VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Tommi Kallio-Kujala

KÄYTTÖLIITTYMÄSUUNNITTELU JA TOTEUTUS

Liiketalous
2020

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Tommi Kallio-Kujala
Opinnäytetyön nimi	Käyttöliittymäsuunnittelu ja toteutus
Vuosi	2020
Kieli	suomi
Sivumäärä	31
Ohjaaja	Päivi Rajala

Opinnäytetyön tarkoituksena on suunnitella ja tehdä käyttöliittymä Unity-pelimoottorilla. Pelinä toimii yhdessä Teemu Karhusen kanssa rakentamamme projekti. Tässä työssä käsitellään käyttöliittymän suunnittelun ja rakentamiseen liittyvää teoriaa.

Työssä selvitetään miten, Unityllä toteutetaan käyttöliittymä. Työssä käsitellään käyttöliittymäobjektien luontia ja niiden ohjausta. Työssä esitellään myös, miten siirretään dataa toisesta käyttöliittymästä toiseen näkymän vaihtuessa.

Opinnäytetyönä tehdyn projektin tuloksena syntyi pelin suunnitelma. Lisäksi käsitellään hieman Collab-työkalun käyttöön liittyviä havaintoja.

Avainsanat Suunnittelu, Ohjelmointi, Unity

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information technology & business

ABSTRACT

Author	Tommi Kallio-Kujala
Title	User Interface Design and Implementation
Year	2020
Language	Finnish
Pages	31
Name of Supervisor	Päivi Rajala

The objective of this thesis was to design and implement a user interface to a videogame project. The project was done in Unity with Teemu Karhunen. This thesis includes theory on how to build and design a user interface.

In this thesis how to produce a user interface using unity was examined and explained. This includes the creation and handling of user interface elements. How to send information from one scene to another during scene change was also reviewed in this bachelor's thesis.

The project that was made during this thesis project created a game design. Collab -tool was also examined to some extent in this work.

Keywords Designing, Programming, Unity

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

1 JOHDANTO

Graafisia käyttöliittymiä on tietokoneissa ollut jo 1960-luvun loppupuolella. Muut kuin graafiset käyttöliittymät ovat poistuneet asiakasmarkkinoilta melko täysin, sekä ohjelmisto- että käyttöjärjestelmätasolla. Tekstipohjaisia käyttöliittymiä käytetään nykyään pääosin vain Linux-palvelinympäristöissä koska, tämä takaa vakaamman toimivuuden ja on alustalleen kevyempi.

Itse olen käyttänyt vuosien varrella paljon erilaisia käyttöliittymiä erilaisten ohjelmistojen ja käyttöjärjestelmien kanssa. Käyttöliittymien suurimmat muutokset alkoivat tapahtua kosketusnäyttöjen yleistymisen mukana.

Opinnäytetyössä käyn läpi kaikki eri vaiheet käyttöliittymän rakentamisprosessissa. Tarkoituksena on suunnitella ja toteuttaa mahdollisimman toimiva sekä selkeäkäyttöinen käyttöliittymä. Opinnäytetyö on laadullinen toiminnallinen tutkimus, jossa rakennetaan pelin käyttöliittymä.

Aloitan tämän opinnäytetyön teoriaosuudella, jossa kerron hieman käyttöliittymistä yleisesti ja niiden historiasta. Tämän jälkeen kerron hieman erilaisista käyttöliittymänelementeistä, joita peliin on toteutettu.

2 KÄYTTÖLIITTYMÄT

2.1 Yleistä

Käyttöliittymä on ohjelmistossa osa, jonka kautta käyttäjä käyttää ohjelmistoa. Käyttöliittymä määräytyy ohjelmiston antamasta viestinnästä ja käytännöllisestä kokonaisuudesta. Tämä kokonaisuus voi sisältää muun muassa valikoita, painikkeita ja näyttöjä.

Käyttöliittymät jaetaan yleensä ryhmiin tyylin mukaan. Näitä ovat esimerkiksi tekstipohjaiset käyttöliittymät (TUI), komentorivi pohjaiset käyttöliittymät (CLI) ja grafiikka pohjaiset käyttöliittymät (GUI). (Frank McCown 2017.)

2.2 Käyttöliittymäsuunnittelu ja Käyttökokemussuunnittelu

Käyttäjä käyttää sovellusta käyttöliittymän kautta. Käyttöliittymä suunnittelun tarkoitus on siis rakentaa tarvittava pohja käyttökokemukselle. (Oryzo 2020.)

Käyttökokemussuunnittelu tarkoittaa tietokoneiden, sovellusten ja muiden elektronisten laitteiden järjestelmäsuunnittelua, jossa tähdätään mahdollisimman korkeaan käytettävyyteen ja käyttökokemukseen. Käyttäjäkokemus keskittyy mahdollisimman hyvin ymmärtämään mitä käyttäjä haluaa, tarvitsee, osaa ja pystyy tekemään. (Oryzo 2020.)

3 KÄYTTÖLIITTYMÄSUUNNITTELUN ASKELEET

Tässä luvussa käyn lävitse käyttöliittymäsuunnittelun askeleita. Näissä käydään lävitse mitä tulee ottaa huomioon suunnittelun alkuvaiheista prototyyppi versioihin ja värien valintaan.

3.1 Tunne käyttäjäsi tai asiakkaasi ja Ymmärrä liiketoiminnallinen tarkoitus

Aluksi on hyvin tärkeää ymmärtää mitä ihmiset tekevät ja mikä siinä on kriittistä. Ensiaskeleet suunnittelussa sisältää käyttäjien luontaiset ja opitut ominaisuudet. Näiden pohjalta tulee ymmärtää miten se vaikuttaa suunnitteluprosessiin. (Galitz 2007.)

Järjestelmän tulee saavuttaa sille tarkoitettu liiketoiminnallinen toiminnallisuus. Tähän päästäkseen suunnittelijan tulee ymmärtää järjestelmän käyttökohde ja mitä tehtäviä järjestelmä suorittaa. Alussa tulisi päättää liiketoiminnalliset perustoiminnallisuudet, käyttäjien aktiviteettien analysointi käyttäjän tehtäväanalyysin kautta, käyttäjän mielellisen mallin ymmärtäminen ja kehittämällä näistä järjestelmän konseptimalli. Konseptimalli antaa ymmärryksen käyttäjästä, käyttökohteesta ja liiketoiminnallisesta tarkoituksesta. Järjestelmän konsepti mallin tulee sopia käyttäjän näkemykseen, miten toiminnallisuus tulisi tehdä. (Galiz 2007.)

3.2 Ymmärrä käyttöliittymä suunnittelunperusteet ja Kehitä järjestelmän valikot

Hyvin suunnitellusta käyttöliittymästä käyttäjä näkee helposti, mitä hän pystyy käyttöliittymän kautta tekemään. Käyttöliittymä tulee myös suunnitella ottaen huomioon käyttökohteen tekniset rajoitukset. Tässä tulee ymmärtää alustan tuomat rajoitteet missä järjestelmää käytetään sekä mahdolliset käyttäjän rajoitteet. (Galiz 2007.)

Graafiset järjestelmät ja verkkosivut ovat usein valikkopainotteisia. Valikkotoimintojen avulla vaikutetaan objektien ominaisuuksiin, ikkunoihin ja asiakirjoihin. Valikot ovat käyttäjäystävällisiä, koska valikot painottuvat ihmisen kykyyn tunnistaa.

Valikkojen kanssa työskennellessä ihmistä muistutetaan mahdollisista toiminnoista ja asioista, joita he eivät muistaneet tai vielä tienneet. (Galiz 2007.)

3.3 Valitse oikeat toiminnan ohjauselementit käyttöliittymään ja Luo tarkoitukseen sopivaa grafiikkaa

On haasteellista valita oikeanlaiset toiminnallisuuden ohjauselementit. Oikeanlaisten valintojen kanssa on järjestelmän käyttö helppoa sekä sujuvaa. Vääränlaiset valinnat aiheuttavat enemmän virheitä, matalamman tuottavuuden ja useimmiten käyttäjälle negatiivisen kuvan tuotteesta. (Galiz 2007.)

Käyttöliittymän sisältämät kuvakkeet ja kuvat ovat olennainen osa graafista suunnitelmaa. Kuvakkeet on selitetty, sisältäen myös dialogin siitä minkälaisia kuvakkeita on olemassa, mikä vaikuttaa niiden käytettävyyteen ja miten ne tulisi suunnitella merkityksellisiksi ja olennaisiksi. (Galiz 2007.)

3.4 Sivujen ja ikkunoiden organisointi

Sivujen ja ikkunoiden tulisi näyttää oikea määrä informaatiota. Liian vähän informaatiota ei ole tehokasta ja liian paljon informaatiota tekee sivusta sekavan. Näytä kaikki informaatio, jota tarvitaan toiminnon tekemiseen yhdellä sivulla, mikäli mahdollista. (Galiz 2007.)

Hallintatyökalut tulisi asettaa seuraavalla tavalla: Kaikista useimmin käytetyt toiminnot tulisi sijoittaa vasempaan yläkulmaan. Säilytä sujuvuus ylhäältä alas ja vasemmalta oikealle. Mikäli jokin toiminto vaikuttaa toisen toiminnon toimivuuteen tulisi tämä sijoittaa kohteen yläpuolelle tai vasemmalle. Sijoita koko ikkunaan vaikuttavat toiminnot sivun alalaitaan horisontaalisesti keskitettynä. (Galiz 2007.)

Navigoinnissa tulisi välttää mahdollisimman vähän vaadittavaa hiiren liikettä. Tulisi myös minimoida mahdollisimman paljon tarvetta näppäimistön ja hiiren välillä vaihteluun. Nämä saadaan tehtyä tasaamalla ja ryhmittelemällä elementtejä. (Galiz 2007.)

Ryhmät tulee rakentaa niin että saman tyyppiset käyttöliittymä elementit ovat ryhmitetty keskenään. Esimerkiksi radiopainikkeet ja kaksi tai useampi samaan aiheeseen liittyvää tekstikenttää. (Galiz 2007.)

3.5 Käytettävyystestauksen tarkoitus

Käytettävyystestaus luo yhteyden kehittäjien ja käyttäjien välille. Testauksen aikana kehittäjä oppii käyttäjän tarpeita, kysymyksiä ja ongelmia. Testauksen aikana käyttäjälle tulee esiin ohjelmiston toiminnallisuudet. Testaus voi myös nostaa esiin mahdollisia ongelmia suunnitelmassa siinä kohtaa, jolloin niitä on vielä helppo muokata. (Galiz 2007.)

3.6 Rajat

Ryhmittelyä voi parantaa lisäämällä ryhmille omat rajat. Yksittäisen elementin tulisi olla rajattu ohuella rajalla, kun taas monta elementtiä sisältävän ryhmän raja tulisi olla paksu. Elementtien ylhäälle ja alhaalle tulee jättää yhden rivin verran tilaa ennen rajaa. (Galiz 2007.)

3.7 Oikeiden sanojen valinta

Ohjelmistossa olevat termit tulisivat olla mahdollisimman lyhyitä ja ennestään tuttuja sanoja. Kannattaa siis välttää tietoteknisiä tai sellaisia sanoja, joilla on jokin muu merkitys ulkoisessa kontekstissa. Sanojen tulisi myös olla täysimittaisia ja tyypiltään positiivisia. (Galiz 2007.)

3.8 Käyttöliittymätestauksen rajat

Käytettävyystestaus tulisi aloittaa mahdollisimman aikaisessa vaiheessa. Paperillisia tai prototyyppi versioita voidaan luoda ja näiden testien avulla muokata tuotetta. Uudelleen suunniteltu käyttöliittymä tulee testata ja uudelleen suunnitella niin kauan, kunnes halutut tulokset on saavutettu. Testauksen tulisi sisältää mahdollisimman paljon käyttäjän tekemiä tehtäviä ja järjestelmän toiminnallisuuksia kuin mahdollista. (Galiz 2007.)

3.9 Prototyypit

Prototyyppi on yksinkertainen ja keskeneräinen versio, jota käytetään testaukseen. Sen tarkoitus on saada käyttäjältä palautetta ohjelmiston nykyisestä käytettävyydestä. Prototyypin mahdollistaa suunnitelman paremman visualisoinnin käyttäjälle, sekä kertoo miltä valmis ohjelmisto voisi näyttää. (Galiz 2007.)

Prototyypin on tarkoitus toimia simulaationa järjestelmästä, joka voidaan luoda nopeasti. Prototyyppi voi olla nopeasti piirretty paperi piirros tai täysin ohjelmistona toteutettu versio, jossa käyttäjä pystyy syöttämään dataa ja selailemaan valikoita. (Galiz 2007.)

3.10 Värit

Väri lisää näyttöön syvyyttä ja realismia. Väri kiinnittää ihmisen huomiota, koska se houkuttelee katsetta. Oikein käytettynä väri erottaa käyttöliittymä elementtejä toisistaan, luo näytöstä mielenkiintoisemman sekä viehättävämmän. Väärin käytettynä värit voivat olla häiritseviä tai visuaalisesti väsyttäviä ja näin häiritä käyttäjää. (Galiz 2007.)

Taulukko 1. Näkyvä spektri, Värit millimikroneina.

Punainen	700
Oranssi	600
Keltainen	570
Keltainen-vihreä	535
Vihreä	500
Sininen-vihreä	493
Sininen	470
Violetti	400

Kaikki värit eivät ole tasaväkisiä ihmisen silmälle. Silmä havaitsee parhaiten värejä väri spektrumin keskivaiheilta eli keltainen ja vihreä. Huonoiten silmä havaitsee ääripään värejä, kuten punaista tai sinistä. (Galiz 2007.)

Tietynlaiset väriyhdistelmän saattavat myös väsyttää silmää. Värin aaltopituus (Taulukko 1), joka tuottaa sinistä väriä keskittyy silmän verkkokalvon etuosaan, kun taas punainen keskittyy silmän takaosaan. Jatkuva vaihtelu punaisen ja sinisen värin välillä saattaa aiheuttaa silmän väsymistä. (Galiz 2007.)

Värien tulisi olla tasaiset eri näyttöjen välillä. Saman väriset näytöt indikoivat niiden liittyvän toisiinsa. Värien tarkoitus ei tulisi myöskään muuttua näyttöjen välillä. Värien tarkoituksen vaihtaminen aiheuttaa käyttäjässä hämmennystä ja käytössä lisää virhetilanteita. (Galiz 2007.)

4 UNITYN KÄYTTÖLIITTYMÄ

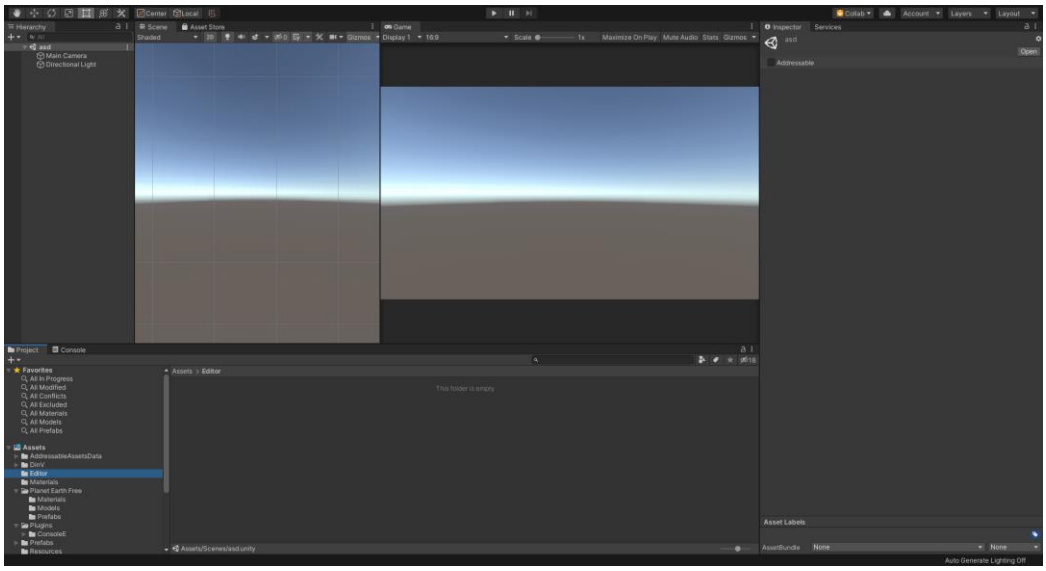
Unityn käyttöliittymä (Kuvio 1) koostuu erilaisista osioista, joilla kaikilla on oma tarkoituksensa. Käyn ne tässä läpi, jolloin muun työn seuraaminen on helpompaa. Tässä esimerkissä käytetään Unityn versiota 2019.4.9f1.

Esimerkki kuvan vasemmassa ylälaidassa on Heirarchy -ikkuna. Hierarchy -ikkuna näyttää tällä hetkellä aktiivisena olevan Scenen sisältävät objektit. Tässä ikkunassa siis hallinnoidaan Scenen sisältäviä objekteja, niiden järjestystä ja ryhmittelyä.

Kuvan alalaidasta löytyy Projektin resurssienhallintaikkuna, jonka kautta päästään käsiksi kaikkiin projektiin kuuluviin materiaaleihin. Samasta ikkunasta löytyy myös Console -niminen välilehti, tätä välilehteä käytetään ohjelman mahdollisten virheviestien näyttämiseen. Console välilehteä voi myös käyttää tulostaakseen koodissa olevia arvoja.

Ruudun keskiosan vasemmalla puolella sijaitsevat Scene ja Asset Store -ikkunat. Tässä ikkunassa näet objektien sijainnin pelimaailmassa. Voit myös käsin säätää niiden kokoa, paikkaa ja suuntaa. Scene ikkuna sisältää paljon hyviä toiminnallisuuksia kuten: Automaattisen tasauksen ja keskityksen objekteja liikutellessa. Asset Store välilehdestä on mahdollista ladata valmiita ilmaisia sekä maksullisia lisäosia omaan projektiin. Ruudun keskiosan oikealta puolelta löytyy Game -ikkuna. Game -ikkuna näyttää sen miltä peli tällä hetkellä näyttäisi käyttäjälle.

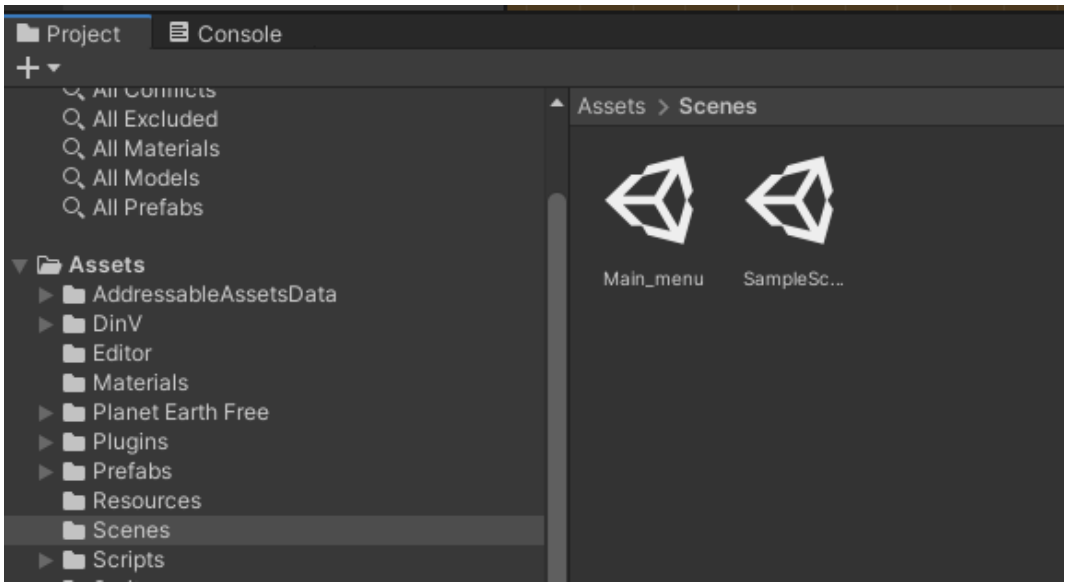
Käyttöliittymän oikeasta laidasta löytyy Inspector -ikkuna, josta säädetään valitun objektin arvoja. Tässä samassa ikkunassa lisätään objekteihin myös komponentit.



Kuvio 1. Unityn käyttöliittymä

5 SCENE

Päävalikon luonti aloitetaan luomalla uusi skene. Uusi Scene luodaan menemällä Unityn alalaidassa olevaan project välilehteen ja navigoimalla Assets > Scenes kansioon (Kuvio 2).



Kuvio 2. Projetin resurssien hallinta

Täällä painamalla oikealla hiiren painikkeella, menemällä Create valikkoon, josta aukeaa Scene mahdollisuus. Mikäli luotu Scene ei aukea automaattisesti voi sen avata tuplaklikkaamalla sitä.

5.1 UI objektit

Kun haluttu Scene on avattu, voidaan vasemmasta laidasta löytyvästä Heirarchy -ikkunasta lisätä siihen halutut käyttöliittymäobjektit. Minimi määrä aloitusvalikon normaaliin toiminnallisuuteen vaadittavat objektit tässä tapauksessa ovat Canvas, Camera, Kaksi kappaletta painikkeita ja yksi täysin tyhjä objekti scriptiä varten. Mahdollista taustakuvaa varten kannattaa lisätä UI -> image objekti.

5.1.1 Tyhjä objekti ja skriptin luonti

Tyhjän objektin ideana on sisältää scripti jonka, funktioita voidaan käsitellä painikkeiden avulla. Painetaan oikealla hiiren painikkeella Hierarchy-ikkunasta ja lisätään siihen tyhjä objekti painamalla Create empty.

Tämän jälkeen luodaan skripti menemällä ruudun alalaidasta Assets / Scripts. Assets > Scripts ikkunassa oikealla hiiren painikkeella painamalla saa Create -valikon, josta löytyy mahdollisuus C# Script. Tämän jälkeen scripti tiedosto avataan ohjelmalla, joka on tarkoitettu ohjelmointiin, esimerkiksi: Visua Studio tai Visual Studio Code ja kirjoitetaan sinne seuraavat funktiot:

```
0 references
public void OnMouseUp()
{

    SceneManager.LoadScene("SampleScene");
}

0 references
public void ExitGame(){

    Application.Quit();
}
```

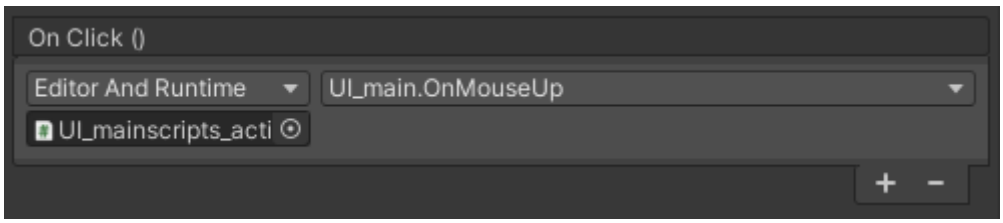
Kuvio 3. Scenen avaamis ja sovelluksen sulkemis scriptit

Mikäli SceneManageria ei suoraan tunnisteta, tulee tiedoston ylhäälle lisätä: `using UnityEngine.SceneManagement;` SceneManager.LoadScene scriptin tarkoitus on ladata haluttu Scene. Mikäli Kuvio 3 skripti linkataan nyt painikkeeseen, vaihtaa se ohjelman Sceneä ”SampleScene” nimiseen sceneen. Heittomerkkien sisään siis laitetaan sen Scenen nimi, johon halutaan siirtyä tätä functiota aktivoidessa.

Kun Scripti on luotu ja tallennettu mennään Unityyn ja valitaan tyhjä objekti Hierarchy-ikkunasta. Kun tyhjä objekti on valittu, mennään sovelluksen oikeassa laidassa sijaitsevaan Inspector -välilehteen ja mennään kohtaan Add Component ja painetaan sitä. Lisätään skripti objektiin valitsemalla script avautuvasta valikosta ja sitten valitsemalla aikaisemmin luotu skripti.

5.1.2 Päävalikon painikkeet

Kun skripti on lisätty tyhjään objektiin, voidaan se linkata painikkeeseen. Tämä tapahtuu valitsemalla haluttu painike Heirarchy -ikkunasta ja menemällä Inspector -ikkunan On Click() kohtaan (Kuvio 4). Painamalla valinta kenttää tai vetämällä objektin Hierarchy- ikkunasta valinta kenttään. Kun objekti on linkattu, tulisi oike-anpuolinen pudotusvalikko aktivoitua. Pudotusvalikosta navigoidaan oman tyhjän objektin nimen kohdalle ja sieltä valitaan se funktio, jonka halutaan aktivoituvan nappia painettaessa.



Kuvio 4. Funktio linkitettynä painikkeeseen

Quit- eli poistumispainike tehdään samalla tavalla. Tosin Quit painikkeen funktio ei toimi editor-moodissa ja sen toiminnallisuus tapahtuu vasta kun peli on kasattu. Pitää muistaa, että iOS käyttöliittymillä kannattaa sovelluksen sulkeminen jättää käyttäjän varaan. iOS käyttöliittymissä ohjelman sulkeminen tällä metodilla saattaa vaikuttaa käyttäjälle kuin ohjelma olisi vain kaatunut. (Unity Technlologies 2020.)

6 PELIN KÄYTTÖLIITTYMÄ

6.1 Tarvittavat lisätyökalut

Ainoana tarvittavana lisätyökaluna käytetään Unityn Addresable Asset työkalua. Addressables Asset työkalun tarkoituksena on mahdollistaa asynkroninen eli rinnakkainen lataus pelissä käytettäville resursseille, käyttäen yksinkertaista tiedosto osoitetta.

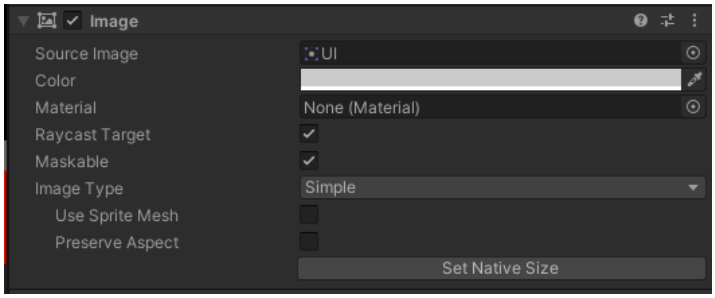
Addressable assets on mahdollista liittää projektiin menemällä Unityn yläpalkissa olevaan Window-valikkoon, valitsemalla Package manager. Tämän jälkeen avautuu Package Manager ikkuna. Ikkunan vasemmasta laidasta valitaan Addressables ja tämän jälkeen Package Manager ikkunan oikeasta alalaidasta install. Asennuksen jälkeen tarvitaan mahdollinen projektin uudelleen avaaminen. Projektia tehdessämme huomasimme, että kaikkien jäsenten tulee ladata tämä työkalu itse.

6.2 Käyttöliittymän tausta

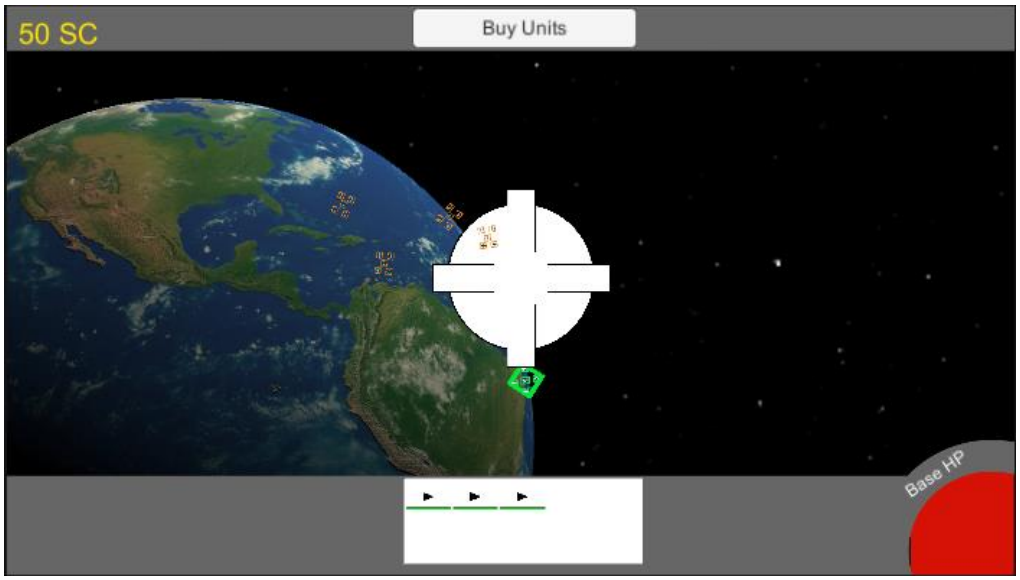
Pelin käyttöliittymällä on hyvä olla tausta, mikäli peliin tulee paljon erilaisia käyttöliittymä elementtejä. Tarkoituksena on siis luoda staattinen kuva, joka istuu pelimaailman päällä, ankkuroituna Canvakseen ja toimii pohjana muille käyttöliittymä objekteille. Tällöin saadaan värimaailmaltaan tasainen alusta eikä pelin värimaailma pääse vaikuttamaan käyttöliittymä objektien näkyvyyteen (Kuvio 7). Taustana kannattaa käyttää PNG-kuvaa, joka tukee läpinäkyvyyttä.

Kuvio 5. Esimerkki PNG kuva

PNG kuva (Kuvio 5) saadaan peliin lisäämällä se pelin resursseihin. Tämän jälkeen voidaan luoda tyhjä objekti Canvakseen, joka sisältää Image-komponentin (Kuvio 6).



Kuvio 6. Objekti, johon on liitetty UI niminen kuva.



Kuvio 7. PNG kuva sijoitettuna peliin

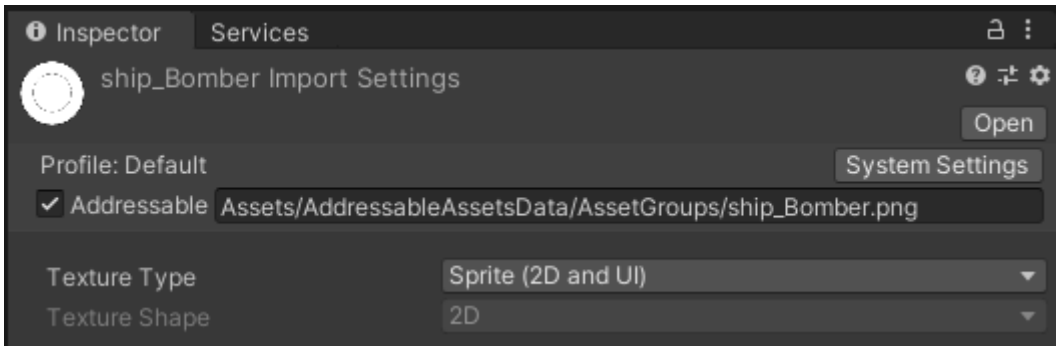
6.3 Kuvien lataaminen Addressable Assets käyttäen

6.3.1 Yleisesti

Addressable Assetsien tarkoitus on mahdollistaa resurssien asynkronisen latauksen, joka mahdollistaa paremman suorituskyvyn. Resursseja on mahdollista ladata joko yksittäin tai ryhmissä. Mikä tahansa resurssi voidaan merkitä Addressalle Assetiksi aina äänistä tekstuureihin ja animaatioihin. (Unity Documentation, 2020)

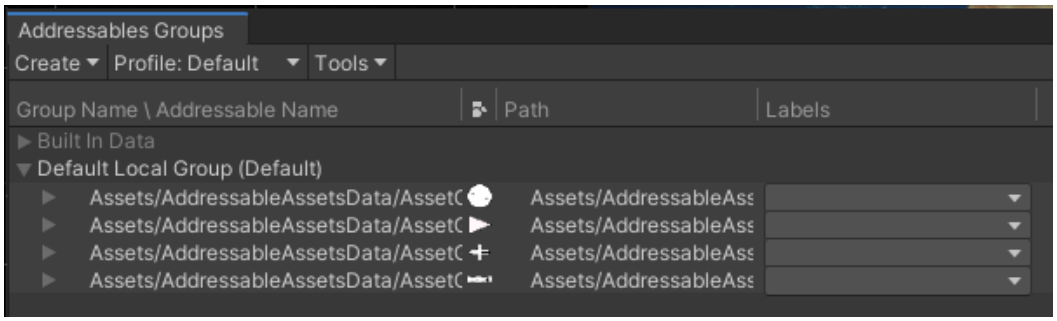
6.3.2 Addressable Assetin luominen

Valitaan haluttu kuva Unityyn liitetyistä kuvista. Inspector -ikkunassa valitaan Addressable kohta (kuvio 8). Tässä kohtaa on myös hyvä tarkistaa, että kuva on laitettu tekstuuri tyypiltään Sprite (2D and UI) koska tekstuuria käytetään 2D-pelissä spriteinä.



Kuvio 8. Addressable Asset Inspector-ikkunassa

Kun resurssiin on liitetty Addressable valinta (Kuvio 8) avataan Addressables Groups ikkuna Tämän ikkuna löytyy ylävalikon Window -> Asset Management -> Addressables -> Groups. Tämän jälkeen vedetään valitut resurssit Default Local Groupin alle (Kuvio 9).



Kuvio 9. Resurssien manageröinti Addressable Asseteissa

6.3.3 Addressable assetin asynchroonine lataaminen koodissa

Aluksi luodaan uusi scripti ja liitetään se tyhjään objektiin kohdan 4.2.1 mukaan. Scriptin alkuun lisätään spriteArray niminen Array muuttuja jonka tarkoituksena on resurssia ladatessa vastaanottaa data komennolla: `public Sprite[] spriteArray;` sekä lista jossa niitä säiltytetään: `public List<Sprite> hold_sprites;`.

Arrayn ja List:in luomisen jälkeen luodaan asynkrooninen funktio (kuvio 10), jota voidaan kutsua ohjelman aloituksessa. Tässä ja seuraavissa esimerkeissä käytän Linq-nimiavaruutta ja sen saa käyttöön lisäämällä scriptin yläosaan: `using System.Linq;`.

```
void LoadSpritesWhenReady(AsyncOperationHandle<Sprite[]> handleToCheck)
{
    if (handleToCheck.Status == AsyncOperationStatus.Succeeded)
    {

        spriteArray = handleToCheck.Result;
        hold_sprites.Add(spriteArray[0]);

    }

}
```

Kuvio 10. Asynkrooninen funktio

Funktion (Kuvio 10) luomisen jälkeen voidaan luoda scriptin void Start(){} funktion alle koodi joka hakee halutut resurssit (Kuvio 11).

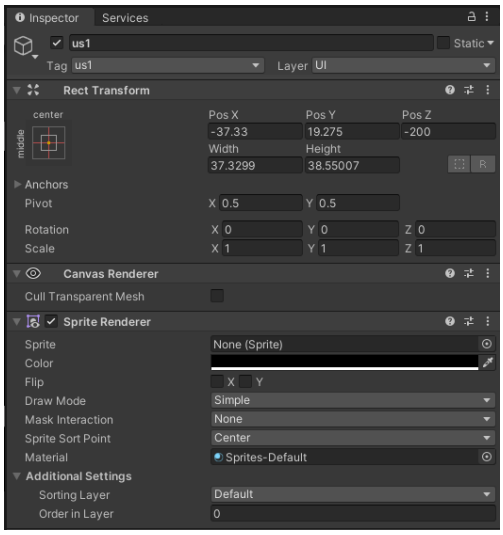
```
AsyncOperationHandle<Sprite[]> spriteHandle = Addressables.LoadAssetAsync<Sprite[]>("Assets/AddressableAssetsData/AssetGroups/ship_Bomber.png");
spriteHandle.Completed += LoadSpritesWhenReady;

spriteHandle = Addressables.LoadAssetAsync<Sprite[]>("Assets/AddressableAssetsData/AssetGroups/ship_fighter.png");
spriteHandle.Completed += LoadSpritesWhenReady;
```

Kuvio 11. Addressable Assets -haku.

Heittomerkkien sisään laitetaan siis sen Addressable assetin tunnus, joka halutaan ladata. Kuvio 11 sisältää myös esimerkin siitä, kuinka voidaan ladata useita eri resursseja peräkkäin.

6.3.4 Addressable assetin kutsuminen ja käyttäminen

Ensimmäiseksi luodaan tyhjä objekti, johon liitetään Sprite Renderer niminen komponentti ja annetaan objektille nimi (kuvio 12).



Kuvio 12. Objekti Sprite Renderer komponentilla.

Kun haluttu objekti on luotu ja siihen on tehty yllä mainitut toimenpiteet, voidaan avata scripti. Scriptin alkuun alustetaan luokka muuttuja komennolla: `public SpriteRenderer spriteRenderer1;` Tämän jälkeen scriptin void Start() {} osion sisään kirjoitetaan komento jossa aikaisemmin luotuun muuttujaan haetaan käytettävä peliobjekti:

```
spriteRenderer1=GameObject.Find("us1").GetComponent<SpriteRenderer>();
```

Näiden toimintojen jälkeen voidaan hakea hold_sprites listasta haluttu sprite tekstuuri Linq nimi avaruutta käyttäen.

```
spriteRenderer1.sprite=hold_sprites.Where(r=>r.name==
"ship_fighter").FirstOrDefault();
```

6.4 Painike logiikka.

Tässä käyn lävitse, miten rakennetaan käyttöliittymään sekuntiperusteinen laskin ja miten sitä voidaan käyttää käyttöliittymä objektien hallinnassa. Tässä kyseisessä projektissa käyttäjälle kertyy rahaa yhden yksikön verran sekunnissa. Rahaa voidaan käyttää ostamaan uusia yksiköitä ja tähän tarvitaan käyttöliittymälogiikkaa, mikäli halutaan estää käyttäjää ostamasta loputtomasti yksiköitä.

Aluksi luodaan tyhjä teksti objekti ja scripti. Avataan scripti ja lisätään tarvittavat muuttujat. Näiden lisäksi luodaan void Start(){} osion sisään rutiini joka alkaa laskemaan rahaa ja hakee peliobjektin(Kuvio 16) johon se sijoitetaan (Kuvio 13).

```
public Text moneytxt;
public int money;
public IEnumerator coroutine;
// Start is called before the first frame update


0 references
void Start()
{
    coroutine = CreateMoney(1.0f);
    StartCoroutine(coroutine);

    moneytxt = GameObject.Find("money").GetComponent<Text>();

}
```

Kuvio 13. Tekstikentän haku ja coroutinen starttaus.

Tämän jälkeen tehdään rutiini funktio (Kuvio 14).

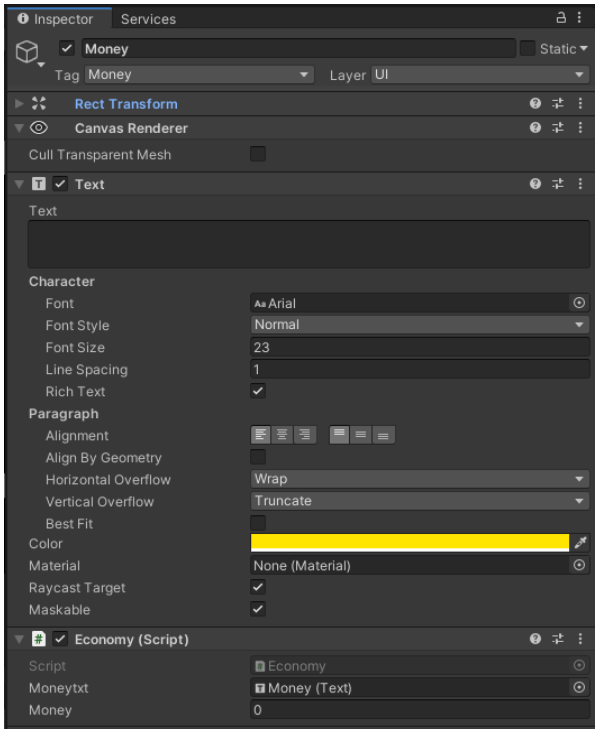
```
1 reference
public IEnumerator CreateMoney(float waitTime)
{
    money = 70;
    while (true)
    {
        yield return new WaitForSeconds(waitTime);
        money++;
    }
}
```

Kuvio 14. Funktio, joka hoitaa rahan tuotannon pelissä.

Näiden jälkeen lisätään void Update(){} osion alle nykyisen rahan määrä ja linkataan scripti teksti objektiin (Kuvio 15).

```
void Update()
{
    moneytxt.text = money.ToString() + " SC";
}
```

Kuvio 15. Nykyisen rahan määrän siirtäminen UI elementtiin.

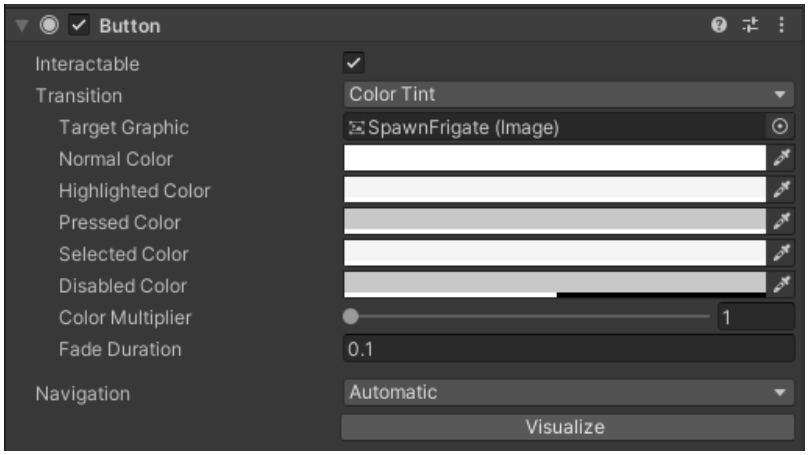


Kuvio 16. Unityn peli objekti, joka näyttää rahan määrän.

Yksiköiden ostamiseen käytetään normaalia painiketta. Painikkeella on ”interactable” attribuutti, jota voidaan vaihtaa sijoittamalla siihen arvo true tai false. Tämä eroaa ”`.gameObject.SetActive`” siten että se ei poista käyttöliittymä elementtiä täysin näkymästä. Alla olevassa Kuviossa 17 painike ”Bomber Squadron” sisältää arvon ”false” Interactable-attribuuttiin.

Kuvio 17. Painike, jossa Interactable-arvo on "false"

Painikkeen Interactable-arvon voi nähdä ja sitä voi vaihtaa Inspector -ikkunasta, kun käyttöliittymä objekti on valittu (Kuvio 18).



Kuvio 18. Painike, jossa Interactable attribuutti on "True"

7 TIEDON SIIRTÄMINEN SKENEJEN VÄLILLÄ

Aluksi luodaan luokka haluttuun scriptiin joka sisältää siirrettävän tiedon (Kuvio 19) ja lisätään nimi avaruus ”`using UnityEngine.SceneManagement;`” Scriptin yläosaan.

```
public static class StaticClass
{
    5 references
    public static string CrossSceneInformation { get; set; }
}
```

Kuvio 19. Luokka, joka säilyttää tiedon.

Tämän jälkeen luodaan funktio, joka asettaa tiedon CrossSceneInformation muuttujaan ja lataa halutun skenen (Kuvio 20).

```
public void DifficultyEasy()
{
    StaticClass.CrossSceneInformation = "Easy";
    SceneManager.LoadScene("Game");
}
```

Kuvio 20. Tiedon tallentava funktio.

Näiden jälkeen siirrytään halutun skenen tässä tapauksessa ”Game” nimisen skenen scriptiin jossa tietoa halutaan käyttää. Alustetaan muuttuja `public string game_difficulty;` ja siirretään tieto tähän (Kuvio 21).

```
game_difficulty = Difficulty.StaticClass.CrossSceneInformation;
```

Kuvio 21. Siirretyn tiedon kutsuminen.

Tässä tapauksessa ”Difficulty” ennen ”StaticClass” osaa tarkoittaa sitä luokkaa, josta tietoa tuodaan (Kuvio 21).

8 LOPPUTULOS

Lopputuloksessa käyn läpi käyttökokemustani uutena käyttäjänä Unitystä, sekä projektin kulusta.

Unity on hyvä työkalu peli ja ohjelmisto kehitykseen tutustuvalle henkilölle. Virallisista dokumentaatioista löytyy hyvin tietoa ja esimerkkejä. Käyttöliittymä itsessään oli minusta hyvin selkeä ja nopeasti opeteltavissa.

Toiminnallisuuksia lisättäessä tulee osata ohjelmoinnin alkeita. Yksinkertaiset asiat olivat koodilla helposti tehtävissä ja scriptit olivat usein tosi lyhyitä. Tarvittavia kolmannen osapuolen kirjastoja tai muita lisätyökaluja ei tarvinnut hakea.

Käyttöliittymää rakentaessa oli muutoksia helppo ja nopea testata. Erilaisia käyttöliittymä tyylejä oli tämän takia vaivatonta rakentaa. Varsinaista suunnitelmaa ei tarvitse rakentaa vaan on hyvin mahdollista kehittää nopealla ja muuttuvalla tyylillä. Skenejen avulla pystyi rakentamaan useita rinnakkaisia versioita samasta näytöstä, mikäli näin halusi tehdä.

Projektissa käytettiin myös Unityn Collaborate-työkalua. Tämä mahdollistaa projektin tekemisen samanaikaisesti useammalle henkilölle. Muutamalla napin painalluksella oli mahdollista ladata toisen henkilön tekemät muutokset projektiin tai lisätä omat muutokset. Työkalun käyttäminen onnistuu, jos ohjelmoijien Unityn käyttöliittymä- ja pelimoottoriversiot ovat samat.

9 YHTEENVETO

Opinnäytetyön tarkoituksena oli suunnitella ja rakentaa käyttöliittymä käyttäen Unityä. Varsinaisia ongelmia ei syntynyt, kunhan löysi oikeat dokumentaatiot. Projektia tehdessä Unityn Collab-työkalu toimi melko moitteettomasti. Ainoa havaittu ongelmakohta tuli, mikäli käyttäjillä oli eri versiot Unitystä.

Käyttöliittymän rakennuksessa itsessään ei tullut mitään suurempia ongelmia, kun ymmärsi Unityn periaatteet. Näyttöjen ja skriptien tekeminen oli helppoa sekä selkeää. Unityn omat dokumentaatiot olivat laajat ja tarjosivat tarvittavan tiedon.

Opinnäytetyö on hyödyllinen, kun rakentaa käyttöliittymää Unityllä. Opinnäytetyö tarjoaa teoreettisen ja käytännön tiedon, kuinka rakentaa käyttöliittymä. Vaikka tutkielman sivussa käytettiin peliä, voi sen sisältöä käyttää myös muissa tarkoituksissa.

Käyttöliittymää pystyi rakentamaan myös siinä järjestyksessä kuin itse halusi. Mikäli jakoi peliä järkeviin osiin, pystyi sen käyttöliittymää rakentamaan haluamallaan logiikalla.

Yritin kerätä opinnäytetyöhön mahdollisimman paljon tietoa käyttöliittymä elementtien hallintaan liittyvää materiaalia. Rajasin tutkimuksesta siis pois perus ohjelmisto logiikkaa pitääkseni sen mahdollisimman helppoluikuisena ja monikäyttöisenä.

LÄHTEET

McCown, F. 2017. COMP 445 – GUI Programming Harding University. Viitattu 30.3.2020. https://www.harding.edu/fmccown/gui/history-gui.pptx

Oryzo 2020. User Interface & User Experience Design | Oryzo | Small Business UI/UX. Viitattu 31.3.2020. https://oryzo.com/user-interface-design/

Unity Technologies. 2020. Unity Documentation. Viitattu 21.9.2020. https://docs.unity3d.com/ScriptReference/Application.Quit.html

Galitz, W. O. 2017. The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques, 3rd Edition. Viitattu 27.9.2020. https://learning.oreilly.com/library/view/the-essential-guide/9780470053423/