LAPIN AMK
Lapland University of Applied Sciences

SUOMENKIELINEN KOULUTUSMATERIAALI UNITY 3D -PELIMOOTTORILLE

Pyykönen Janne

Opinnäytetyö

Tieto- ja viestintätekniikka
Insinööri (AMK)

2022

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tieto- ja viestintätekniikka
Insinööri (AMK)

Tekijä	Janne Pyykönen	Vuosi	2022
Ohjaaja	Toni Westerlund		
Toimeksiantaja	FrostBit-laboratorio		
Työn nimi	Suomenkielinen koulutusmateriaali Unity 3D -pelimoottorille		
Sivu- ja liitesivumäärä	42 + 4		

Opinnäytetyössä toteutettiin Unity 3D -pelimoottorille suomenkielinen koulutusmateriaali, myös niille joilla ei ole aikaisempaa ohjelmointi- tai pelimoottorikokemusta. Tarkoituksena on madaltaa kynnystä oppia Unity 3D:n käyttöä ja ohjelmointia.

Toteutettu koulutusmateriaali julkaistaan FrostBit-laboratorion verkkosivuilla. Ohjeistuksen toteuttamisessa käytettiin WordPress-julkaisujärjestelmää. Ongelmaksi osoittautui se, ettei siihen voinut lisätä videoita, vain kuvia. Myös aiheiden järjestys perehdytyksessä oli haastavaa.

Tutoriaali koostuu teksteistä, selittävistä kuvioista ja nettisivujen linkeistä. Kuvioita on hyvin paljon ja linkkien tiedot lyhyitä. Suurin osa ohjelmoinnista perustuu näihin lähteisiin.

Tutoriaalin lopussa tehtiin pieni peli. Siinä tehtiin visuaaliset osiot ja animaatioiden ohjaus ohjelmoimalla sekä eri toimintojen ohjelmointia.

Avainsanat peliobjekti, tutoriaali, Unity 3D, WordPress

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Study Programme in Information'and Communication Technology
Bachelor of Engineering

Author Janne Pyykönen Year 2022
Supervisor Toni Westerlund
Commissioned by Lapland University of Applied Sciences
Subject of thesis Finnish Education Material for Unity 3D Game Engine (FrostBit Laboratory)
Number of pages 42+4

The aim of this thesis study was to create a Finnish education tutorial for Unity 3D game engine. The material was aimed at everyone, even those who do not know anything about programming and/or making games with Unity 3D.

The tutorial was planned to lower the threshold to learn about using Unity 3D and programming and even to inspire to study information technology. The tutorial was published on the websites of Frostbit using Word Press program. The problem turned out to be that videos could not be used, only pictures. The order of different topics was also challenging.

As a result, the tutorial consists of text, explaining figures and links to websites. There are many figures and information on the websites is presented briefly. Most of the programming is based on these links. The theory is mostly presented in this thesis. In the end of the tutorial there is a small game which teaches visual things, animation controlling by programming and different functions.

Key words Unity 3D, WordPress, game object, tutorial

SISÄLLYS

ALKUSANAT

Haluan kiittää Unity 3D -ohjelman tekijöitä ja yritystä; ne mahdollistivat tämän tutoriaalin kaikille käytännön ohjeillaan ja ohjelmoinnillaan. Tein myös Unityn nettisivun tutoriaalin noin vuosi sitten harjoittelussa. Tämä auttoi huomattavasti opinnäytetyön tekemisessä. Myös Moakley Brianin ja Nitin nettisivuista oli todella paljon hyötyä.

Haluan myös kiittää Frostbitin ohjaajia Toni Westerlundia, Petri Nissiä ja Pertti Rauhalaa; heiltä sain hyviä neuvoja ja kannustusta työn loppuun saattamiseksi. FrosBitin ulkopuolelta hyödyksi olivat myös Tuija Haapasalmen neuvot kirjalliseen osaan. Sirpa Torvinen oli hyvin kannustava. Kiitän erityisesti Toni Westerlundia ja Tuija Haapasalmea. Lopuksi haluan kiittää Piia Koskelaa. Hän muokkasi tutoriaalin nettisivuille ja antoi hyviä ehdotuksia opinnäytetyön kehittämiseen.

1 JOHDANTO

Opinnäytetyön alussa täytyi ottaa selvää tarvittavien ohjelmien (esimerkiksi Unity 3D) toiminnoista, jotta pystyin arvioimaan niiden kestoa projektisuunnitelmassa. Suunnittelin aluksi ohjeet, joilla käyttäjä voi oppia kaikki avoinna olevat toiminnot Unity 3D -pelimoottorissa. Nämä olivat helpoimpia asioita.

Perehdyin eri ohjelmien käyttämiseen projektissa. Aloitin kuvanmuokkausohjelmista, kuten Paint.net ja Snipping Tool. Piti myös tutustua Gitin ja Github Desktopin käyttöön. Gitin ja Github desktopin käyttämistä varten katsoin Youtube-videoita.

Suunnitelman teossa oli hyvin aikaa. Aloitin työn tekemisen suunnittelemalla sen Microsoft Word -tiedostoon. Lopullisen rakenteen tein Microsoft Exelin Gantt -tiedoston pohjalle. Siinä näkyvät suunnitellut opinnäytetyön sprintit, joiden välillä osioiden tulee olla tehtynä.

Varsinaisen toimeksiannon kohdalla oli aluksi vaativaa selittää perehdytettävät asiat. Kun nämä oli kuitenkin tehty, muodostui haasteeksi nettisivuille laitettavan tekstin rakenne. Lähinnä se, missä järjestyksessä asiat tulivat käsitellyiksi, jotkin asiat piti olla selitettyinä ennen toista.

Unity 3D -pelimoottorin vaihtoehdoista piti selittää paljon, esimerkiksi peliobjektit, colliderit ja triggerit. Lisäksi Inspector-valikosta löytyvät peliobjektin komponentit, joita ovat transform, collider, material, rigidbody ja koodi. Näitä kaikkia käytettiin ohjelmoinnissa kutsumalla niitä.

Ohjelmoinnissa käytiin läpi lyhyitä ja yksinkertaisia koodeja. Niissä käytettiin yksinkertaista laskentaa, 3D-maailman arvoja ja ohjaamista näppäimistöllä. Näissä käytettiin if()-ehtolauseita ja for()-looppeja sekä yleisiä Start()- ja Update()-metodeja. Samantyyppisiä käytettyjä metodeja olivat OnCollisionEnter(), OnCollisionExit(), OnTriggerEnter() ja OnTriggerExit(). Käytettyjä Unityn metodeja olivat Transform.Translate, Input.GetKey, Rigidbody.AddForce, Vector3, Instantiate.Transform ja Random.Range.

Lopuksi tehtiin peli, jossa oli tavoitteita, joita kaikkia ei kuitenkaan toteutettu. Vain helpoimmin ymmärrettävät osat tehtiin, kuitenkin niin, että saatiin toimiva ja innostava peli. Pelissä liikutaan 3D-ympäristössä ja ammuskellaan Ghost-peliobjekteja. Se sisältää myös animaatioiden käyttöä.

Opinnäytetyön raportti koostuu samoista asioista, mutta kuvia on vähemmän ja teksti on teoreettista. Tarkoituksena ei ole edetä askel askeleelta, vaan aihepiireittäin. Kuitenkin tarkoitus on edetä aiheittain, niin että se vastaisi toimeksiantoa, tai muuten loogisesti.

2 OHJELMAT

Opinnäytetyössä käytettiin FrostBitin versionhallintaa. Tämän kautta voitiin ladata kaikki tiedostot GitHub-työpöytäohjelmalla. Tämä tapahtui lisäämällä projektin URL GitHub-työpöytäohjelmaan ja kloonaamalla projekti tietokoneelle.

Käyttämällä GitHub-työpöytäohjelmaa ei tarvita terminaalin komentoja. Sitä käytetään koodin versionhallinnassa, koodit ja tekstit tallentuvat sinne. Muutokset eri tekstien ja koodien vaiheissa näkyvät myös siellä.

Kaikki tiedostot tallennettiin OneDrive-pilvipalveluun, jonne laitettiin isot tiedostot, kuten Unity 3D -projektit ja Build-tiedostot sekä kuvat. Näitä ei laiteta muualle kuin mahdollisesti kovalevylle ja USB-muistitikulle.

Kaikki koodit luotiin ja avattiin Visual Studio Code -ohjelmalla. Visual Studio Code -ohjelmalla voidaan kirjoittaa C#-kooditiedostoja, joita käytetään Unity 3D -pelimoottorissa.

Paint-, Paint.net- ja Snipping tool -ohjelmilla tehtiin pääsääntöisesti perehdytykseen ja opinnäytetyöhön neuvoa antavat kuviot. Ohjelmien avulla kohdistettiin kuvia pienemmiksi ja havainnoitiin tärkeät valikot. Voitiin esimerkiksi rajata jonkin perehdytyksessä valittu kohta. Aina ei voitu käyttää Snipping Tool -työkalua, silloin käytettiin Print Screen -toimintoa ja leikattiin Paint.net-ohjelmalla.

Microsoft Word on tekstinkäsittelyohjelma. Microsoft Word -ohjelman avulla pystyi myös hahmottamaan toimeksiannon. Opinnäytetyöhön lisättiin myös kuvia, mutta ei niin paljon kuin toimeksiannossa. Toinen käytetty Microsoftin ohjelma oli Excel, joka on taulukko-ohjelma. Opinnäytetyössä sitä käytettiin Gantt-kaavioon ja ajanseurantaan.

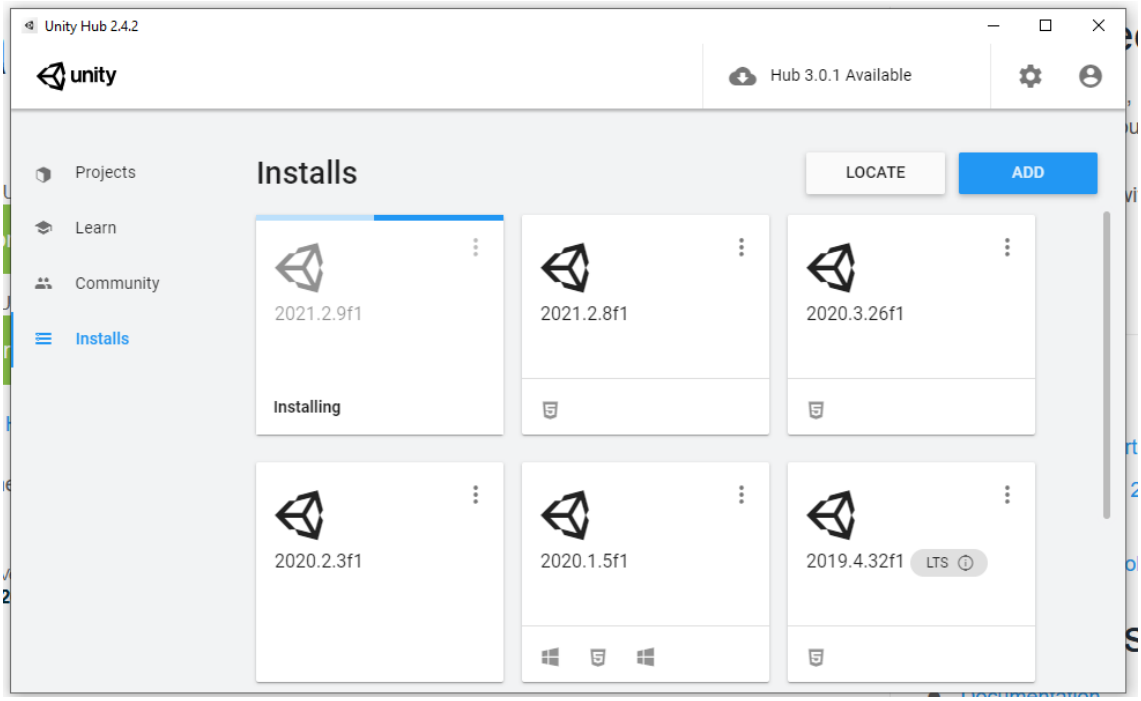
Unity 3D -ohjelmalla toteutettiin tutoriaali. Ohjelmassa pinottiin yhteen koodit, materiaalit, GameObjectit ja lopuksi peli. Käyttäjä käyttää perehdytyksessä tätä ohjelmaa.

3 KEHITYSYMPÄRISTÖ

3.1 Unity 3D -ohjelmien asentaminen

Aluksi pitää mennä Unityn verkkosivuille. Täältä täytyy valita Download for Windows, tämän jälkeen tallennetaan *UnityHubSetup.exe*. Sen jälkeen ladataan taas Unity Hub -ohjelma. Asennusprosessin aikana on seurattava ohjeita Unity-version valinnassa ja lataamisessa, versio voi olla uusin tai jokin muu versio. (Unity Technologies 2022a.)

Asennetut Unity 3D -ohjelman versiot näkyvät Unity Hubissa (Kuvio 1). Uusin niistä on kuviossa asentumassa, ja paras vaihtoehto on uusin Long Time Support (LTS). Jos on asennettuna juuri uusin versio, ei sitä tarvitse enää päivittää. Versiota voidaan vaihtaa, jos halutaan LTS-versio. LTS tarkoittaa sitä, että versio on hyvä pidemmän aikaa. Se on hyödyllistä, jos käytettävä projekti kestää pitkään. Oikeassa alakulmassa on yksi ladatun Unity 3D:n LTS-versio. Kaikkia versioita voi käyttää, kun luodaan projektia.



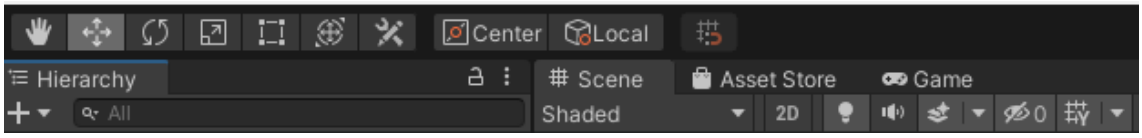
Kuvio 1. Unity 3D -versioita

Lisäksi tarvitaan Visual Studio Code -ohjelma itse ohjelmointia varten. Käyttäjän pitää valita käyttöjärjestelmä. Se on yleensä Windows. (Microsoft 2022.)

3.2 Tutustuminen Unity 3D -pelimoottoriin

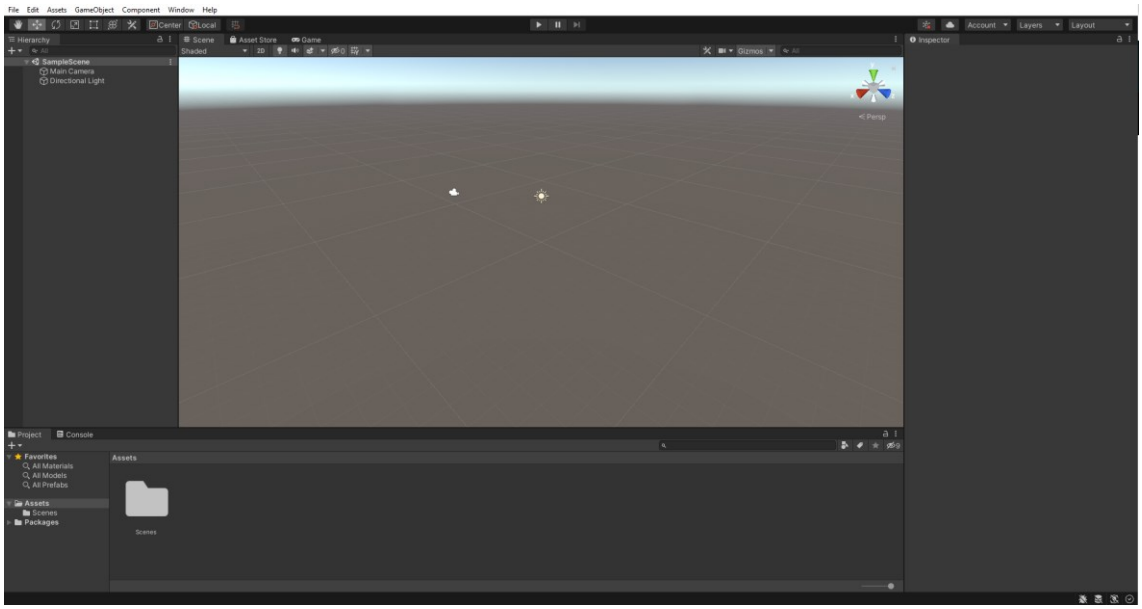
Projektin aloittaessa on edessä tyhjä scene, jossa on vain main camera ja directional light. Tällaisia voi luoda enemmänkin ja valittuna on new scene. Siellä voi myös tallentaa scenen ja projektin. Valittuun sceneen voi lisätä myös cuben tai muun GameObjectin.

Hahmottamisen avuksi on suositeltavaa pyörittää scene-näkymää jatkossa olevin neuvoin. Hanskan kuvassa voidaan mennä tilaan, jossa pyöritetään näkymää pisteessä, jossa ollaan, ja neljässä nuolessa voi siirtää kappaleita 3D-tilassa. Ympyrässä on mahdollista kiertää kappaletta sen mukaisessa paikassa. Neliössä suurennetaan kappaletta koordinaateissa x, y, tai z. Keskeltä voi venyttää kaikkia 3D-arvoja samassa skaalassa. Kuudennessa kohdassa voidaan yhtä aikaa muokata ja skaalata haluttuun kokoon ja pyörittää kappaletta haluttuun asentoon. Sen jälkeisessä kuvassa muokataan kappaleen collideria. Kaikki nämä kappaleessa mainitut toiminnot näkyvät kuviossa 2.



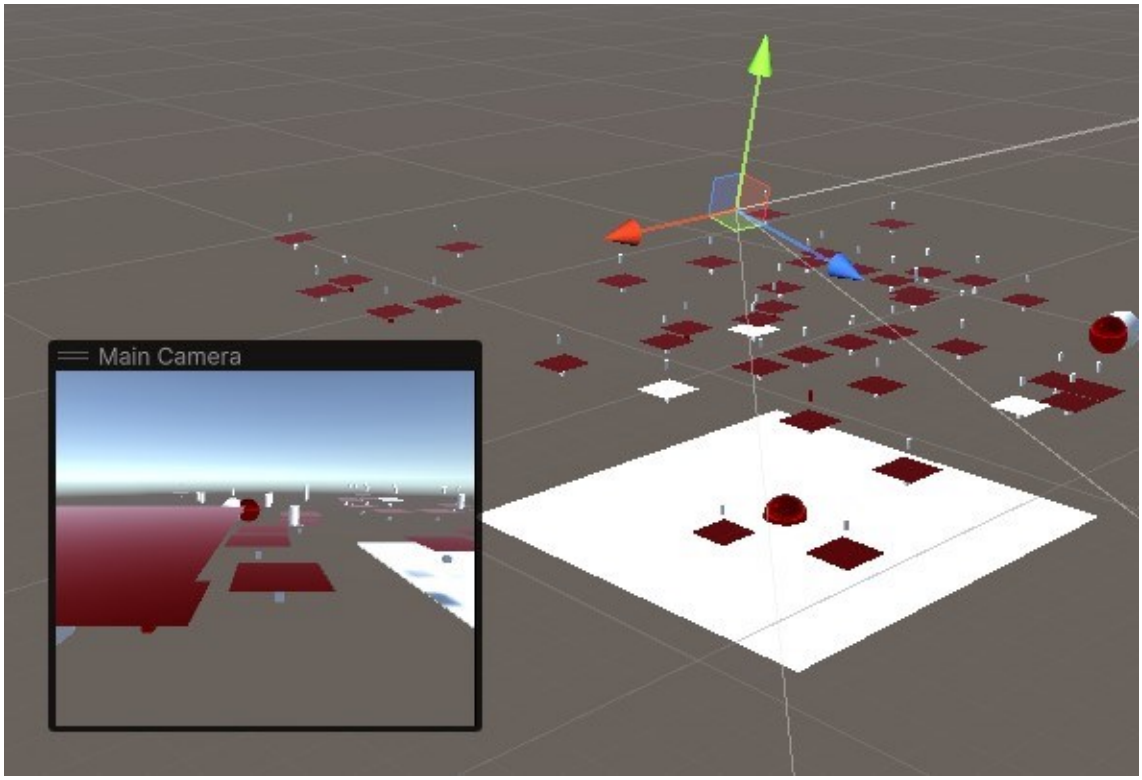
Kuvio 2. Navigointivalinnat

Layers-valintoja ovat default (Kuvio 3), tall, wide, 2 by 3 ja 4 split. Näillä voi valita, missä muodossa kaikki valintaikkunat näkyvät: scene, game, hierarchy, project ja inspector. Console näkyy oletuksena vain Default Layerillä, joka on kuviossa 3. Layers-valintaikkuna löytyy pelimoottorin oikeasta yläkulmasta.



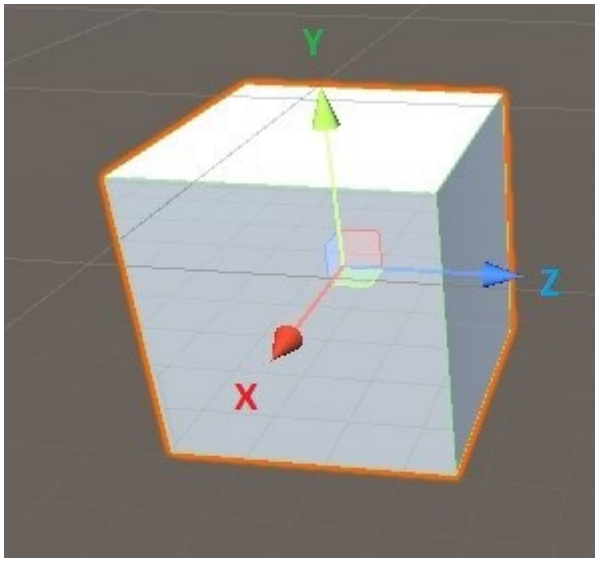
Kuvio 3. Layer Default

Game view -osiossa näkyy pelin näkymä, joka on main cameran suunnasta tuleva kuva kuviossa 4. Game view -esikatselu tapahtuu scene viewin sisällä, ja game view -näkymää käyttämällä voi kokeilla varsinaista pelin näkymää. Näin ei tarvitse aina ajaa varsinaista peliä play-tilassa.



Kuvio 4. Game view scenessä

Koordinaateissa, joissa x on leveyssuunta, arvot ovat positiivisia oikealla ja vasemmalla, jos ne ovat negatiivisia. Pelissä pelaajan katsomat koordinaatit ovat local, nollapiste on GameObjectin keskipisteessä. Koordinaattijärjestelmän ollessa global-tilassa keskipiste on 3D-avaruuden keskipisteessä, tässä kohdassa siksi, että pelaajan näkymä on suunnassa z. Koordinaatit x, y ja z näkyvät kuviossa 5 kuution sisällä ja koordinaatti z on siis lähellä tai kaukana kamerasta, x taas oikealla tai vasemmalla. Koordinaatti y määrittää korkeuden, ylöspäin tai alaspäin.



Kuvio 5. Unity 3D -koordinaatit

Katsotaan, miten eri suuntien koordinaatit muuttuvat. Esimerkiksi eteenpäin (forward) arvoilla (x = 0, y=0 ja z = 1) vain z kasvaa. Kun z=0 niin se on vakio, sen arvo ei siis muutu. Negatiivisena (x=0, y=0 ja z=-1) se kulkee taaksepäin (back). (Unity Technologies 2022b.)

Forward kulkee eteenpäin z-vektoria ja back taaksepäin sitä. Mennessään vektoria right päin liikutaan positiivisesti x:n mukaan, mennessä vektoria left päin liikkuessa x-arvo taas on negatiivinen. Ylöspäin mennessä (up) kuljetaan y-vektoria positiiviseen suuntaan ja alas (down) taas negatiivisesti. Tilassa one lisätään kaikkia vektoreita yhdellä ja zero taas on nollapisteessä tai ei liiku ollenkaan. Kolmessa viimeisessä syötetään suorat x-, y- ja z-arvot suoraan. Nämä kaikki näkyvät kuviossa 6. (Unity Technologies 2022b.)

Static Properties

back	Shorthand for writing Vector3(0, 0, -1).
down	Shorthand for writing Vector3(0, -1, 0).
forward	Shorthand for writing Vector3(0, 0, 1).
left	Shorthand for writing Vector3(-1, 0, 0).
negativeInfinity	Shorthand for writing Vector3(float.NegativeInfinity, float.NegativeInfinity, float.NegativeInfinity).
one	Shorthand for writing Vector3(1, 1, 1).
positiveInfinity	Shorthand for writing Vector3(float.PositiveInfinity, float.PositiveInfinity, float.PositiveInfinity).
right	Shorthand for writing Vector3(1, 0, 0).
up	Shorthand for writing Vector3(0, 1, 0).
zero	Shorthand for writing Vector3(0, 0, 0).

Properties

magnitude	Returns the length of this vector (Read Only).
normalized	Returns this vector with a magnitude of 1 (Read Only).
sqrMagnitude	Returns the squared length of this vector (Read Only).
this[int]	Access the x, y, z components using [0], [1], [2] respectively.
x	X component of the vector.
y	Y component of the vector.
z	Z component of the vector.

Kuvio 6. Käytettävissä olevat vektoriarvot (Unity Technologies 2022a)

3.3 Ohjelmien testaaminen

Seuraavaksi tehdään ohjelman nopea testaus. Tämä löytyy ylhäältä keskiosassa ruutua, toimintoina play ja pause. Vähemmän oleellinen toiminto on oikealla, eteneminen yksi askel kerrallaan.

Kuvion 7 ehto on metodin Update() sisällä. Ohjelma sulkeutuu, kun käyttäjä painaa välilyöntipainiketta. Tämä määritetään kuvion 7 koodiriveissä. Varsinainen pelin paketointi ja ajaminen onnistuu Build and run -toiminnolla, joka löytyy filen alta. Tätä voi kutsua ajamiseksi. Ennen pelin päälle laittamista pitää olla koodi, jolla mahdollistetaan poistuminen run-tilasta. Hyvä paikka koodille on tyhjä objekti (oikea hiirin nappi / create empty).

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour
{
    void Update()
    {
        if(Input.GetKeyDown(KeyCode.Space))
         Application.Quit();
    }
}
```

Kuvio 7. Poistuminen pelistä (Unity Technologies 2022c)

3.4 Build-asetukset

Build-asetuksetkin ovat vasemmassa yläkulmassa valikossa file. Sen alla on build and run, jonka käynnistyksen jälkeen voi ajaa pelin nykymuodossaan. Valittuna build settings tulee valikko, jossa lukee vasemmassa yläkulmassa build settings. Tässä valikossa ovat vaihtoehdot. Käyttöjärjestelmänä on yleensä Windows. Prosessiarkkitehtuuri on nykyään 64-bittinen.

Valittava nappi player settings sisältää project settingsin, tärkeimpänä on input manager. Sieltä voi katsoa horizontal- ja vertical-asetusten tiedot. Oikealta alakulmasta löytyy kaksi nappia, toisessa on mahdollista valita build ja toisessa build and run. Build tarkoittaa pelin rakentamista ja run sen pyörittämistä.

3.5 Ohjelmoinnin perusrakenteet

Koodeja tehdessä on Unityssä hyvä luoda scripts-niminen kansio ja laittaa sen sisälle kaikki koodit. Siellä oikeaa hiiren nappia painaen create ja sen sisältä c# script. Kuviossa olevaa koodia muokkaamalla saadaan luotua muitakin lyhyitä koodeja, perusrakenne on sama kuin kuviossa 7.

Tehdäkseen muutoksia koodiin Unity 3D:tä käytettäessä pitää koodiin lisätä ”using UnityEngine”. Tämän lisäksi pitää c#-koodissa olla ”using System.Collections;”- ja ”using System.Collections.Generic;”-kirjastot. Toteutettujen luokkien, jotka halutaan lisätä Unity 3D -pelimoottorissa peliobjektin komponentiksi, tulee

periytyä monobehaviour-luokasta. Koodill pitää olla tämä ”class”, joka sisältää koodin.

Metodi ”void Start()” tapahtuu kerran ohjelman käynnistyessä. Tässä osiossa käytetään ”void Update()”-metodia, joka pyörii loputtomasti ohjelman ollessa päällä. Alla olevassa koodissa näkyy edellä mainitut metodit, molemmissa näkyy aaltosulut ”{” ja ”}”. Näiden välissä ovat komennot, jotka suoritetaan. Edellisessä kuviossa poistumiskoodissa ”if()” ei tarvitse aaltosulkuja, koska komentoja on vain yksi rivi, joka on ”Application.Quit();”-komento. Muuten ehtolauseessa aaltosulut toimivat samalla lailla kuin metodeissa, class-luokissa tai vastaavissa. Nämä ja komennot ovat aina aaltosulkujen välissä. Tämänkin kappaleen rakenne näkyy kuvioissa 7 ja 8. (Unity Technologies 2022d.)

Aaltosulku ”{” aloittaa toiminnat oikealta vasemmalle tai näyttäen siltä, että ne menevät ylhäältä alas. Aaltosulku ”}” taas sulkee toiminnot kaikkein viimeisenä. Esimerkiksi ”Update()”-metodi jatkaa alusta: ”{”, kun koodi on saapunut loppuun, lisätään: ”}”. Ne näkyvät myös kuvioissa 7 ja 8. (Unity Technologies 2022d.)

3.6 Ohjelmoinnin transform-toimintoja

Tämä koodi luodaan samaan assets-alkuvalikkoon, jossa materiaalit ovat. Se on objektissa tai objekteissa. Kuviossa 8 objekti kulkee eteen ja ylöspäin loputtomasti yksi koordinaatti sekunnissa, koska se sijaitsee ”void Update()” sisällä. Käytössä suuntia on monia, koodit voi kommentoida pois käytöstä, jolloin kohde ei esimerkiksi mene ylöspäin (vector3.up). Voidaan myös vaihtaa ensimmäiseen riviin vector3.back tai toiseen vector3.down. (Unity Technologies 2022d.)

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

public class Koodi_Transform : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        transform.Translate(Vector3.forward * Time. Time.deltaTime);
        transform.Translate(Vector3.up * Time.deltaTime, Space.World);
    }
}
```

Kuvio 8. Liike viistosti (Unity Technologies 2022d)

Kuvion 9 riveissä on erona liikkumisen suunta ja toisella koodirivillä liikkuminen 3D-maailmassa, sen omien 3D-arvojen sijaan. Kun space.world ei ole lisättynä transform.translate()-metodiin, niin se liikkuu omien x-, y- ja z-akselien halutun muutoksen mukaan. Kun se on lisätty, liikkuu kohde 3D-maailman akselien mukaan.

Kuviossa 9 gameobjekti kulkee z-akselia kohteessa ja y-akselia avaruudessa koordinaatin sekunnissa. Koodi voidaan liittää myös toiseen GameObjectiin. Niissä pystyy näkemään koodin liikkeen.

```
void Update()
{
    transform.Translate(0,0, Time.deltaTime);
    transform.Translate(0, Time.deltaTime, 0, Space.World);
}
```

Kuvio 9. Samantyyppinen liike (Unity Technologies 2022d)

Tässä koodissa on käytetty suoria 3D transform -koordinaatteja. Arvot ovat Transform.translate(x,y,z)-muodossa. Toinen vaihtoehto on Addforce(), jolloin li-

sättäisiin rigidbodyyn voimaa ja liikuttaisiin näin 3D-avaruudessa. Tästä kerrotaan lisää luvussa ”5.1 Fysiikka”. Tähän pystyy myös lisäämään transform-tyyppisen liikkeen x-koordinaatin mukaan (Unity Technologies 2022e). Näissäkin erona on koordinaattien määräytyminen, koordinaatissa x kohde liikkuu oikealla.

Kameran z-arvo on negatiivinen kameran ollessa kohteen takana. Mikä tahansa muuttuja voidaan muuttaa negatiiviseksi lisäämällä miinusmerkki muuttujan arvon eteen. Näin käy myös maailmassa kameran ollessa kohteen takana. Kameran z-arvo on negatiivinen luku, muuttuja voidaan muuttaa negatiiviseksi lisäämällä miinusmerkki sen eteen. Koodissa sitä voi hyödyntää käyttämällä sitä time.deltatimen edessä. Silloin x-koordinaatissa kohde liikkuu vasemmalle, samoin koordinaatissa y kohde liikkuu alas ja z taaksepäin. Jos molemmat luvut ovat negatiivisia ja ne kerrotaan, tulee niistä positiivinen luku matemaattisten sääntöjen mukaisesti.

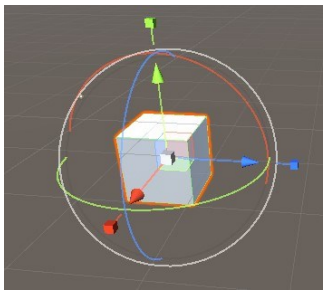
Liikkuminen näppäimistöllä toimii samoin kuin koodin lopetus (application.quit()). Painamalla jotain nuolinäppäintä siirrytään siihen suuntaan, kuin on ohjelmoitu. Kuviossa 10 mennään ylös menevästä nuolesta UpArrow eteenpäin kohdassa vector3.forward. Taakse mennään käyttämällä DownArrowia ja vector3.backia. (Unity Technologies 2022c; 2022m.)

```
if(Input.GetKey(KeyCode.UpArrow))
{
    this.transform.Translate(Vector3.forward * Time.deltaTime);
}
```

Kuvio 10. Liikkuminen näppäimellä input.getkey() (Unity Technologies 2022c; 2022d)

Kuviosta 11 näkee hyvin rotate()-arvot eli miten akselien ympäri kierretään. Lisäksi koko kappaleen havainnointi näkyy samassa kuviossa 11. Kaikkia pyörimissuuntia katsotaan kamerasta päin. Akselin y ympäri kierrettäessä kierretään 3D-maailmaa horisontaalisesti, positiivisesti oikealle ja negatiivisesti vasemmalle. Akselin x ympäri pyörittäessä kuljetaan rengasmaisesti koordinaatin keskiössä positiivisella arvolla eteenpäin ja negatiivisella taaksepäin. Akseli z pyörii

kuin kello edestäpäin katsottuna, positiivisella arvolla ajassa eteenpäin ja negatiivisella arvolla ajassa taaksepäin. Akselien yhtäaikaista käyttöä ei käydä tässä työssä läpi. Kuvion 11 ympyröiden ulkopuolella olevilla erivärisillä kuutioilla x, y ja z voi skaalata (scale) peliobjektia. Hiiren rullalla taas voi skaalata objektia niin, että arvot säilyvät oikeassa suhteessa toisiinsa. Nuolilla voi muuttaa kohteen paikkaa 3D-ympäristössä.



Kuvio 11. Transform tool

Samanlaisella ehdolla saa objektin kääntymään y-akselin mukaan, mikä toteutuu käyttämällä sitä koodissa vector3(0,10,0). Tämä siis toteutuu laittamalla transform.rotaten transform.translaten tilalle. Koodin time.deltatime tilalla on asteiden arvojen pyörimisen nopeus. Nämä näkyvät kuviossa 12 ja tässä tapauksessa oikean ja vasemman inputin ehtolauseen tilalla. Pyöriminen vasemmalle saa negatiivisen arvon -y ja oikealle positiivisen arvon y. Nämä toteutuvat painaen vasenta ja oikeaa näppäintä, niin kuin kuviossa 10. Käyttämällä transform.Rotate() ovat kaikki arvot asteissa. (Unity Technologies 2022f.)

```
this.transform.Rotate(Vector3.up, 10);
```

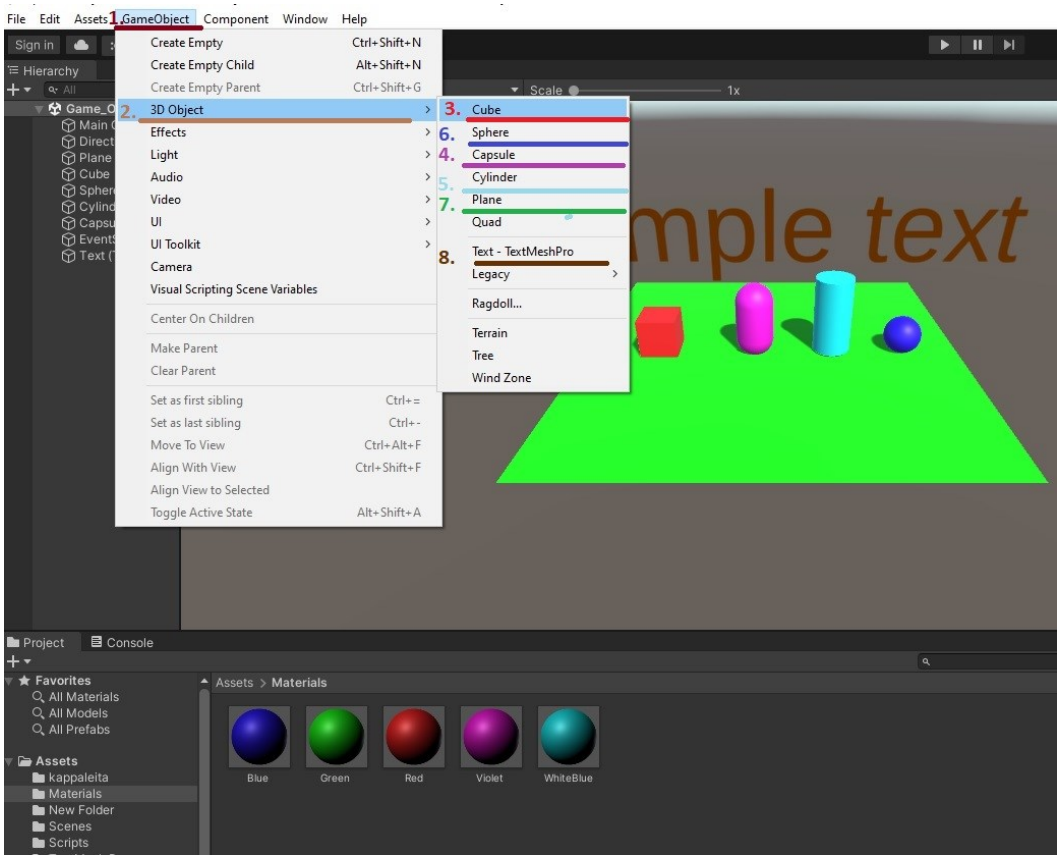
Kuvio 12. Horisontaalisesti oikealle kääntyminen (Unity Technologies 2022f)

4 PELIOBJEKTIT

4.1 3D-objektit, prefabit ja materiaalit

Merkitykselliset objektit löytyvät valikosta kohdasta 1. GameObject ja 2. 3D Object. Ne näkyvät visuaalisesti kuviossa 13. Geometrisistä muodoista käytännöllisimpiä ovat seuraavat: 3. cube (kuutio), 4. capsule (kapseli), 5. sylinder (sylinteri), 6. sphere (pallo) ja 7. plane(taso). Numerot viittaavat kuvion 13 peliobjektien luontikohtiin. Ne on merkitty niiden materiaalien väreillä, materiaalit löytyvät alhaalta kuviossa 13, pallon näköisinä kuvakkeina, kansiossa materials. Samat luetellut muodot löytyvät myös collider-, sylinder- ja plane-muotoja lukuunottamatta.

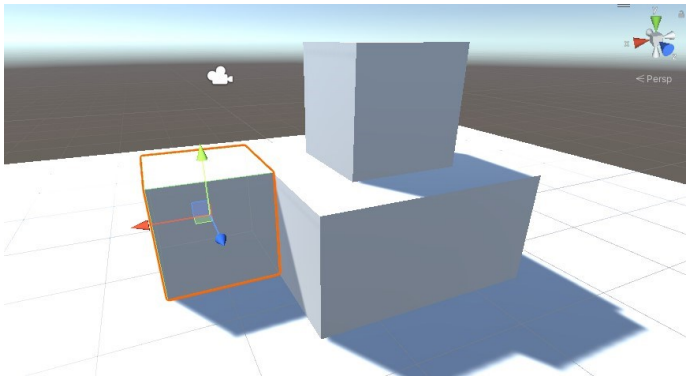
Kuvion 13 kohdassa 8. viitataan luotuun tummanruskeaan 3D-gameobjektiin TMPro (Textmeshpro), joka on tekstiä. TMpro-teksti voi näkyä scenessä ja gameviewissä takaapäin peilikuvana. Tämä näkyy gameviewissä, kun kamera on asetettu tekstin taakse ja kohdistettu siihen. Scenessä se näkyy, kun koordinoinnin näkymä on asetettu TMpro:n taakse.



Kuvio 13. Materiaalit GameObjectissa

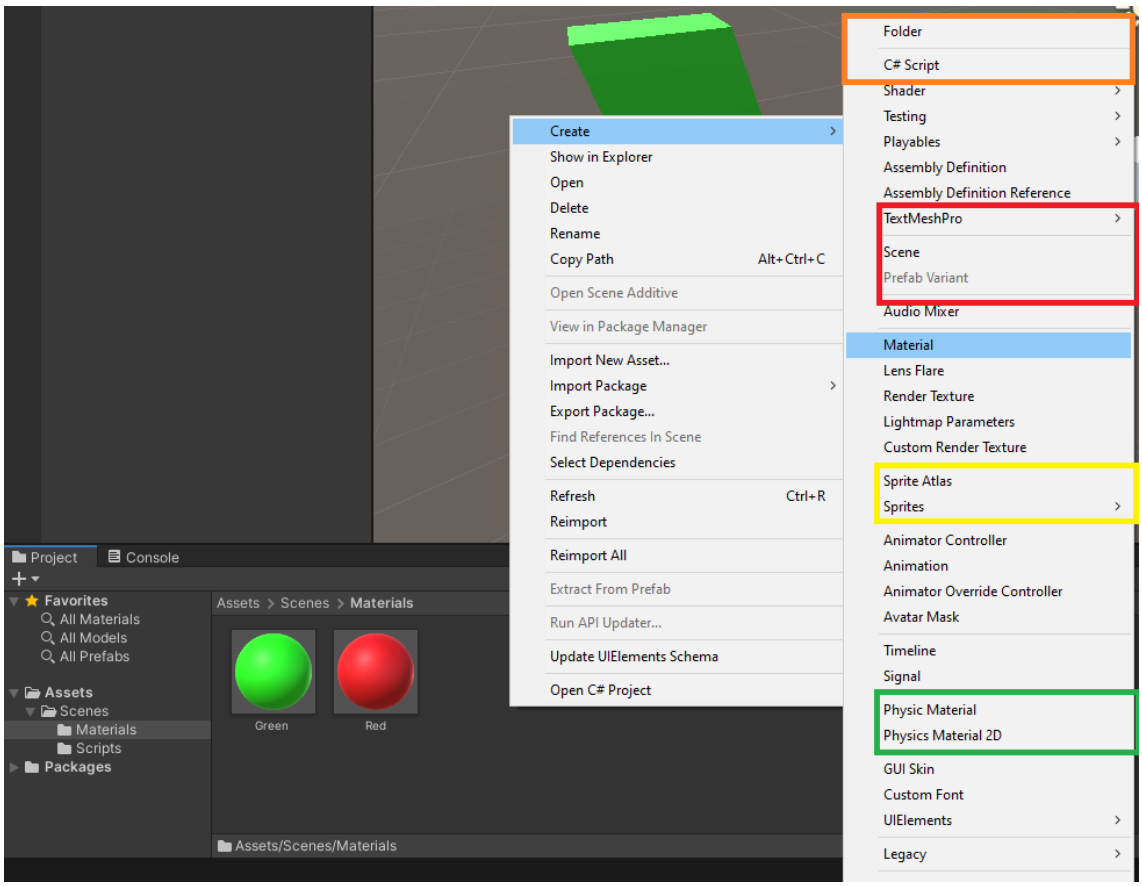
Kuviossa 14 näkyvä rakennelma koostuu kuution muotoisista GameObjecteista. Kuutiot ovat skaalalla (1,1,1), korkeus y on 0.5 ja rotaatioon muutoksia oletuksesta ei ole. Kuutioiden paikat koordinaateissa x, y ja z ovat seuraavat: (0,0.5,0), (1,0.5,0), (0,0.5,1), (1,0.5,1), (0.5,1.5,0.5) ja (2,0.5,0). Ensimmäinen kuutio sijaitsee x:n ja z:n nollapisteessä. Neljä ensimmäistä määrittelevät kuutiot, jotka muodostavat isomman rakennelman. Viides laitetaan niiden päälle ja keskelle koordinaatteina (0.5, 1.5, 0.5). Kuudes havainnoi koordinaattien käyttöä, kun mennään kauemmaksi kuutiojoukosta. Tämän koordinaatit ovat (2,0.5, 0).

On hyvä syöttää arvot suoraan transform-muuttujaan, jos muuttaa jotain manuaalisesti hiirellä, ei saa haluttuja arvoja niin helposti. Tämä tapahtuu, kun yrittää saada muutettua kokonaislukuihin tai tiettyyn desimaalin.



Kuvio 14. Rakennelma kuutioobjekteista

Kuviossa 15 näkyvä valikko näkyy klikkaamalla hiiren oikealla napilla projektin jossain kansiossa. Projektinäkymään voi lisätä seuraavaa: materiaalit, folder, c# script, TextMeshPro, scene, sprite ja fysiikkamateriaalit. Materials-kansio on hyvä lisätä ja siihen halutut materiaalit. Kun muokkaa materiaalia inspector-näkymässä, väri voidaan lisätä albedo-kohdassa. Inspector näkyy oletuksena käyttöliittymässä oikealla.



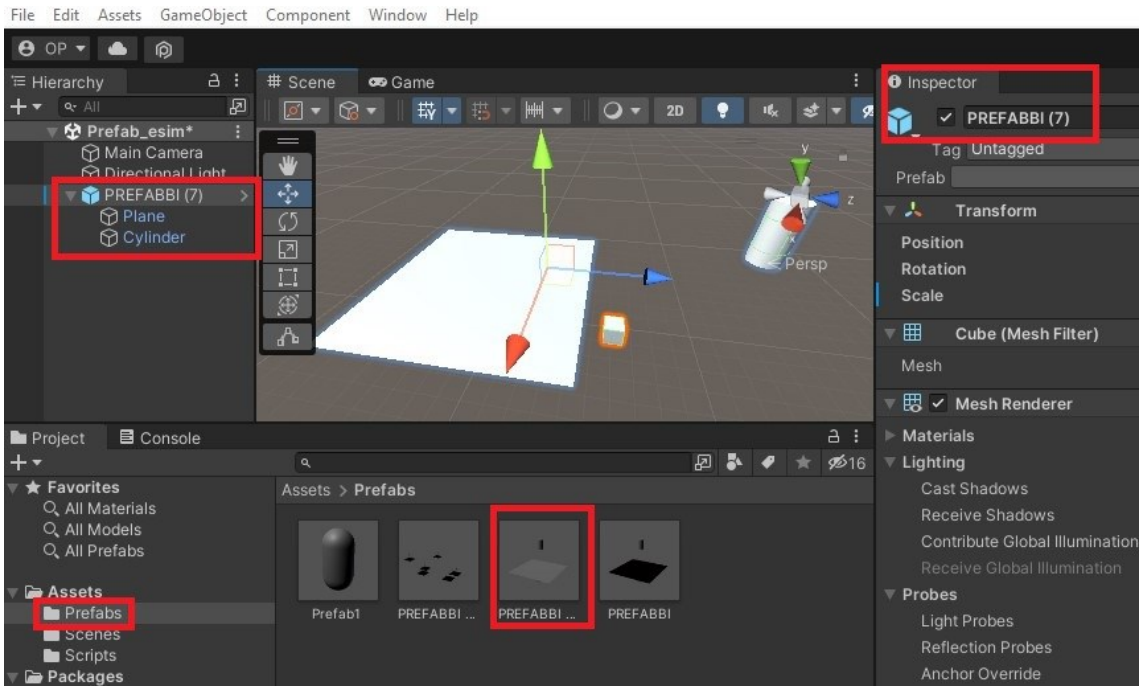
Kuvio 15. Hiirellä tulevat vaihtoehdot

Peliobjektit ovat tyhjiä säiliöitä, joihin lisätään komponentteja. Komponentit sallivat suunnitella geometriaa, luoda valoa, toimia kamerana tai lisätä toimintoja koodeilla. Ne voivat myös olla enemmän kansion tyyppisiä, jolloin ne sisältävät toisia peliobjekteja. Kun peliobjekti poistetaan, poistuu se koko scenestä. (Moakley 2019a.)

Moakley Brianin mukaan ”kaikki hierarkiassa on tyyppiä GameObject” (Moakley 2019b). GameObject sisältää 3D-avaruuteen sijoittumisen arvot. Sitä voi muuttaa myös lisäämällä siihen komponentteja, kuten materiaali, rigidbody, koodi tai äänitehoste. GameObjectit voivat olla todella monimutkaisia joukkoja. Yksinkertainen tällainen voisi olla kolme geometristä kuutiota tai esimerkiksi kuvion 14 rakennelma. Nämä olisivat siis yhtenäisenä joukkona hierarkiassa, toistensa sisällyttäminä. Ne voisivat olla myös muotoa prefab. Se on yleensä parempi valinta, sillä ne on helpompi kopioida koodissa tai ohjelmassa. (Moakley 2019b.)

Prefabin voi luoda esimerkiksi project-osion sisällä, sen sisällyttämät objektit taas vain hierarchy-valikosta. Project-valikon kansiossa on sininen kuvake, mutta kun hierarkiavalikosta lisätään objekti, muuttuu se objektin näköiseksi. Sininen kuvio näkyy kuitenkin hierarchy-näkymässä.

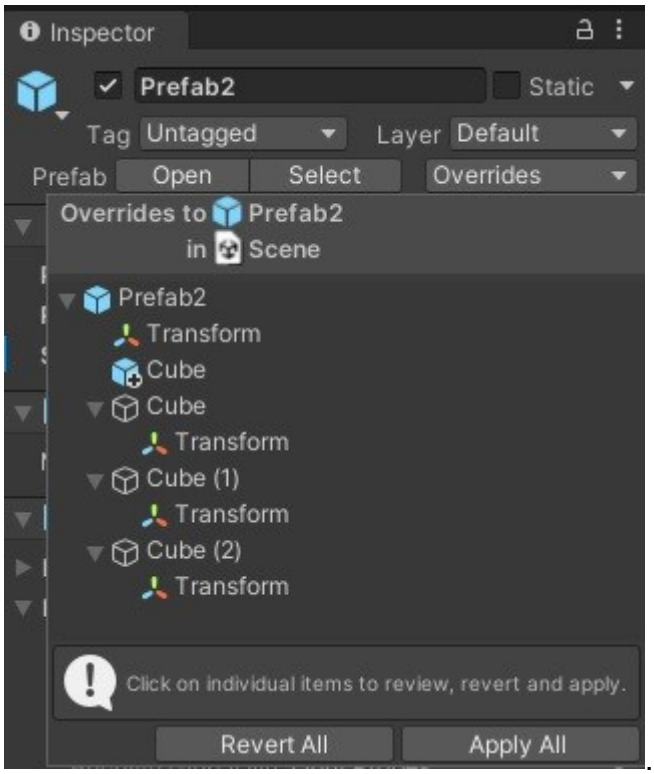
Aluksi kannattaa luoda prefabs-kansio, niin kuin esimerkiksi koodeille scripts. Projektin sisällä voi sitten luoda elementtiobjektin. Se saa 3D-objektin vain hierarkiavalikossa luodusta GameObjectista. GameObject ei näy, jos se ei sisällytä 3D-objektia, esimerkiksi kuutiota, tai UI:ta, esimerkiksi tekstiä. Kuviossa 16 näkyvät kaikki prefabeihin liittyvät näkymät ja valinnat.



Kuvio 16. Prefabs-valinnat punaisella

Jos objektissa on sinistä tekstiä, kertoo se, että se on linkitettynä prefabiin. Teksti on ruskeaa, jos linkki on katkennut. Prefabit ovat hyviä siinä mielessä, että kun projektissa olevan objektin elementin mallia muuttaa, muuttuvat myös kaikki sen kopiot hierarkiassa ja scene view -näkymässä. Hierarkiassa voi kuitenkin muuttaa prefab-peliobjekteja ilman, että alkuperäinen projektin sisäinen prefab muuttuu. Lopuksi on luotu prefab, jonka voi monistaa projektista sceneen lähes niin monta kertaa kuin haluaa. Tämä siis riippuu kopioidusta GameObjectista. (Moakley 2019b.)

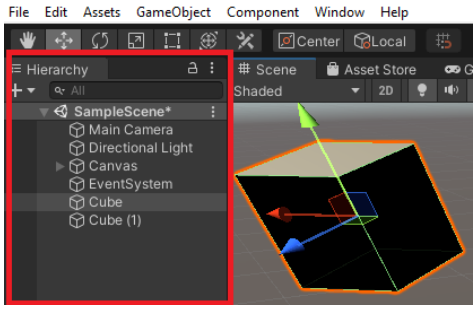
Alla olevassa kuviossa 17 näkyy prefabin avaus (open), valinta (select) ja yliajo (overrides), jonka kohteet ja vaihtoehdot ovat valintojen alapuolella. Valitsemalla revert all peruutetaan muutokset ja apply all asetetaan uudet muutokset. (Moakley 2019b.)



Kuvio 17. Prefab ja inspector

4.2 Hierarkia

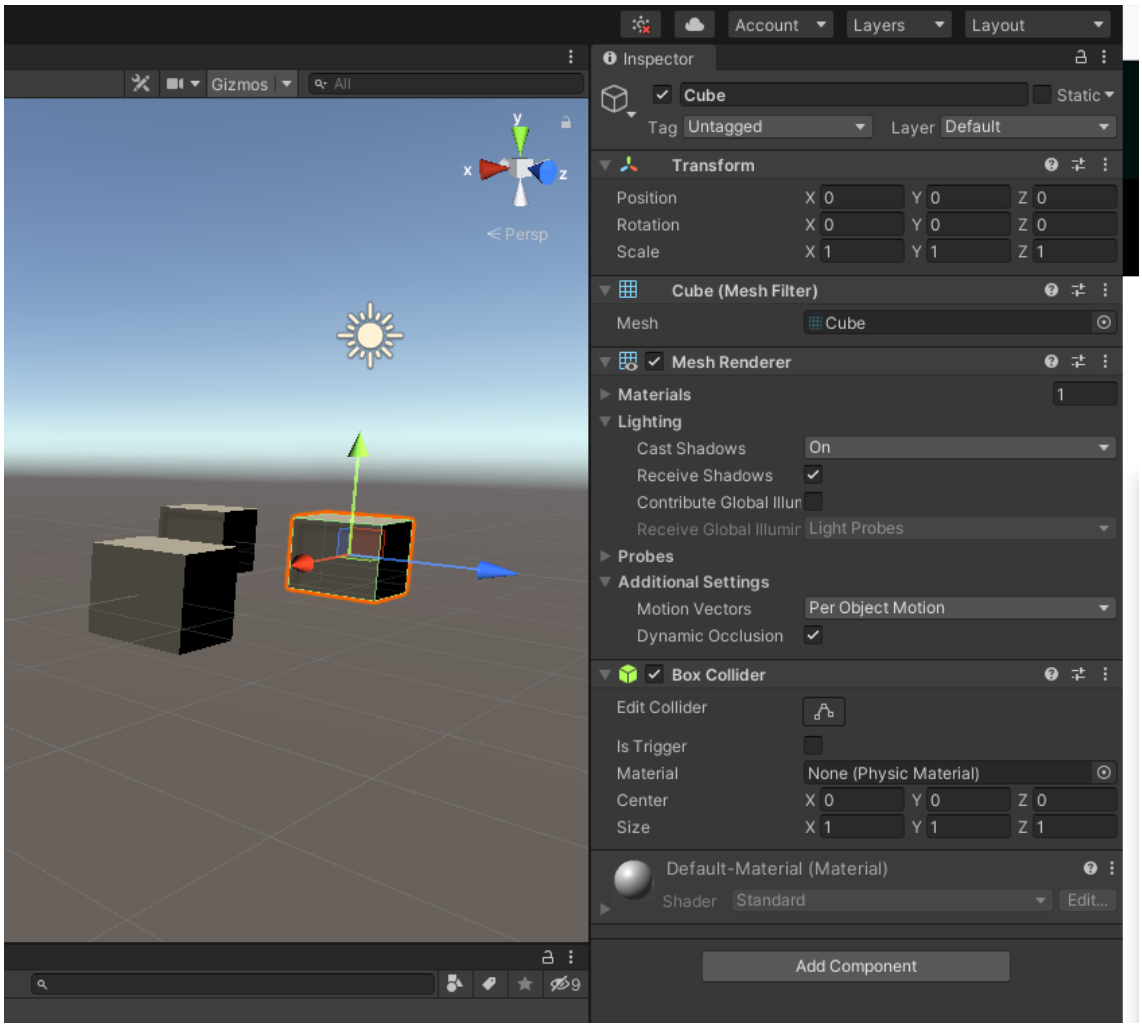
Kaikki peliobjektit näkyvät hierarchy-ikkunassa kuviossa 18. Ne löytyvät pelimoottorin vasemmasta laidasta. Siellä objekteja voi luoda painamalla hiiren oikeaa nappia Objectissa, samoin kuin GameObject-valikossa. Niitä voi leikata, liittää ja liittää lapseksi. Niitä voi myös uudelleen nimetä, käyttää toimintoa duplicate, poistaa ja näyttää objektin tiedot erilliseen ikkunaan. Tiedot näkyvät myös inspector-ikkunassa kaksoisklikkaamalla vasenta hiiren nappia objektin päällä.



Kuvio 18. Hierarkiavalikko

5 KOMPONENTIT

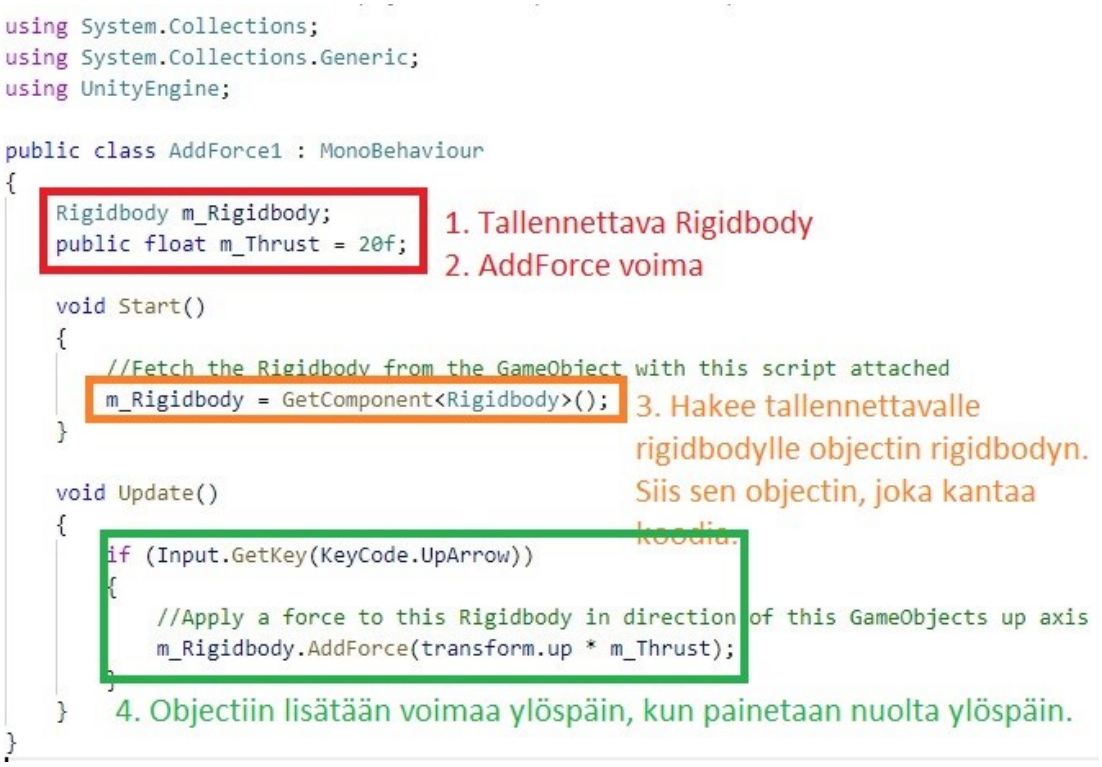
Peliobjektit sisältävät komponentit. Komponentteja ovat esimerkiksi colliderit, rigidbody, materiaalit, koodit, animator ja transform-muuttujat. Ne tulevat näkymään inspectoriin (Kuvio 19). Kuviossa 19 on kuutio valittuna, siinä on oranssit reunat. Inspectorissa näkyy tämän kuution sisältö, toisin sanoen sen eri komponentit. Niitä ovat transform, mesh filter, mesh renderer, lightning, probes, additional settings, box collider ja Default–material. Sinne ilmestyvät kaikki lisätyt komponentit, kuten rigidbody, material, animator ja eri koodit.



Kuvio 19. Inspector-näkymä oikealla

5.1 Fysiikka

Fysiikka tarvitsee collideriin rigidbodyn, ettei GameObjectin collider ole staattinen (Moakley 2019b). Seuraavassa kuviossa 20 havainnollistetaan voiman lisäämistä peliobjektiin. Ensin luodaan rigidbody-muotoa oleva muuttuja. Lisättävä arvo on fysiikassa oleva addforcen voiman suuruus, sitä voi muuttaa halutuksi (m_Thrust). Seuraavaksi kohdassa 3 haetaan objectin rigidbodyn kohdassa ensiksi luotuun muuttujaan ja viimeisessä kohdassa taas lisätään fysiikan voimaa samalla tapaa kuin transorm.Translate()-komennolla. (Unity Technologies 2022e; 2022g.)



Kuvio 20. Rigidbody.AddForce() (Unity Technologies 2022e)

5.2 Colliderit

Kaikissa tavallisissa 3D-objekteissa on niiden mukainen collider, esimerkiksi kuutiossa on box collider. Se näkyy inspectorissa. Vihreän kuvakkeen vieressä on valintalaatikko, jossa sen voi ottaa käyttöön (on) tai poistaa käytöstä (off). Is trigger -osiossa on myös samanlainen valintalaatikko.

Silloin sitä pystyy ohjelmoimaan rigidbodyn lähettämillä viesteillä: Collider.OnTriggerEnter() (Kuvio 21), Collider.OnTriggerExit() ja Collider.OntriggerStay(). Nämä viestit tulevat, kun rigidbody tulee tai poistuu objektista, joka on asetettu is trigger -muotoon. (Unity Technologies 2022h.)

Box colliderilla toteutettava koodin kantavan objektin tuhoaminen toteutuu, kun liikutettava objekti ”Collision collision” törmää siihen. Tämä tapahtuu kuviossa 21. Sillä pitää olla rigidbody. Aluksi ei käytetä painovoimaa. Sen sijaan voidaan käyttää myös metodia OnCollisionExit(). (Unity Technologies 2022i; 2022j.)

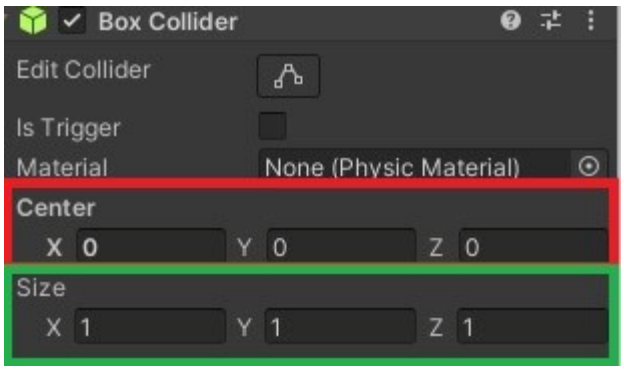
```
void OnCollisionEnter(Collision collision)
{
    Destroy(gameObject);
}
```

Kuvio 21. Tuhoutuminen törmätessä collideriin (Unity Technologies 2022i)

Kuvion 21 koodia on käytetty myös tämän tutoriaalin pelissä (Kuvio 34). Sillä poistetaan haamut (viholliset) ja lasketaan, kuinka monet niistä on tuhottu.

Kuviossa 22 punainen alue määrää box colliderin paikan suhteessa peliobjektiin. Se on oletuksena paikalla ”0,0,0,” (x,y,z), silloin se on kantavan gameobjektin kohdalla. Esimerkiksi jos y=10, niin se on GameObjectin yläpuolella. Vihreässä alueessa on colliderin koko (size), oletuksena on koko ”1,1,1” (x,y,z).

Size on rajattu kuviossa 22 vihreällä neliöllä, size on melkein sama asia kuin scale gameobjektin inspectorissa. Skaalaus venyttää pintaa ja sen kuvioita, esimerkiksi tekstuureja. Jos ei ole mitään kuvioita pinnassa, skaalaus ei näy. Skaalauksella venytetään alun perin annettuja objektin koon arvoja.



Kuvio 22. Box collider inspectorissa

Kuvion 23 uusi koodi lisätään alussa tehtyihin koodeihin, joissa käytettiin metodia transform.Translate(). Se voidaan tehdä myös uuteen ja erilliseen c#-tiedostoon. Collider other on liikutettava objekti, joka sisältää rigidbodyn. Sen törmätessä is trigger -objektiin toteutuu funktion komento. Tässä tapauksessa float-arvo speed muuttuu negatiiviseksi, jolloin kulkusuunta muuttuu päinvastaiseen suuntaan, toisella kertaa suunta taas muuttuu takaisin. (Unity Technologies 2022h.)

```
//Upon collision with another GameObject, this GameObject will reverse direction
private void OnTriggerEnter(Collider other)
{
    speed = speed * -1;
}

//https://docs.unity3d.com/ScriptReference/Collider.OnTriggerEnter.html
```

Kuvio 23. OnTriggerEnter() (Unity Technologies 2022h)

6 OHJELMOINTIA

6.1 Instantiate, Random.Range ja GameObject[]-lista

Kuviossa 24 lisätään objekti tiettyyn paikkaan, käyttäen instantiate-komentoa. On suositeltavaa lisätä myös prefabin, jonka ei tarvitse olla valmiina hierarkiassa (hierarchy), vain projektissa.

Luodaan tyhjä objekti ja määritellään sille muuttuja transform, niin kuin kuviossa 24. Ensimmäisessä osassa on *public Transform paikka;*. Kuviossa 24 on myös instantioitu (instantiate) kymmenen objektia, jotka ovat alkuperäisen klooneja.

Tässä on käytetty for loop -toimintoa, siinä lukua i, jossa kohdassa 1 annetaan arvo 0. Kohdassa 3 lisätään aina yksi luku lukuun i, niin kauan kun i on pienempi kuin 10. Koodi muodostaa 10 gameobjectin rivin. Korkeus (y) ja syvyys (z) ovat nollia, ne eivät muutu. (Unity Technologies 2022k.)

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

0 references
public class Ilmestyminen : MonoBehaviour
{
    1 reference
    public Transform paikka;
    // Start is called before the first frame update
    0 references
    void Start()
    {
        for (int i = 0; i < 10; i++)
        {
            Instantiate(paikka, new Vector3(i * 2.0F, 0, 0), Quaternion.identity);
        }
    }
}

//https://docs.unity3d.com/ScriptReference/Object.Instantiate.html
```

Kuvio 24. Instaniate for() loopissa (Unity Technologies 2022k)

Edellisen rakenteen mukaisesti voi luoda myös peliobjekteja satunnaisille paikoille, ei siis riviin niin kuin edellisessä. Tässä osassa vain korkeus (y) on vakio. Kuviossa 25 lisätään koodiin peliobjektit, jotka halutaan luoda. Edellisen kuvion mukaisesti lisätään for looppiin satunnainen paikka käyttäen random.rangea paikoilla x ja z. Koodin rivit ovat tekstimuodossa: *var pos = new Vector3(Random.Range(-10.0f, 10.0f), 0, Random.Range(-10.0f,10.0f));* instantiate käskyyn: *Instantiate(Randomi, pos, Quaternion.identity);* (Unity Technologies 2022k; 2022l.)

On mahdollista lisätä GameObject Randomin tilalle GameObject-listaan (Kuvio 25), joka hakee satunnaisesti elementit, jotka lista sisältää. Eriväriset kuutiot siirretään listaan inspectorissa koodin kohdalla (element 0 - 4). Tämä näkyy kuviossa 25. Käytetty määritelmä koodissa on *public GameObject[] kuutiot;*. Kuviossa on jätetty Randomi vihreisiin kirjanmerkkeihin.

Arpominen on toteutettu for loopissa laittamalla GameObject Randomin tilalle *public kuutiot[Random.Range(0,5)]* (Kuvio 25, kohta 3), edellinen luku 0 on listan lähtökohta. Luku 5 taas on listan maksimipituus. Elementtien määrä on näiden lukujen etäisyys toisistaan, satunnaiset GameObjectin kloonit muodostuvat nämä Instantiatiella luomisen aikana. Tässäkin kohdassa korkeus(y) on vakio. (Unity Technologies 2020b.)

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

public class RandomOjectIlmestyminen : MonoBehaviour
{
    //public GameObject Randomi;
    public GameObject[] kuutiot;

    void Start()
    {
        for (int i = 0; i < 10; i++)
        {
            var pos = new Vector3(Random.Range(-10.0f, 10.0f),4,Random.Range(-10.0f,10.0f));
            Instantiate(kuutiot[Random.Range(0,5)], pos, Quaternion.identity);
        }
    }
}
//https://answers.unity.com/questions/915759/how-to-instantiate-a-random-object-from-an-array.html
```

2. GameObject Randomi poistetaan käytöstä

1. Lisätään GameObject lista nimellä kuutiot. Se täytetään Unity pelimoottorin sisällä. Tässä tapauksessa niitä tehdään viisi.

3. Poistetun Randomi tilalle laitetaan satunnainen GameObject.

Kuvio 25. Random.Range() (Unity Technologies 2022l)

6.2 Hahmon assetit

Asset-tiedostot löytyvät Kansiosta Asset. Seuraavassa kuviossa Enari_Housut-materiaalit osaan lisätään neljä osaa, joista kerrotaan lisää myöhemmässä vaiheessa. Albedo Einari_Housut_Albedo on materiaalin pääväri, ja se menee kohtaan albedo. Värinä on valittu väri, yleensä valkoinen. Tämä voidaan esimerkiksi tehdä punaisesta materiaalista. Materiaalin erottaa muista se, että se on ympyrän muotoinen, tekstuurit näkyvät pelimoottorissa taas neliöinä. Seuraavat osat näkyvät kuviossa 26.

Vaalea osa on occlusion. Sillä voidaan vaihtaa koko materiaalin värejä tummasta vaaleaan. Se erottuu vaaleina neliöinä. Metallic smoothness on metallista materiaalin kiiltoa. Se taas erottuu mustana neliön värinä. Normal map on kaksiulotteisen pinnan pieniä muutoksia, ne ovat pinnan kohoumia tai syventymiä. Todellisuudessa ne ovat tekstuureja, jotka näyttävät pinnalla 3D-objektilta. (Unity Technologies 2022m.) Tekstuurit erottuvat vaaleansinisenä tai violettina neliön värinä pelimoottorissa. Ylhäältä alas laskettuna ensimmäinen paikka on albedo, toinen metallic, kolmas normal map ja neljäs occlusion. Tässä ei käytetä height map -eikä detail mask -kohtia. Kuviossa 26 vasemmalla on materiaali, joka sisällyttää edellä olevat paikat ja värit 1 – 4.

Kuvio 26. Einari_Housut ja effektit

Päähahmon assettien materiaalit ovat kuviossa 27 ylhäältä alas ensimmäisenä housut, toisena kauluri, kolmantena lakki, neljäntenä silmät, viidentenä iho ja kuudentena takki. Alla olevassa kuviossa on ryhmitelty materiaalit ja niiden eri osilla ryhmittäin eri väriä olevien neliöiden sisälle. Neliön muotoiset tekstuurit laitetaan materiaaleihin, jotka taas ovat ympyränmuotoisia. Nämä kokonaisuudet laitetaan

esimerkiksi Einari_Lakki GameObjectiin viemällä valmiin materiaalin projekti näkymästä hiirellä GameObjectiin. Samalla tavalla määritellään myös haamun materiaalit.

Kuvio 27. Visuaalisesti valmis päähahmo

6.3 Hahmon ohjelmointi

Ohjelmointi aloitetaan tekemällä aseeseen kuviossa 28 oleva koodi. Koodi sisältyy aseobjektin inspectoriin. Siellä taas koodi sisältää julkiset (public) GameObjects. Näitä ovat *public float thrust, public Rigidbody rb, public Transform piippu* ja *public GameObject Luoti_etsitty.*

```
void Update()
{
    if(Input.GetKeyDown("space"))
    {
        GameObject Luodin_lahto = GameObject.Find("Ase");
        Rigidbody uusiluoti = Instantiate(Luoti_etsitty,piippu.position,Quaternion.
        identity).GetComponent<Rigidbody>();
        uusiluoti.transform.position = piippu.position;
        uusiluoti.AddForce(piippu.forward * thrust1);
    }
}
```

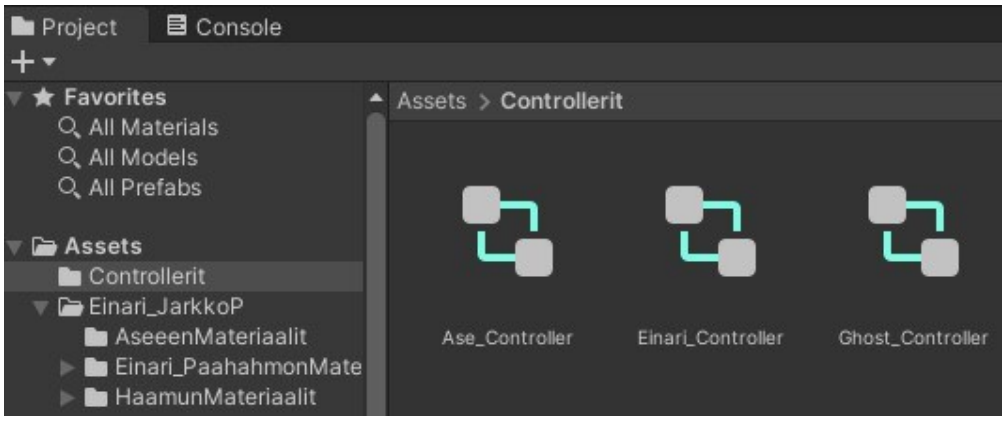
Kuvio 28. Luoti_etsitty gameobjectin instantiointi (Hannula 2021; Unity Technologies 2022c; 2022e; 2022k)

Ampuminen_Sphere-koodissa on Luoti_etsitty-muuttuja, johon lisätään luotina käytettävä prefab Luoti. Törmätessään Ghost GameObjektiin tuhoutuu tämä Ghost, koska se kantaa kuvion 29 koodin.

```
void OnCollisionEnter(Collision collision)
{
    Debug.Log(collision.gameObject.name);
    Destroy(gameObject);
}
```

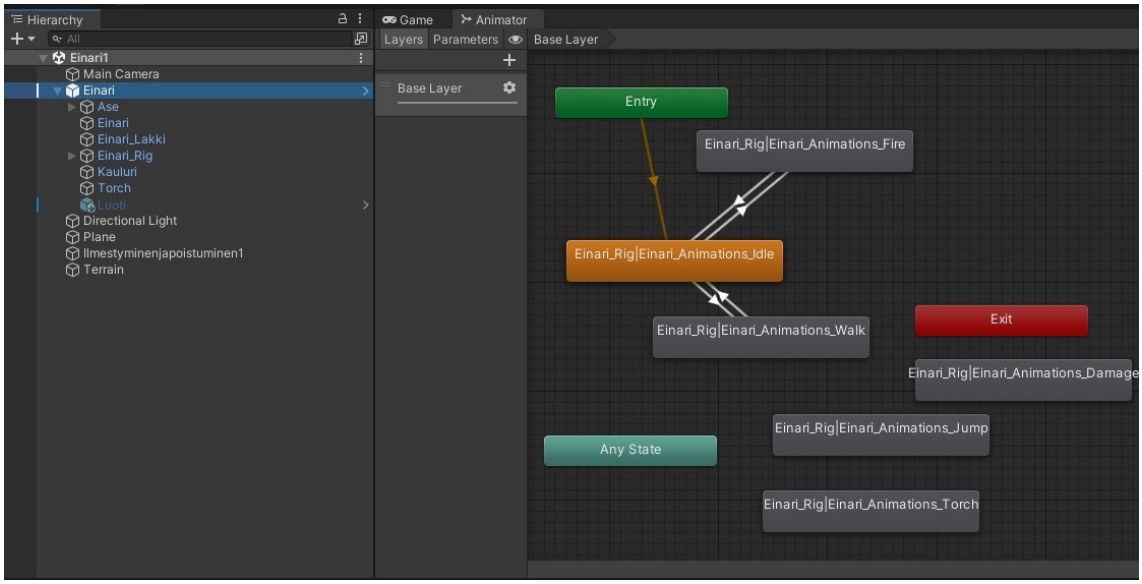
Kuvio 29. Objektin tuhoava Luoti (Unity Technologies 2022i)

Animator controlloreita muokatakseen pitää project-näkymän olla päällä. Siellä on assets-kansio ja sen sisällä controllerit. Kuviossa 30 oikealta löytyvät animator controllers. Vasenta hiiren näppäintä kaksoisklikkaamalla saa näkymään kunkin hahmon animator controllerin. Animator ilmestyy scenen tai gameviewin viereen.



Kuvio 30. Animator-controllerit

Pelissä käytetään päähahmon kolmea animaatioita: *“Einari_Rig|Einari_Animations_Idle”, ” Einari_Rig|Einari_Animations_Walk”* ja *”Einari_Rig|Einari_Animations_Fire”*. Loppuja animaatioita ei käytetty. Oranssiin animaatioon pitää olla kahdet Transition-nuolet, kahdesta muusta animaatiosta. Nuolet saa painamalla hiiren oikealla napilla jotain laatikkoa (Make Transition). Alla olevassa kuviossa 31 on Einari-hahmon Animation Controller. (Unity Technologies 2022n.)



Kuvio 31. Einari-päähahmon animaatiot (Unity Technologies 2022n)

Eteen (uparrow) ja taakse (downarrow) menemisessä käytetään Einari_Rig|Einari_Animations_Walk -animaatiota. Kun painaa space-nappia, kutsutaan Einari_Rig|Einari_Animations_Fire -animaatiota. Oletuksena toimii Einari_Rig|Einari_Animations_Idle, joka on tila, jossa ei ole toimintoja. Koodissa on Einari_Anim-niminen lisätty animator. Start()-metodissa liitetään siihen koodin kantaman animator-komponentti. Haluttuja animaatioita kutsutaan näppäimillä. Tämä tapahtuu if-ehdoilla. Koodi näkyy liitteessä 1, kuviossa 36.

Ghost-hahmon animator controller löytyy samasta paikasta kuin päähahmollakin. Se on nimeltään Ghost_Controller. Valinta tapahtuu tuplaklikkaamalla sitä vasemmalla hiiren napilla.

Kooditiedostona on käytetty ”Vektorikoodia3.cs”. Ghost-hahmoon on laitettu oletuksena animaatio Ghost_Rig|Ghost_Move. Hahmo on koko ajan liikkumistilassa, ja animaatio pyörii loputtomasti metodissa Update(). Siirtääkseen Ghosteja liikutetaan hahmoa komennolla *transform.Translate(Vector3.forward * Time.deltaTime);*. Tässä koodissa on void Start()-metodissa muutettu y-akselin rotaatiota x = 0, y = 90 ja z = 0, tällöin Ghostit kulkevat haluttuun suuntaan. Näitä kulkusuuntia voi muokata halutuksi, esimerkiksi y = 180. Samalla lailla on luotu Ghosteille animator-muuttuja ja etsitty siihen Start()-metodissa inspectorista animator-komponentti. Kaikki kappaleen koodit ovat liitteessä 2.

Ghost-hahmoon lisätään värit materiaalin avulla. Materiaali sisältää Mainmaps Assets/*HaamunMateriaali*t, jossa ovat *Albedo: None_AlbedoTransparency, Metallic: None_MetallicSmoothness, Normal Map: None_Normal, Occlusion: None_AO.* (Hannula 2021.)

Mainmenua varten ovat canvas, mainmenu, TMPro teksti(Aika) sekä kaksi nappia Play_Button ja Quit_Button ja niiden sisältämät tekstit. Hierarkiassa voi huomata, että teksti on näkyvä ja harmaat napit ovat näkymättömiä. Kuviossa 32 ja 38 näkyy pelin alku, jossa canvaksesta näkyy vain tekstimuotoinen GameObject Aika. Aluksi on vain GameObject Kills -laskurille ja siinä tekstikenttä Kills, johon lasketaan ammutut haamut. Se on Canvaksen sisällä oleva TMPro-teksti Kills. Objektiin on lisätty kuvion 38 koodi Killsa.cs.

Kuvio 32. Peliin tarvittava canvas

Ohjelmoimalla on piilotettu ja asetettu napit näkyviksi. Samalla on piilotettu sekuntikello. Kun peli on ollut 30 sekuntia päällä, tulevat napit esiin. Ne ovat koodissa nimillä gameobjekti button1 ja button2. Buttonit ovat aluksi Start()-metodissa arvossa false. Koodiin kirjoitettuna ne ovat button1.SetActive(false) ja but-

ton2.SetActive(false). Ajan päätyttyä on napit tuotu esiin seuraavanlaisella koodilla button1.SetActive(true), samat asiat on tehty button2.SetActive(true)-napille. Koodit ovat liitteessä 3.

Napeista on luotu myös kahdet Button-tyyppiset muuttujat. Näistä on luotu Button btn ja Button btn2. Niitä on kutsuttu koodilla *Button btn = yourButton.GetComponent<Button>(); btn.onClick.AddListener(TaskOnClick);*. Niiden sisällä ensimmäisestä napista käynnistetään peli ja toisesta poistutaan ohjelmasta (Liite 3). (Unity Technologies 2016; 2020e.) Koodi on liitteessä 3.

Lisättynä on uusi teksti laskuri koodiin (Kuvio 33), haamujen koodien kautta. Tämä tapahtunut siis koodilla *int go = GameObject.Find("Kills").GetComponent<Killsa>().counter++;*.(Unity Technologies 2020f.) Tällä on haettu counter toisesta koodista nimeltä Killsa.cs. Se sisältää objektit *public int counter=0;*, *public TextMeshProUGUI text1;* ja *void Update()* sisällä *text1.text = "Kills" + counter.ToString();*. Koodiin on lisätty myös kirjasto TMPro.

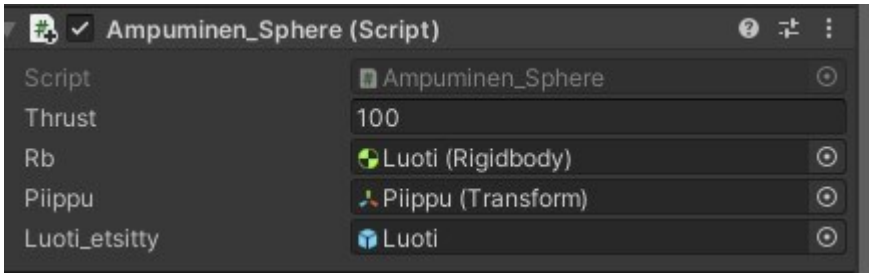
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using System;

public class Killsa : MonoBehaviour
{
    public int counter = 0;
    public TextMeshProUGUI text1;
    // Start is called before the first frame update


    // Update is called once per frame
    void Update()
    {
        //Debug.Log("Koodia Killsa kutsutaan, kills: " + counter);
        text1.text = "Kills: " + counter.ToString();
    }
}
```

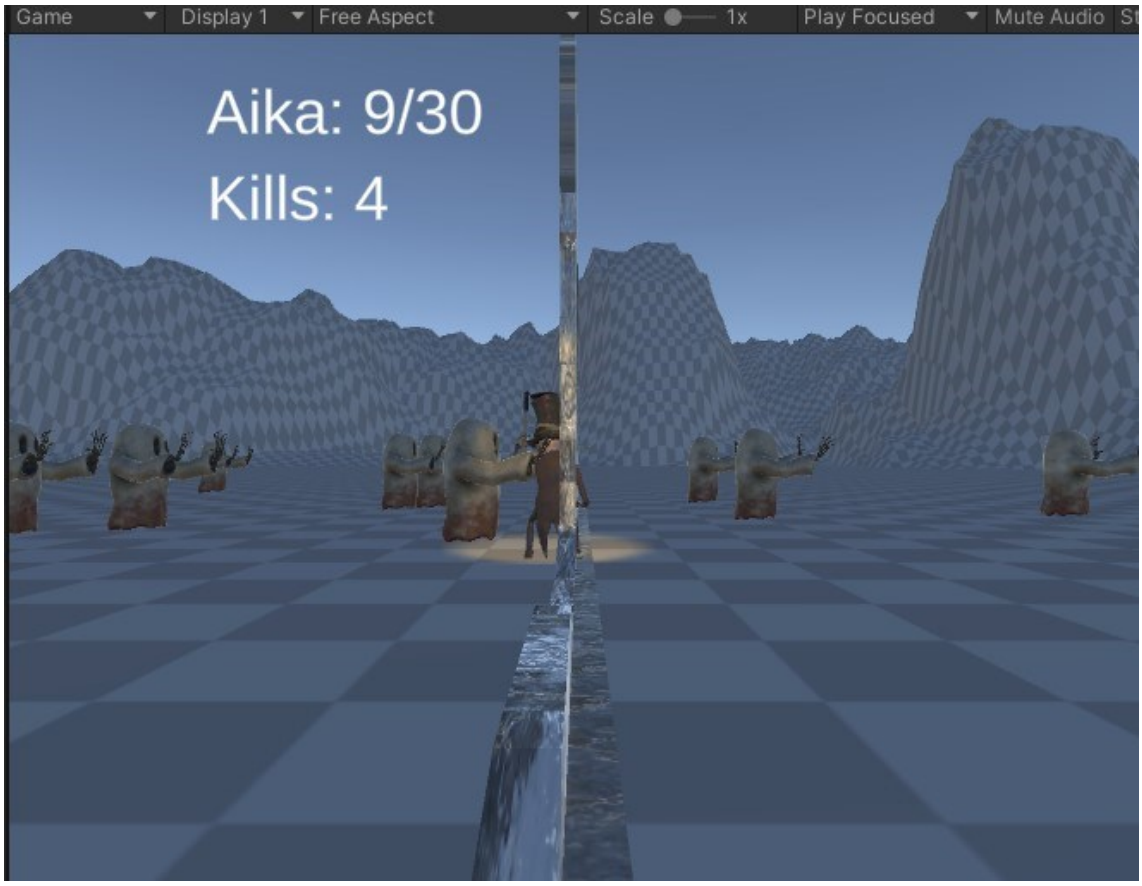
Kuvio 33. Counterin lisääminen tekstiin

Arvossa Thrust pitää olla paljon voimaa. Tällöin luoti kulkee tarpeeksi nopeasti (Kuvio 34). Rb:n arvo sisältää Luodin sisältämän rigidbodyn. Piippu on kohta, josta Luoti lähtee. Kohtaan ”Luoti_etsitty” laitetaan Luoti-prefabin. Tästä on siis otettu myös rigidbodyn. Nämä arvot näkyvät inspectorissa, koodin Ampuminen_Sphere-komponentissa, niin kuin kuviossa 34.



Kuvio 34. Ampumisen sisältävä koodi

Einarin sisälle on lisätty main camera, joka seuraa hahmoa. Kameran pitää olla lisättynä suoraan Einarin päälle, ei sen ja jonkun osan väliin. Jos tulee viesti, jonka nimenä on ”Cannot restructure Prefab instance”, valitaan ”Cancel”.



Kuvio 35. Peli play-tilassa

7 POHDINTA

Olennaista työssä on tehdä hyvä, selkeä ja helppo tutoriaali (perehdytys) Frostbitin nettisivuille. Tarkoituksena on ohjata ohjelmien asentamisessa ja niiden käynnistämisessä. Tällaisten asioiden selittäminen käyttäjille, jotka eivät välttämättä tiedä asioista mitään, on haastavaa. Varsinkin kun pitää selittää asiat tekstinä ja ei saa käyttää ohjeissa videoita.

Tutoriaalissa käydään läpi yleisimpiä Unity 3D -pelimoottorin toimintoja ja niihin liittyviä lyhyitä ja yksinkertaisia koodeja. Näitä asioita ei käyttäjän tarvitse osata ollenkaan. Tarkoitus on oppia nämä kaikki asiat tutoriaalissa. Näissä kaikissa asioissa on onnistuttu hyvin. Kieli on selkeää, yksinkertaista ja havainnoivaa, kuvia on paljon, ja ne hahmottavat asioita hyvin.

Käyttäjä oppi käyttämään Unity 3D -pelimoottoria ja ohjelmointia askel askeleelta, helpoimmat asiat alussa ja vaikeimmat lopussa. Lisäksi asioiden läpikäyminen muuttaa niiden järjestystä, esimerkiksi jos ohjelmoinnissa tarvitaan jotain pelimoottorin peliobjekteja, kuten prefab tai komponentteja, kuten rigidbody.

Toimeksiantoa voi hyödyntää monin tavoin. Sitä voi käyttää kuka vain, verkkosivujen ohjeiden mukaisesti. Sitä voi käyttää myös kurssien tai vastaavien opintojen ohella. Tarkoitus on myös hyödyntää sitä FrosBit-laboratoriassa, muun muassa koulun harjoittelijoiden perehdyttämisessä. Sitä voivat käyttää myös koulun ulkopuolelta tulevat. Näitä ovat yliopisto-opiskelijat, työelämässä olevat tai peruskoululaiset, jotka suorittavat TET-harjoittelua, erityisesti FrosBit-laboratoriolla. Tarkoitus on innostaa käyttäjiä IT-alaan, varsinkin Lapin ammattikorkeakoulun tieto- ja viestintätekniikan alalle.

Opinnäytetyön tiedot ovat hyvin luotettavia, koska pelimoottorin tiedot näkyvät sen näkymässä. Koodit on haettu Unity Technologiesin omalta sivulta. Kaikkea on kokeiltu käytännössä, ne siis toimivat. Opinnäytetyön raporttiosuus taas perustuu toimeksiantoon.

Toimeksiantoa voi jatkaa. Sen pitää vain mennä laajemmalle ohjeissa, esimerkiksi hiiren käyttämisessä. Pelimoottorissa on myös haastavampia toiminnallisuuksia, esimerkiksi monimutkaisten animaatioiden käyttö. Lisää vaativuutta ohjelmoimiseen löytyy todella pitkään, jopa vuosiksi. Tutoriaalia voisi myös tutkia tutkimusaiheena. Siihen sopisi esimerkiksi IT-alan tai koulutusalan osaaja. Siinä voisi tutkia teknistä oppimista ja asioiden hahmottamista.

Opinnäytetyössä suuressa osassa, jopa tarkoituksena, on pelin tekeminen. Sen on tarkoitus olla kannustava tekijä tutoriaalissa. Käyttäjä tekee oppimillaan asioilla lopuksi pelin.

LÄHTEET

Hannula P. 2021. Unity3d – Perusteet – Osa8-Einari liikkuu enemmän. Viitattu 6.9.2022 https://www.youtube.com/watch?v=WAv0-eL1FD4.

Microsoft 2022. Download Visual Studio Code. Viitattu 31.3.2022 https://code.visualstudio.com/download.

Moakley, B. 2019a. Introduction to Unity: Getting Started-Part 1/2. Viitattu 10.3.2022 https://www.raywenderlich.com/7514-introduction-to-unity-getting-started-.

Moakley, B. 2019b. Introduction to Unity: Getting Started-Part 2/2. Viitattu 14.3.2022 https://www.raywenderlich.com/9175-introduction-to-unity-getting-started-.

Unity Technologies 2016. Button.onClick. Viitattu 28.10.2022 https://docs.unity3d.com/530/Documentation/ScriptReference/UI.Button-onClick.html.

Unity Technologies 2022a. Unlock your creativity. Viitattu 6.4.2022 https://unity.com/download.

Unity Technologies 2022b. Vector3. Viitattu 31.1.2022 https://docs.unity3d.com/ScriptReference/Vector3.

Unity Technologies 2022c. KeyCode. Viitattu 1.6.2022 https://docs.unity3d.com/ScriptReference/KeyCode.html.

Unity Technologies 2022d. Transform.Translate. Viitattu 31.1.2022 https://docs.unity3d.com/ScriptReference/Transform.Translate.

Unity Technologies 2022e. Rigidbody.AddForce. Viitattu 14.2.2022 https://docs.unity3d.com/ScriptReference/Rigidbody.AddForce.

Unity Technologies 2022f. Transform.Rotate. Viitattu 18.4.2022 https://docs.unity3d.com/ScriptReference/Transform.Rotate.

Unity Technologies 2022g. Rigidbody. Viitattu 7.3.2022 https://docs.unity3d.com/ScriptReference/Rigidbody.

Unity Technologies 2022h. Collider.OnTriggerEnter(Collider). Viitattu 23.3.2022 https://docs.unity3d.com/ScriptReference/Collider.OnTriggerEnter.

Unity Technologies 2022i. Collider.OnCollisionEnter(Collision). Viitattu 18.3.2022 https://docs.unity3d.com/ScriptReference/Collider.OnCollisionEnter.html.

Unity Technologies 2022j. Collider.OnCollisionExit(Collision). Viitattu 18.3.2022 https://docs.unity3d.com/ScriptReference/Collider.OnCollisionExit.

Unity Technologies 2022k. Object.Instantiate. Viitattu 11.3.2022 https://docs.unity3d.com/ScriptReference/Object.Instantiate.

Unity Technologies 2022l. Random.Range. Viitattu 28.3.2022 https://docs.unity3d.com/ScriptReference/Random.Range.html.

Unity Technologies 2022m. Normal map(Bump mapping). Viitattu 23.8.2022 https://docs.unity3d.com/Manual/StandardShaderMaterialParameterNormal-Map.

Unity Technologies 2022n. Animator Controller. Viitattu 1.9.2022 https://docs.unity3d.com/Manual/class-AnimatorController.

LIITTEET

Liite 1. Päähahmon animaatioiden kutsut

```
using UnityEngine;
using System.Collections;

// Git and Github for beginners - Crash Course 16:55 (kansio git:(master) la
public class Liikkuminen : MonoBehaviour
{
    public Animator Einari_Anim;
    void Start()
    {

        Einari_Anim = GetComponent<Animator>();
    }
    void Update()
    {
        if(Input.GetKey(KeyCode.UpArrow))
        {
            Einari_Anim.Play("Einari_Rig|Einari_Animations_Walk");
            this.transform.Translate(Vector3.forward * Time.deltaTime * 10);

        }
        if(Input.GetKey(KeyCode.DownArrow))
        {
            Einari_Anim.Play("Einari_Rig|Einari_Animations_Walk");
            this.transform.Translate(Vector3.back * Time.deltaTime * 10);
        }
        if (Input.GetKey(KeyCode.LeftArrow))
        {
            this.transform.Rotate(Vector3.up, -1);
        }
        if (Input.GetKey(KeyCode.RightArrow))
        {
            this.transform.Rotate(Vector3.up, 1);
        }
        if(Input.GetKeyDown("space"))
        {
            Einari_Anim.Play("Einari_Rig|Einari_Animations_Fire");
        }
    }
}
```

Liite 2. Ghost-animaatio koodissa

```
using UnityEngine;
using System.Collections;

// Git and Github for beginners - Crash Course 16:55 (kansio git:(master)
public class Vektorikoodia3 : MonoBehaviour
{
    public Animator ghost_Anim;

    void Start()
    {
        transform.Rotate(0,90,0);

        // Start is called before the first frame update

        ghost_Anim = GetComponent<Animator>();
    }


    void Update()
    {
        ghost_Anim.Play("Ghost_Rig|Ghost_Move");
        // Move the object forward along its z axis 1 unit/second.
        transform.Translate(Vector3.forward * Time.deltaTime);


        // Move the object upward in world space 1 unit/second.
        //transform.Translate(Vector3.down * Time.deltaTime, Space.World);


    }
}
```

Liite 3. MainMenu-koodin ensimmäinen osa

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using System;
using UnityEngine.SceneManagement;

public class MainMenu_koodi : MonoBehaviour
{
    public float score = 0.0f;
    public TMP_Text KelloTeksti;
    public int seconds;
    public int maxAika =30;
    public GameObject button1;
    public GameObject button2;

    public Button yourButton;
    public Button yourButton2;
    // Start is called before the first frame update
    void Start()
    {

        KelloTeksti.text = "Aika";
        button1.SetActive(false);
        button2.SetActive(false);
        Button btn = yourButton.GetComponent<Button>();
        btn.onClick.AddListener(TaskOnClick);
        Button btn2 = yourButton2.GetComponent<Button>();
        btn2.onClick.AddListener(TaskOnClick2);

    }
```

Liite 4. MainMenu-koodin toinen osa

```
    //https://answers.unity.com/questions/1038757/how-to-make-a-timer-that-counts-up-in-seconds-as-a.html
    void Update()
    {
        if(seconds <= 29){
        print(seconds);
        score += Time.deltaTime;
        seconds = (int)(score % 60);
        Debug.Log("sekunnit: " + seconds);
        KelloTeksti.text = "Aika: "+seconds.ToString()+"/"+maxAika;
        }if(seconds == maxAika)
        {
            KelloTeksti.text="Aika loppui!";
            button1.SetActive(true);
            button2.SetActive(true);
            KelloTeksti.text = "";
            Debug.Log("30 sekunttia taynna");

        }
    }

    //https://docs.unity3d.com/530/Documentation/ScriptReference/UI.Button-onClick.html
    //https://answers.unity.com/questions/1369393/how-to-switch-between-scenes.html
    void TaskOnClick()
    {
        Debug.Log ("You have clicked the button1!");
        SceneManager.LoadScene("Einari1");
    }
    void TaskOnClick2()
    {
        Debug.Log ("You have clicked the button1!");
        Application.Quit();
    }

}
```