkeintä on, että ryhmän henkilöt käyttävät kyseistä sovellusta ja viestit kulkeutuvat heille vaivattomasti. Tällöin on helppo sopia palavereita, ilmoittaa muuttujista tai pyytää apua ongelmatilanteissa nopeasti.

Nykypäivänä Discord on suosittu alusta projektinhallinnan avuksi ja viestintään. Discordiin voi luoda useita teksti- ja puhekanavia eri tarpeisiin ja määritellä kenellä on oikeus tehdä mitäkin. Sinne voi kiinnittää esimerkiksi linkit pilvipalveluissa sijaitsevaan dokumentointiin. Tämän kaltainen käytäntö mahdollistaa, että tieto on saatavilla helposti ja selkeästi kaikille ryhmänjäsenille. Discordin huono puoli on, että ilmoitukset eivät välttämättä tule perille

reaaliajassa kaikille laitteille. Lisäksi tärkeä tieto saattaa kadota viestitulvan uumeniin, jolloin on kannattavaa kirjata ne projektin dokumentointiin. Kuvassa (Kuva 6) on esimerkki soveltavassa projektissa käytetystä kanavajaottelusta Discord-ympäristössä.

Kuva 6. Esimerkki soveltavan projektin työalustana käytetystä Discordista ja sen kanavista, jotka sijaitsevat kuvan vasemmassa reunassa.



Toinen hyvä vaihtoehto on Microsoft Teams yhdistettynä Microsoft OneDriven kanssa, joka on suosittu yrityksien käytössä ja asiakkaiden kanssa viestimiseen. Tiedostojenhallinta on selkeää kansiomaisella rakenteella, dokumentointia ja muita tiedostoja voi muokata suoraan näkymässä samanaikaisesti muiden kanssa. Sen kautta voi luoda palavereita myös projektin ulkopuolisten henkilöiden kanssa.

Hyvä kombinaatio on yksi pikaviestin, esimerkiksi Whatsapp-sovellus, jossa tiedotetaan lyhyesti tärkeät asiat. Yksi työskentelyä varten, josta löytyy kytkentä kaikkialle projektissa. Tätä varten Discord on erittäin ylivoimainen etenkin, jos ryhmäntyöskentely toteutetaan etänä. Yksi alusta dokumentaatioille ja muille tiedostoille, esimerkiksi Microsoft Onedrive, Google Drive tai Github, sekä tarvittaessa oma alusta peliprojektille riippuen käytettävästä pelimoottorista.

3.4 Projektikansio

Projektikansiolla tarkoitetaan paikkaa, jossa on kaikki projektiin liittyvät dokumentit sekä materiaalit tallessa ja järjestyksessä. Tällöin on helppo löytää, mitä milloinkin tarvitaan. Monesti projektien tiedostot ajelehtivat paikasta toiseen ja versioita on useita, joista on

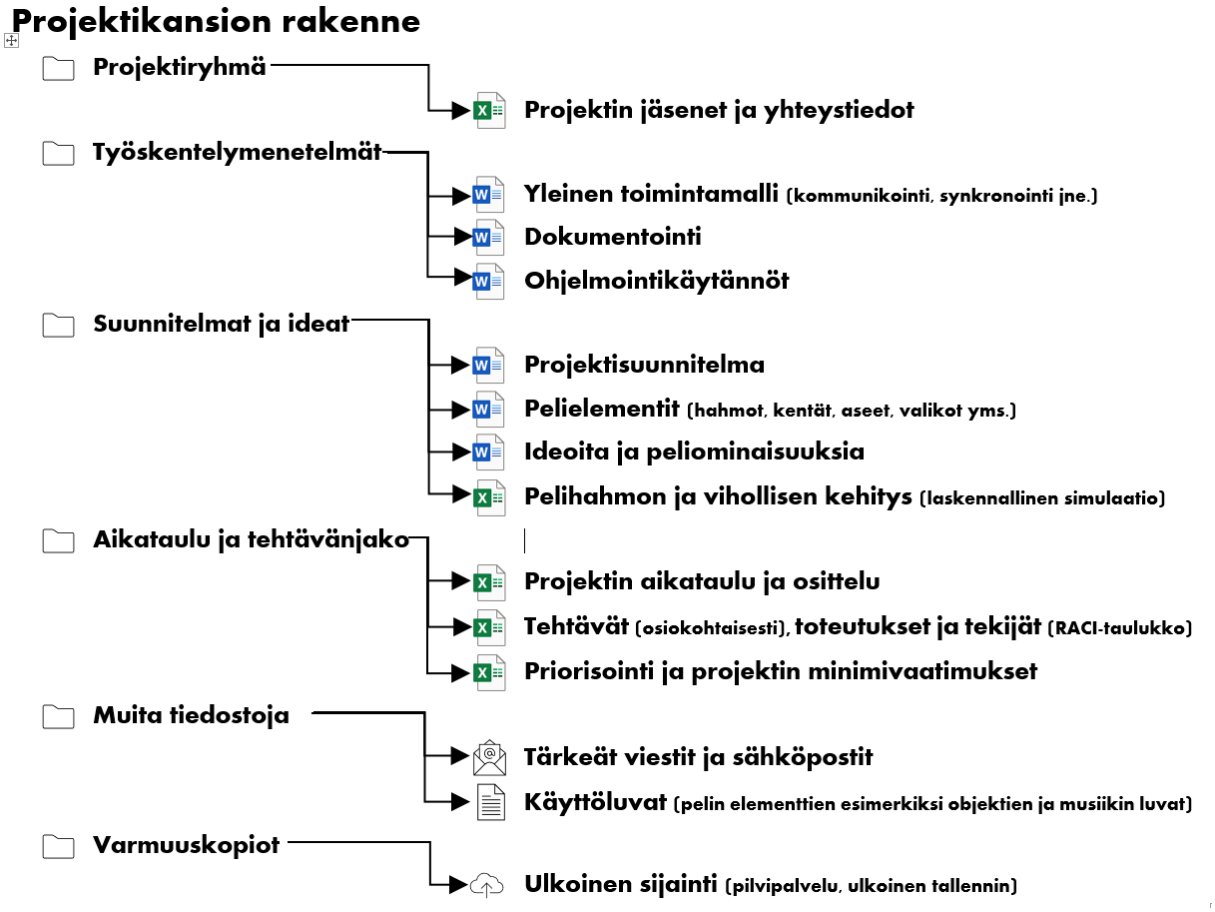
lopuksi vaikea hahmottaa, mikä on viimeisin versio ja onko siinä kaikki ajan tasalla. Myös irtonaiset sähköpostit ja viestit, joissa on jotain oleellista tietoa, kannattaa lisätä projektikansioon. (Lehtimäki, 2006, ss. 45-46)

Yhteisprojekteissa kannattaa turvautua pilvipalveluihin tai muihin vastaaviin ohjelmistoihin, joissa voi jakaa dokumentit ajantasaisena ja on mahdollisuus palata edelliseen versioon. Hyviä alustoja ja ohjelmistoja löytyy dokumentointia varten ilmaiseksi. Hyviä vaihtoehtoja ovat esimerkiksi Google Drive ja Microsoft Onedrive, joissa tiedostoja on mahdollista muokata suoraan selaimen kautta. Itse peliprojekti kannattaa kuitenkin hallinnoida sille soveltuvalla erillisellä alustalla. Esimerkiksi Unityn tiedostot, jotka tallentuvat binäärisessä muodossa. Harvat pilvipalvelut tukevat binääristen tiedostojen tallentamista ja käyttämistä tehokkaasti pelikehityksen näkökulmasta, vaikka niiden tallentaminen on mahdollista, jos pilvipalvelun sallittu yksittäisen tiedoston maksimikoko ei ylity.

Dokumentointi tuntuu useasti ajanhaaskaamiselta, mutta se on tärkeä osa projektia. Hyvin toteutettuna dokumentointiin käytettävä aika maksaa itsensä takaisin projektin edetessä. Tämän ansiosta myös jokaisella ryhmän jäsenellä on tiedossa mitä on työn alla ja mitä on tekemättä. Kun dokumentointi on kunnossa, tiedetään mitä on tutkittu ja miksi se on hyvä toteuttaa tai miksi näin ei kannata tehdä.

Jokainen projekti on erilainen, joten dokumentointitarpeet vaihtelevat. Pienissä projekteissa riittää useasti muutamaan tiedostoon kirjaaminen, mutta selkeyden vuoksi ne on jaettu esimerkissä useampaan dokumenttiin. Kuten kaikessa, varmuuskopiointia on hyvä suorittaa säännöllisesti. Seuraavassa kuvassa (Kuva 7) on esimerkki projektikansion rakenteesta.

Kuva 7. Esimerkki peliprojektin projektikansiosta



Tämä on vain karkea esimerkki projektikansion toteutukselle. Seuraavassa luvussa käydään läpi muutama työkalu, joita käytetään projektihallinnassa.

3.5 Projektihallinnantyökalut

Projektihallintaan on olemassa paljon erilaisia työkaluja ja niiden käyttäminen tehostaa työskentelyä, jos vain ymmärtää miten ne toimivat. Työvälineet kannattaakin valita projektin mukaan, pieneen projektiin voi riittää Excel-taulukko ja muutama dokumentti. (Lehtimäki, 2006, s. 196)

Microsoft Office tarjoaa monipuolisia ohjelmistoja erilaisiin tilanteisiin ja monet pilvipalvelut tukevat näiden tiedostojen käyttöä suoraan selaimella. Dokumentaatioita voi toteuttaa Microsoft Wordilla. Microsoft Excel on kevyt ja tehokas taulukkolaskentajärjestelmä, jonka

avulla on nopea toteuttaa aikataulut ja tehtävälistat. Excel on myös useasti käytetty työkalu, kun luodaan ja simuloidaan pelissä tapahtuvia muutoksia, esimerkiksi hahmojen kehitystä ja vaikeustason muutoksia.

Gitlab on pienimuotoiseen työskentelyyn ilmainen verkkotyökalu, jolla voi hallita projektia kuin projektia. Peliprojekteissa työkalun hyödyt keskittyvät pääasiassa dokumentointiin, tehtävien seurantaan ja aikataulun hallintaan. Gitlabin käyttöönottoon ja hallintaan projektia varten menee kokeneeltakin käyttäjältä reilusti aikaa, joten se kannattaa huomioida aikataulussa.

Pelikehityksen näkökulmasta työkalun valinta riippuu siitä, miten projektia työstetään ja miten tarkka suunnitelma on olemassa. Gitlabin kaltainen työkalu on liioittelua, jos ryhmässä on vain muutama henkilö tai jos projektia ei pilkota pieniin kokonaisuuksiin. Useasti pelikehityksessä runko luodaan ennen projektin aloitusta ja lopullinen muoto muodostuu projektin edetessä. Useasti pienissä projekteissa tehokkain vaihtoehto on yksinkertaistettu versio, joka koostuu tekstistä ja taulukoista.

3.6 Action List - toimenpidelista

Ryhmässä työskenneltäessä on todennäköistä, että jäsenet työstävät eri asioita samanaikaisesti. Tällöin käyttöön otetaan Action List eli toimenpidelista, jossa näkyy tehtävät, joita työstetään, kuka työstää tehtävää, mitkä osiot on toteutettu valmiiksi ja kenen toimesta. (Lehtimäki, 2006, s. 119)

RACI-taulukko, on yleinen standardi, joka on luotu tätä varten. RACI tarkoittaa R, responsible eli kenelle kyseinen tehtävä kuuluu. A, accountable eli tämä henkilö seuraa, että kyseinen tehtävä tulee tehdyksi ajallaan. C, consulted eli henkilö, jolta saa apua ja jolla on mahdollisesti aikaisempaa kokemusta vastaavasta. I, informed eli henkilö, joka pidetään ajan tasalla tehtävän tilanteesta. Yleensä I-henkilö tarkkailee yhden osa-alueen suurempaa kokonaisuutta ja mahdollisesti yhdistää tuotetut palaset yhteen. (Lehtimäki, 2006, ss. 119-120)

Seuraavassa pelkistetyssä (Taulukko 3) on esimerkki RACI-taulukon käyttämisestä peliprojektin ensimmäisessä vaiheessa.

Taulukko 3. RACI-taulukolla toteutettu tehtävien jako.

Tehtävä	Matti	Pekka	Toni	Anna
Päähahmo Grafiikka	A/R	C	I	
Päähahmon Aseet	A/R	C	I	
Päähahmon perusliikkeiden ohjelmointi	A	R/C	I	
Testikentän luominen			A/R	I
Testikentän interaktiivisuus			A	R

RACI-taulukosta on olemassa myös laajennettu versio RACIVS, joka tuo kaksi merkintää lisää. V, Verify, eli kuka tarkastaa tehtävän suorituksen ja onko se kelpaava projektille. S-henkilö suorittaa virallisen hyväksynnän, tämä henkilö on tavallisesti projektijohtaja. (Lehtimäki, 2006, s. 121)

RACI-taulukon käytön voi yhdistää samaan taulukkoon, jolla seurataan projektin aikataulua ja sen etenemistä sekä tehtävien priorisointia.

3.7 Projektin aikataulutus ja laatutason ylläpito

Aikataulussa pysyminen on helppoa, täytyy seurata luotua suunnitelmaa ja pitäytyä siinä. Suunnitelman luominen tarkasti on haastavaa ja tähän tarvitaan paljon kokemusta, jotta pystytään arvioimaan työtunnit kutakin tehtävää varten. Menetelmä, jolla helpotetaan aikataulun hallintaa, on jakaa yksi iso projekti pienempiin kokonaisuuksiin. Näitä osioita voidaan työstää mahdollisuuksien mukaan päällekkäin tai rinnakkain. (Lehtimäki, 2006, ss. 156-157)

Otetaan esimerkkinä peliprojekti, joka on jaettu osiin. Käytän esimerkkinä toteutettua soveltavaa peliä Teddy Bear Adventure. Seuraavassa kuvassa (Kuva 8) projekti on jaettu

kahdeksan viikon ajanjaksolle, koska tämä oli pelikurssin pituus. On selvää, että tavoite ja aikataulu eivät kohtaa, joten paljon suunniteltua jouduttiin jättämään pois. Tämä oli tiedostettu etukäteen, mutta emme tienneet, kuinka paljon ehtisimme rakentamaan annetussa ajassa. Suunnittelulle ja Unityn tekniikoiden oppimiselle varasimme ensimmäiset kaksi viikkoa, tänä aikana kehitimme pelin idean ja suurimman osan ideoista, mitä haluamme sisällyttää peliin.

Kuva 8. Soveltavan peliprojektin suunnitelma jaettuna 8 viikon ajanjaksolle.

Projektissa kannattaa pitää kiinni sovituista laatuvaatimuksista. Huonon laadun lisäksi on mahdollista törmätä liian hyvään laatuun. Jos tavoitteena on jokin tietty laatuaste, kokonaisuuden pitäisi pysyä sen sisällä aikataulun ja kokonaisuuden takia. Liian hyvä laatu tarkoittaa enemmän työtunteja, jolloin voidaankin puhua huonosta laadusta, kun se suhteutetaan käytettyyn aikaan ja tarpeeseen. Turhaa hieromista kannattaa välttää, ei ole järkevää ajankäyttöä luoda jotakin, jota vain itse tekijä osaa arvostaa, hyvä laatu riittää. (Lehtimäki, 2006, ss. 77-78)

Esimerkkinä voi mainita, jos pelissä esiintyy pieniä grafiikkaan tai fysiikkaan liittyviä epäkohtia, onko kannattavaa käyttää niiden korjaamiseen paljon työtunteja. Vaihtoehtoisesti voidaan esitellä kyseistä ongelmaa henkilölle, joka ei ole tuttu kyseisen peliprojektin kanssa. Vaikka itse pelintekijä pitäisi asiaa häiritsevänä, normaali pelaaja ei välttämättä kiinnitä asiaan huomiota. Erittäin isojen yrityksien peleissä esiintyy samanlaista oikomista, elementit menevät toisien sisään, näkymättömiä seiniä rajaamassa pelialuetta ja niin edelleen. Jos se ei aiheuta suoranaista ongelmaa itse pelaamiseen, asia kannattaa merkata ylös, jotta asiaan voidaan palata, jos aikataulu tämän sallii.

3.8 Priorisointi

Priorisointi projektin alussa ja uudelleen priorisointi projektin edetessä on elintärkeää. Peliprojekteissa voi tulla vastaan yllätyksiä, joihin menee huomattavasti enemmän aikaa kuin oli suunniteltu. Pelkästään Unity-pelimoottori saattaa aiheuttaa ongelmia, joihin pelikehittäjä ei ole varautunut. Se saattaa viivästyttää projektin etenemistä, joten ei kannata asettautua liian optimistisesti aikataulun pitävyyteen, vaan muutoksiin tulee varautua, jolloin esille nousee priorisointi.

Priorisoidun tehtävälistauksen aikana loimme myös minimivaatimukset, jotka täytyy saada valmiiksi aikataulun mukaisesti. Peliprojektin eri osat jaettiin pienempiin kokonaisuuksiin, joita työstettiin yhtäaikaisesti. Nämä sijoitettiin tärkeysjärjestyksessä pitkin aikajanaa. Järjestys määräytyi täysin sen mukaisesti, mitä täytyy tehdä ensin, että seuraava vaihe on mahdollista toteuttaa. Jos aikataulussa pysyminen vaikutti mahdottomalta, tiputettiin laatutaso alas, jolla pelaaminen onnistuu mukavasti ja eteneminen on mahdollista, mutta ei niin täydellisesti kuin olisi toivottu. Jos suunniteltuun ajanjaksoon jää aikaa, paranneltiin aikaisempia ratkaisuja tai luotiin vähemmän tärkeitä elementtejä peliin eli asioita, jotka olivat pienemmällä prioriteettiasteella.

3.9 Työnjako ja peliprojektin aloitus

Unitylla, kuten muillakin pelimoottoreilla, pelien tekeminen aloitetaan yleensä luomalla prototyyppi pelin ideasta, josta muodostuu useasti samalla testikenttä, jossa uusia

elementtejä ja mekaniikkoja voi testata. Testikentän tarkoituksena on antaa ymmärrystä mittasuhteista ja graafisesta ilmeestä sekä mekaniikasta, jolloin kenttiä on helpompi luoda, kun vertailukohde on olemassa. Prototyypin toteuttamisen jälkeen ryhmä jaetaan toteuttamaan eri osa-alueita.

Peliprojektit koostuvat useista osa-alueista. Näitä ovat muun muassa hahmot, pelikentät, musiikki, ääniefektit, visuaaliset efektit, ohjelmointi, interaktiivisuus ja käyttöliittymät. Ryhmän jäsenmäärän mukaan jaetaan ryhmä toteuttamaan eri osa-alueita samanaikaisesti. Tärkeintä on projektin alussa luoda pelihahmon perusosat, pelikenttä ja käytettävät fysiikan lait, jotka yleensä toteutetaan prototyyppiä luodessa. Seuraavassa vaiheessa luodaan uusia pelialueita ja menetelmä, jolla toteutetaan tiedonsiirto ja siirtyminen kentästä toiseen. Tiedonsiirto ja tallennusmenetelmä on hyvä luoda projektin alussa kuntoon, koska niiden lisääminen voi olla työlästä, jos projekti on edennyt pitkälle. Grafiikan voi jättää myöhemmäksi ja korvata tilapäisesti puuttuvat tekstuurit ja elementit väliaikaisilla objekteilla. Ääni, musiikki ja visuaaliset efektit voidaan myös lisätä tai viimeistellä myöhemmin, mutta ne kannattaa huomioida pelinrakennusvaiheessa, jotta niiden lisääminen on yksinkertaista toteuttaa. Esimerkiksi samat äänet ja visuaaliset efektit sisältäviin elementteihin sisällytetään prefabs -tiedosto, jota muokkaamalla saadaan tuotua lisäominaisuudet useaan objektiin samanaikaisesti. Näitä voivat olla esimerkiksi ammuksien, askelten, hyppyjen aiheuttamat efektit eri pinnoilla.

Unitylla projektin yhtäaikainen käsittely voi osoittautua erittäin haasteelliseksi, koska asiat ovat kytköksissä toisiinsa. Tämä tarkoittaa käytännössä sitä, että muutoksien tekeminen johtaa usein yhteentörmäyksiin ja muutokset saattavat kadota synkronoinnin yhteydessä. Tästä syystä on hyvä olla olemassa useampi projekti, paikallisesti jokaisella ryhmän jäsenellä oma sekä virallinen versio. Tällöin peliä voi rakentaa rauhassa ja tuoda sovitulla aikataululla muutokset viralliseen projektiin, jolloin vältetään yhteentörmäykset.

4 Projektin organisointi ja hallinta Unityssa

Unityssa, kuten muissakin pelimoottoreissa, järjestelmällisyys on erittäin suuressa asemassa. Myös se, miten projekti pidetään turvattuna mahdollisten ongelmien varalta, on

avainasemassa. Seuraavaksi käydään läpi hieman versionhallintaa yleisesti ja miten Unityn hierarkiaa voidaan toteuttaa. Jokainen projekti on kuitenkin erilainen, joten toimintamallit vaihtelevat.

4.1 Versionhallinta

Versionhallinnalla tarkoitetaan lyhyesti projektin ja tiedostojen varmuuskopiointia, mahdollisten laiteongelmien ja käyttäjistä johtuvien ongelmien ehkäisyä. Ilman versionhallintaa koko projekti saattaa mennä pilalle tai kadota muutaman virheklikkauksen takia. Versionhallinnan toteuttamiseen on monia tapoja, joista osa on maksullisia ja osa ilmaisia. Suosittuja tapoja on paikalliset tallenteet projektista ja pilvitallennus. (Lehtimäki, 2006, ss. 101-102)

Versionhallinnan sopivuus pelimoottorille ja muille tiedostoille vaihtelee palveluntarjoajan toimintatavan mukaan. Versionhallinnan toinen tärkeä tarkoitus on luoda mahdollisuus palata mahdollisimman helposti edelliseen versioon, jos todetaan ongelmia tehdyissä muutoksissa. Tämä auttaa myös, jos useampi ryhmän jäsen on tehnyt päällekkäisiä muutoksia, jolloin tietyissä versionhallinta menetelmissä on mahdollista yhdistää tehtyjä muutoksia ilman, että tietoa katoaa. Seuraavaksi tarkastellaan tarkemmin yleisiä tapoja toteuttaa versionhallintaa, mitkä ovat hyviä vaihtoehtoja Unityn binääripohjaisen projektien tallentamiselle.

4.1.1 Versionhallinta sovelluksia

Ohjelmistokoodeihin ja pieniin tiedostoihin soveltuu hyvin esimerkiksi GitHub versionhallintajärjestelmä, joka automaattisesti tarkistaa onko tehty päällekkäisiä muutoksia ja näyttää, mitä muutoksia käyttäjä on tehnyt ja auttaa yhdistämään koodin yhtenäiseksi. Tämä perusrakenne ei tosin sovellu binääripohjaiseen tallennukseen. Se on hyvä vaihtoehto dokumentaatioille ja ohjelmointikoodille, koska tällöin muutokset ovat nähtävissä ja lisäksi on mahdollisuus palata edelliseen versioon. Tähän on myös saatavilla Git LFS -laajennus, jonka avulla on mahdollista tallentaa useita gigan kokoisia tiedostoja. (GitHub, n.d.)

Binäärimuotoisten tiedostojen tallentamiseen soveltuu erittäin harva versionhallintajärjestelmä. Testien jälkeen parhaimmaksi yleisesti binäärien tallentamiseen valikoitui Perforce helix-core, joka on ilmainen viidelle käyttäjälle ja 20 työpisteelle. (Perforce, 2022)

Etenkin Unityn kanssa Plastic SCM cloud edition on hyvä vaihtoehto, joka on Unityn oma versionhallintajärjestelmä. Se tarjoaa ilmaisen käytön kolmelle käyttäjälle ja 5GB tallennustilan. (Unity, 2022) Hyvin hallittu pieni Unityn peliprojekti mahtuu pääsääntöisesti 5GB ilmaiseen tilaan, mutta jos assetteja ja elementtejä lisätään ilman harkintaa, projektin koko nousee nopeasti.

Kaikessa binääripohjaisessa versionhallinnassa voi esiintyä paljon erilaisia haasteita. Useasti ongelmia tuottaa ryhmätyöskentely, koska binääripohjaiset tiedostot toimivat kokonaisuuksina. Tästä syystä kokonaisuuden hallinta ja työnjako kannattaa pitää selkeänä tai vaihtoehtoisesti pitää virallinen projekti erillään ja käyttää väliaikaisia projekteja työskentelyyn.

4.1.2 Versionhallinta paikallisesti

Versionhallinta voidaan hoitaa myös paikallisesti ja manuaalisesti. Tämä tosin tarkoittaa sitä, että edelliseen versioon ei voi vain palata, vaan vanhempi versio projektista tulee ottaa kokonaisuudessaan työn alle. Tämä myös vaikeuttaa ryhmässä työskentelyä, koska virallinen projekti löytyy vain joltakin ryhmän jäseneltä. Myös ison tiedoston jakaminen ja lataaminen toistamiseen vie paljon aikaa. Tämä menetelmä soveltuukin parhaiten yksintyöskentelyyn tai jos yksi henkilö hallitsee peliprojektia ja muut tuottavat hänelle osia ja elementtejä peliin. Vaikka käytettäisiin palveluntarjoajan versionhallintaa, kannattaa varmuuskopioita ottaa myös paikallisesti mahdollisten ongelmatilanteiden varalta.

Varmuuskopiot on hyvä tallentaa ulkoiseen tallennuspaikkaan. Lisäksi voi tehdä dokumentaatio versioiden välisistä muutoksista. Versionhallinta tai pikemminkin varmuuskopiointi kannattaa suorittaa esimerkiksi kerran päivässä tai tarpeen mukaan

riippuen projektin etenemisestä. Myös projektin muista dokumentaatioista kannattaa tehdä varmuuskopiot säännöllisesti, joko paikallisesti tai pilvipalveluun.

4.2 Resurssienhallinta ja perusteet Unity-ympäristössä

Käydään läpi Unityn sisällä toteutettavaa resurssienhallintaa yleisellä tasolla ryhmätyöskentelyn näkökulmasta. Unity-projekteissa on usein paljon tiedostoja, koska eri toiminnot jaetaan usein omiin tiedostoihin, jotta niiden käyttäminen on tehokkaampaa.

4.2.1 Unityn näkymät ja peruskäsitteet

Unityssa on useita eri elementtejä, jotka ovat pakollisia toimivuuden kannalta. Project eli projekti, joka pitää sisällään kaiken projektiin tuodun sisällön. Useasti pelikehityksessä hyvä käytäntö on luoda vähintään kaksi projektia. Toinen projekti testailuun ja lisäosien latailuun sekä toinen pelin pääprojektiksi, jonka sisältöä ja rakennetta valvotaan tarkasti. Viralliseen projektiin ei kannata ladata esimerkiksi assetteja suoraan Assets storesta, vaan tarvittavat komponentit kannattaa ladata testiprojektiin, jonka jälkeen tarvittavat elementit siirretään prefab -pakettina varsinaiseen peliprojektiin. Tämä menetelmä mahdollistaa paremman resurssienhallinnan, kun vain tarvittavat elementit lisätään peliprojektiin ja hierarkia toteutuu hallitusti. Lisäksi menetelmä pitää projektin koon hallinnassa, joka tuo rahallisen säästön pilvipalveluiden käytössä.

Scene eli näytös tai näkymä, näitä voi olla yhdessä projektissa useita riippuen pelin koosta ja sen rakennustavasta. Yhdellä scenellä voi luoda kokonaisen pelin, mutta on selkeämpää käyttää useampaa sceneä. Myös scene voi sisältää muita scenejä, ne ovat kätevä tapa tuoda ja poistaa isoja sisältöjä pelistä pelin aikana. Tällaisia voi olla esimerkiksi erilaiset valikot ja muut vastaavat, joihin on tarve päästä käsiksi useasta eri scenestä. Main scene eli päänäkymä pitää sisällään kaikki elementit, jotka näkyvät ruudulla ja ovat käytettävissä kyseisessä tilanteessa. Scenen ymmärtäminen on yksi tärkeimpiä asioita Unityssa, koska sitä käytetään monipuolisesti. Jos esimerkiksi vaihtaa scenestä toiseen, tiedot katoavat, ellei niitä erikseen tallenneta ja siirretä. (Docs Unity3D, 2022c)

4.2.2 Nimeäminen ohjelmoinnissa

Pelit ja ohjelmistokoodit koostuvat yleensä sadoista tai jopa miljoonista erilaisista elementeistä, joista jokaiselle pitää antaa asiaa kuvaava nimi. Hyvä nimeäminen on tärkeää kaikenlaisessa ohjelmoinnissa, koska se ensinnäkin helpottaa koodin myöhempää tarkastelua ja toimintojen hahmottamista, lisäksi se tekee koodista selkeän. Hyvin nimetyt muuttujat maksavat vaivan takaisin. (Karaś, 2020)

Ennen tavoitteena oli kirjoittaa tiivistä koodia, jota lyhennettiin mahdollisimman paljon. Tällöin koodi vei vähemmän tilaa ja koodin kirjoittaminen oli hieman nopeampaa. Tämä tapa on jäänyt näiltä ajoilta käyttöön ja tavasta hankkiudutaan pikkuhiljaa eroon, koska nykypäivänä tallennustilaa on reilusti saatavilla ja ohjelmointiin käytettävät apuohjelmat nopeuttavat koodin kirjoittamista ja tilaa on käytettävissä runsaasti.

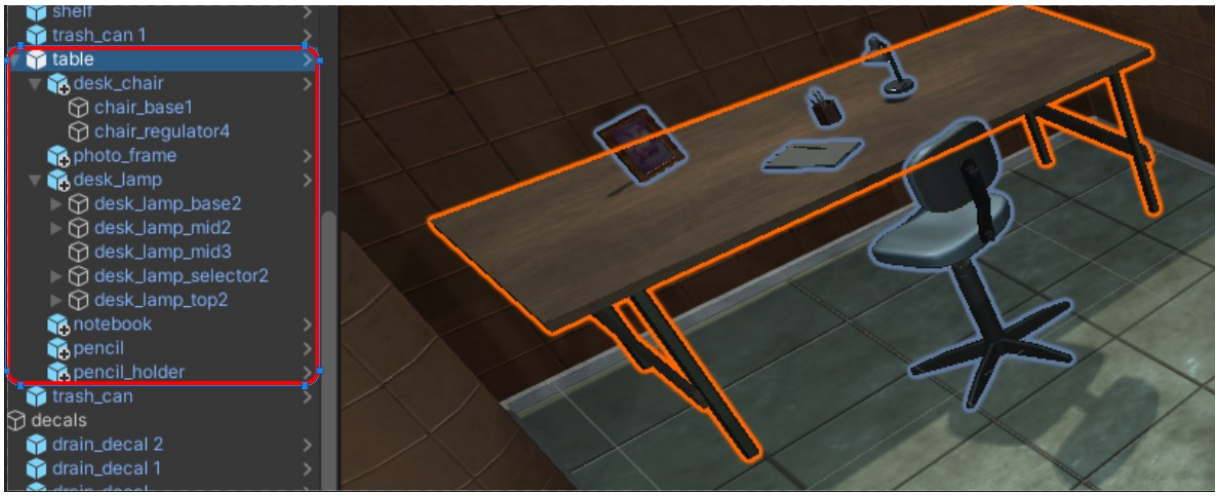
Koodin kuvausta voi laajentaa kommentoimalla koodia esimerkiksi funktioiden yhteyteen. Lyhenteitä ei tulisi käyttää, jos kyseistä muuttujaa kutsutaan ja käytetään myös muualla koodissa. Selkeän tarkoituksen omaavia lyhenteitä voi käyttää ohjelmoinnissa paikallisesti, esimerkiksi funktion sisällä, kunhan funktion nimestä tulee selville, mikä on kyseisen funktion tarkoitus. Tulee muistaa, että koodin tulisi olla selkeää myös muille ryhmäläisille. (Karaś, 2020)

4.2.3 Hierarkia ja elementtien nimeäminen Unityn scenessä

Hierarkian hallinta ja selkeä elementtien nimeäminen on tärkeässä roolissa. Hierarkian looginen rakenne ja nimeäminen on tärkeää etenkin kenttien suunnittelussa ja rakentelussa. Loogista on, että talon sisään on sijoitettu huone ja huoneessa olevalla pöydällä olevat tavarat on sijoitettu aina isomman elementin alielementteihin. Esimerkiksi pääelementtiin on liitetty myös siihen kuuluva tuoli kuten kuvassa (Kuva 9). Tämä mahdollistaa alempitasoisten elementtien sijainnin, koon ja kulman muuttamisen ylemmän elementin

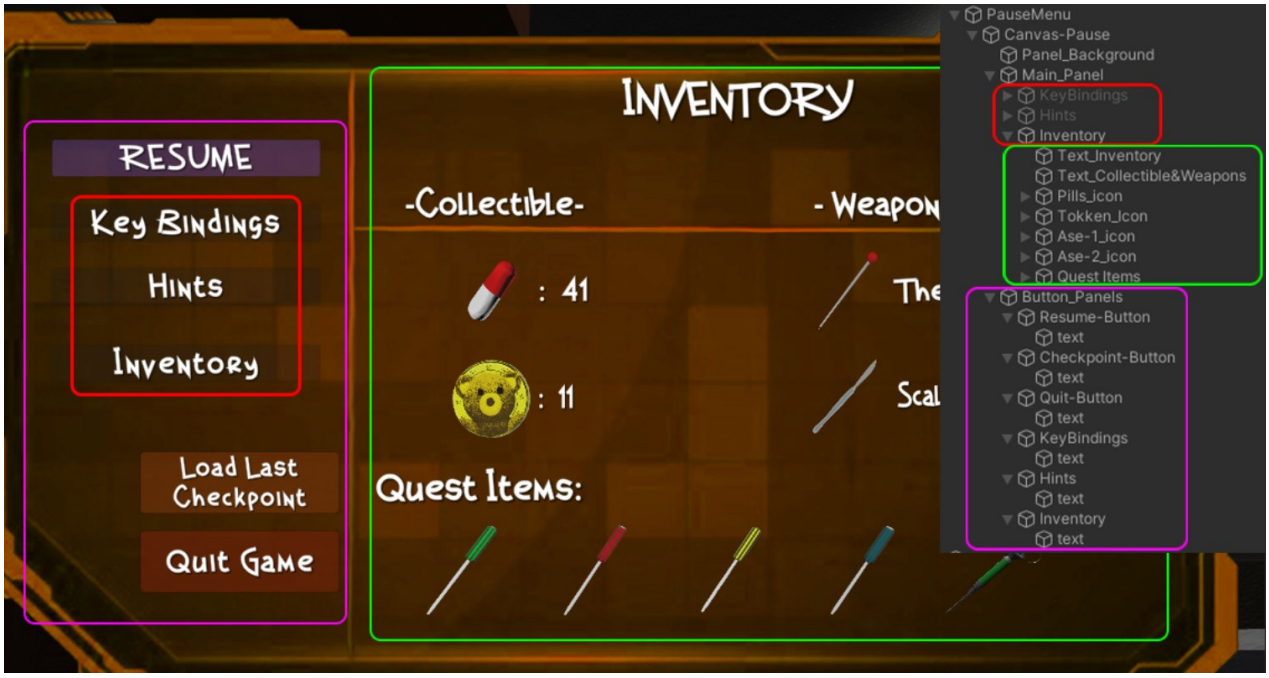
avulla. Alemman tason elementti on kiinnitetty aina ylemmän tason elementtiin ja muuttuvat tällöin ylempää elementtiä muutettaessa.

Kuva 9. Elementtien nimeäminen ja hierarkia Unityn scenessä.



Esimerkiksi UI eli käyttöliittymä, joka on 2D-näkymä ruudulla. Tällöin myös järjestyksellä on merkitys, koska alempi taso näkyy käyttäjälle oletuksena ensimmäisenä. Kuvassa (Kuva 10) on soveltavasta pelistä oleva pause -valikko, joka näkyy ruudulla 2D-näkymänä. Elementit on nimetty selkeästi hierarkiassa. Kuvassa on aktiivisena Inventory eli tavaraluettelo, tästä syystä keybindings ja hints valikko näkyy piilotettuna hierarkia valikossa.

Kuva 10. Soveltavassa pelissä käytetty valikko ja sen hierarkia. Väreillä selkeytetty rakenteen sijoittelua.



Pelielementtien nimeämisessä auttaa myös hierarkia. Tällöin ylempien tasojen nimeäminen ilmaisee alielementin paikan pelissä, jolloin elementin nimi voi olla yksinkertaisempi ja kyseistä elementtiä kuvaava.

4.2.4 Unityn resurssienhallinta näkymä

Unityn resurssienhallinnasta löytyy kaikki projektissa olevat elementit ja hallinnan tärkeys korostuu projektin kasvaessa suuremmaksi. Hierarkian ja hallinnan toteuttamisen voi tehdä monella eri tavalla. Käytettävä menetelmä riippuu pitkälti siitä, kuinka paljon ja minkälaisia elementtejä pelin projekti sisältää. Seuraavassa esimerkissä (Kuva 11) tarkastellaan soveltavassa peliprojektissa käytettyä toimintamallia.

Kuva 11. Soveltavassa peliprojektissa käytetty toimintamalli Unityn resurssienhallintaan.

Sisältö on jaettu aluksi kahteen kategoriaan. Pelin elementteihin (a Game Assets) ja Scenes eli pelin päänäkymät sisältävään kansioon. Pelin elementit on jaettu kuuteen osaan: pelihahmot (Characters), testityökalut (Developer tools), elementit (Elements), kerättävät esineet (PickUps), scenejen välisten tietojen ja hahmon siirtäminen (Teleports) ja käyttöliittymät (UI).

Resurssienhallinnan helpottamiseksi kaikki kansiot ja rakenteet noudattavat samaa logiikkaa. Kaikista elementeistä on luotu prefabs -tiedosto, joka sisältää kaiken tarvittavan, jotta objekti käyttäytyy halutulla tavalla pelissä. Tällöin objektien uudelleen käyttäminen pelialueita rakentaessa on mutkatonta.

4.3 Uudelleen käyttäminen

Re-Use on termi, jolla tarkoitetaan luotujen osien uudelleen käyttöä. Siitä puhutaan paljon, mutta todellisuudessa sitä ei tehdä kovin usein, ellei kyseessä ole vastaava projekti tai jatkoa

edelliseen projektiin. Kuitenkin useimmiten uudelleenkäyttämistä tapahtuu kyseisen projektin sisällä. (Lehtimäki, 2006, ss. 168-169)

Unity-pelimoottorissa luodaan paljon asioita, joita voidaan käyttää uudelleen helposti. On mahdollista pakata esimerkiksi koko pelialue tai päähahmo ja siirtää nämä kaikkien ominaisuuksien kanssa toiseen peliin. Unityssa käytetään paljon Prefab-paketointia, joka sisältää aivan kaiken tarpeellisen koodista lähtien. Tietenkin silloin, kun projekti on rakennettu kokonaisuudessaan loogisesti ja selkeästi. Toisin sanoen pitämällä projektin tallessa, sitä voi hyödyntää erittäin vahvasti uudessa projektissa etenkin, jos kyseessä on jatko-osa, jolloin voidaan käyttää paljon ominaisuuksia edellisestä pelistä. Peliä luodessa, elementtejä tulee ja kannattaa käyttää uudestaan. Esimerkiksi viholliset, huonekalut, rakennukset ja niiden rakenteet ovat hyviä elementtejä, joita kannattaa käyttää uudelleen. Hieman ulkoasua tai materiaalia muuttamalla elementit vaikuttavat erilaisilta. Esimerkiksi voisi ottaa soveltavassa pelissä olevan vihollisen (Kuva 12), jonka väriä ja ominaisuuksia on muutettu.

Kuva 12. Vihollisen rungosta luotu kolme eri vihollista, yksinkertaisesti scriptiä muokkaamalla ja vaihtamalla pintamateriaali.

Tässä kohtaa voisi nostaa ison pelialan yrityksen pelisarjan esille. Tomb Raider ja sen pelit vuosilta 2010–2020, jotka ovat peruspohjaltaan täysin yhteneväiset, vaikka pelinkehittäjä vaihtui 2017 vuoden jälkeen (”Tomb Raider”, 2022). Vaikuttaa siltä, että Re-use on ollut erittäin suurena tekijänä jatko-osia luodessa. Ainoa selkeä muutos näissä peleissä on pelattava pelialue. Tämän lisäksi päähahmo on saanut uusia taitoja aikaisemmista peleistä tuttujen lisäksi ja valikoiden grafiikka sekä tunnelma on muutettu. Henkilökohtaisesti pelisarjan pelit tuntuvat yhdeltä ja samalta peliltä, jossa hahmo kadottaa taitonsa jatkaessaan seikkailuja uudessa pelissä. Tämä toimintamalli on erittäin hyvä tapa säästää paljon työtunteja kierrättämällä aikaisemmin rakennettuja elementtejä.

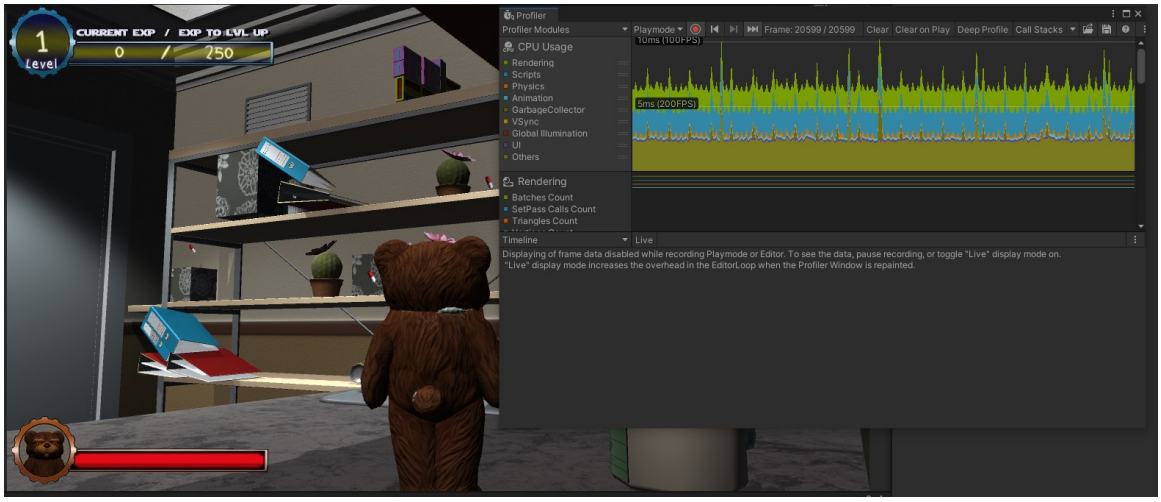
5 Pelin suorituskyvyn ja tallennustilan hallinta Unityssa

Seuraavaksi käsitellään murto-osa perusasioita, jotka vaikuttavat pelin suorituskykyyn Unityssa, jotka on jätetty useasti huomioimatta Unity-pelimoottorilla luoduissa peleissä. Näiden tekijöiden huomioiminen riittää useasti pienimuotoisessa pelinkehityksessä.

Unity-pelimoottorista löytyy monipuolinen Profiler-työkalu, jolla voidaan tarkastella pelin suorituskykyä ja selvittää mahdollisia ongelmakohtia, jotka vaativat korjausta optimoinnin näkökulmasta (Kuva 13). Soveltavassa peliprojektissa huomioitiin tässä kappaleessa käytettyjä optimoinnin peruspilareita, jonka lopputuloksena saavutettiin peli, jonka pelaaminen luonnistuu sujuvasti peruskäyttöön tarkoitetulla tietokoneella.

Profiler-työkalu sisältää erittäin paljon tarkkaa informaatiota, joten sitä ei avata syvällisemmin tässä opinnäytetyössä.

Kuva 13. Unity-sovelluksen profiler-työkalu soveltavassa peliprojektissa.



5.1 Suorituskyky ja sen tehostaminen Unityssa

RAM(Random-access memory) laitteen välimuisti ja VRAM(video random-accesss memory) kuvamuisti, joka tallentaa fyysisesti nähtäviä asioita laitteen näytöllä. Eri ohjelmointikielissä ja pelimoottoreissa hallitaan muistimoduulien käyttöä eri tavoin, joka pitää ottaa huomioon pelin luomisessa. Vanhoissa ohjelmointikielissä muistista täytyy varata paikka etukäteen. (Techcolleague, n.d.). Nykypäivän laitteet ovat tehokkaita, joten ei ole suurta merkitystä varaako koodia kirjoittaessa mahdollisimman pienen vai hieman isomman muistialueen, kun luodaan pieniä tai keskisuuria pelejä.

Suorituskyvyn optimoinnin tärkeys korostuu, kun peli suunnitellaan mobiililaitteille, koska mobiililaitteiden suorituskyky on huomattavasti alhaisempi kuin keskiverto tietokoneiden. Mobiililaitteiden tehokkuus on noin kymmenen prosenttia pöytäkoneiden tehokkuudesta, etenkin näytönohjaimella toteutettavan työn osalta. Tällöin on tärkeää varmistaa, että kaikki elementit ovat optimoituja ja peli ei käytä resursseja huonosti optimoituun sisältöön. Seuraavissa luvuissa käydään Unityn optimoinnin perusasioita, jotka kannattaa huomioida peliä tehtäessä. Maksimaaliseen optimoinnin saavuttamiseen vaikuttaa moni asia seuraavien lisäksi, mutta seuraavilla toimenpiteillä pärjää useammissa tilanteissa.

5.2 Välimuistin ja laskentatehon käyttäminen Unityssa

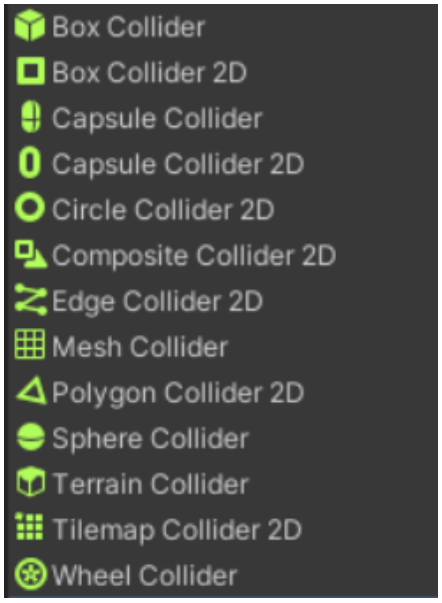
Unity ja käytetty ohjelmointikieli C Sharp (C#) hallitsee itsenäisesti välimuistin käyttöä ja tilan varaamista tehokkaasti. Pelikehittäjä voi tosin luoda epähuomiossa kasoittain muistia kuluttavia elementtejä peliin, jotka voitaisiin toteuttaa tehokkaammin. Peleissä on yleisesti paljon erilaisia elementtejä, joista erittäin monet eivät ole interaktiivisessa käytössä.

Static element eli ei interaktiivisessa käytössä oleva elementti (staattinen elementti) tarkoittaa pelissä olevaa objektia, joka ei reagoi fysiikkaan. Staattiset elementit eivät voi liikkua ja pääsääntöisesti ne ovat vain koristamassa pelikenttää. Objektit kannattaa muuttaa staattisiksi, jolloin ne kuluttavat vähemmän resursseja. Jokainen törmäys (collider) kuluttaa resursseja, vaikka objekti olisi liikkumaton. (Brusca, 2022, ss. 140-142)

Colliderin käyttö mahdollistaa törmäyksen elementtien välillä ja estää niitä läpäisemästä toisiaan. Seuraavassa kappaleessa tarkastellaan esimerkkejä ajansäästön, ongelmien ehkäisemiseksi ja optimoinnin parantamiseksi colliderin käytön kanssa.

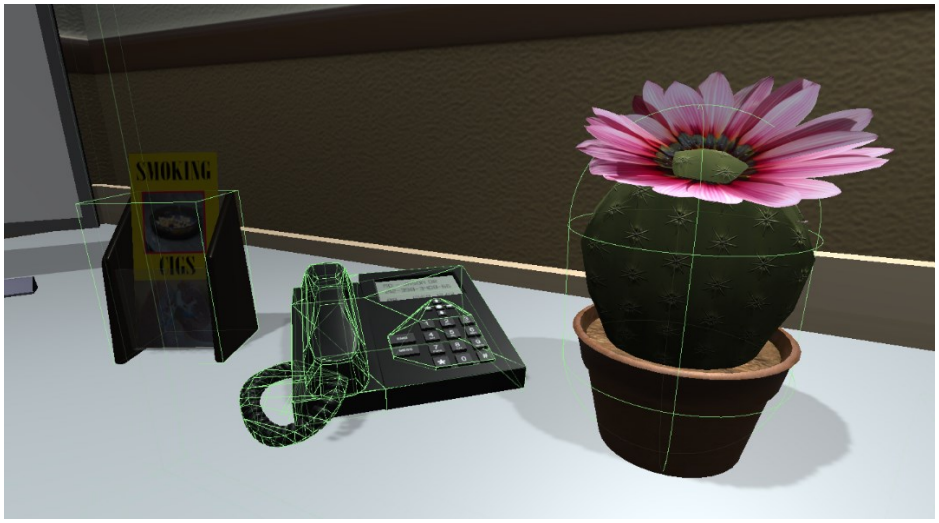
Yksinkertaisissa muodoissa, kuten pahvilaatikko tai pöytä, joka koostuu yksinkertaisista nelikulmion kaltaisista muodoista, voidaan käyttää colliderin luomiseen elementin omia muotoja. Monimutkaisia muotoja sisältävien elementtien kanssa on suuri vaikutus, käytetäänkö elementin omia muotoja vai rakennetaanko elementille manuaalisesti collider. Unity tarjoaa useita erilaisia valmiita collider vaihtoehtoja (Kuva 14), joita voi käyttää suoraan tai niiden avulla voidaan rakentaa elementille toimiva ratkaisu.

Kuva 14. Unityssa saatavilla olevia collider pintoja.



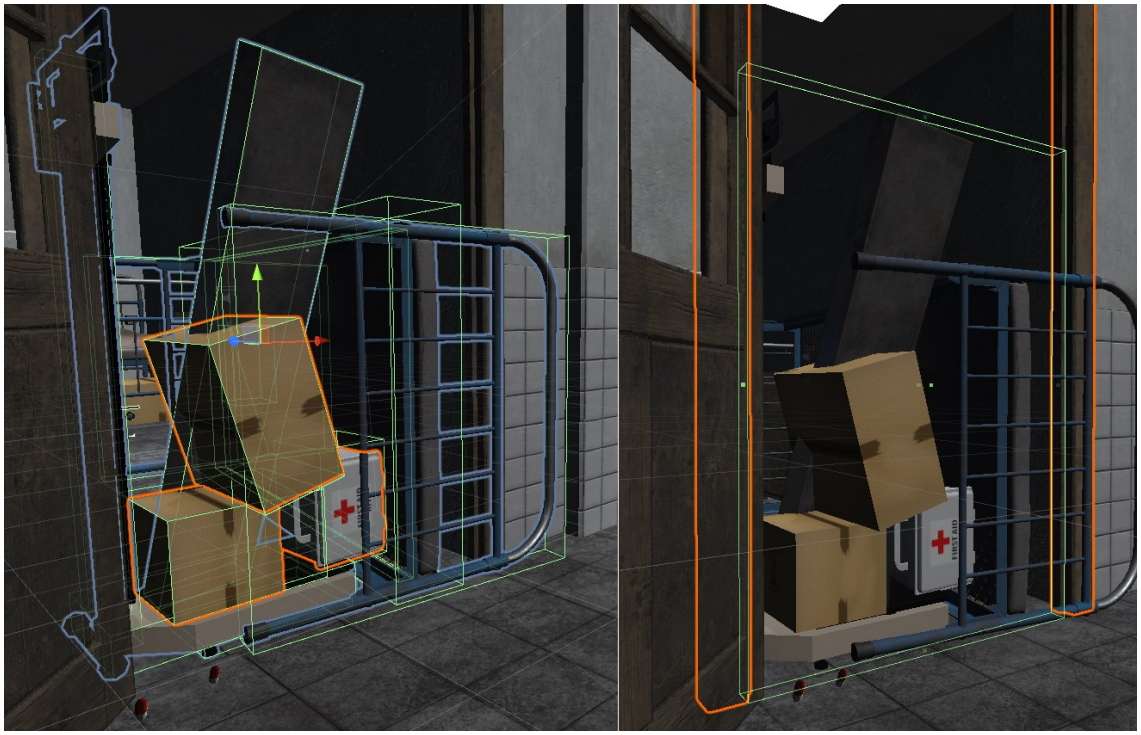
Yleisemmin käytettyjä ovat Box Collider, Mesh Collider ja Capsule Collider (Kuva 15). Kuvan Box Collider vasemmalla, käyttää kuusi pintaa. Mesh Collider keskellä, käyttää 542 pintaa, koska elementin muodot eivät sovellu käytettäväksi rakenteen muotoilua käyttävää Mesh Collider rakennetta. Oikeassa reunassa Capsule Collider, joka käyttää 12 pintaa kapselin muodon luomiseen, joka oli vähäpintaisin vaihtoehto pyöreille muodoille.

Kuva 15. Unity-sovelluksen Box Collider, Mesh Collider ja Capsule Collider.



Esimerkiksi elementit, jotka on luotu ainoastaan estääkseen hahmon eteneminen. Tällöin kannattaa luoda yksi näkymätön collider eli kosketuspinta, jolla estetään pelaajan liikkuminen, kuten (Kuva 16) oikealla puolella. Pelikentän pelattavat alueet kannattaa rakentaa mahdollisimman yksinkertaisista muodoista, jolloin pelinkehittäjällä on parempi hallinta siitä, miten pelissä voidaan toimia. Tällä ehkäistään esimerkiksi pelihahmon juuttumista ja niin sanottujen graafisien bugien hyödyntämistä edukseen. Tämä on myös optimoinnin kannalta parempi toimintatapa.

Kuva 16. Vasemmalla puolella oviaukosta kulku on estetty, käyttäen yksittäisien objektien collidereita ja oikealla puolella käytetty yksittäistä collideria, pelaajan liikkumisen estämiseksi. Collider näkyy kuvassa vihreinä ääriviivoina.

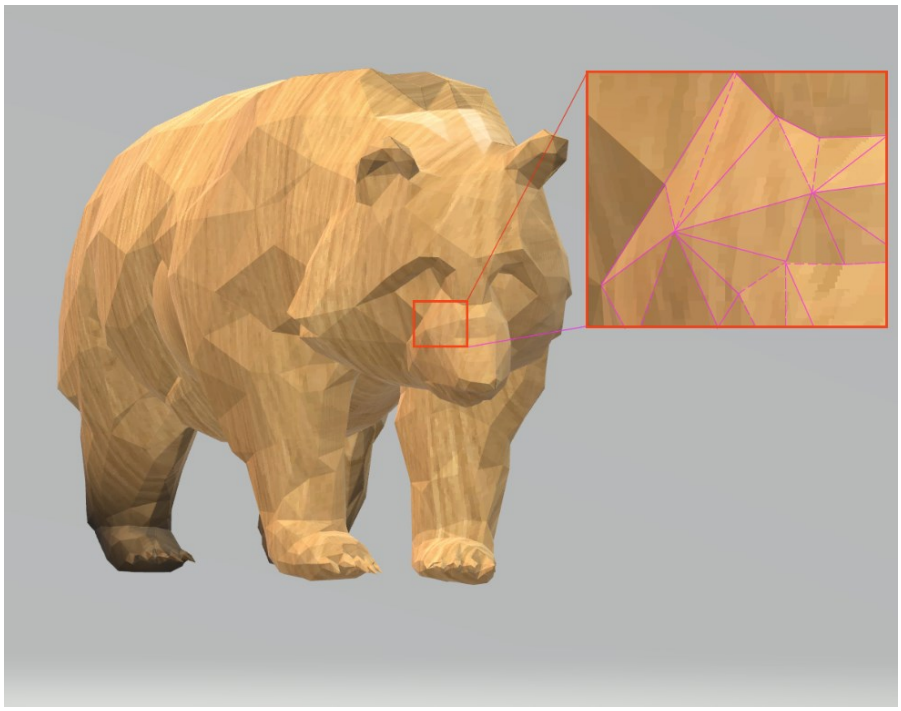


5.3 Polygonien vaikutus suorituskykyyn

Pelikehityksessä tulee ottaa huomioon elementeissä olevat polygonien määrät ja kuinka paljon polygoneja on ruudulla yhtä aikaa näkyvillä. Termi polygoni tarkoittaa monikulmiota,

mutta tietokonegrafiikassa ja yleisesti pelinkehityksen yhteydessä puhuttaessa sillä tarkoitetaan kolmesta suorasta viivasta koostuvaa kolmiota monikulmion pinnalla (Kuva 17). Erittäin useasti muodot näytetään nelikulmaisina muotoina ohjelmistoissa, jolloin yksi nelikulmainen muoto sisältää laskennallisesti kaksi polygonia. Jokainen kulma elementissä muodostaa lisää polygoneja, joten mitä yksityiskohtaisemmin 3D-malli on muotoiltu, sitä enemmän polygoneja se sisältää.

Kuva 17. Esimerkki karhusta ja sen pinnalle muodostuvista kolmioista, joista käytetään pelinkehityksessä termiä polygoni.

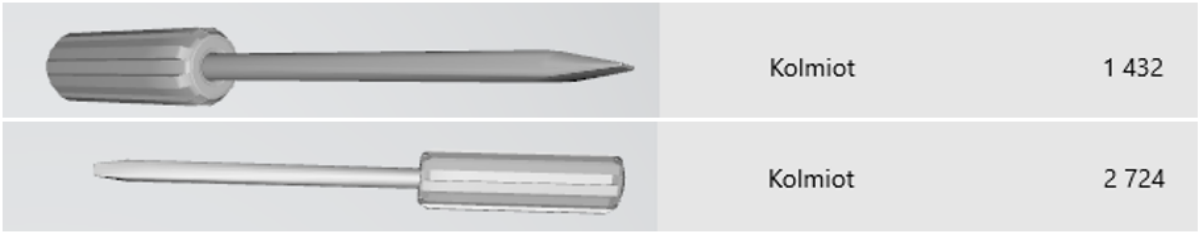


Yleistä suositusta polygonien määrälle on vaikea määritellä elementti kohtaisesti, mutta mitä vähemmän polygoneja sen parempi on pelin suorituskyky. Suorituskykyyn vaikuttaa eniten se, kuinka paljon renderöitävää on ruudulla kerrallaan näkyvissä. Myös reaaliaikaisen valaistuksen ja heijastuksien vaikutus suorituskykyyn korostuu, mitä enemmän polygoneja on ruudulla yhtäaikaisesti.

Etenkin valmiita 3D-malleja käytettäessä on tärkeää huomioida ja tarkistaa onko ne optimoitu pelikäyttöä varten. Yksinkertainenkin elementti voi sisältää tuhottoman paljon turhia polygoneja huonosti optimoituna. Esimerkiksi soveltavaa projektia varten haettu

valmis malli ruuvimeisselistä (Kuva 18), joka sisälsi 2724 polygonia. Säilyttäen mallin täysin identtisenä ja kevyesti optimoimalla 3ds Max -ohjelmistolla polygonien määrä vähentyi lähes puolella 1432 polygoniin.

Kuva 18. Yläpuolella optimoitu ja alapuolella optimoimaton 3D-malli, sekä polygonien eli kolmioiden määrät. Tiedot haettu käyttäen Paint 3D -ohjelmistoa.



Unity-ohjelmisto mahdollistaa myös polygonien määrän muuttamisen piirtoetäisyyden mukaan. Tällöin mitä lähempänä kohdetta ollaan pelissä, sitä tarkempi 3D-malli piirretään kyseiseen sijaintiin. Tämä mahdollistaa rakennuksien yksityiskohtaiset muotoilut ilman, että suorituskyky kärsii, jos esimerkiksi rakennuksia katsotaan kauempaa sijainnista, josta on mahdollista nähdä kymmeniä rakennuksia samanaikaisesti. 3D-mallista luodaan useampi malli vähentäen samalla polygonien määrää radikaalisti, jonka jälkeen Unityssa luodaan elementille LOD Group eli yksityiskohtien tasojen ryhmä. (Docs Unity3D, 2022d)

5.4 Object pooling

Object pooling eli objectien varastoiminen välimuistiin. Tämä tarkoittaa sitä, että tietyt objektit, jotka syntyvät ja tuhoutuvat nopeasti, pidetään valmiina odottamassa. Esimerkiksi viholliset ja ammukset ovat objekteja, joita ilmestyy ja tuhoutuu paljon. Nämä kannattaa tallentaa talteen uusiokäyttöä varten, jolloin niiden uudelleen luominen ei kuluta prosessoritehoja. Esimerkiksi ammuksia kannattaa varastoida sen verran kuin lippaassa on ammuksia, ja latauksen aikana ne siirtyvät takaisin odotustilaan uudelleen käyttöä varten. Tämä toteutetaan scriptillä, jossa luodaan klooneja halutusta objektista. Ihannetilanne on pitää niitä objektivarastossa vain silloin, kun niille voi olla tarve. Jos alueella on tiettyjä vihollisia, ne otetaan talteen varastoon ja kun alueelta poistutaan, nämä poistetaan

varastosta esimerkiksi latausruudun tai muun siirtymän aikana. Varastoon lisäämisen ja tuhoamisen voi tarvittaessa porrastaa. (Unity Learn, 2022)

6 Yhteenveto

Opinnäytetyön päätavoitteena oli saada syvempää oppimista projektinhallinnasta ja sen menetelmistä ohjelmisto- ja pelikehityksen osalta. Näiden lisäksi syventää osaamista Unity- ja 3D-ympäristöissä. Useimmiten ryhmätyöskentely ja projektihallinta tuottivat haasteita erilaisissa kouluprojekteissa. Ne kuluttivat myös runsaasti resursseja pelituotanto-moduulissa, jossa toteutettiin opinnäytetyöhön liitetty soveltava projekti. Olen toiminut useasti toteutetuissa projekteissa vetäjänä, koska tavoitteeni oli oppia paljon ja saada projektit kiitettävästi maaliin. Soveltavan projektin peli Teddy Bear Adventure toteutettiin Unity-pelimoottorilla neljän hengen ryhmässä kahdeksan viikon ajanjaksolla, joka sisälsi myös Unityn opiskelun. Tässä kokonaisuudessa ilmenneet ongelmat ja haasteet toimivat ajatuksena opinnäytetyöhön ja sen sisältöön, koska paljoin asioita jäi selvittämättä perusteellisesti tiukan aikataulun takia.

Opinnäytetyön toteuttaminen aiheesta osoittautui erittäin haastavaksi, koska suoranaista asiaa käsitteleviä lähteitä ei löytynyt useista yrityksistä huolimatta. Useat kirjat, joiden nimestä ja johdannosta voisi päätellä sisältävän oleellista tietoa, keskittyivät kuitenkin Unityn mekaniikkoihin ja pelin toteutukseen tai vaihtoehtoisesti, miten projektin budjetointi toteutetaan tehokkaasti. Lähteiden sivuavasta sisällöstä huolimatta niistä löytyi kuitenkin käyttökelpoista informaatiota, joista oli hyötyä opinnäytetyön luomisessa.

Työn aikana opin Unity-pelimoottorin kanssa työskentelyn lisäksi erittäin paljon yleisesti projektityöskentelystä ja menetelmistä, jotka mahdollistavat tehokkaan työskentelyn ja aikataulun hallinnan. Opin erittäin laajasti projekteihin liittyvistä tekijöistä, jotka jäivät kuitenkin opinnäytetyön ulkopuolelle, näitä olivat muun muassa budjetointi, ryhmän toiminnan ylläpitäminen ja työskentelyilmapiirin säilyttäminen. Käsittelin erittäin paljon erilaista materiaalia etsiessäni soveltuvia lähteitä opinnäytetyöhön, joita löytyi loppujen lopuksi erittäin vähän. Näiden avulla paransin kuitenkin omaa käsitystäni toimintamenetelmistä.

Omat tavoitteet opinnäytetyön työskentelyn osalta täytin mielestäni onnistuneesti, löysin vastaukset kysymyksiin, jotka olivat epäselviä aikaisemmin, mutta itse opinnäytetyöhön en saanut tuotua informaatiota niin laajasti ja kattavasti kuin olisin toivonut. Tosin tämä olisi vaatinut kokonaisen kirjan kirjoittamista, jotta olisi ollut mahdollista avata kaikki oleellinen syvällisesti. Tavoite opinnäytetyön sisällön osalta täyttyi kuitenkin hyvin, koska siinä tuodaan esille tarvittava tieto, mitä tarvitaan pienimuotoisessa pelikehityksessä antaen vahvan pohjan etenkin Unity-pelimoottorin kanssa työskentelemiselle. Halusin jättää opinnäytetyön ulkopuolelle pelisuunnittelun kokonaisuudessaan ja suurimman osan Unity-pelimoottorin ominaisuuksista, koska aiheesta löytyi satoja opinnäytetöitä, verkkokursseja ja muuta kirjallisuutta, keskittyen vähemmän käsiteltyihin, mutta tärkeisiin tekijöihin projektin kokonaisvaltaisen onnistumisen kannalta.

Opinnäytetyössä käsiteltiin erittäin niukasti itse soveltavaa projektia, koska itsessään se on käytännössä pelkästään pelisuunnittelua. Tarkoituksena oli tuoda soveltavaa projektia enemmän esille, mutta tiedonhankinnan haasteiden, opinnäytetyön pituuden ja sen kokonaisuutta sivuavan sisällön takia jätin sen lopulta pois opinnäytetyöstä.

Soveltava peliprojekti toteutettiin itselleni ennalta tuntemattomien henkilöiden kanssa. Peliprojekti onnistui loppujen lopuksi hyvin, mutta haasteita esiintyi etenkin ryhmätyöskentelyn, kommunikoinnin ja aikataulun kanssa. Unity oli meille kaikille lähes tuntematon pelimoottori, mutta osaamista 3D-mallinnuksesta ja animointipuolelta löytyi jonkun verran. Pelin elementtejä luotiin erittäin paljon itse projektin alussa, johon kului erittäin paljon aikaa. Soveltavan projektin kautta opin erittäin laajasti Unitysta ja sain erittäin vahvat taidot erityylisien pelien luomiseen ja mekaniikkoihin. Soveltavan projektin päätavoitteena oli kuitenkin Unityn oppiminen, jonka myötä peli sisälsi erilaisia tekniikoita, jotka eivät olleet tarpeellisia pelin kannalta. Näiden tekniikoiden avulla pelistä olisi voinut luoda erittäin mielenkiintoisen ja hauskan kokonaisuuden. Näin olisi todennäköisesti tapahtunut annetun aikarajan puitteissa, jos tässä opinnäytetyössä käsitelty tieto olisi ollut saatavilla, kun aloitimme pelin rakentamisen.

Opinnäytetyö antaa hyvän lähtökohdan yleisesti projektienhallintaan ja Unitylla toteuttamiseen käsitellen useita tekijöitä, joiden tulisi olla itsestään selviä kaikille

osapuolille. Käytännössä punaista lankaa ei anneta tässäkään opinnäytetyössä suoraan, miten projekti täytyy toteuttaa, vaan jokaisen tulisi rakentaa oma polkunsa, koska jokainen projekti on hieman erilainen. Standardeja miten toimia eri tilanteissa on kuiteinkin saatavilla monipuolisesti. Kuten tässä opinnäytetyössä käsittelin hieman laajemmin menetelmiä ja toin esimerkkejä toteuttaa asioita, joiden pohjalta on hyvä rakentaa sopiva toimintamalli erilaisissa tilanteissa.

Lähteet

Brusca, V. (2022). *Advanced Unity Game Development. Build Professional Games with Unity, C#, and Visual Studio* (1 p.). Apress.

Docs Unity3D. (2018). *Unity Documentation.* https://docs.unity3d.com/2018.2/Documentation/ScriptReference/EventSystems.EventSystem.html

Docs Unity3D. (25.3.2022a). *Unity Documentation.* https://docs.unity3d.com/Manual/Tags.html

Docs Unity3D. (26.3.2022b). *Unity Documentation.* https://docs.unity3d.com/2022.2/Documentation/Manual/Prefabs.html

Docs Unity3D. (16.4.2022c). *Unity Documentation.* https://docs.unity3d.com/Manual/CreatingScenes.html

Docs Unity3D. (15.5.2022d). *Unity Documentation.* https://docs.unity3d.com/Manual/LevelOfDetail.html

GitHub. (n.d.). *GitHub*. Haettu 11.6.2022 osoitteesta https://git-lfs.github.com

Karaś, C. (29.6.2020). *Naming 101 programmers guide on how to name things.* https://www.elpassion.com/blog/naming-101-programmers-guide-on-how-to-name-things

Lehtimäki, T. (2006). *Ohjelmistoprojektit käytännössä* (1 p.). Readme.fi.

Perforce. (2022). *Perforce.* Haettu 25.5.2022 osoitteesta https://www.perforce.com/blog/vcs/version-control-for-binary-files

Techcolleague. (n.d.). *Techcolleague.* Haettu 15.5.2022 osoitteesta https://techcolleague.com/vram-vs-ram

Tomb Raider (18.6.2022). Wikipedia-artikkeli.
https://en.wikipedia.org/w/index.php?title=Tomb_Raider&oldid=1093715209

Tuliper, A. (8.2014). *Unity: Developing Your First Game with Unity and C#. Docs Microsoft:* https://docs.microsoft.com/en-us/archive/msdn-magazine/2014/august/unity-developing-your-first-game-with-unity-and-csharp

Unity. (24.2.2021). *Unity Documentation.* https://docs.unity3d.com/2020.1/Documentation/Manual/GettingStartedUnityHub.html

Unity. (2022). *Unity Plastic SCM.* https://unity.com/products/plastic-scm

Unity (game engine) (16.7.2022). Wikipedia-artikkeli.
https://en.wikipedia.org/w/index.php?title=Unity_(game_engine)&oldid=1098555325

Unity Learn. (18.5.2022). *Unity Learn.* https://learn.unity.com/tutorial/introduction-to-object-pooling?uv=2019.4