Noora Angelva
R54T19S

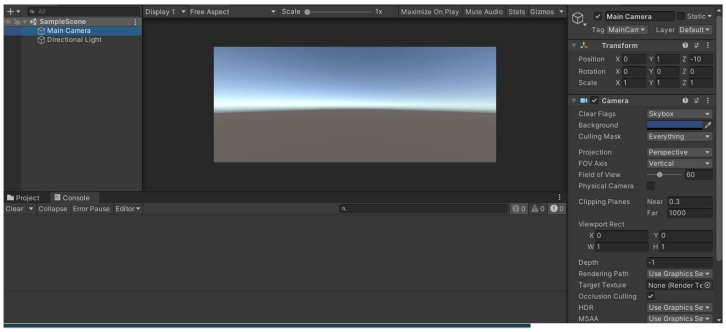
Oppimispäiväkirja
Pelimoottorit #2020 Unity

6.10.2020

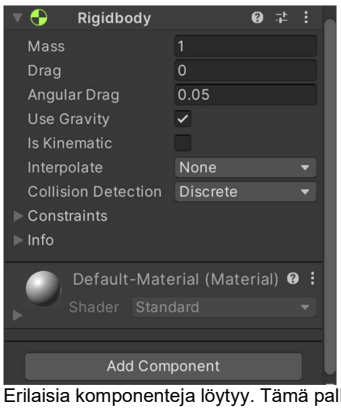
Pelimoottorit #2020 Unreal

Ensimmäinen luento 6.10.2020

Seurasin unityn lataamisen, mutta olin sen jo ehtinyt ladata aikaisemmin ohjelmistotekniikan tehtävää varten. Unity pelimoottori on ulkonäöllisesti hyvin saman näköinen kuin unreal. details unrealissa on sama kuin unityn inspector esineelle. pelin assettien lista on vasemmalla puolella. Valitsemalla assetin ja painamalla f-nappia kamera fokusoituu esineeseen.



Skaalaukset ja kääntelyt ja sen sellaiset löytyvät vasemmasta yläkulmasta.



Erilaisia komponenteja löytyy. Tämä pallolle jotta saadaan se liikkeelle.
Alaosassa komponentien lisäys nappi.

Scripteille olisi hyvä tehdä oma kansionsa. painamalla hiiren oikeaa nappia ja valitsemalla create -> c#-script voi luoda koodi-tiedostoja. Scriptin saa lisättyä komponentiksi assetille add component napista -> scripts -> haluttu koodi-tiedosto.

Noora Angelva
R54T19S

Oppimispäiväkirja
Pelimoottorit #2020 Unity

6.10.2020

Ensimmäinen koodini unitylla ja sen console viestit.

Toinen luento 8.10.2020

Luennolla käydään läpi debuggausta, muuttujia ja yleisimpiä virheitä ja kuinka niitä löydetään.

Debug.Log on sama kuin Console.Log esim. c-kielessä. Break Point pysäyttää simulaation kun asetettuun Break Pointtiin päästään koodissa. Jos ei meinaa toimia muista tarkistaa että unity on yhdistetty vs codeen. Toinen asia liitä unity debugger lisäsosa vs codeen -> add attachement. Debuggaus vs codesta päälle ja peli päälle.

Muuttujien osion seurasin ainoastaan sillä aihe on jo tuttu muilta kursseilta ja asiaa on harjoiteltu olio-ohjelmoinnissa. Muuttujat mielellään privatena ja get seteillä ulos. Toinen tapa jolla muuttujat näkyvät unityssä on publicin poisto ja muuttujalle lisäys [serializefield].

Erroreiden löytäminen koodista on helppoa. Unity ilmoittaa punaisella huutomerkillä consoleen errorista ja sen sijainnista. Klikkaamalla erroria se kursivoi erroin paikan koodista. Unity kertoo huonosti mikä virhe on etenkin ; kohdalla. Muista siis katsoa edellistä riviä koodista. Pidä myös sulje parit samoilla kohdilla ja sisennykset samoina, jotta teksti on helpompaa lukea ja ja virheet löytyy nopeaa.

Kolmas luento 21.10.2020

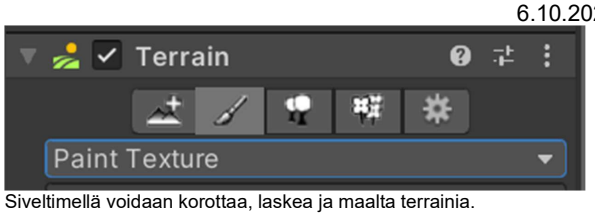
Luennolla käydään läpi assetteja, terrainia, puita ja tekstuureja. Assetteja voi tuoda myös omista tiedostoista viemällä ne yksinkertaisesti peliprojektin assets kansioon.

Terrain, millä kannattaa aloittaa pelin maailman luonti, löytyy 3D Object -> Terrain. Näin luodaan plane, jota voidaan lähteä muokkaamaan sen jälkeen halutunlaiseksi.

Noora Angelva
R54T19S

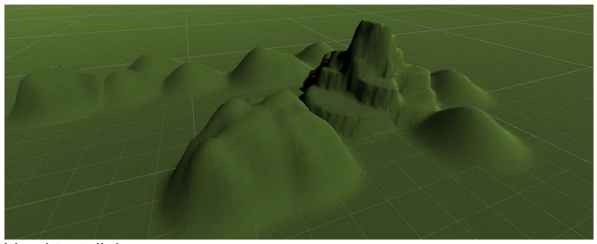
Oppimispäiväkirja
Pelimoottorit #2020 Unity

6.10.2020



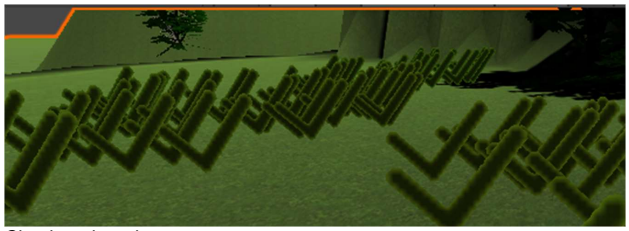
Siveltimellä voidaan korottaa, laskea ja maalta terrainia.

Edit terrain layer… luodaan uusi terrain (create new terrain), joka peittää planen. vuorten ja kuoppien luonti tapahtuu valitsemalla “raise or lower terrain”. Siveltimen koolla hallitaan muokattavan alueen kokoa. left-klikkaamalla maasto nousee ja painamalla lisäksi shift painamalla maasto syvenee. Esim. Stamp Height voidaan määrittää tietty korkeus ja sen jälkeen vain “lätkiä menemään” saman korkuisia kuoppia tai vuoria maastoon.


Vuori testailuja.

Puita lisätään valitsemalla paint trees ja lisäämällä edit Trees -> add trees kautta halutun puun/puut. Jos esim. puut näyttää väärältä, voi jokin osa olla kateissa.

Ruohon tekstuureja voi muokata paint detailsista samaan tyyliin lisäämällä detailsin.


Checkmark-ruoho.

Assets storesta löytyy monesti ilmaisia assetteja. Tonin tehtävissä on jo käytetty assetteja storesta. Storesta ladattu assetti avataan unityssä, ladataan ja importataan projektiin.

Clouberryllä ei ollut rigiä, koska se ei ole hahmo, mutta hamolta se löytyy. (Näitäkin olen jo käynyt Tonin viedoilla niin en kirjoita niistä kauhean tarkkaan).

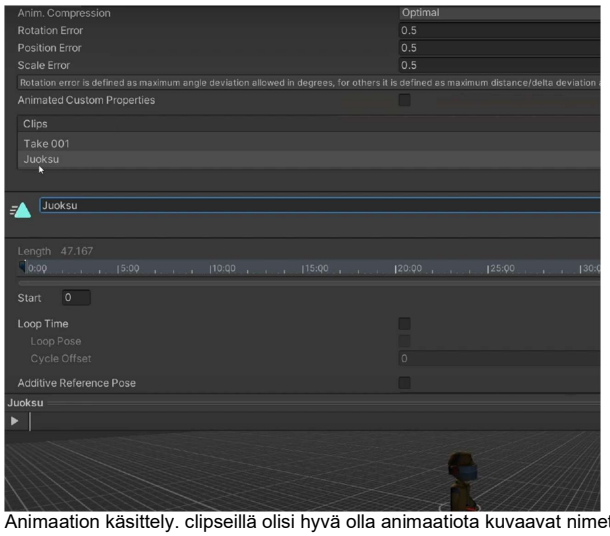
Neljäs luento 26.10.2020

Noora Angelva
R54T19S

Oppimispäiväkirja
Pelimoottorit #2020 Unity

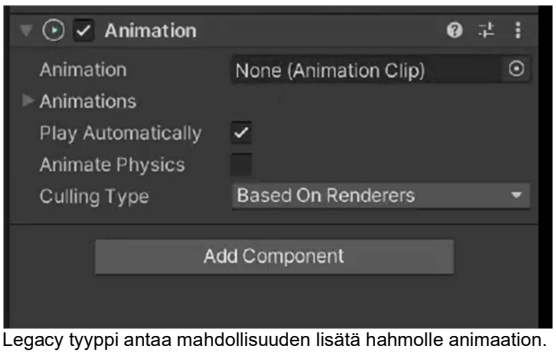
6.10.2020

Luennon aiheita: Legacy animaation käyttöönotto pelihahmolle, liikkumisscripti, luokka CharacterController, yksinkertainen tapa saada liikutettua hahmoa, hahmon pyörittäminen vasen-oikea liikkumisen sijaan, Unity dokumentaatio ja Kameran parentointi.



Animaation käsittely. clipseillä olisi hyvä olla animaatiota kuvaavat nimet.

Yhdestä animaatiosta leikattiin useampi tekeminen kuten juoksu ja idle-tila. Kun animaatiot on tehty. Painamalla applyta unity luo animaatiot. Yksi animaatio voi käydä useampaan hahmoon, jos luusto on hahmoissa sama. Animaatiossa rig tyyppi legacyksi kuten myös hahmoille.



Legacy tyyppi antaa mahdollisuuden lisätä hahmolle animaation.

Jos animaatio esimerkiksi hyppii play moodissa, voi mallinnuksesta olla jäänyt sijainti vääräksi, sen voi korjata hahmolta. Animaation voi laittaa pelaamaan loopissa, kun valitsee wrap moden loopiksi animaation muokkauksesta.

Noora Angelva
R54T19S

Oppimispäiväkirja
Pelimoottorit #2020 Unity

6.10.2020

Hahmon liikkeelle luotiin scripti hahmoliike. Scriptin nimen olisi hyvä kuvastaa koodin toimintoja! Scriptin voi liittää hahmo gameobjectiin add commponentilla.

```
transform.position = transform.position+(transform.forward*0.01f);
```

koodi, jolla hahmo liikkuu eteenpäin. (“oikeaan suuntaan” eteenpäin) ei pelaajan inputilla.

```
float eteenpainliike = Input.GetAxis("Vertical");
transform.position = transform.position+(transform.forward*0.01f*eteenpainliike);
```

pelaaja painaa w näppäintä (liike -> eteenpäin)

Toinen tapa liikuttaa hahmoa on tallentaa muuttujaan characterController.pelaajan inputista otetaan silti yhä liikkeen suunta. CharacterController törmää fysikaalisiin objekteihin josne tulevat vastaan edellisessä tyylissä hahmo olisi kulkenut asioiden läpi. (tarkista että collider on hyvin aseteltu hahmolle)

```
1 reference
CharacterController perttikontrolleri;

// Start is called before the first frame update
0 refer void PerttiLiike.Start()
void Start()
{
    perttikontrolleri = GetComponent<CharacterController>();
}
```

muuttujaan tallennus.

```
perttikontrolleri.Move( transform.forward*0.01f*eteenpainliike );
```

pertti hahmon liikutus.

Hahmolle luodaan painovoima tarkistamalla onko pelaaja maassa, jos ei niin y-akselin arvo laitetaan pienenemään (lasku.y +=) painovoiman avulla ja se kerrotaan Time.deltaTime, joka kertoo kuinka paljon on aikaa kulunut edelisestä framesta. pelaajaa liikutetaan laskun lopputuloksella: Move(lasku).

Hahmolle luodaan hyppy. Hypyssä pitää huomioida ettei tule tupla tai tripla hyppyjä ilmasta. Painovoima on jo valmiina joten täytyy ainoastaan luoda ylöspäin suuntautuva liike.

```
if ( Input.GetButtonDown("Jump") && pelaajaonmaassa )
    {
    pelaajanmaanvetovoima.y = Mathf.Sqrt( hyppyvoima * -3.0f * maanvetovoima );
    }
```

Hyppy koodi.

Noora Angelva
R54T19S

Oppimispäiväkirja
Pelimoottorit #2020 Unity

6.10.2020

Hahmon pyöräytys.

```
float sivuttaisliike = Input.GetAxis("Horizontal");
transform.Rotate( 0f, sivuttaisliike, 0f );
```

Koodi pyörimiselle.

Kameran laittaminen hahmon perään tätä seuraamaan. Asetetaan kamera haluttuun kohtaan pelaajan taakse ja säädetään näkymä mieleiseksi. Tämän jälkeen drag and dropataan main care pelaajahahmon alle (tässä tapauksessa Pertin).

Animaatioiden käyttö koodissa. Luodaan componentille Animation muuttuja.

```
if ( eteenpainliike == 0f )
    {
    perttianimaatiot.Play("Hengailu1");
    }
else
    {
    perttianimaatiot.Play("Juoksu");
    }
```

liikkeestä tapahtuva animaation “soitto”.

Viides luento

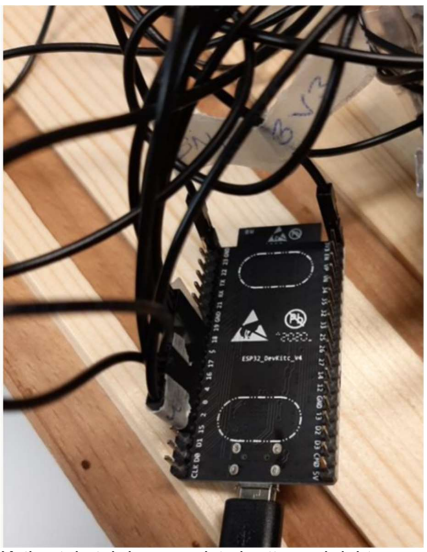
Luennon aiheena on ESP32:sen käyttö peliohjaimena ja sen yhdistys pelimoottori unityyn wifin kautta. Käytän itse ESP32, usb-piuhaa ja arduinon koodaus sovellusta koodin tekemiseen ja ESP:hen laittamiseen.

Olin jo aikaisemmin laittanut ESP32 “lisäosat”. webbi osoite: https://www.espressif.com/dl/package_32_index.json, paikkaan preferences -> additional boards…. Muista laittaa Seria.begin(115200). Unityn ja ESP32 on oltava samassa wifi-verkossa.

Noora Angelva
R54T19S

Oppimispäiväkirja
Pelimoottorit #2020 Unity

6.10.2020



Kytkentöjä tehdessä muista irrottaa usb-johto.

Sarjamonitorista (serial monitor) voi tarkastella “consolelogi” tekstiä.

```
#define SERVICE_UUID        "70f4ca56-7dcd-4154-9312-06124b33c84c"
#define CHARACTERISTIC_UUID "a13de6eb-54da-40ba-8c62-e55279cae349"

enum ButtonMask{
  buttonPoiminta = 0b0001,
  buttonJump = 0b0010,
  buttonKirja = 0b0100,
  buttonPause = 0b1000,
  buttonOikea = 0b0011,
  buttonVasen = 0b0110,
  buttonEteen = 0b1100,
  buttonTaakse = 0b1001
};

int poimintaButton = 15;
int jumpButton = 2;
int kirjaButton = 0;
int pauseButton = 4;
int vasenButton = 16;
int oikeaButton = 17;
int eteenButton = 5;
int taakseButton = 18;
```

nappien maskien ja pinnien määrittely. muista tarkistaa että pinni käy hommaan!

Web socket server demon osoite: https://github.com/larkin/ESP32-Websocket , otetaan zip-tiedosto ja ladataan sketch -> include libary -> add zip library.

Unityn päässä luodaan c#-koodi esim. espOhjain.cs ja luodaan koodille tyhjä gameobject. Koodi tarvitsee kirjastot: System.Net.WebSockets, Systems.Threading ja Systems.

Noora Angelva
R54T19S

Oppimispäiväkirja
Pelimoottorit #2020 Unity

6.10.2020

```
void Start()
{
    var sokettiyhteys = Task.Run(()=>ESPYhteys()  );
    sokettiyhteys.Wait();

}
```

Luodaan startissa asynkroninen yhteys.

```
private static async Task ESPYhteys()
{
    using( ClientWebSocket soketti = new ClientWebSocket() )
        {
        Uri serveriosoite = new Uri("ws://127.0.0.1:80");

        }

}
```

Metodi yhteyden muodostamiselle. ws://arduinoIP:portti

```
var yhteyssource = new CancellationTokenSource();
yhteyssource.CancelAfter(5000);
await soketti.ConnectAsync( serveriosoite, yhteyssource.Token );
Debug.Log("Await ohitettu");
while( soketti.State == WebSocketState.Open )
    {
    Debug.Log("Socketti yhteys luotu");
```

Metodissa (ESPYhteys) tallennetaan arvot yhdistymisestä ja annetaan yhdistymiselle yritys aikaa 5 sekunttia.
Looppi joka pyörii niin kauan kuin yhteys on validi.

Kuudes luento

Unity tulee kaatuilemaan kunnes kuudennen luennon asiat on tehty.

While looppiin luodaan viestin vastaanotto.Viestit otetaan bitti muodossa. Arduinon puolella laitetaan koodi lähettämään tavaraa jos kättely on onnistunut.

```
while ( true )
    {
    ArraySegment<byte> vastaanotetut =
        new ArraySegment<byte>( vastaanotto, offset, dataayhdessapaketissa );
    WebSocketReceiveResults tulokset = await soketti.ReceiveAsync( vastaanotetut );
    }
```

```
await soketti.ReceiveAsync( vastaanotetut, yhteyssource.Token );
```

Async metodi. Muista laittaa yhteyssource.token mukaan!

Kun viesti on vastaanotettu kokonaisuudessaan, “if(tulokset.EndOfMessage), poistutaan loopista ja siirrytään odottamaan

uutta viestiä. Viesti pitää encodata , jotta se olisi luettavaa, “Encoding.UTF8….”.

Viestin lähetys serverille.

```
string viestiserverille = "HELLO";
ArraySegment<byte> viesti =
    new ArraySegment<byte>(Encoding.UTF8.GetBytes(viestiserverille));
```

Viesti ja viestin “kirjekuori”. Viestin muuttaminen byteiksi.

Viesti lähetetään await soketti.SendAsync metodilla.

```
SendAsync(viesti, WebSocketMessageType.Text, true, yhteyssource.Token)
```

(viesti, millainen lähetys esim. kuva, streami tai teksti, true, tokeni)

Jotta peli ei menisi jumiin kun koodi odottelee viestejä, luodaan Thread, joka pyörittää sillä välin peliä. Nimetään Start metodi, joka sisältää soketti yhteyden käynnistyksen uusiksi ja luodaan uusi Start-metodi.

```
void Start()
{
    esphandlerThread = new Thread( TaskinKaynnistys);
    esphandlerThread.Start();

}
```

Käynnistetään entinen start metodi.

Thread.Sleep(10)-toiminnolla luodaan pieni tauko koodiin kun while loopista poistutaan kun viesti on vastaanotettu.

Jos dataa halutaan lähettää arduino json:nin sisässä, pitää hakea arduino koodiin kirjasto “arduinoJson.h”. Json mahdollistaa arvojen käytön esim. muissa luokissa.

```
int potikanarvo = analogRead( potikkapinni );
dataToSend = String(potikanarvo, DEC );
StaticJsonDocument<200> jsondokkari;
jsondokkari["potikanarvo"] = dataToSend;
jsondokkari["napin1arvo"] = "3";
string lahetettavadata;
serializeJson( jsondokkari, lahetettavadata );
webSocketServer.sendData(lahetettavadata);
```

Esimerkki.

Noora Angelva
R54T19S

Oppimispäiväkirja
Pelimoottorit #2020 Unity

6.10.2020

```
public int potikanarvo;
0 references
public string petrintesti;

public Espdatat espidatat = new Espdatat();
```

```
espidatat = JsonUtility.FromJson<Espdatat>( vastaanotettuarvo);
```

Vastaanotettu "chunky" data laitetaan json-muotoon. data tallentuu potikanarvoon ja petrintestiin.

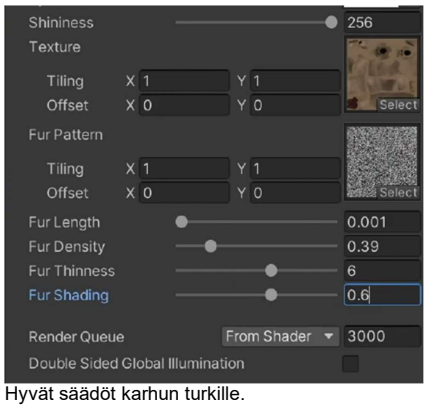
```
{"potikanarvo":"912","napin1arvo":"petrintesti"}
UnityEngine.Debug:Log(Object)
```

Lopputulos.

Seitsemäs luento

Luennon aiheet: Vaihdetaan liikkuminen fysiikkapohjaiseksi, inputtien lukeminen, hungarian notaatio, laitetaan 3d-mallille fur shader käyttöön. Käytetään Petrin jakamaa Animals.zip-pakettia ja sen sisältämiä assetteja. Siirretään halutut assetit projektin asset-kansioon copy-pastella.

Valitaan karhu assetti pelaajaksi. Karhu liikkuu kuitenkin lopuksi fysiikoiden avulla. Githubista löytyy furshader goolesta hakemalla ja tällä saadaan karhusta vähän paremman näköinen. Pura furshader projekti ja ota vain kansiot sieltä ulos ja importtaa omaan kansioosi unity projektissa. Karhun Bear osissa piirretään karhu. Vaihdetaan Bear materiaalissa textureksi jokin furshader assetti. Turkin väritystä ja kiiltoa saa säädettyä colorista ja specularista. furlength: 0.001 on sopivan kokoinen. fur pattern kohtaan kannattaa laittaa furshaderin noise.



Hyvät säädöt karhun turkille.

Noora Angelva
R54T19S

Oppimispäiväkirja
Pelimoottorit #2020 Unity

6.10.2020

Karhun laittaminen hahmoksi tapahtuu ensiksi deaktivoimalla edellisen pelaaja hahmon. Kannattaa nimetä uusi hahmo pelaajaksi, helpottaa tunnistamista. Sen jälkeen peljaalle lisätään rigidbody ja sphere collider. Muista lisätä kamera ja sen scripti! Unity wikistä löytyy hyvä koodeja.

Karhun liikkuminen fysiikoilla. Rigidbody vastaa liikkumisesta. Constrainteista saa säädettyä akselien liikkumista. Liikkeelle tehdään oma scripti ja se liitetään hahmoon. Scriptillä vaikutetaan rigidbodyyn. Rigidbodyn dragiin kannattaa lisätä arvoksi vaikka 10 ja angular drag 15 niin pelaajan impulsseilla on vastaliike.

```
public class KarhunLiikkuminen : MonoBehaviour
{
    [SerializeField]
    1 reference
    float m_karhunNopeus = 200f;

    [SerializeField]
    1 reference
    SerializeField
    float m_karhunkaantyminen = 200f;

    3 references
    Rigidbody m_karhunliikkuminen;
    2 references
    float m_xakseli = 0f;
    2 references
    float m_yakseli = 0f;
    // Start is called before the first frame update
    void Start()
    {
        m_karhunliikkuminen = GetComponent<Rigidbody>();
    }

    // Update is called once per frame
    0 references
    void Update()
    {
        m_xakseli = Input.GetAxis("Horizontal");
        m_yakseli = Input.GetAxis("Vertical");
    }
```

koodin alustus ja inputtien otto.

```
void FixedUpdate()
{
    // ( 0, 0, 1 )*-1*500 -> 0, 0, -500
    m_karhunliikkuminen.AddRelativeForce( Vector3.forward*m_yakseli*500f );
    m_karhunliikkuminen.AddRelativeTorque( Vector3.up*m_xakseli*500f);

}
```

FixedUpdate jolla lisätään voimaa rigidbodyyn -> syntyy liikettä.

Kahdeksas luento

Luennolla käydään läpi pelimoottorin kanssa käytettävää matematiikkaa. Kuten pisteiden laskentaa ja vektorilaskentaa. Luenolla käydään läpi myös peliobjektin instantointia.

Pisteiden laskun tekeminen aloitetaan luomalla SerializeField muuttuja kaikkipisteet (kannattaa laittaa lähyöarvoksi 0). Kerättävät asiat olisi hyvä tehdä myös samanlaisiksi muuttujiksi kuin pisteet. Ajan kannaattaa olla float tyyppinen.

Noora Angelva
R54T19S

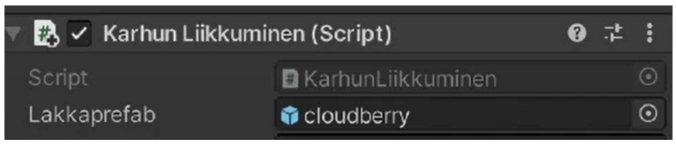
Oppimispäiväkirja
Pelimoottorit #2020 Unity

6.10.2020

```
void Update()
{
    if ( Input.GetKeyDown("q"))
        {
        kaikkimarjat = kaikkimarjat+1;
        }
    if ( Input.GetKeyDown("e"))
        {
        kaikkisienet += 1;
        }
    if ( Input.GetKeyDown("r"))
        {
        aikapisteet = 3.7f;
        }
    kaikkipisteet = kaikkisienet + kaikkimarjat;
    // 9 = 5 + (int)4.2
    kaikkipisteet = kaikkipisteet + (int)aikapisteet;
```

esimerkki pisteiden laskusta. float castataan intiksi.

Seuraavaksi hillan istutusta suhteessa pelaajaan. Gameobjecti luodaan pelaajan scriptiin. Luo hillasta originali kun viet sen kentään.



Muista laittaa asseti kiinni scriptiin.

Luodaan lakka kenttään ja asetetaan se karhuun nähden sen eteen ja maan tasolle raycastilla laskemalla maan pinnan korkeus halutussa pisteessä (koska maa on epätasainen). Koodi on updatessa eli karhu osaa istuttaa hilloja nyt.

```
void Update()
{

    if ( Input.GetKeyDown("y"))
        {
        GameObject luotulakka = (GameObject)Instantiate(m_lakkaprefab);
        Vector3 lakanpaikkakarhunetupuolella = transform.position+(transform.forward*4f);

        RaycastHit rh = new RaycastHit();
        Vector3 raynalkupiste = lakanpaikkakarhunetupuolella + (Vector3.up*100f);
        if ( Physics.Raycast(raynalkupiste, Vector3.down, out rh, 300f ))
            {
             lakanpaikkakarhunetupuolella = rh.point;

            }

        luotulakka.transform.position = lakanpaikkakarhunetupuolella;
        }
```

Kommentoinut [1]: Todella hyvin sinulla näemmä onnistuu myös Unity koodin kanssa toimiminen! Jos jotain pientä parannettavaa, niin melkein kaipasi sellaista lyhyttä yhteenvetoa miten kaikki toimii (esim. kun on tehty pitkä pätkä koodia). Erittäin hyvin olet dokumentoinut kaiken tekemisen. Kuvat selkeyttävät tehdyistä asioista kertovia tekstejä. Etkä päätynyt lasarettiin hyppyä laudalla testatessa :D.