Taru Vikström

Kehitysohjeistus videopelituotan-noissa

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Tieto- ja viestintätekniikan tutkinto-ohjelma
Insinöörityö
3.9.2024

Tiivistelmä

Tekijä:	Taru Vikström
Otsikko:	Kehitysohjeistus videopelituotannoissa
Sivumäärä:	46 sivua + 2 liitettä
Aika:	3.9.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Pelisovellukset
Ohjaajat:	Lehtori Antti Laiho

Insinöörityön tavoitteena oli kertoa videopelituotannoissa toistuvista monenlaisista pelikehitykseen liittyvistä aihealueista, joista kehitystiimien oli järkevää tehdä ohjeistusdokumentointia ja perustella sen tekemisen hyötyjä. Insinöörityö pohjautuu ohjeistusdokumenttiin, joka tuotettiin Svenska Ylelle, BUU-klubben-peliprojektille.

Työssä esitellään monia pelialan standardiohjeistuksia, kuten ohjelmointi-, dokumentointi- ja pelimoottorikäytänteitä. Tämän lisäksi esiteltiin laaja kirjo erilaisia työkaluja, joita voidaan hyödyntää videopelituotannoissa. Peliprojekteissa on tärkeää noudattaa sovittuja työskentelytapoja, jolloin voidaan vaikuttaa pelin laatuun, projektin eheyteen sekä ammattimaiseen ja sujuvaan tiimityöskentelyyn.

Insinöörityön tuloksena tuotettiin kehitysohjeistusdokumenttipohja, jossa on valmiina tässä työssä esiteltyjä aihealueita. Pelikehitystiimit voivat hyödyntää dokumenttipohjaa esimerkiksi, kun uutta peliprojektia aloitetaan ja on hyödyllistä sopia yhteisistä projektikäytänteistä ja -toimintaohjeista.

Avainsanat: videopelikehitys, kehitysohjeistus, kehityskäytänteet

---

Abstract

Author:	Taru Vikström
Title:	Development Instructions in Video Game Productions
Number of Pages:	46 pages + 2 appendices
Date:	3 September 2024
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Game Applications
Supervisors:	Antti Laiho, Senior Lecturer

The objective of this study was to identify recurring topics essential for video game development teams and to demonstrate the value of documenting these topics into guidelines. The study is based on a guideline document that was produced for Svenska Yle, for the BUU-klubben game project.

This study reviews various industry-standard guidelines related to programming, documentation, and game engines. Additionally, it presents a broad range of tools that can be utilized in video game production. In game development projects, adhering to agreed-upon practices is crucial, as it directly impacts the quality of the game, and the integrity of the project, and fosters professional and efficient teamwork.

As a result of this study, a template for a development guidelines document was created, incorporating the key topics discussed in this work. Game development teams can utilize this template when initiating new game projects, facilitating the establishment of common practices and procedures relevant to the project.

Keywords: video game development, development instructions, development practices

Sisällys

Määritelmät ja termistö

Asset: Peliprojektissa käytettävä resurssi.

Sprite: 2D-grafiikkaobjekti, jota käytetään erityisesti videopeleissä.

Inspector: Unity-editorin ikkuna, johon tulee tietoja ja asetuksia valitusta peliobjektista tai assetista.

Mesh: Peliobjektin verkko, joka koostuu kolmioista 3D-avaruudessa.

Prefab: Peliobjektityyppi, joka on tallennettu ja sitä voidaan monistaa.

Repository: Tiedostovarasto, jonka sisältöä ylläpidetään joko paikallisesti tai palvelimessa.

Branch: Eriytetty haara tiedostovarastosta, johon voi tehdä muutoksia.

Commit: Talletusoperaatio, joka lähettää tiedostomuutokset tiedostovarastoon.

Pull&Push: Muutoksien hakeminen ja lisääminen tiedostovarastoon.

PR: Pull Request eli pyyntö muutoksien liittämisestä tiedostovarastoon.

Merge: Muutoksien yhdistäminen tiedostovaraston haaraumasta toiseen.

Riggaus: 3D-mallin valmistelu animointia varten.

1 Johdanto

Tämä insinöörityö käsittelee kehitysohjeistuksia pelituotannoissa. Työ sai inspiraationsa Svenska Ylen tuottamalle BUU-klubben-sovellukselle luodusta kehitysohjeistudokumentista (liite 1). Kehitysohjeistusdokumentaation luomisen toimeksiantaja on Svenska Yle, jossa insinöörityön tekijä työskenteli noin vuoden määräaikaisena sovelluskehittäjänä, vuosina 2022–2023.

Yleisradio Oy eli lyhennettynä Yle, on Suomen valtion omistama valtakunnallinen mediayhtiö. Ylen omilla nettisivuilla yrityksen toimenkuvaa kuvaillaan seuraavasti: ”Ylen julkisen palvelun tehtävään kuuluu tuottaa, luoda, kehittää ja säilyttää kotimaista kulttuuria, taidetta ja virikkeellistä viihdettä.”. (1) Yle kehittää sisältöä usealla eri kielellä suomen lisäksi ja erityisesti ruotsinkielistä sisältöä tuotetaan paljon. Sen takia ruotsinkielisen sisällön tuottaminen on eriytetty omaksi yksikökseen, Svenska Yleksi. (1)

Svenska Ylen tuottaman BUU-klubben-lastenohjelman pohjalta on kehitetty mobiilisovellusta, jossa ohjelmassa tutut hahmot seikkailevat (2). Sovelluksen kehitys aloitettiin vuonna 2014 Unity-pelimoottorilla, ja siitä lähtien sen sisältöä on kasvatettu sekä uudistettu useaan otteeseen. Kehitystyötä tehtiin yhteistyössä Ylen Pikku Kakkonen -sovelluksen kehittäjien kanssa. Vuosien saatossa BUU-klubben-sovellukseen on tuottanut sisältöä laaja kirjo eri kehittäjiä. Svenska Ylen omien sovelluskehittäjien lisäksi sovellus sisältää opiskelijoiden toteuttamia minipelejä sekä ostopalveluna tuotettuja sisältöjä. (3.)

Sovelluksen laajuudesta ja pitkästä kehityskaaresta huolimatta sovelluksen kehitysohjeistukset sekä dokumentaatio olivat kuitenkin jääneet tekemättä. Pelikehittäjän näkökulmasta oli haastavaa ja aikaa vievää omaksua vuosien aikana kertyneen massiivisen projektin sisältö ja sen toiminnallisuudet dokumentaation puutteen vuoksi. Insinöörityön tekijä tuotti yhteistyössä kehitystiimin kanssa kehitysohjeistusdokumentaation, joka muodosti tämän insinöörityöprojektin. BUU-klubben-projektille tehty kehitysohjeistusdokumentti on insinöörityön liitteenä (liite 1).

Dokumentaatio ja kehitysohjeistus pelituotannoissa on hyvin olennaista ja tuo paljon hyötyjä kehitystiimille, mutta se luo myös omat haasteensa. Projektien dokumentaation luominen ja ylläpitäminen voi olla kehittäjille hyvin aikaa vievää ja pitkäveteistä. Se on kuitenkin välttämätön ja kriittinen osa kehitysprosessia, jotta kaikilla projektin osapuolilla on ajantasaista tietoa projektin tilasta, sisällöstä ja ohjeistuksista peliä kehitettäessä. (4; 5.)

Kehitystiimin kesken sovittiin yhteisistä käytänteistä ja keskusteltiin kehitysohjeistuksista sekä dokumentoitiin ne. Tässä insinöörityössä esitellään yleisimpien pelituotantoon liittyvien käytäntöjen ja ohjeistuksien lisäksi myös BUU-klubben-sovelluksen projektikohtaisia sisältöjä. Ylelle tehdyssä dokumentaatiossa (liite 1) on annettu yksityiskohtaisia ohjeistuksia sovelluksen sisäisistä teknologioista, jotka ovat tärkeitä tulevaisuudessa projektin parissa työskenteleville. Insinöörityön tuloksena luotiin lisäksi toinen dokumentti (liite 2), joka toimii mallipohjana muiden videopeliprojektien ohjeistuksien luomiseen.

2 Ohjeistus videopelituotannoissa

Tässä luvussa esitellään pelikehitysalan yleisimpiä pelikehittäjien keskuudessa hyväksi todettuja käytäntöjä ja niiden ohjeistuksia. Tarkoituksena on ammentaa ohjeistuksia akateemisesta kirjallisuudesta sekä alan yleisimmistä ohjesäännöistä. Monet aihealueet, kuten tekniset toteutustavat ja käytettävät työkalut, ovat tiimin sisällä sovittavia, eikä niihin ole vain yhtä oikeaa tapaa, jonka mukaan pitäisi menetellä. Kyseiset aihealueet on kuitenkin lisätty omiksi luvuikseen, koska ne toistuvat jokaisessa peliprojektissa ja niistä täytyy olla jokin yhteisesti sovittu linjaus sekä kirjallinen ohjeistus kehitystiimille.

2.1 Videopelituotantojen kieli- ja nimeämiskäytänteet

Videopelituotannoissa on tärkeää pitää mielessä hyvät kieli- ja nimeämiskäytänteet. Kun sovittuja ja käyttötarkoitukseen soveltuvia käytänteitä noudatetaan, pysyy projekti kokonaisuudessaan selkeänä, johdonmukaisena ja helposti ylläpidettävänä. (6.)

Englannin kieltä pidetään standardikielenä pelikehitysalalla ja sen käyttämistä suositellaan useista syistä. Englanti on teknologiateollisuudessa hallitseva kieli, joten sitä käytettäessä yhteistyö ja projektien ymmärtäminen helpottuu, sillä yhdessä projektissa voi olla kehittäjiä useista eri maista. Kun dokumentaatio sekä koodien kommentoinnit on tehty englanniksi, myös uusien tiimijäsenten perehdyttäminen helpottuu. Englannin kielellä tehty dokumentaatio antaa projekteille läpinäkyvyyttä ja mahdollistaa eri sidosryhmien ymmärryksen projektin sisällöstä ja toiminnallisuuksista. Kun projekti on tehty kielellä, jota kaikki ymmärtävät, väärinkäsitysten ja virheiden määrä vähenee. (7.)

Ohjelmointiin liittyy monia hyviä käytänteitä, joita on hyvä noudattaa. Johdonmukainen hyvien käytänteiden noudattaminen edesauttaa koodin luettavuutta, virheenjäljitystä (eng. debugging) ja muokattavuutta. Koodissa esiintyvät sisennykset, rivivälit ja suljemerkkien sijainnit ovat kuitenkin monesti

mielipidekysymyksiä, mutta tärkeintä on, että tiimin sisällä päästään yhteisymmärrykseen projektikohtaisista sovituista koodikäytänteistä. (8, luku 28.)

Ohjelmoidessa on suositeltavaa käyttää kuvailevia nimiä muuttujille, funktioille sekä luokille, jotta koodista tulee mahdollisimman itsestään selvää. Käytettävien nimien tulee olla helposti äännettäviä ja muuttujien nimet eivät saa olla arvoituksellisia lyhenteitä. Hyvin nimetyt elementit vähentävät kommentoinnin tarvetta koodissa ja helpottavat luettavuutta. Koodin kommentointi on kuitenkin edelleen tärkeää, erityisesti monimutkaisissa osissa koodia. (9; 10.)

Nimeämiskäytänteisiin liittyy olennaisesti myös koodin muuttujien sekä pelimoottorin peliobjektien nimeämistavat, joissa käytetään case-tyylisiä nimeämistapoja. Jokaiselle tyylille löytyy erilaisia hyviä käyttötapoja, joiden tehtävänä on parantaa luettavuutta ja johdonmukaisuutta sekä helpottaa koodin ylläpitämistä. (10.) Seuraavaksi esitellään yleisimpiä case-nimeämistyylejä:

1. snake_case

   Snake_case-tyylissä kirjoitetaan kaikki sanat pienellä ja erotellaan alaviivalla. Tätä tyyliä käytetään useissa ohjelmointikielissä muuttujille.

   Käyttöesimerkki: “player_score”, “enemy_max_health”.

2. camelCase

   CamelCase-tyylissä sanat kirjoitetaan yhteen. Ensimmäisen sanan alkukirjain kirjoitetaan pienellä ja seuraavat alkavat aina isolla alkukirjaimella. Tästä tyylistä käytetään myös nimeä lowerCamelCase. Tätä tyyliä käytetään useissa ohjelmointikielissä muuttujille ja joissain tapauksissa myös funktioille.

   Käyttöesimerkki: ”playerScore”, ”enemyMaxHealth”.

3. PascalCase

   PascalCase-tyylissä sanat kirjoitetaan yhteen ja kaikki sanat alkavat isolla alkukirjaimella. Tyylistä voi nähdä käytettävän myös nimeä Upper-CamelCase. Tätä nimeämistyyliä käytetään yleisimmin funktioiden ja luokkien nimeämisessä, rajapinnoissa (eng. interface) ja enumeroinnissa.

   Käyttöesimerkki: ”MyEnemyClass”, ”PlayerStatsEnum”.

4. kebab-case

   Kebab-case-tyylissä sanat alkavat pienellä alkukirjaimella ja erotellaan väliviivalla. Tätä tyyliä käytetään harvoin peliohjelmoinnissa, mutta sitä näkee käytettävän mm. tunnuksissa (eng. identifier tai ID), URL-osoitteissa ja CSS-luokkanimissä.

   Käyttöesimerkki: ”player-score”, ”enemy-max-health”. (11.)

Nimeämisessä voidaan myös käyttää etu- ja jälkiliitteitä. Yleisimmin näitä käytetään kuvaamaan muuttujien tai pelin sisäisten resurssien ominaisuuksia tai tyyppiä tunnistamisen nopeuttamiseksi. Koodissa voi boolean-tyyppisen muuttujan nimen edessä olla b-kirjain (esim. ”bLevelCleared”) tai projektin sisäisen materiaalin nimen edessä voi olla ”mat”-lyhenne (esim. ”mat_LevelGrass”). (12.)

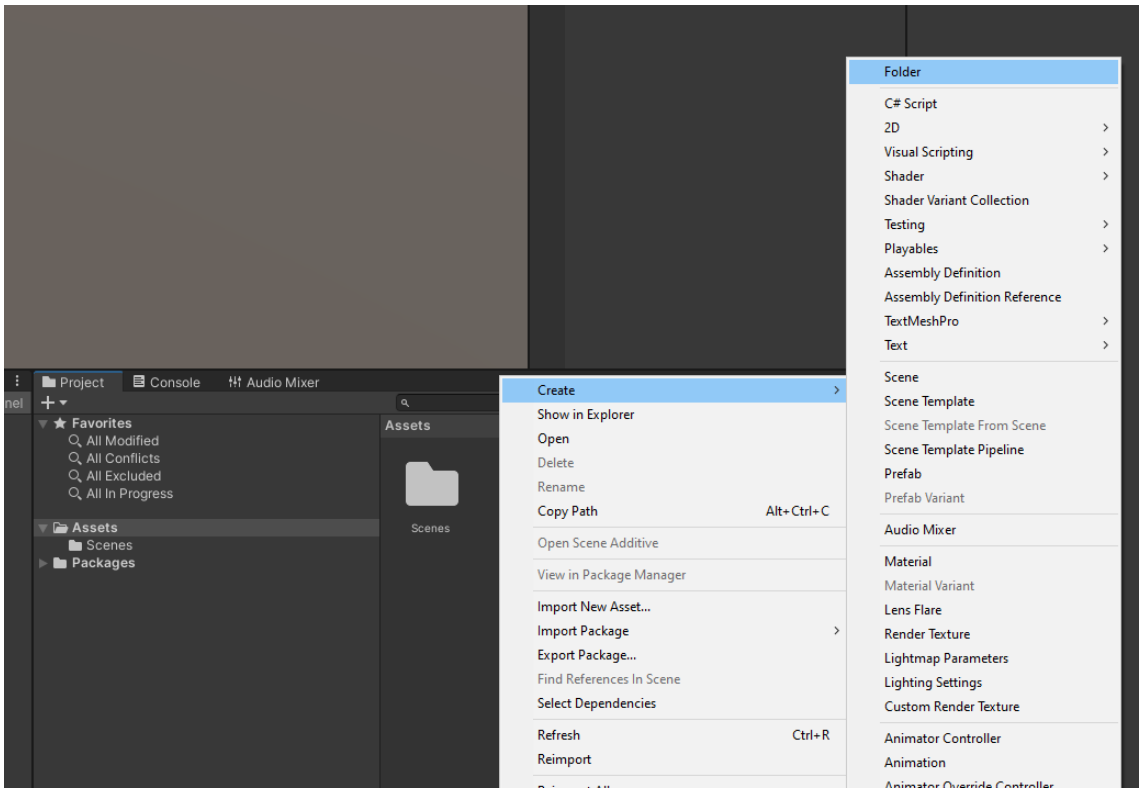
2.2 Pelimoottoriin liittyviä ohjeistuksia

Pelimoottoreita on monia, mutta useimmissa niistä on monia samoja ominaisuuksia. Tässä luvussa käydään läpi yleisimmät ominaisuudet ja niihin liittyvät käytännöt, jotka tuovat pelikehitysprosessiin selkeyttä ja auttavat pitämään projektin sisäisten resurssien hallinnan johdonmukaisena.

Peliprojektin kansio- ja peliobjektihierarkia

Peliprojektissa on monenlaisia resursseja, jotka on lajiteltava kansioihin, jotta kansiohierarkia pysyy järjestelmällisenä. Hyvä kansiojärjestys edesauttaa projektin ylläpitoa ja helpottaa pelin parissa työskentelevien työnkulkua, kun resurssit löytyvät helposti. Selkeä kansiojärjestys voi vaikuttaa myös tuotteliaisuuteen. (13.)

Kun Unity-pelimoottorissa luodaan uusi projekti, luodaan projektin juurikansioon myös Assets-kansio eli kansio projektin sisäisille resursseille. Assets-kansion sisälle generoituu myös automaattisesti heti Scenes-kansio projektin luomisen yhteydessä. Scenes-kansioon talletetaan kaikki projektin pelinäkymät. Assets-kansion sisälle voidaan luoda lisää kansioita tuleville projektiresursseille. Kuvassa 1 näkyy, kuinka painamalla hiiren oikealla näppäimellä Assets-ikkunan sisällä avautuu valikko, jota kautta voi luoda lisää kansioita.



Kuva 1. Kansioiden luominen Unityssä.

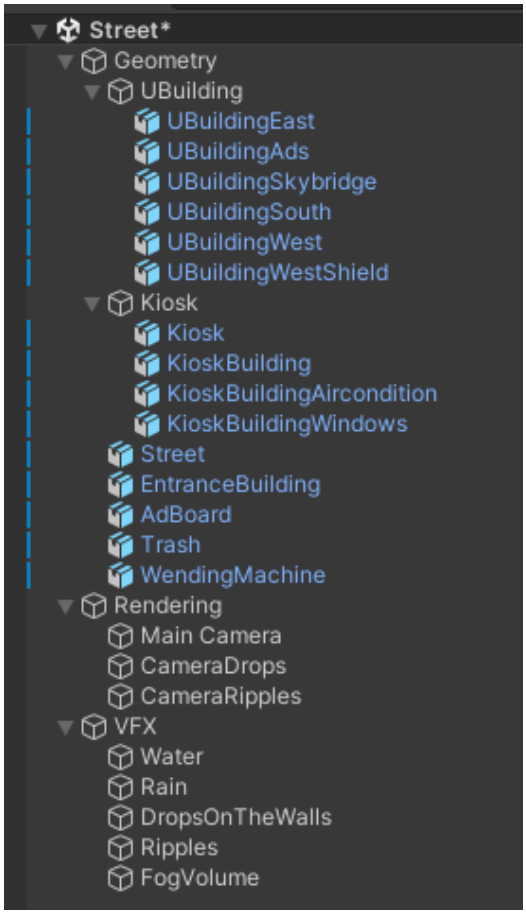
Peliprojekteissa esiintyy monenlaisia resursseja, joita on hyvä lajitella omiin kansioihinsa. Yleisimmin kansiot tehdään resurssikansion juureen, pelinäkymien lisäksi, ohjelmakoodille, grafiikalle, materiaaleille, äänille sekä prefabeille eli monistettaville peliobjekteille. (14; 15, luku 1.4.) Joissain tapauksissa projekti voi sisältää useampia toisistaan erillisiä kokonaisuuksia, jolloin voi olla järkevää tehdä resurssikansiot jokaiselle kokonaisuudelle erikseen. Esimerkiksi BUU-klubben-projektissa näin menetellään kaikkien minipelien resurssien osalta, eli jokaiselle minipelille tehdään pääkansio, jonka sisälle sijoitetaan kaikki kyseiseen minipeliin liittyvät resurssit omiin kansioihinsa. Jokaisessa projektissa ja tiimissä voidaan erikseen sopia tilanteeseen sopivista ja käytännöllisimmistä kansiorakenteista ja hierarkioista, jotka palvelevat projektin parissa työskentelevien tarpeita. (15.) Alla näkyy esimerkkirakenne peliprojektin sisäisestä kansiohierarkiasta ja mitä kansioihin yleensä tallennetaan.

```
Assets
+---Art
|   +---Materials
|   +---Models      # FBX and BLEND files
|   +---Textures    # PNG files
+---Audio
|   +---Music
|   \---Sound       # Samples and sound effects
+---Code
|   +---Scripts     # C# scripts
|   \---Shaders     # Shader files and shader graphs
+---Docs            # Wiki, concept art, marketing material
+---Level           # Anything related to game design in Unity
|   +---Prefabs
|   +---Scenes
|   \---UI
\---Resources       # Configuration files, localization text and other
user files.
```

Esimerkki hyvästä kansiohierarkiasta (16).

Peliprojekteissa hyviin käytäntöihin kuuluu myös peliobjektien johdonmukainen ja siisti järjestäminen pelinäkymien hierarkiassa. Peliobjektien hierarkia kannattaa pitää mahdollisimman tasaisena ja välttää syviä peliobjektipuita. Yleisimpiä ohjenuoria ovat, että yhdellä peliobjektilla ei olisi enempää kuin 8 lapsiobjektia ja ettei peliobjektin sukupuu menisi 4 astetta syvemmäksi. Näillä toimilla voidaan vähentää merkittävästi laskentarasitusta, kun peliobjektin kokoa tai sijaintia muutetaan pelin ajon aikana. (17.)

Peliobjektien järjestäminen toiminnallisuuksien mukaan tuo hierarkiaan selkeyttä. Esimerkiksi käyttöliittymä, staattiset tasot ja dynaamiset elementit kannattaa jakaa omiin peliobjektiryhmiinsä. Tyhjien peliobjektien käyttäminen on hyvä väline peliobjektien jaotteluun ja nopeuttaa objektien löytämistä hierarkiasta. Hyvien käytänteiden noudattaminen helpottaa pelin kehitystyötä ja parantaa pelin suorituskykyä. (17.) Kuvassa 2 on erään pelinäkymän peliobjektien hierarkia, jossa on selkeää objektiryhmittelyä.



Kuva 2. Esimerkki selkeästä peliobjektihierarkiasta pelinäkymässä.

Peliobjektien lajittelutasot ja tunnisteet

Unityssä on useita erilaisia tapoja ryhmitellä, lajitella ja asettaa tasoille objekteja. Ensimmäisessä lajitteluominaisuudessa peliobjekteille annetaan nimetty taso (eng. layers). Projektiin luodaan automaattisesti sarja yleisimpiä tasoja, jonka lisäksi kehittäjät voivat lisätä tasoja tarpeen mukaan. Tasoja voi

yksittäisessä projektissa olla 8–31. Tasoja käytetään peliobjektien törmäysten havaitsemiseen sekä renderöinti ja ryhmittelytarpeisiin. Tasoja kannattaa lisätä peliin maltillisesti ja miettiä tarkkaan niiden käyttötarkoitus sekä nimeäminen. Tasojen geneerinen nimeäminen on hyvä tapa, jolloin samoja tasoja voidaan käyttää useissa eri tilanteissa, eikä uusia tasoja tarvitse luoda joka kerta. (18.)

Peliobjekteille voidaan antaa nimikkeitä (eng. tag) ryhmittely- ja tunnistustarpeisiin. Peliobjekteilla ei ole nimikkeitä, ellei nimikettä erikseen määritellä. Peliobjektilla ei voi olla kuin yksi nimike kerrallaan. Yleisimpiä käyttötarkoituksia nimikkeille on objektien ryhmittely ja tarkastelu ohjelmakoodeissa. Esimerkkikoodissa 1 on yksinkertainen funktio, jossa tarkistetaan peliobjektin nimike. Yleisimpiä pelikehityksessä käytettäviä nimikkeitä ovat mm. ”Player”, ”Enemy” ja ”Ground”. (19.)

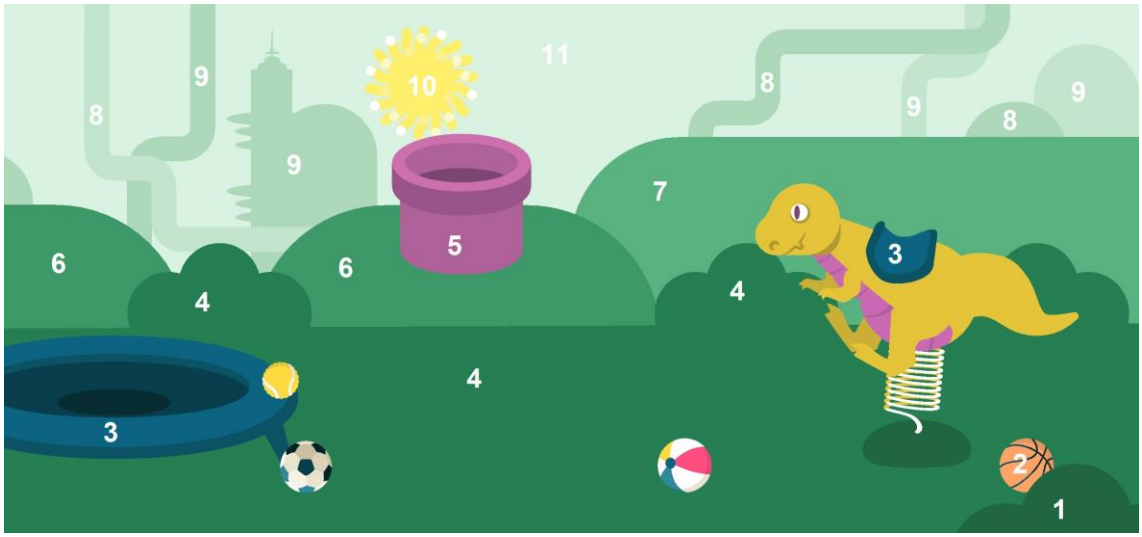
```
void OnCollisionEnter (Collision col)
{
          if (col.gameObject.CompareTag("Enemy"))
          {
              Debug.Log ("Törmättiin viholliseen.");
          }
}
```

Esimerkkikoodi 1. Peliobjektin nimikkeen tarkistus törmäyksen sattuessa.

Sprite-grafiikoille on Unityssä omat renderöintitasot ja -järjestykset. Pelimoottorissa grafiikkaa voi ryhmitellä eri lajittelutasoille (eng. sorting layers) ja sen lisäksi antaa tason sisällä järjestyksiä (eng. order in layer). Nämä helpottavat grafiikkaresurssien renderöintijärjestyksen hallitsemista pelinäkymissä. Lajittelutasoja voi luoda lisää, ja ne on hyvä nimetä käyttötarkoituksen mukaisesti ja selkeästi, mahdollisuuksien mukaan niin, että lajittelutasoja voidaan käyttää muissakin pelinäkymissä samaan tapaan. (20; 14, luku 5.)

Hyvänä esimerkkinä lajittelutasojen tärkeydestä toimii BUU-klubben-sovelluksen Leikkipinta-alueella toteutettu illuusio syvyydestä, jossa 2D-grafiikkaa on lajiteltu usealle eri lajittelutasolle ja tasojen puitteissa on tehty tasojen järjestämistä ja lajittelua. Kuvassa 3 näkyy numeroituna, kuinka monessa eri tasossa

grafiikkaa on. BUU-klubben-projektissa lajittelutasoja on nimetty mm. Far Back, Back, Middle Back, Middle, Middle Front, Front ja Close Front.



Kuva 3. BUU-klubben-sovelluksessa 2D-grafiikan renderöintitasoja numeroilla havainnollistettuna.

Kun kuvassa 3 esiintyviä grafiikkaelementtejä on lisätty pelinäkymään, on lajittelutasolla pitänyt tehdä myös järjestämistä. Kuvassa näkyvät tummanvihreät kumpareet, jotka on numeroitu 6 ja 7, ovat kaikki samalla lajittelutasolla ”Middle Back”, mutta jotta numerolla 7 merkitty kumpare piirtyy kumpareen numero 6 taakse, on sille täytynyt antaa pienempi indeksitaso järjestyksessä. Lisää kuvassa esiintyvästä Leikkipinta-kokonaisuudesta kerrotaan luvussa 3.3.

Projektiresurssien tuonti- ja pakkausasetukset

Resurssien tuominen (eng. import) projektiin on suoraviivaista, ja siihen on selkeät ohjeet Unityn verkkosivuilla. Resursseihin ja niiden pakkaamiseen projektissa liittyy kuitenkin useita käytäntöjä ja ohjenuoria resurssin tyypin mukaan. Kun asetukset määritellään oikein ja pitämällä mielessä mihin ja miten resurssia käytetään, parannetaan pelin suorituskykyä ja laatua. (28.) Peliprojektissa käytettäviä resursseja kutsutaan yleisemmin asseteiksi.

2D- ja tekstuurigrafiikka

Kun peliprojektiin tuodaan grafiikkaelementtejä, täytyy niiden asset-asetukset käydä läpi pelimoottorin Inspector-ikkunassa. Yksi yleisimmistä säännöistä 2D-grafiikassa on se, että kuvan sivujen kokojen tulee olla kahden potensseja. Unity pakkaa parhaiten kahden potenssissa olevia 2D-kuvia (29). Pakkaamattomat grafiikka-assetit ovat korkealaatuisimpia, mutta ne kasvattavat pelin kokoa, ja niiden lataaminen pelissä kuluttaa paljon laitteen muistia. Assetin voi pakata Inspectorin asetuksissa kolmella eri laadulla:

1. ”None” eli ei pakata lainkaan, jolloin grafiikka renderöidään täysin samalla laadulla kuin sen alkuperäisessä koossa.

2. ”Low quality” eli alhainen laatu, joka vähentää assetin kokoa huomattavasti, mutta saattaa tuottaa huomattavia vääristymiä grafiikkaan.

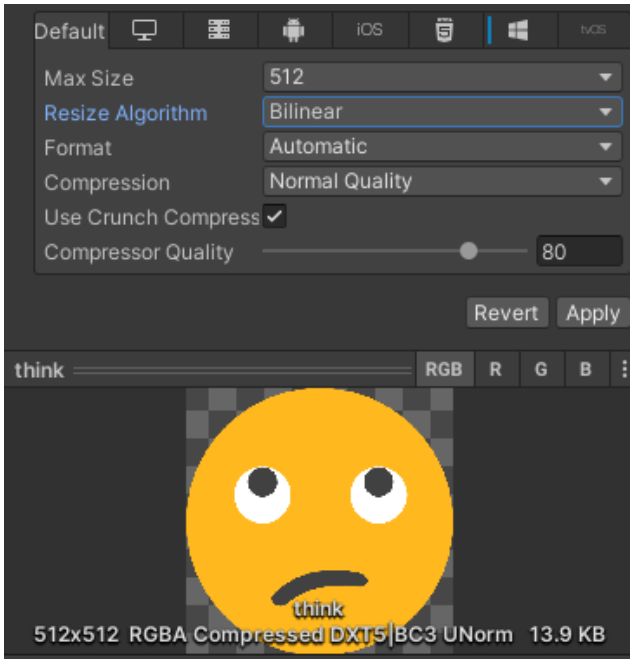
3. ”Normal quality” eli tavallinen laatu, joka on hyvä valinta, kun halutaan vaihtoehto, jossa assetin koko pienentyy, mutta laatu pysyy vielä hyvänä.

4. ”High quality” eli korkea laatu, jossa assetin koko ei pienene yhtä paljon kuin edellisissä pakkausvaihtoehdoissa, ja grafiikka on laadultaan erittäin hyvää. (30.)

Kuva-assetin lopullinen koko määräytyy pakkausasetuksien perusteella ja riippuu paljon siitä, mitä pakkausformaattia käytetään. Pakkausformaatteja on useita (DTX, ETC, PVRTC, ASTC ja BCn) ja sopivimman vaihtoehdon valitseminen vaatii aina tarkkaa tutustumista kehitettävän pelin kohdealustan ja pakkausformaatin yhteensopivuuteen. Valinta riippuu myös pakattavan kuva-assetin formaatista, koska kaikki pakkausformaatit eivät ole yhteensopivia. Kun peli tehdään usealle eri pelialustalle, voidaan Inspectorissa valita eri alustoille soveltuvat pakkausasetukset. (30.)

Kuvan suurimman mahdollisen resoluution määrittely suoraan pakkausasetuksissa on nopea ja hyödyllinen tapa pakata kuvaa lähemmäs todellista

kokotarvetta. Graafikot työstävät yleensä kuvia suuressa koossa, koska kuvan koon ja laadun pienentäminen käy helpommin, kuin toisinpäin. Kun kuvan koko on suurempi kuin asetettu maksimikoko, Unity voi käyttää pienentämiseen kahta tapaa, Mitchelliä tai Bilinearia. Näistä Mitchell on standardivalinta, mutta Bilinear taas toimii paremmin kuviin, joissa halutaan säilyttää mahdollisia yksityiskohtia. Crunch compression -asetusta kannattaa käyttää, kun pelin kokonaiskokoa halutaan pienentää niin paljon kuin mahdollista. Tämä pakkaustapa on kuitenkin ns. lossy-tyylinen, eli kuvan laatu kärsii tästä pakkaustavasta. Inspectorissa on liukusäädin, jonka avulla on helppo valita sopiva pakkaustaso. (31.) Kuvassa 4 näkyy Unityn 2D-assetin pakkausasetukset Inspectorissa.

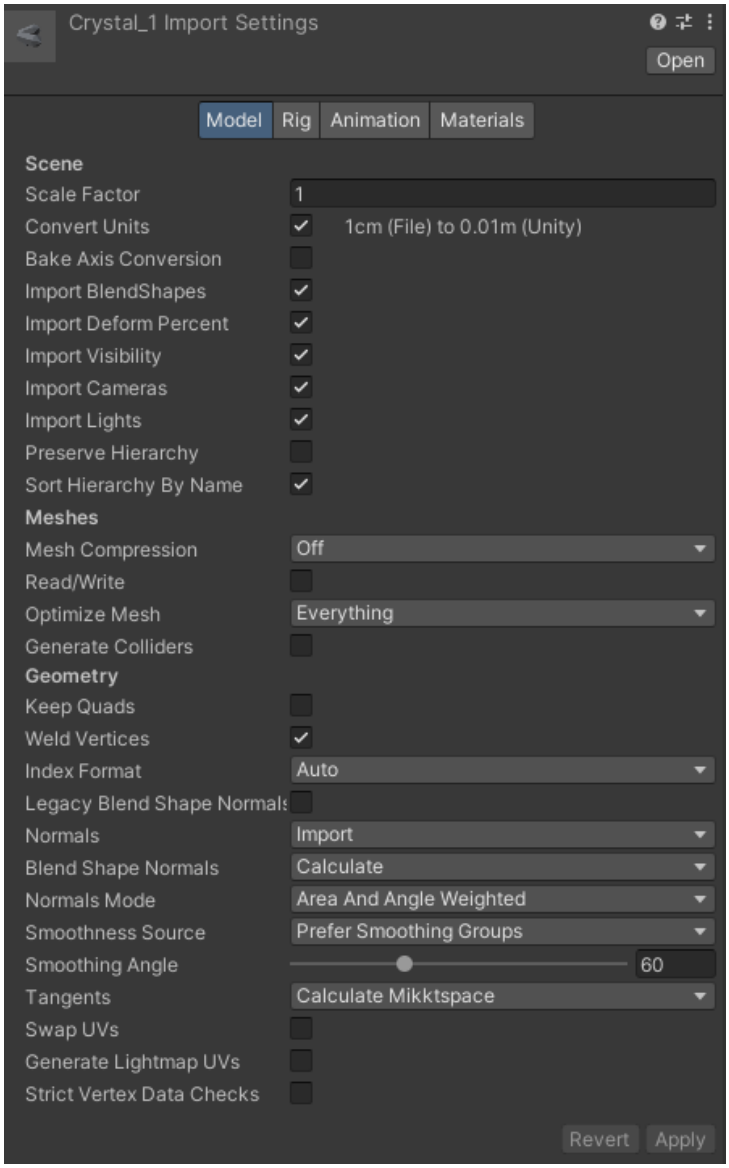


Kuva 4. 2D-assetin pakkausasetukset Unityn Inspector-ikkunassa.

Bitmappeja, eli pakkaamattomia rasteritiedostoja, voidaan pakata pelimoottorin sisällä vielä yhdellä tavalla, käyttämällä Sprite Atlaksia. Sprite Atlakseen pakataan useampi bitmap-formaatin assetti, jolloin Unity käsittelee sitä yhtenä isona tekstuuri-assettina. Yhteen Sprite Atlakseen pakataan vain saman pelinäkymän sisällä käytettäviä grafiikka-assetteja. Tämän pakkaustapa voi nopeuttaa latausaikoja huomattavasti. (32.)

3D-mallit

Unity tukee useita erilaisia 3D-formaatteja, joita voidaan tuoda projektiin. 3D-mallin mukana voi tulla monenlaista muutakin tietoa tiedostosta, esimerkiksi materiaaleja, tekstuureja ja meshejä. Meshit ovat kolmioista koostuvia 3D-peliobjektiverkkoja. (35.) Unityn Inspector-ikkunassa pystyy vaikuttamaan mallin tuontiasetuksiin, joihin kuuluvat myös mallin valot, geometria, ym. (36.) Kuvassa 5 näkyy 3D-assetin asetuksia Unityn Inspector-ikkunassa.



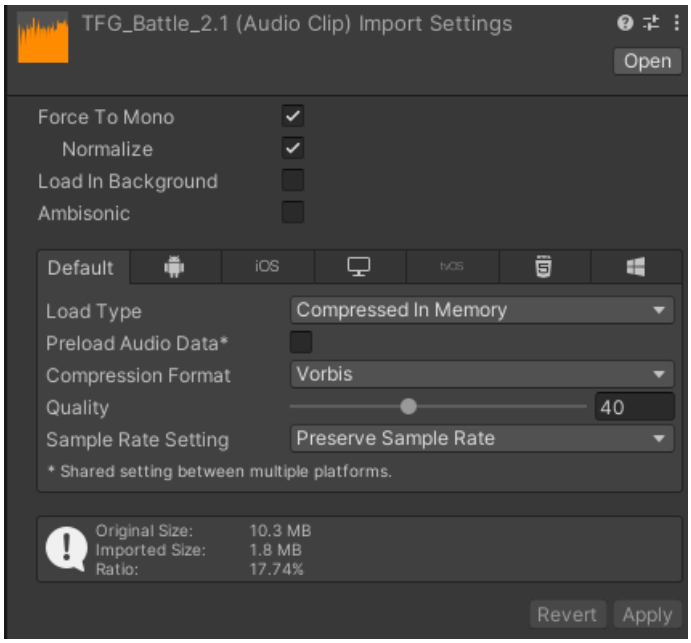
Kuva 5. 3D-mallin asetukset Unityn Inspector-ikkunassa.

Myös 3D-mallien tiedostokokoihin pystytään vaikuttamaan, niin Unityssä kuin jo mallia luodessa toisella ohjelmalla. 3D-assetin kokoon projektissa liittyy polygonien määrä, joka tulee pitää mielessä, kun tehdään 3D-mallia. Polygonien määrä tulee pitää mahdollisimman pienenä. 3D-maailma vaatii paljon malleja, mutta niissä ei tarvitse olla paljon yksityiskohtia, toisin kuin esimerkiksi pelattavalla hahmolla, jonka näyttävyyteen kannattaa panostaa.

Malleja luodessa tulee selvittää, mikä on käytettävän 3D-mallinnusohjelman yhden yksikön mitta Unityyn siirrettäessä. Malleja tehdessä pyritään aina tekemään malli siinä koossa, kuin se tulee olemaan myös pelissä ilman, että skaalausta tarvitsee tehdä Unityssä. Unityssä 3D-mallin koko on 1/yksikkö:1/metri. Yksikköasetuksia voidaan muuttaa myös jälkikäteen 3D-mallin Inspector-ikkunassa Unityssä. 3D-assettien kokoa voidaan saada pakattua pienemmäksi entisestään käyttämällä Unityn Vertex- tai Mesh Compression -toiminnallisuuksia. Samaan Mesh-komponenttiin ei voi käyttää molempia pakkaustoimintoja samanaikaisesti. (33; 35.)

Äänet

Ääni-assettien tuominen projektiin tapahtuu samalla tavalla kuin muidenkin assettien kohdalla. Unity tukee WAV-, MP3-, OGG- ja AIFF-äänitiedostoformaatteja. Äänitiedostot voivat olla peliprojektien suurimpia tiedostokokoja. Siksi ääniasetuksien konfigurointi on tärkeää, jos haluaa pitää pelin koon maltillisena. (34; 37.) Ääni-assetin Inspector-ikkunassa on asetuksia, joilla voidaan pienentää merkittävästi tiedoston kokoa. Nämä asetukset näkyvät kuvassa 6.



Kuva 6. Äänitiedoston asetukset Unityn Inspector-ikkunassa.

Mikäli ääni ei ole spatiaalinen, kannattaa muuttaa se stereosta monoksi. Jo pelkästään tämä voi pienentää tiedoston kokoa jopa puolella. Unity tarjoaa kolme erilaista pakkaustapaa: PCM, Vorbis ja ADPCM. PCM ei oikeastaan pakkaa äänitiedostoa lainkaan, jolloin tiedoston laatu pysyy koskemattomana, mutta tämä ei myöskään vähennä tiedoston kokoa. PCM:n etuna on, että sen purkaminen on nopeaa eikä vaadi paljon resursseja. Vorbis-pakkaus antaa käyttäjälle mahdollisuuden laskea äänitiedoston laatua prosenttiliukurilla, joka voi pienentää äänitiedoston kokoa huomattavasti. Laatua voi laskea hyvin 70 %:iin asti ilman, että voidaan huomata eroa. Äänen käyttötarkoituksen mukaan laatua voidaan hyvin laskea entisestään jopa 40 %:iin ilman kuultavaa laadun huononemista. Tiedoston purkamiseen voi kuitenkin kulua paljon suorittimen tehoja ja latausaika pitenee. ADPCM-pakkausformaatti pienentää tiedoston kokoa tehokkaasti, ja sen etuna on myös varsin nopea purkamisaika, joka tekee siitä houkuttelevimman pakkausvaihtoehdon. Ainoana ongelmana tässä pakkaustavassa on, että äänessä on huomattu vääristymiä puretussa äänitiedostossa, jos äänitiedostossa on korkean taajuuden ääniä. (28; 37.)

Asetuksissa pystytään vaikuttamaan äänitiedoston pakkauskoon lisäksi siihen, milloin tiedosto ladataan. Latausasetuksien osalta on tärkeää ymmärtää, mitä

erilaiset yhdistelmät tekevät. Jos asetuksista valitaan ”Load In Background”, aletaan äänitiedostoa lataamaan, kun sen toistamista kutsutaan ensimmäisen kerran, jonka jälkeen ääni toistetaan vasta, kun se on täysin ladattu. Ongelmana tässä asetuksessa on, että jos äänitiedosto on suuri, voi lataus kestää pidempään kuin on toivottavaa ja aiheuttaa viiveen äänen toistolle. Seuraavilla toistoilla viivettä ei ole. Tämä asetus soveltuu esimerkiksi taustaäänille (eng. ambience), joissa ei ole niin tärkeää, että ääni toistetaan heti, kun sitä kutsutaan. Jos taas valitaan ”Preload Audio Data”, äänitiedosto ladataan samalla, kun pelinäkymää ladataan. Tämä voi pidentää latausruudun kestoa, jos sellainen on pelinäkymien vaihtuessa, tai näyttää siltä, että peli olisi ei-responsiivinen, koska lataus kestää odotettua kauemmin. Tätä asetusta suositellaan käyttämään äänille, joita voidaan tarvita heti pelinäkymän avautuessa. Jos molemmat asetukset valitaan, alkaa peli lataamaan äänitiedostoa jo pelinäkymän latauksessa. Jos pelinäkymän lataus on valmis ennen kuin äänitiedoston lataus on valmis, jatkuu tiedoston lataaminen ladatun pelinäkymän aikana. Tätä asetusta kannattaa käyttää äänille, joita ei tarvita heti pelinäkymän avautumisen jälkeen, mutta joiden halutaan kuitenkin olevan valmiina nopeaan toistamiseen, kuten taistelun efektiäänet. Jos kumpaakaan asetusta ei valita, tiedostoa aletaan lataamaan vasta, kun sen soittamista kutsutaan ensimmäisen kerran. Tässä asetuksessa peli tulee käyttämään lataamiseen suorittimen pääsäiettä (eng. main thread), jonka käyttäminen tässä tapauksessa voi aiheuttaa pelissä ns. jäätymistä. Tämä asetus kannattaa olla valittuna pelkästään hyvin pienille äänitiedostoille, joita ei yleensä tarvitse toistaa samaan aikaan. (28; 39; 40.)

Esiteltyjen asetuksien lisäksi pystytään määrittämään, miten lataus tai äänitiedoston purkaminen tapahtuu, kun äänen toistoa kutsutaan. Unity tarjoaa tähän kolme erilaista vaihtoehtoa. Ensimmäisenä on ”Decompress On Load” -asetus, joka käyttää suoritinta äänitiedoston purkamiseen toistettavaan muotoon ennen kuin lataa sen muistiin. Tämä asetus vie muistista verrattain paljon tilaa, koska äänitiedosto säilötään muistiin sen pakkaamattomassa muodossa. ”Decompress On Load” -asetusta käytetään yleensä pienen tiedostokoon äänille, joita käytetään usein, kuten pelaajan askeläänet. Seuraavaksi on asetus ”Compress In Memory”, joka pitää ääni-assettia pakatussa muodossa muistissa, jolloin

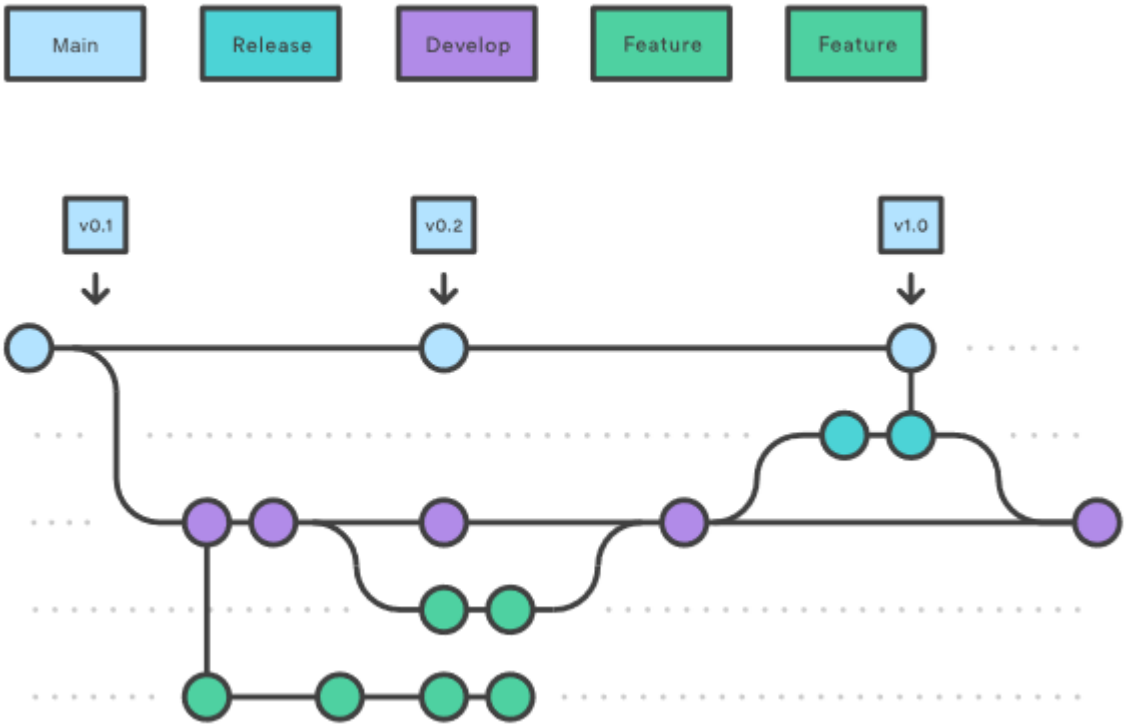
tiedosto vie vähemmän tilaa. Tämän asetuksen äänet ovat nopeita toistaa, mutta äänen toistamiseen käytetään suoritinta joka toistolla, kun ääni pitää purkaa toistettavaan muotoon. Tämä asetus soveltuu hyvin esimerkiksi äänille, joita tarvitaan usein, mutta ei jatkuvasti, ja jotka pitää saada toistettua nopeasti kutsumisesta. Suoratoistoasetus (eng. streaming) ei pura ääni-assettia lainkaan muistiin, mutta käyttää jatkuvasti suoritinta äänen purkamiseen ja toistamiseen. Tätä asetusta ei suositella käytettäväksi kuin enintään kahdelle äänelle pelinäkymässä, koska se vie paljon tehoja prosessorilta. Suoratoisto on kuitenkin toimiva vaihtoehto suurikokoisille äänitiedostoille, kuten musiikkiraidoille, jotka veisivät purettuna suuren määrän muistista. (28; 38.)

2.3 Versionhallinta pelikehityksessä

Hyvillä versionhallintastrategioilla ja -käytänteillä voidaan pitää yllä laadukasta projektinhallintaa sekä välttyä suurimmilta riskeiltä, joita peliprojektin kehityksessä voi tapahtua. Versionhallinnan käyttäminen pienissäkin peliprojekteissa, joissa on vain yksi kehittäjä, on monella tapaa hyödyllistä. (21, s. 5–6; 22.) Seuraavaksi esitellään yleisimpiä suositeltuja versionhallintakäytäntöjä.

Haarautumisstrategia

Hyvin määritelty tiedostovaraston haarautumien (eng. branching) mahdollistaa projektin ominaisuuksien ja osa-alueiden työstämisen samanaikaisesti muiden kehittäjien kanssa, ja projektinhallinta selkeytyy. Projektin juurihaara on yleensä nimetty main- tai master-nimiseksi, mistä käytetään nyt nimikettä päähaara. Päähaarassa säilytetään viimeisintä toimivaa versiota projektista. Päähaarasta haarautuu tarpeen mukaan useita projektin kehitykselle tarkoitettuja kehityshaaroja, joissa työstetään pelin toiminnallisuuksia. Tarpeen mukaan näistä haaroista voidaan tehdä vielä lisähaaroja toiminnallisuuksien kehitykseen. (23; 21, s. 47.) Kuvassa 7 havainnollistetaan haarautumisrakennetta.



Kuva 7. Selkeä rakenne-esitys versionhallinnassa usein käytetystä haarautumistavasta (23).

Kehityshaarojen (eng. develop) muutoksia kerätään uuteen julkaisuhaaraan (eng. release), josta seuraava julkaisu, tai laajempi päivitys, koostuu. Julkaisuhaarat liitetään sitten taas päähaaraan, jolloin päähaarasta tulee uusi versio. Päähaaran päivittämisen jälkeen kaikkien tulevien haarojen tulee periytyä uusimmasta päähaaran versiosta. (23; 21, s. 47.) Haaroja on hyvä poistaa sitä mukaa, kun ne on onnistuneesti liitetty ylemmän tason haaroihin, mikäli niiden sisällölle ei ole enää jatkossa käyttöä (24).

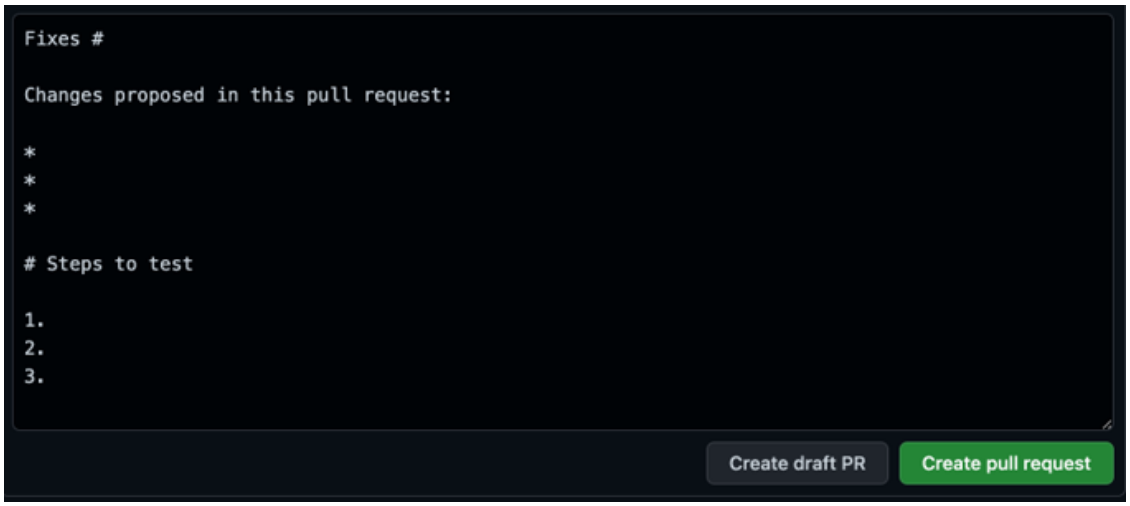
Talletuskäytännöt

Muutokset on parasta pitää pieninä ja tehdä niitä usein, jotta yksittäisiä muutoksia saadaan koteloitua. Tämä helpottaa mm. virhetilanteiden sattuessa ongelman löytämistä sekä projektin tehtävien etenemisen seuraamista. Talletuksille (eng. commit) tulee aina laittaa muutoksien sisältöä selvästi ja lyhyesti kuvaava selite. Esimerkki selkeästä talletusselitteestä voi olla ”Pelaajan tuplahyppy -

toiminnallisuus lisätty PlayerManager-ohjelmakoodiin”. Hyvin toteutetut talletuskäytännöt luovat myös samalla hyvää dokumentaatiota projektista ja sen etenemisestä. Tiimien kesken määritellään, kuinka usein ja miten haarojen yhdistämistä (eng. merge) suoritetaan, jotta projekti pysyy selkeänä ja hyvin hallinnassa. (24; 25.)

Vetopyynnöt

Vetopyyntöjen (eng. pull request/PR) tekeminen on tapa, jota on hyvä pitää yllä kehitystiimissä. Vetopyynnön kautta muu tiimi saa ajantasaista tietoa ja selkeän käsityksen muiden tiimin jäsenten osuuksien etenemisestä, ja vetopyynnön tekijä saa palautetta työstään ja koodistaan. Hyvässä vetopyynnössä käy selvästi ilmi muutoksien mukana tuleva sisältö, mahdollinen linkki projektinhallintatyökalussa sijaitsevaan tehtävään, johon vetopyyntö liittyy, sekä ohjeet muutoksien sisällön testaukseen. Kuvassa 8 näkyy pohja tyypillisen vetopyynnön sisällölle.



Kuva 8. Esimerkki selkeän vetopyynnön rakenteesta.

On hyödyllistä, että vetopyyntöjen käsittelijät tekevät myös testausta muutoksien sisällön osalta, jolloin testausta tapahtuu aina ennen kuin muutokset hyväksytään yhdistettäväksi korkeamman tason haaraan. Vetopyynnön voi tehdä myös luonnoksena jo hyvissä ajoin, ennen kuin kokonaisuus on valmis, mikä helpottaa vetopyynnön päivittämistä sitä mukaa, kun muutoksia tulee.

Luonnosvaiheessa voi myös pyytää muita tiimiläisiä keskustelemaan mahdollisista ongelmakohdista tai erilaisista ratkaisuista. Vetopyyntöjen sisällöille voidaan ajaa myös automaattisia testejä erilaisilla CI/CD-työkaluilla (mm. Jenkins). (25; 26.)

Tiedostovarastojen hallinta ja suojaus

Suurissa tiimeissä on järkevää rajoittaa tiedostovarastojen (eng. repository) ja niiden sisäisten haarojen käyttöoikeuksia. Vähintäänkin päähaarat yleensä suojataan. Suojausasetuksia on monenlaisia, mutta yksi yleisimmistä suojauksista on asetus, jossa valittuun haaraan ei voida yhdistää muutoksia ilman, että muutokset lähetetään vetopyynnön kautta ja hyväksytään. Isoissa tiimeissä tiedostovarastojen hallitsemisoikeuksia kannattaa jakaa maltillisesti ja vain tarpeeseen. Oikeuksia tulee muistaa ylläpitää ja poistaa säännöllisin väliajoin. (25; 27.)

2.4 Dokumentaatio- ja projektinhallintatyökalut pelikehityksessä

Peliprojekteissa tuotetaan monenlaista dokumentaatiota, kuten esimerkiksi pelisuunnitelmadokumentti (eng. game design document/GDD), kaavioita ja palaverimuistioita, joille kehitystiimi tarvitsee säilytyspaikan. Suosituimpia tapoja säilöä näitä dokumentteja on tallentaa ne pilveen eli internetissä isännöityihin tallennuspalveluihin. Näistä tunnetuimpia ovat OneDrive, Google Drive ja DropBox. Nämä palvelut tarjoavat tallennuksen lisäksi ympäristön järjestellä, seurata ja muokata digitaalisia dokumentteja tehokkaasti. Peliprojekteissa on yleistä, että pelimoottorin ulkopuolella kehitetyt assetit talletetaan tiedostovaraston lisäksi johonkin näistä pilvitallennuspalveluista, joka on tiimin käytettävissä. (41.)

Pelikehityksessä projektinhallintatyökalut tulevat tarpeeseen pelin koosta riippumatta. Projektin kokonaisuuden hahmottuminen ja hallitseminen helpottuu, kun käytössä on joku tai useampi monista tarjolla olevista projektinhallintatyökaluista. (42.)

Ensimmäisenä esimerkkinä on Atlassianin tarjoama Trello-työkalu, jonka avulla projektin kokonaisuus ja eteneminen visualisoidaan Kanban-taululla. Trello on vain yksi monista Kanban-taulun tarjoavista työkaluista. Trellossa luodaan työtila (eng. workspace), joka on yleensä projektikohtainen. Työtiloihin luodaan Kanban-taulu. Taulun tilan voi jakaa useampaan osaan, jotka toimivat yleensä tehtävien tilan määrittelyinä. Tehtävätiloja luodaan yleisimmin seuraavanlaisiin osiin: Backlog, To Do, In Progress, Stuck, Testing ja Done. Kuvassa 9 näkyvät mustat laatikot kuvastavat tehtävien tiloja, joihin tehtäviä asetetaan.

Kuva 9. Havainnollistava kuva Kanban-taulu projektinhallintatyökalusta.

Tehtäville luodaan kortit, joihin voidaan laittaa monenlaista tietoa tehtävän sisällöstä. Yleisimmin tehtäviin voidaan nimittää tehtäväntekijä, selite, etikettejä, aikatauluja, linkkejä yms. (43.)

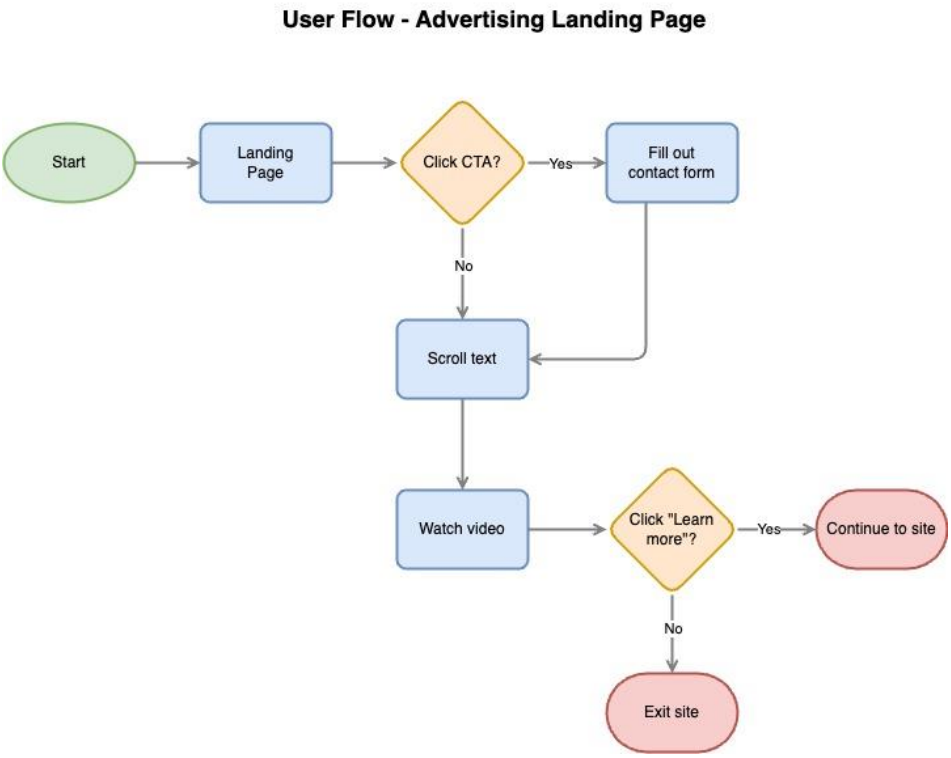
Useisiin projektinhallinta työkaluihin on olemassa myös hyödyllisiä plugineja eli lisäosia. Esimerkiksi GitHub-lisäosan avulla tehtäviin voidaan liittää peliprojektin GitHub-tiedostovaraston haaroja ja talletuksia. Tämän lisäksi suosituimpia lisäosia ovat OneDrive, Google Drive ja Slack. Trellon Drive-lisäosien avulla voidaan tehtäviin lisätä vaivatta suora polku pilvitallennussijaintiin. Slack auttaa yhdistämään tehtävät keskusteluihin Slackissä ja lisäämään muistutuksia tehtävien aikatauluista. (44.)

2.5 Tärkeimmät työkalut pelikehityksessä

Peliprojektit ovat yleensä useamman kuin yhden hengen projekteja, jolloin tarpeeseen tulee jokin kommunikointiin soveltuva työkalu. Tiimityöskentelylle tyypillisimpiä tarpeita ovat mahdollisuus laittaa viestejä ja kokouksien pitäminen ääni- tai videopuheluitse. Tunnetuimpia tähän tarkoitukseen sopivia palveluita ja sovelluksia ovat Slack, Google Meet, Microsoft Teams sekä Zoom. (66.)

Suunnittelutyökaluja, joiden avulla voidaan tukea tiimin työpajamaista yhteistyötä, on tarjolla monia. Esimerkksi Miro-sovellus tarjoaa mahdollisuuden reaaliaikaiseen tiimityöskentelyyn virtuaalisessa työtilassa. Miron hiekkalaatikko-tyylinen (eng. sandbox) työtila tarjoaa valtavasti erilaisia käyttötapoja. Pelikehityksessä Miroa voidaan käyttää esimerkiksi peli-ideoiden aivoriiheen, UI/UX-suunnitteluun ja palaverien jäsentelyyn. Miro tarjoaa lisäksi myös aiemmin esitellyn Kanban-taulu-toiminnallisuuden. Miro tarjoaa valmiita taulupohjia moniin tarkoituksiin, joten käyttäminen on tehty helpoksi ja nopeaksi. (45.)

Pelisuunnittelussa tarvitaan myös työkaluja, joilla tuottaa vaivatta monenlaisia diagrammeja. Diagrammeja käytetään esimerkiksi peleissä tilakoneiden toiminnallisuuden suunnitteluun, jonka avulla toiminnallisuuden ohjelmointi on helpompaa. Työkalua voidaan käyttää myös pelin käyttöliittymän toiminnallisuuksien havainnollistamiseen, kuten kuvassa 10. (46.)



Kuva 10. Esimerkki User Flow-diagrammista, joka on toteutettu Draw.io-työkalulla (47).

Peleissä käytettävien 2D- ja 3D-assettien luomiseen tarvitaan pelin tarpeisiin soveltuvia työkaluja ja sovelluksia. Tunnetuimpia 2D-assettien tuottamiseen tarkoitettuja sovelluksia ovat Adoben Photoshop ja Illustrator, GIMP ja Aseprite. Adoben Photoshop tukee monia formaatteja, ja sillä pystyy tekemään mm. spritejä, tekstuureja ja konseptitaidetta. Photoshop tarjoaa laajan kirjon myös työkaluja, kuten suuren valikoiman erilaisia siveltimiä, suodattimia ja muita kuvanmuokkaukseen liittyviä toimintoja. (48.) GIMP (Gnu Image Manipulation Program) on ilmainen ohjelma, jolla voidaan luoda digitaalista taidetta ja muokata kuvia, mutta sen toiminnallisuudet eivät ole yhtä hienostuneita ja laajoja kuin Photoshopissa. (49.) Adobe Illustrator on työkalu erityisesti vektorigrafiikan tuottamiselle, kun taas Aseprite on nimenomaan pikseligrafiikan tuottamiselle tarkoitettu ohjelma. (50; 51.)

Myös 3D-assettien suunnittelulle ja luomiselle on olemassa useita sovellusvaihtoehtoja, joista eniten käytettyjä ovat Blender, Autodesk Maya ja ZBrush. Näistä

vaihtoehdoista Blender on ilmainen ja hyvin monikäyttöinen kaikenlaisiin 3D-pelin assettitarpeisiin, kuten mallintamiseen ja animointiin. Blenderillä on lisäksi vahva käyttäjäyhteisö, josta on paljon tukea käyttäjille ja sovelluksen kehittäjille. (52.) Mayan vahvuuksiin kuuluu, mallintamisen lisäksi, hienostuneiden animaatioiden tekeminen, visuaaliset tehosteet ja riggaus eli 3D-mallin valmistelu animointia varten. Myös Mayassa on laajat muokkausmahdollisuudet MEL- ja Python-ohjelmoinnin avulla. (53.) ZBrush-ohjelmaa käytetään erittäin tarkkaan muotoiluun ja digitaaliseen maalaamiseen. Se tarjoaa lisäksi työkaluja dynaamisten meshien kehittämiseen ja tekstuurien maalaamiseen. (55.)

Kuten mainittu, monet 3D-mallinnusohjelmat tukevat animaatioiden tuottamista, mutta erillisiä animointi- ja riggausohjelmia löytyy myös. Esimerkiksi 3D-mallien animoinnille ja riggaukselle soveltuva ohjelma on Autodesk MotionBuilder. (56.) Myös 2D-grafiikan animoinnille löytyy ohjelmia, esimerkiksi Spine (57). Ohjelmien valitsemiseen vaikuttaa se, minkälaisia assetteja peliin ollaan tekemässä. Työkalujen valintaan vaikuttaa usein hinta tai työnantajan tarjoama työkalupakki, sillä monet ohjelmista ovat maksullisia.

Ääni-assettien suunnittelulle, tuottamiselle ja peleihin lisäämiselle löytyy niin ikään sovelluksia, jotka ovat tärkeässä osassa pelikehityksessä. Suosituimmiksi sovelluksiksi on nostettu FMOD Studio, Wwise, Audacity, Adobe Audition sekä Logic Pro X. Kaikilla sovelluksilla pystytään vähintään äänittämään ja editoimaan äänitiedostoja. (58.) FMOD Studio sekä Wwise erottuvat joukosta adaptiivisilla ominaisuuksillaan, mitkä helpottavat työprosessia, sillä sovelluksissa on lisäosa pelialan eniten käytetyille pelimoottoreille, Unitylle sekä Unreal Enginelle (59; 60). Adobe Audition sekä Logic Pro X ovat korkean tason työkaluja ammattimaiseen äänikehitykseen. Audacity on monen Indie-kehittäjän suosima ohjelma, jolla voi äänittää ja muokata äänitiedostoja. (58.)

Tekoälytyökaluja on alkanut ilmestyä markkinoille koko ajan enemmän, ja niistä monet on suoraan suunnattu pelikehityksen tueksi. Eniten tekoälyä käytetään luovien prosessien tueksi, kuten pelin tarinan ja hahmojen ideointiin ja kehittämiseen. 2D- ja 3D-assetteja voidaan luoda esimerkiksi Scenariolla ja Meshyllä.

Nämä tekoälytyökalut voivat säästää työaikaa huomattavasti. Assetteja voi jatkojalostaa halutessaan, käyttää mallina tai lisätä peliin sellaisenaan. Pelien tarinoiden ja hahmojen ideointiin sekä tekstintuottamiseen soveltuvia työkaluja ovat AI Dungeon sekä Charisma.ai (61).

Kuten huomataan, työkalujen tarjonta on laaja, ja niitä on mahdollista hyödyntää videopelituotannoissa moniin eri tarpeisiin. Pelikehitystiimin projektiohjeistuksissa on kannattavaa esitellä käytettävissä olevat työkalut ja ohjeistukset niiden käyttöön. Videopelituotannossa on tärkeää myös pysyä ajan tasalla uusimmista työkaluteknologioista, jotta kehittäjillä on parhaat mahdolliset resurssit sujuvaan työskentelyyn kaikissa pelikehityksen vaiheissa.

3 Peliprojektikohtaiset ohjeistukset Ylen BUU-klubben-sovelluksessa

Tässä luvussa esitellään BUU-klubben-sovellusta ja projektin teknisiä kokonaisuuksia. Kuten johdanto luvussa kerrottiin, sovellus on Svenska Ylen tuottama mobiilisovellus lapsille, mikä pohjautuu lastenohjelmaan nimeltä BUU-klubben. Sovellus sisältää projektikohtaisia teknisiä ratkaisuja, kuten BUUCore ja Leikkipinta, joiden ympärille koko sovellus sisältöineen on kehitetty.

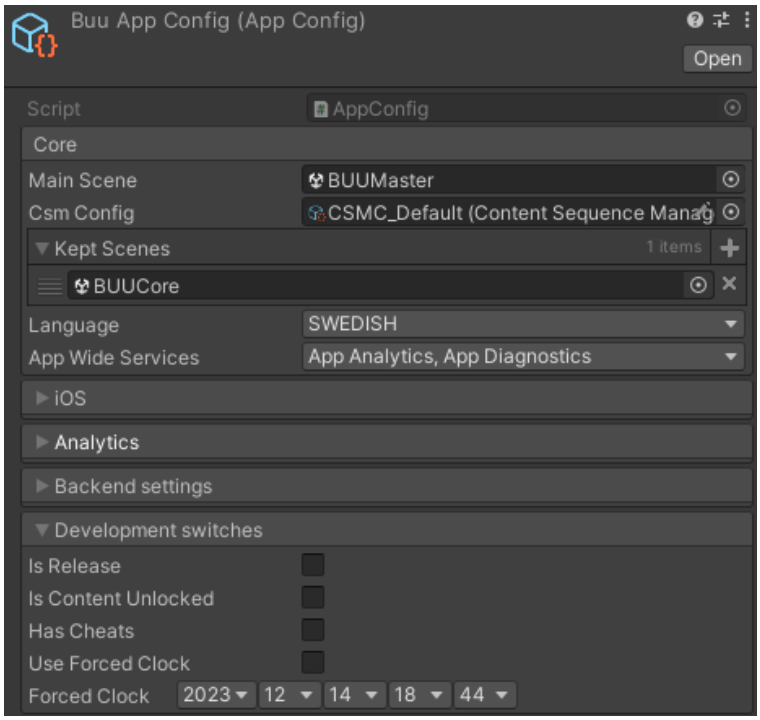
3.1 Ylen määrittelemät ohjeistukset tuotteille

Sovelluksen kehityksessä on otettava huomioon Ylen määrittelemät eettiset periaatteet ja Ylen strategia, aivan kuten muissakin Ylen tuottamissa mediasisällöissä. Yleisradiolaki edellyttää Yleä tuottamaan monipuolista ja kattavaa sisältötarjontaa, joka on kaikkien saatavilla. Sisältöjen tulee olla tarjolla yleisissä viestintäverkoissa, ja niitä tulee olla tarjolla kahdella kotimaisella kielellä, suomeksi ja ruotsiksi. Yle tuottaa sisältöjä ja palveluita myös saamenkielisille sekä muille vähemmistöille ja erityisryhmille. (62.)

Lapsille suunnattuun sovelluskehitykseen liittyy vahvasti Ylen lapsille kohdistettujen sisältöjen periaatteet, jotka ovat mainoksettomuus, turvallisuus ja oma kieli. Kun kehitetään lapsille tarkoitettuja palveluita, täytyy ottaa lisäksi huomioon tietosuojalainsäädäntö sekä ilmoittaa sovelluskauppasivuilla selvästi, minkälaista tietoa käyttäjästä kerätään. (63.) BUU-klubben-sovellus ei vaadi minkäänlaista sisäänkirjautumista, eikä kerää käyttäjätietoja, joiden perusteella käyttäjän voisi tunnistaa. Sovellus kuitenkin kerää analytiikkatarpeisiin seuraavia tietoja käyttäjästä: käyttäjä-ID, jota käytetään aktiivisten käyttäjien laskemiseen, laite-ID tai muu tieto laitteesta, jolla sovellusta käytetään, likimääräinen sijainti, sovelluksen virhetilaraportteja sekä sovelluksen sisällä laukaistuja toimintoja, joilla voidaan analysoida käyttäjien toimintaa sovelluksessa. (64.)

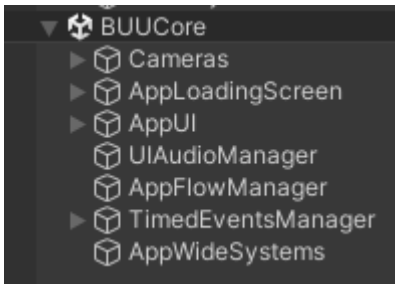
3.2 BUU-klubben-sovelluksen ydintoiminnallisuus eli BUUCore

BUU-klubben-sovellus on rakennettu sovellukselle kehitetyn ydintoiminnallisuuden ympärille, jota kutsutaan BUUCoreksi. BUUCoren logiikka määrittelee heti sovelluksen käynnistymisen jälkeen, mitä sovelluksessa tapahtuu seuraavaksi. BUUCore käyttää BUU App Config -scriptable -objektissa määriteltyjä toiminnallisuuksia, mitkä näkyvät kuvassa 11. Scriptable-objekti on Unityn tarjoama tietovarasto-objekti.



Kuva 11. BUUCoren käyttämä Buu App Config -scriptable -objekti.

BUUCore on pelinäkymä, joka liikkuu aina toisen pelinäkymän mukana, vaikka pelinäkymä vaihtuisi. Sovellus käynnistyy aina BUUIntro -pelinäkymästä, jossa BUUCore -pelinäkymä myös sijaitsee. Intro-animaation jälkeen BUUCore käynnistää latausruudun. Latauksen jälkeen käyttäjälle aukeaa sovelluksen Leikkipinta-näkymä. Latauksen aikana ladataan myös käyttäjän laitteesta mahdolliset tallennetut tiedot, joita käytetään heti ensimmäisessä Leikkipinta-pelinäkymässä. Kuvassa 12 näkyy BUUCoren sisältämät peliobjektit pelinäkymän objektihierarkiassa.



Kuva 12. BUUCore ja sen objektit, joissa on komponentteina monenlaisia toiminnallisuuksia.

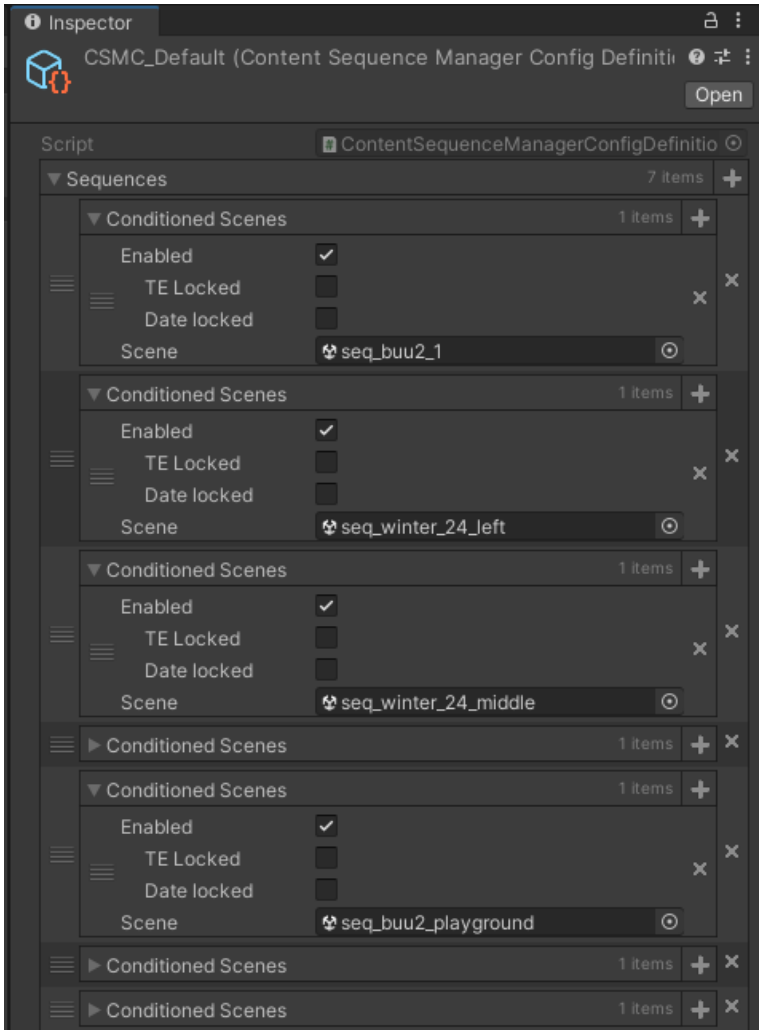
BUUCoressa on oma käyttöliittymäpohja (eng. canvas), jonka avulla käyttäjä voi navigoida minipeleistä takaisin Leikkipinnalle. Myös käyttöliittymän äänet ovat osana BUUCorea. Uusiin pelinäkymiin siirtyminen tehdään aina erillisellä ohjelmakoodilla, jossa kutsutaan BUUCoren latausruututoiminnallisuutta, jonka jälkeen haluttu pelinäkymä aukeaa. BUUCore hallitsee myös ajastettuja tapahtumia sovelluksessa, joiden mukaan sovelluksen latausruutu sekä Leikkipinnan sisältö määritellään.

3.3 BUU-klubben-sovelluksen päänäkymä eli Leikkipinta

Leikkipinta on BUU-klubben-sovelluksen päätoiminnallisuus, joka avautuu käyttäjälle, kun sovellus avataan. Leikkipinta sisältää useita pelinäkymiä, joita kutsutaan sekvensseiksi, joiden sisältö on rakennettu käyttöliittymäpohjalle. Sekvenssit muodostavat horisontaalisesti vieritettävän pelimaailman, joissa on monenlaista interaktiivista sisältöä käyttäjälle.

Leikkipinta kasautuu BUUMaster-pelinäkymän sisään, joka koostuu kamerasta, Leikkipinta-käyttöliittymäpohjasta ja erillisestä SunMoon-käyttöliittymäpohjasta sekä ContentSequenceManagerista (CSM), joka järjestää sekä lataa sekvenssit Leikkipinnalle. CSM hakee tiedot ContentSequenceManagerConfig (CSMC)-scriptable -objektista, jossa on määritelty sekvenssien esiintymisjärjestys, kun Leikkipinta-pelinäkymää vieritetään. Kuvassa 13 näkyy, kuinka CSMC:ssa sekvenssit voidaan järjestää, aktivoida ja deaktivoida sekä määritellä ajastetuiksi tapahtumiksi. Kausiluontoiset sekvenssit, kuten Halloween tai Joulukalenteri

ovat ajastettuja tapahtumia, eli ne ilmestyvät vieritettävälle Leikkipinnalle vain määriteltyinä aikoina.

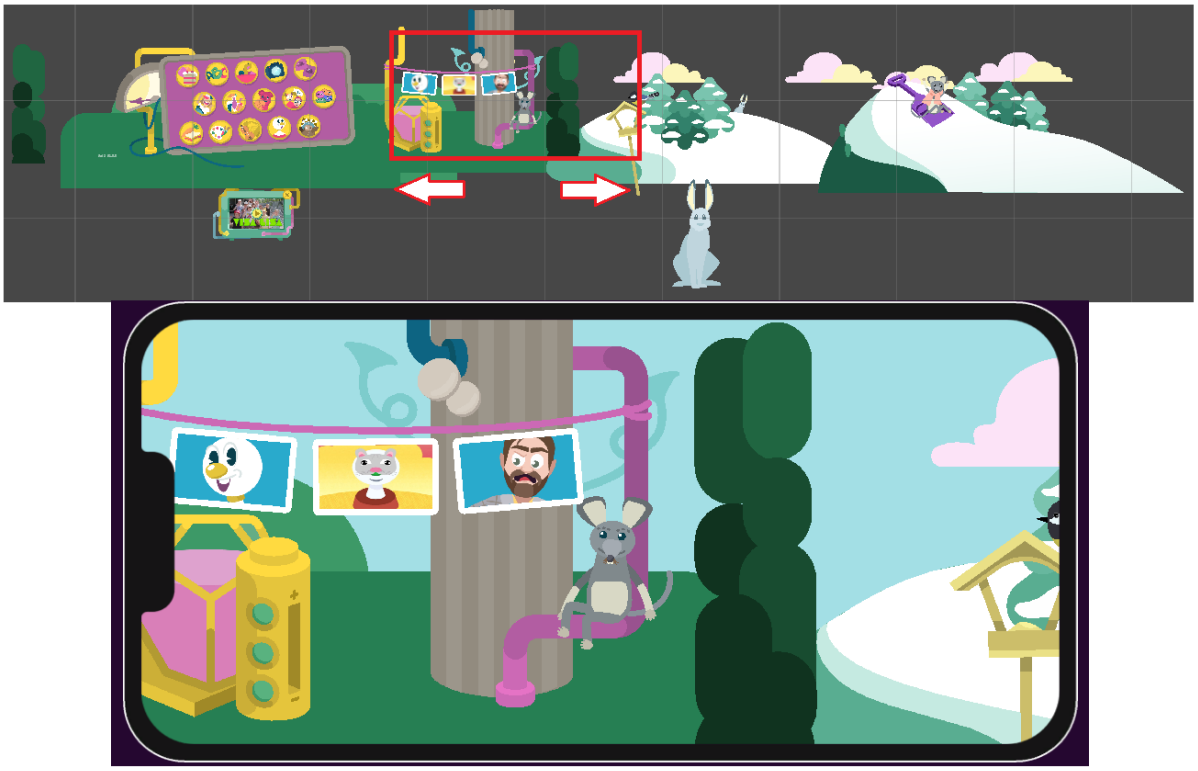


Kuva 13. CSMC-scriptable-objekti, jossa Leikkipinnan sekvenssit määritellään.

SunMoon-käyttöliittymäpohja toimii Leikkipinnan taustana, jossa käyttäjä voi vaihtaa päivän ja yön välillä. Tämä tausta pysyy koko ajan kameran keskellä, vaikka käyttäjä liikkuisikin eri sekvensseihin.

Jokainen sekvenssi on yksi kokonaisuus Leikkipinnalla. ContentSequence-Managerissa ensimmäiseksi määritelty sekvenssi on ensimmäinen kokonaisuus, joka näkyy käyttäjälle. Manageriin laitetaan listana ensimmäisen sekvenssin alle kaikki sitä seuraavat sekvenssit, jotka esiintyvät Leikkipinnalla, kun sitä vieritetään oikealle. Sekvenssit aktivoidaan sitä mukaan, kun kamera on

sitä edeltävän tai aiemman sekvenssin päällä. Sekvenssit deaktivoituvat sitä mukaan, kun sekvenssejä vieritetään. Jos käyttäjä vierittää oikealle niin kauan, että listan sekvenssit loppuvat, aktivoituu listan ensimmäinen sekvenssi taas viimeisen perään. Vierittäminen toimii tietenkin myös vasemmalle, jolloin sekvenssit tulevat käänteisessä järjestyksessä, alkaen listan viimeisimmästä. Kuvassa 14 havainnollistetaan sekvenssien järjestymistä vierekkäin sekä havainnollistetaan kuvassa renderöity sisältö, jota BuuCoren kamera kuvaa kyseisellä hetkellä.



Kuva 14. Kuvankaappaus Unitystä, jossa näkyy useampi sekvenssi. Kameran renderöimä alue on kehystetty punaisella. Alapuolella on Leikkipinnan kameran näkymä.

Sekvenssit, jotka ovat pelinäkymiä, ovat käyttöliittymäpohjalle rakennettuja kokonaisuuksia, jossa sisältö on jaettu yleensä kolmen eri tason sisälle: Background, Middle ja Foreground. Nämä eivät pelkästään lajittele sekvenssin sisältöä eri tasoille vaan mahdollistavat sekvenssin parallaksi-toiminnallisuuden. Parallaksi-toiminnallisuus vaikuttaa eri tasoilla olevien objektien

liikkumisnopeuteen, kun Leikkipintaa vieritetään. Kauimpana olevien objektien grafiikka liikkuu hitaasti, lähimpänä nopeasti ja keskellä normaalilla vieritysnopeudella. Tämä luo Leikkipinnalle illuusion näkymän syvyydestä.

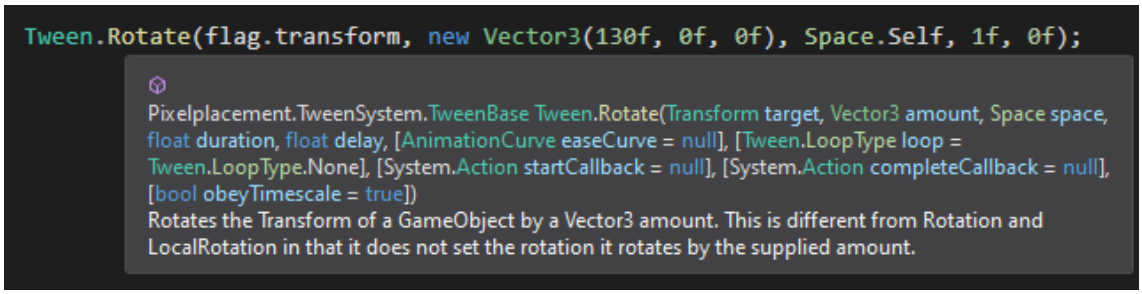
3.4 BUU-klubben-projektin sisäiset työkalut ja laajennukset

Unity-pelimoottorissa on mahdollista luoda omia työkaluja osaksi projekteja (64) ja näitä itsemukautettuja työkaluja on tehty myös BUU-klubben-projektiin. Projektin editoriin on tehty lisäikkunoita, joista yksi on analytiikkatoiminnallisuuden tarkastelulle, jossa näytetään selkeästi lähtevät analytiikkatapahtumat ja niiden parametrit. Tämä helpottaa analytiikan lisäämistä sekä testaamista uusissa toiminnallisuuksissa. Projektiin on tehty rajoitteita, jotka estävät paikallisen koontiversion (eng. build) tekemisen pelistä Unity-editorin omalla koontiominaisuudella, joten tätä varten on kehitetty oma koontiversion muodostaja (eng. builder). Julkaisuvalmiit koontiversiot laukaistaan Jenkinsin kautta, joka on CI/CD-työkalu.

Työkaluista löytyy ratkaisuja usein toistuviin tarpeisiin. Player Datan, eli peleissä tallennettavan tiedon, poistamiseen on kehitetty työkalu, joka poistaa datan yhdellä napin painalluksella sen sijaan, että kehittäjän tarvitsisi aina manuaalisesti navigoida resurssienhallinnan kautta. Toinen hyvin usein käytetty työkalu on BUUMasterin käynnistäminen, jota käytetään pelin testiajoon editorin sisällä, kun työskennellään Leikkipinnan parissa. Tämä nopeuttaa testaamista, sillä sekvenssejä muokataan aina omassa pelinäkymissään. Ilman tätä pikanäppäintä tulisi muuten aina erikseen navigoida BUUMaster- tai BUUIntro-pelinäkymään, jotta peliä, ja uuden sekvenssin rakentumista Leikkipinnalle, voisi testata.

Monissa projekteissa käytetään lisäksi kolmannen osapuolen tuottamia laajennuksia, jotka mahdollistavat erilaisia toimintoja projektissa. Myös BUU-klubben-projektissa on ulkopuolisten tuottamia laajennuksia, joita käytetään sovelluksessa laajasti. Yhtenä esimerkkinä on analytiikkaa keräävät laajennukset, joita on projektin olemassaolon aikana ollut useita. Tällä hetkellä projektissa on käytössä Adobe Analytics ja Unity Analytics.

PixelPlacement-laajennusta käytetään myös hyvin paljon. Kyseinen laajennus mahdollistaa peliobjektien ns. tweenaamisen. Tweenit ovat yksinkertaisia liikkeitä, joita annetaan peliobjekteille ohjelmakoodissa tai Inspector-ikkunassa nähtävien Tween-komponentin vaihtoehtojen avulla. PixelPlacement on käytännöllinen vaihtoehto, kun halutaan välttää paljon resursseja käyttävän animaattorin käyttöä. Kuvassa 15 näkyy yksinkertainen esimerkki siitä, kuinka PixelPlacement-laajannuksen Tween.Rotate-funktiota käytetään ja mitä parametreja se tarvitsee.



Kuva 15. Esimerkki PixelPlacement-laajennuksen tarjoamasta funktiosta peliobjektin kiertämiseen.

Tässä luvussa oli esiteltynä vain osa eniten käytetyistä työkaluista ja laajennuksista, joita käytetään BUU-klubben-projektissa. Projektien sisäiset työkalut ja niiden käyttö on hyvä dokumentoida huolellisesti projektien kehitysohjeistuksiin, koska ne eivät ole osa pelieditorin standardisisältöä.

4 Ohjeistusdokumentti videopeliprojektien tuotantojen aloittamiseen

Videopeliprojekteihin liittyy paljon sovittavia asioita, eikä yksikään projekti ole samanlainen. Tämän insinöörityön tuloksena syntyi valmis pohja ohjeistusdokumentille (liite 1), jota voidaan käyttää videopelituotannossa. Insinöörityön tekijän luoma ohjeistusdokumentti (liite 2) BUU-klubben-sovelluskehitystiimille on toiminut vahvana esimerkkinä ja motivaationa tämän insinöörityön sisällölle ja luodulle ohjeistusdokumenttipohjalle.

4.1 Ohjeistusdokumentin sisältö

Ohjeistusdokumentti sisältää valmiiksi otsikoidut osiot aiheille, joita tässä insinöörityössä on käsitelty luvussa 2. Jokaisessa peliprojektissa on mainittujen aiheiden lisäksi myös sisältöjä, joita on mahdotonta määritellä etukäteen, mutta niitä on kannattavaa avata dokumentissa. Hyvänä esimerkkinä erittäin projektikohtaisista, ja ohjeistusta vaativista, sisällöistä ovat BUU-klubben-videopeliprojektin BUUCore- sekä Leikkipinta-toiminnallisuudet, joita on käsitelty tämän työn luvussa 3. Kyseiset teknologiat ja ratkaisut ovat olennaisia kokonaisuuksia, jotka kyseisen peliprojektin parissa työskentelevän kehittäjän on tiedettävä. Ohjeistusdokumentti sisältää siis tärkeää tietoa peliprojektin parissa työskenteleville, ja se on erityisen hyödyllinen etenkin, jos tiimiin liittyy myöhemmin lisää jäseniä, jolloin ohjeistusdokumentti tukee projektiin perehtymistä.

Tässä insinöörityössä on kuitenkin jätetty kaksi hyvin tärkeää ja laajaa aluetta kokonaan pois, jotka ovat pelin testauskäytänteet ja laadunvarmistus sekä pelin analytiikan kerääminen ja analytiikan tärkeimmät mitattavat parametrit. Nämä osa-alueet ovat itsessään laajoja sekä usein hyvin projektikohtaisia, jolloin niistä on vaikea kertoa lyhyesti, mutta tarpeeksi kattavasti tämän työn puitteissa. Osiot on kuitenkin lisätty ohjeistusdokumenttiin, sillä ne ovat tärkeäitä dokumentoitavia kokonaisuuksia pelikehitysprojekteissa.

4.2 Dokumentin määritteet

Ohjeistusdokumenttipohja kattaa tässä luvussa esitellyt yleisimmät peliprojekteihin liittyvät aihealueet. Lisättyjen aihealueiden lisäksi voidaan tiimissä tarvita lisäosioita projektikohtaisille kokonaisuuksille, jotka vaativat lisätietoja sekä ohjeistuksia.

Yleiset tiedot projektista

Tämä osio on lisätty dokumenttiin, jotta siinä avattaisiin peliprojektin pääpiirteitä ja sisältöä. Kyseisessä osiossa voidaan kertoa myös projektin toteutuksesta ja kehityssuunnitelmasta.

Kieli- ja nimeämiskäytänteet projektissa

Dokumentista on hyvä löytää sovitut periaatteet kieli- ja nimeämiskäytänteille, joita noudatetaan projektissa. Hyviä sovittavia alueita ovat yhtenäiset nimeämistavat koodin funktioille ja muuttujille sekä pelimoottorissa peliobjekteille ja asseteille.

Työkalut

Työkalut -luvussa voidaan esitellä tiimin valitsemia työkaluja kaikenlaisiin projektin tarpeisiin. Muutamia yleisimpiä kokonaisuuksia, joihin tarvitaan työkaluja peliprojekteissa ovat: pelimoottori, dokumentointi ja dokumenttien säilytys, projektisuunnittelu, assettien luominen, versionhallinta, projektinhallinta, yhteistyötarpeet (palaverit ja keskustelut) ja analytiikka.

Pelimoottoriin liittyviä ohjeistuksia

Dokumentista on hyvä löytyä ohjeistukset ja sopimukset assettien tiedostoformaateista, assettien koista sekä niille määritellyt pakkausasetukset, joita noudatetaan. Dokumenttiin on hyödyllistä lisätä kaikki pelimoottorissa käytössä olevat työkalut ja laajennukset sekä ohjeistukset niiden käyttöön.

Versionhallinta

Versionhallintaohjelmistoista ja niiden käyttämisestä voidaan kertoa lyhyesti dokumentista. Oletuksena lienee, että jokainen tiimin jäsen osaa käyttää versionhallintaa, mutta kehitystiimeissä saattaa olla eroja haarautumisstrategioissa sekä vetopyyntö-, talletus-, veto- ja puskukäytänteissä, joista on hyvä antaa ohjeistus dokumentissa.

Projektinhallinta

Projektinhallintaosiossa voidaan esitellä valittu työkalu tai ratkaisu, jota tiimissä käytetään projektin jäsentelyä varten. Projektinhallintatyökalun käyttäminen voi vaatia myös ohjeistusta ja sovittuja tapoja, jotka voidaan esitellä tässä osiossa.

Pelin laadunvarmistus ja testaus

Tässä osiossa voidaan kertoa tiimissä sovituista testauskäytänteistä ja pelin laadunvarmistuksesta- ja valvonnasta (eng. quality assurance/QA).

Pelin analytiikka

Pelissä kerättävä analytiikka on tärkeä ja laaja kokonaisuus, joka on esiteltävä ohjeistusdokumentissa. Ohjeistuksissa voidaan kertoa mittareista, joita käytetään, valituista parametreista ja niiden tulkitsemisesta. Lisäksi ohjeistuksessa voidaan kertoa valitun analytiikkatyökalun käytöstä ja analytiikkatapahtumien (eng. event) kutsumisesta koodissa.

4.3 Ohjeistusdokumentin hyödyntäminen kehitysprosessissa

Ohjeistusdokumenttia voidaan käyttää peliprojektien aikana, erityisesti alussa, ohjaamaan tiimejä pohtimaan ja dokumentoimaan projektiin liittyviä käytänteitä, joista on järkevää sopia. Projekti voi sisältää myös erilaisia ohjeistusta vaativia sisältöjä, joita tulee noudattaa projektissa.

Ohjeistusdokumentin tarkoitus on selkeyttää projektin kokonaisuutta ja dokumentoida sovittuja asioita, joihin voi aina palata tarvittaessa. Ohjeistusdokumentin on tarkoitus muotoutua sitä käyttävän kehitystiimin tarpeiden mukaan. Dokumentin tekeminen voi tuntua työläältä, mutta sen olemassaolo todennäköisesti tulee tukemaan tiimin tuotteliaisuutta ja johdonmukaisuutta sekä tekee sen luomisesta ja ylläpitämisestä arvokkaan kokonaisuuden osana pelikehitysprosessia.

Dokumentin mahdollistaessa selkeät kiintopisteet kaikkiin ohjeistuksien osa-alueisiin se voi auttaa tiimiä arvioimaan ohjeistuksien sisältöä kriittisesti ja tehdä muutoksia tarpeen mukaan. Hyvin ylläpidetystä ja työstetystä ohjeistusdokumentista on hyötyä myös tuleviin projekteihin, sillä se tarjoaa käytännöllisen pohjan käytänteiden jälkitarkastelulle aina projektien päätyttyä. Hyväksi todetut käytänteet ja ohjeistukset voidaan uudelleen käyttää tulevissa projekteissa, joissa toistuu samat kokonaisuudet, jolloin dokumentin luomisen työmäärä vähenee.

5 Yhteenveto

Insinöörityössä esiteltiin yleisimpiä kehitysohjeistuksia, joita videopelien tuotannossa tarvitaan. Hyvät kehityskäytänteet, joista on dokumentointi sekä ohjeistus, auttavat kehitystiimiä pitämään projektin toimintatavat johdonmukaisena ja sisällöt selkeänä. Videopeliprojekteissa käytetään yleisesti pelialan standardikäytänteitä, mutta projektit sisältävät myös paljon tiimin kesken sovittavia aihealueita, joissa ei ole vain yhtä oikeaa tapaa toimia.

Työssä esiteltiin kieli- ja nimeämiskäytänteitä pelialla. Alan standardina pidetään englannin kieltä eli sekä koodissa että pelimoottorissa työskentelykieli on silloin englanti. Dokumentoinnin ja työkielen suhteen voi olla käytössä eri kieliä, mutta näissäkin tapauksissa suositellaan koodissa ja pelimoottorin sisällä pysymään englannissa, jotta mahdolliset sidosryhmät tai myöhemmin tiimiin liittyvät jäsenet pystyvät hyötymään dokumentaatiosta.

Nimeämiskäytänteisiin liittyy esimerkiksi koodien muuttujien ja funktioiden nimeäminen sekä pelimoottorissa assettien ja peliobjektien nimeäminen. Näissä käytänteissä ei ole mitään yhtä ja oikeaa tapaa toimia, mutta selkeyden vuoksi on hyvä sopia tiimissä toimintatavoista ja dokumentoida ne. Nimeämiskäytänteissä törmätään usein case-tyyleihin, joita on esitelty omassa osiossaan tarkemmin.

Pelimoottoriin liittyviä asetuksia ja toimintatapoja on kannattavaa avata myös tiimissä luotavaan ohjeistusdokumenttiin. Ensimmäiseksi voidaan dokumentoida sovitut pelimoottorin sisäiset hierarkiat ja kansiorakenteet. Johdonmukainen ja selkeä kansiorakenne helpottaa kehitystiimin navigointia projektin resursseissa. Pelinäkymissä taas peliobjektien hierarkiassa kannattaa ottaa huomioon mahdollisimman matalat peliobjektipuut. Tasojen, tunnisteiden sekä lajittelutasojen ja niiden järjestyksen ylläpitäminen selkeänä kokonaisuutena on ensiarvoisen tärkeää, ottaen etenkin huomioon, ettei tasoja voi luoda loputtomasti.

Pelimoottorissa käytetään monenlaisia tapoja pakata assetteja, jolloin pelin kokoon voidaan vaikuttaa. Ei ole yhdentekevää minkälaisilla asetuksilla assetteja pakataan, koska asetuksiin vaikuttavat assettien käyttötapa- ja tarkoitus. Nämä ovat myös kokonaisuuksia, joissa ei ole yhtä ja oikeaa tapaa toimia, joten tiimissä sovitut menettelytavat kannattaa dokumentoida.

Peliprojektien versionhallinnan osalta kannattaa sopia kehitystiimille sopivat menettelytavat. Tiimi voi päättää joustavasti versionhallinnan haarautumisstrategiastaan. Voi olla tarpeellista tehdä vetopyyntöjä, joissa käydään valmiita kokonaisuuksia läpi ja pyydetään ensin hyväksyntä muilta ohjelmoijilta ennen muutoksien liittämistä ylemmän tason haaraan. Vetopyyntöjen avulla muu tiimi pysyy paremmin tietoisena muiden tekemistä kokonaisuuksista ja saavat mahdollisuuden palautteen antamiselle.

Insinöörityössä esiteltiin erilaisia peliprojekteissa hyödynnettäviä työkaluja, joita käytetään mm. assettien luomiseen, dokumentaatioon, tiedostojen säilytykseen, suunnitteluun ja kommunikointiin. Työkalujen käyttö ei yleensä vaadi erillisiä ohjeistuksia, mutta joidenkin työkalujen käytölle voi olla hyödyllistä tehdä ohjeistusdokumentaatiota, esimerkiksi projektinhallintatyökaluille kuten Trellolle.

Monessa projektissa voi olla sisältöjä, joista kannattaa tehdä oma dokumentaationsa, kuten BUU-klubben-projektin BUUCore. Dokumentaation puute tämän tyylisistä sisällöistä voi olla erityisen ongelmallista tilanteessa, jossa kehitystiimiin liittyy uusia jäseniä tai kokonaisuuksien kehittäjät eivät ole enää tiimissä.

Kehitysohjeistusdokumentin luominen kehitystiimin käyttöön voi olla aikaa vievä prosessi, mutta siitä hyödytään pitkällä aikavälillä monin tavoin. Hyvin ohjeistetussa tiimissä työskentely on johdonmukaista ja projektin sisältö ei ajaudu kaaokseen. Olemalla järjestelmällisiä ja noudattamalla sovittuja käytänteitä voidaan välttyä väärinkäsityksiltä ja reflektoida projektin sisäisiä toimintotapoja, kun kaikki on kirjallisena. Hyvin tehtyä dokumenttia voidaan uudelleen käyttää ja päivittää myös seuraavien projektien tarpeisiin soveltuvaksi.

Lähteet

1 Mitä Ylen kulttuuritehtävä tarkoittaa? Verkkoaineisto. Yleisradio Oy. <yle.fi/aihe/s/10003317>. Luettu 7.6.2024.

2 BUU-klubben saa oman mobiilisovelluksen. 2014. Verkkoaineisto. Yleisradio Oy. <yle.fi/aihe/artikkeli/2014/08/13/buu-klubben-saa-oman-mobiilisovelluksen>. Luettu 7.6.2024.

3 Höstman, Eva-Marie. 2022. Konseptisuunnittelija, Svenska Yle, Helsinki. Keskustelu 6.11.2022.

4 The Crucial Role of Documentation in Software Development. Verkkoaineisto. Mach One Digital Corporation. <machonedigital.com/blog/the-crucial-role-of-documentation-in-software-development>. Luettu 15.6.2024.

5 Documentation standards to live by. Verkkoaineisto. Atlassian. <atlassian.com/work-management/knowledge-sharing/documentation/standards>. Luettu 15.6.2024.

6 Code Documentation: How to Do It Right. 2023. Verkkoaineisto. Skill Reactor. <skillreactor.io/blog/the-importance-of-code-documentation-and-how-to-do-it-right/>. 16.9.2023. Luettu 15.6.2024.

7 Montezzana, Mariana. 2023. The Importance of English for Technology + 3 Ideas to Boost Fluency in Tech Teams. Verkkoaineisto. Voxy. <voxy.com/blog/english-for-technology/>. Päivitetty 10.10.2024. Luettu 15.6.2024

8 McConnell, Steve. 2004. Code Complete. 2. painos. Redmond, Washington: Microsoft Press.

9 Dyankov, Dimitar. 2023. Clean Code Principles: Best Practices for Writing High-Quality Software. Verkkoaineisto. The Coding Hub. <thecodinghub.com/articles/clean-code-principles-best-practices-for-writing-high-quality-software>. 24.5.2023. Luettu 15.6.2024.

10 Amoah, Gervais Yao. 2023. A Guide to Clean Code: The Power of Good Names. Verkkoaineisto. DEV Community. <dev.to/gervaisamoah/a-guide-to-clean-code-the-power-of-good-names-3f6i>. 20.10.2023. Luettu 15.6.2024.

11 Programming Naming Conventions (Pascal, Snake, Camel & More). 2022. Verkkoaineisto. Title Capitalize. <titlecapitalize.com/programming-case-styles/>. 17.5.2022. Luettu 16.6.2024.

12 Coding Standard. 2024. Verkkoaineisto. Unreal Engine -dokumentaatio. Epic Games Inc. <dev.epicgames.com/documentation/de-de/unreal-engine/epic-cplusplus-coding-standard-for-unreal-engine>. 2024. Luettu 16.6.2024.

13 Buttfield-Addison, Paris; Manning, Jon & Nugent, Tim. 2019. Unity Game Development Cookbook. O'Reilly Media, Inc.

14 Mendez, Jose. 2023. Why folder structures matter. Verkkoaineisto. Unity. <unity.com/blog/engine-platform/why-folder-structures-matter>. 28.12.2023. Luettu 25.6.2024.

15 Prefabs. 2022. Unity Manual. Unity. <docs.unity3d.com/Manual/Prefabs.html>. 2022. Luettu 20.6.2024.

16 The Ramen Unity Style Guide. 2018. Verkkoaineisto. GitHub -tiedostovarasto. GitHub Inc. <github.com/stillwwater/UnityStyleGuide?tab=readme-ov-file#folders>. 13.9.2018. Luettu 20.6.2024.

17 Bonet, Rubén Torres. 2019. Unity Scene Hierarchy: Catch that Performance Thief! (Part 2). Verkkoaineisto. The GameDev Guru. <thegamedev.guru/unity-performance/scene-hierarchy-catch-that-performance-thief-part-2/>. 19.11.2019. Luettu 25.6.2024.

18 Layers. 2022. Verkkoaineisto. Unity Manual. Unity. <docs.unity3d.com/Manual/Layers.html>. 2022. Luettu 25.6.2024.

19 Tags. 2022. Verkkoaineisto. Unity Manual. Unity. <docs.unity3d.com/Manual/Tags.html>. 2022. Luettu 25.6.2024

20 2D Sorting. 2022. Verkkoaineisto. Unity Manual. Unity. <docs.unity3d.com/Manual/2DSorting.html>. 2022. Luettu 27.6.2024.

21 Unity Technologies. 2022. ABM Version Control and Project Organization in Unity. E-kirja. Unity.

22 What is version control? 2024. Verkkoaineisto. Atlassian. <atlassian.com/git/tutorials/what-is-version-control>. 2024. Luettu 3.7.2024.

23 Gitflow workflow. 2024. Verkkoaineisto. Atlassian. <atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>. 2024. Luettu 3.7.2024.

24 Van Kemenade, Hugo. 2023. Senior Developer, Digia. Helsinki. “git tips” -esitys.

25 Pull requests documentation. 2024. Verkkoaineisto. GitHub, Inc. <docs.github.com/en/pull-requests>. 2024. Luettu 7.7.2024.

26 Azure Repos Git Documentation. 2024. Verkkoaineisto. Microsoft. <learn.microsoft.com/en-us/azure/devops/repos/git/?view=azure-devops>. 2024. Luettu 7.7.2024.

27 GitHub Repository Best Practices. 2024. Verkkoaineisto. DEV Community. <dev.to/pwd9000/github-repository-best-practices-23ck>. Päivitetty 17.8.2024. Luettu 7.7.2024.

28 Optimize Your Games In Unity – The Ultimate Guide. 2022. Verkkoaineisto. Awesome Tuts. <awesometuts.com/blog/optimize-unity-game/>. 2022. Luettu 7.7.2024.

29 Textures. 2022. Verkkoaineisto. Unity Manual. Unity. <docs.unity3d.com/Manual/ImportingTextures.html>. 2022. Luettu 7.7.2024.

30 Recommended, default, and supported texture formats, by platform. 2022. Verkkoaineisto. Unity Manual. Unity. <docs.unity3d.com/Manual/class-TextureImporterOverride.html>. 2022. Luettu 9.7.2024.

31 Texture Import Settings. 2022. Verkkoaineisto. Unity Manual. Unity. <docs.unity3d.com/Manual/class-TextureImporter.html#platform>. 2022. Luettu 9.7.2024.

32 Sprite Atlas. 2022. Verkkoaineisto. Unity Manual. Unity. <docs.unity3d.com/Manual/sprite-atlas.html>. 2022. Luettu 26.8.2024.

33 Importing Model Files. 2022. Verkkoaineisto. Unity Manual. Unity. <docs.unity3d.com/Manual/ImportingModelFiles.html>. 2022. Luettu 12.7.2024.

34 Audio Overview. 2022. Verkkoaineisto. Unity Manual. Unity. <docs.unity3d.com/Manual/AudioOverview.html>. 2022. Luettu 12.7.2024.

35 Models. 2022. Verkkoaineisto. Unity Manual. Unity. <docs.unity3d.com/Manual/models.html>. 2022. Luettu 12.7.2024.

36 Model tab. 2022. Verkkoaineisto. Unity Manual. Unity. <docs.unity3d.com/Manual/FBXImporter-Model.html>. 2022. Luettu 12.7.2024.

37 Indrayana, Made. 2020. Understanding Audio Compression Settings in Unity. Verkkoaineisto. <medium.com/@made-indrayana/understanding-audio-compression-settings-in-unity-e879a821023f>. 15.4.2020. Luettu 15.7.2024.

38 Indrayana, Made. 2020. Choosing the Right Load Type in Unity's Audio Import Settings. Verkkoaineisto. <medium.com/@made-indrayana/choosing-the-right-load-type-in-unitys-audio-import-settings-1880a61134c7>. 6.5.2020. Luettu 15.7.2024.

39 Indrayana, Made. 2020. Verkkoaineisto. Preload Audio Data & How Unity Decides Which Audio Assets to Load to a Scene. <medium.com/@made-indrayana/preload-audio-data-how-unity-decides-which-audio-assets-to-load-to-a-scene-a440a654e7e2>. 28.5.2020. Luettu 17.7.2024.

40 Indrayana, Made. 2020. Verkkoaineisto. Load in Background - Optimizing Audio Load Times in Unity. <medium.com/@made-indrayana/load-in-background-optimizing-audio-load-times-in-unity-33d3cc04c47e>. 18.6.2020. Luettu 17.7.2024.

41 Mitchell, Robert. 2023. Verkkoaineisto. 10 top file-sharing services: Dropbox, Box, Google Drive, OneDrive, and more. IDG Communications, Inc. <computerworld.com/article/1615178/top-file-sharing-services-dropbox-box-google-drive-onedrive-more.html>. 23.9.2023. Luettu 22.8.2024.

42 8 Best Project Management Tools for Game Developers. 2024. Verkkoaineisto. Teamhub Group Ltd. <teamhub.com/blog/best-project-management-tools-for-game-developers/>. 2024. Luettu 25.7.2024.

43 What is a kanban board? 2024. Verkkoaineisto. Atlassian. <atlassian.com/agile/kanban/boards>. 2024. Luettu 25.7.2024.

44 Power-Ups. 2024. Verkkoaineisto. Atlassian. <trello.com/power-ups/all>. 2024. Luettu 25.7.2024.

45 Product Overview. 2024. Verkkoaineisto. Miro. <miro.com/product-overview/>. 2024. Luettu 31.7.2024.

46 5 Top Free Diagramming Tools for Software Development. 2023. Verkkoaineisto. DEV Community. <dev.to/evergrowingdev/5-top-free-diagramming-tools-for-software-development-a4a>. Päivitetty 29.6.2023. Luettu 31.7.2024.

47 Schmitt, Bastian. 2023. Verkkoaineisto. I 🧡 UX Diagrams – Create user flow diagrams in Confluence. JGraph Ltd. <drawio-app.com/blog/i-%F0%9F%A7%A1-ux-diagrams-how-to-create-ux-diagrams-inside-confluence-or-jira/>. 13.10.2023. Luettu 26.7.2024.

48 Tutustu Adobe Photoshopin uusiin ominaisuuksiin. 2024. Verkkoaineisto. Adobe. <adobe.com/fi/products/photoshop/features.html>. 2024. Luettu 3.8.2024.

49 The Free & Open Source Image Editor. 2024. Verkkoaineisto. GIMP's Team. <gimp.org/>. 2024. Luettu 3.8.2024.

50 Tutustu Adobe Illustratorin uusiin ominaisuuksiin. 2024. Verkkoaineisto. Adobe. <adobe.com/fi/products/illustrator/features.html>. 2024. Luettu 3.8.2024.

51 Animated Sprite Editor & Pixel Art Tool. 2024. Verkkoaineisto. Igara Studio S.A. <aseprite.org/>. 2024. Luettu 3.8.2024.

52 Blender 4.2 LTS. 2024. Verkkoaineisto. Blender. <blender.org/>. 2024. Luettu 3.8.2024.

53 Autodesk Maya: Create expansive worlds, complex characters, and dazzling effects. 2024. Verkkoaineisto. Autodesk Inc.<autodesk.com/eu/products/maya/overview?term=1-YEAR&tab=subscription>. 2024. Luettu 3.8.2024.

54 Autodesk 3ds Max: Create massive worlds and high-quality designs. 2024. Verkkoaineisto. Autodesk Inc <autodesk.com/eu/products/3ds-max/overview?term=1-YEAR&tab=subscription>. 2024. Luettu 3.8.2024.

55 The industry standard for digital sculpting and painting. 2024. Verkkoaineisto. Maxon Computer GMBH. <maxon.net/en/zbrush>. 2024. Luettu 3.8.2024.

56 Autodesk MotionBuilder: Bring characters and creatures to life. 2024. Verkkoaineisto. Autodesk Inc <autodesk.com/eu/products/motionbuilder/overview?term=1-YEAR&tab=subscription>. 2024. Luettu 3.8.2024.

57 2D animation for games. 2024. Verkkoaineisto. Esoteric Software LLC. <esotericsoftware.com/>. 2024. Luettu 3.8.2024.

58 Reaber, Grant. 2024. Verkkoaineisto. Top Sound Effects Tools for Indie Game Developers: A 2024 Guide. Respeecher. <respeecher.com/blog/top-sound-effects-tools-for-indie-game-developers-a-2024-guide>. 4.6.2024. Luettu 4.8.2024.

59 Welcome to FMOD Studio. 2024. Verkkoaineisto. Firelight Technologies Pty Ltd. <fmod.com/docs/2.03/studio/welcome-to-fmod-studio.html>. 2024. Luettu 4.8.2024.

60 Where Sound Awakens New Worlds. 2024. Verkkoaineisto. Audiokinetic Inc. <audiokinetic.com/en/>. 2024. Luettu 4.8.2024.

61 Shanjidah, Jafar. 2023. Verkkoaineisto. The 12 Most Useful AI Tools for Game Developers in 2024. Squash Labs Inc. <squash.io/the-12-most-useful-ai-tools-for-game-developers-in-2024/>. Päivitetty 12.12.2023. Luettu 4.8.2024.

62 Ylen strategia – kaikille yhteinen, jokaiselle oma. 2020. Verkkoaineisto. Yleisradio Oy. <yle.fi/aihe/strategia>. 19.5.2020. Luettu 8.8.2024.

63 Lapsi verkossa – Näkökulmia lasten oikeuksiin ja tietosuojaan digitaalisessa ympäristössä. 2019. Verkkoaineisto. Lastensuojelun Keskusliitto. <lskl.fi/wp-content/uploads/Lapsi-verkossa.pdf>. Luettu 10.8.2024.

64 Google Play -verkkosivut. 2024. Verkkoaineisto. BUU-klubben-sovellussivu. Yleisradio Oy. <play.google.com/store/apps/data-safety?id=fi.yle.buu&hl=en>. 2024. Luettu 10.8.2024.

65 Using Custom Editor Tools. 2022. Verkkoaineisto. Unity Manual. Unity. <docs.unity3d.com/Manual/UsingCustomEditorTools.html>. 2022. Luettu 21.8.2024.

66 Leading virtual meeting platforms to consider in 2023. 2023. Verkkoaineisto. Slack Technologies. <slack.com/blog/collaboration/best-virtual-meeting-platform>. 4.12.2023. Luettu 22.8.2024.

Liite 1. Project Guidelines for Buu 2.0.pdf

Updated 12/2023

Project Guidelines for BUU app

Disclaimer:
The document is written with the assumption that its readers know the basics of Unity and game development.
You might see that not all guidelines or conventions have not been practised in the project. That is exactly why this documentation has been made. There have been

Liite 2. Videopeliprojektien ohjeistusdokumentti -pohja.docx

VIDEOPELIPROJEKTIEN OHJEISTUS

SISÄLLYS

YLEISESTI PROJEKTISTA

Tässä luvussa esitellään peliprojekti yleisesti. Esittely on hyvä pitää tiiviinä, sillä tämä ei ole GDD (Game Design Document).

Tässä dokumentissa keskitytään kirjaamaan tietoja projektista, jotka liittyvät ohjeistuksiin pelin tekemisen aikana eikä niinkään pelin sisältöön.

Tätä dokumenttia on hyvä ylläpitää projektin aikana. Aina, kun jotain sovitaan yhteisesti tai jokin aiemmin sovittu käytäntö muuttuu, tulee dokumenttiin tehdä päivityksiä. Käyttäkää dokumenttia hyödyksi myös reflektointiin projektin päätyttyä. Kerran täytetty dokumentti toimii hyvin pohjana myös tuleville projekteille.

KIELI- JA NIMEÄMISKÄYTÄNTEET

Tässä luvussa annetaan kehitystiimille selkeä ohjeistus projektin kieli- ja nimeämiskäytänteistä. Kirjatkaa dokumenttiin selvästi käytettävä kieli ja kuinka sitä käytetään missäkin peliprojektin osuudessa. Alla on kysymyksiä, joita voi käyttää apuna osion kirjoittamiseen.

- Mikä on projektissa käytettävä kieli?
- Pysyykö kieli samana projektin kaikissa alueissa?
    - Pelimoottorin sisällä?
    - Koodissa?
    - Tiedostojen ja assettien nimeämisessä?
    - Suunnitteludokumenteissa?
    - Tiimin välinen kommunikointi?
    - Onko mahdollista, että tiimin ulkopuoliset tarkastelevat projektia ja millä kielellä dokumentti olisi silloin hyvä olla?
- Miten peliprojektin sisällä nimetään objekteja ja assetteja?
    - ➔ Laittakaa esimerkkejä case -tyyleistä eri käyttötarkoituksista

PELIMOOTTORI

Tässä luvussa voidaan ohjeistaa seuraavissa pelimoottoriin liittyvissä asioissa:

- Pelimoottorin sisällä asetettavat asetukset
- Haluttu kansiohierarkia, jonka voi havainnollistaa esimerkiksi alla esitetyllä tavalla

```
Assets
+---Art
|   +---Materials
|   +---Models      # FBX and BLEND files
|   +---Textures    # PNG files
+---Audio
|   +---Music
|   \---Sound       # Samples and sound effects
+---Code
|   +---Scripts     # C# scripts
|   \---Shaders     # Shader files and shader graphs
+---Docs            # Wiki, concept art, marketing material
+---Level           # Anything related to game design in Unity
|   +---Prefabs
|   +---Scenes
|   \---UI
\---Resources       # Configuration files, localization text and other
user files.
```

- Pelinäkymien peliobjektihierarkia säännöt
- Tagit, layerit, sorting layerit ja order in layer (nimeämiset, käytänteet, käyttötavat)

• ASSETTI-OHJEISTUKSET

Pelin assetteihin liittyvät ohjeistukset on hyvä kirjata selvästi.

- Assettien formaatit (2D-grafiikka, 3D-mallit, ääni, fontit)
- Koko ohjeistuksia
- Pakkausasetukset pelimoottorissa

• PELIMOOTTORIN TYÖKALUT JA LAAJENNUKSET

Tässä luvussa esitellään työkaluja ja laajennuksia, joita käytetään pelimoottorissa tämän peliprojektin osalta. Työkalujen ja laajennuksien käyttämistä voi joutua ohjeistamaan, mikäli ne eivät ole standardeja tai yleisesti tuttuja kehitystiimin kesken.

VERSIONHALLINTA

Tässä osiossa voidaan kertoa tiimissä sovituista versionhallintakäytänteistä.

Esimerkiksi:

- Branching-strategia
- Käyttötavat

PR-politiikka ja esimerkit hyvästä PR:stä

TYÖKALUT

Tässä luvussa määritellään projektissa käytettävät työkalut.

Työkaluja ja niiden käyttöä voidaan määritellä seuraaville kokonaisuuksille:

- Pelimoottori
- Dokumentointi ja säilytys
- Suunnittelu
- Assettien luominen
- Versionhallinta
- Projektinhallinta
- Yhteistyö (palaverit ja keskustelut)
- Analytiikka

PROJEKTINHALLINTA

Tässä luvussa voidaan kertoa tiimissä sovitusta projektinhallinnasta ja ohjeistuksista siihen liittyen.

- Esimerkiksi, jos tiimillä on käytössä Trello, voidaan luvussa ohjeistaa Trellon käyttöä, selventää jokaisen omaa roolia projektinhallinnassa ja ylläpidossa.

QA JA TESTAUS

Tässä luvussa kerrotaan, kuinka tiimi on sopinut pelin testausmenettelyistä.

PELIN ANALYTIIKKA

Tässä luvussa kerrotaan, kuinka pelissä kerätään analytiikkaa ja millä työkalulla. Ohjeistuksessa on hyvä käydä ilmi tärkeimmät parametrit ja data, joita pelistä halutaan sekä selkeä ohjeistus analytiikkatapahtumien kutsumiseen koodissa.