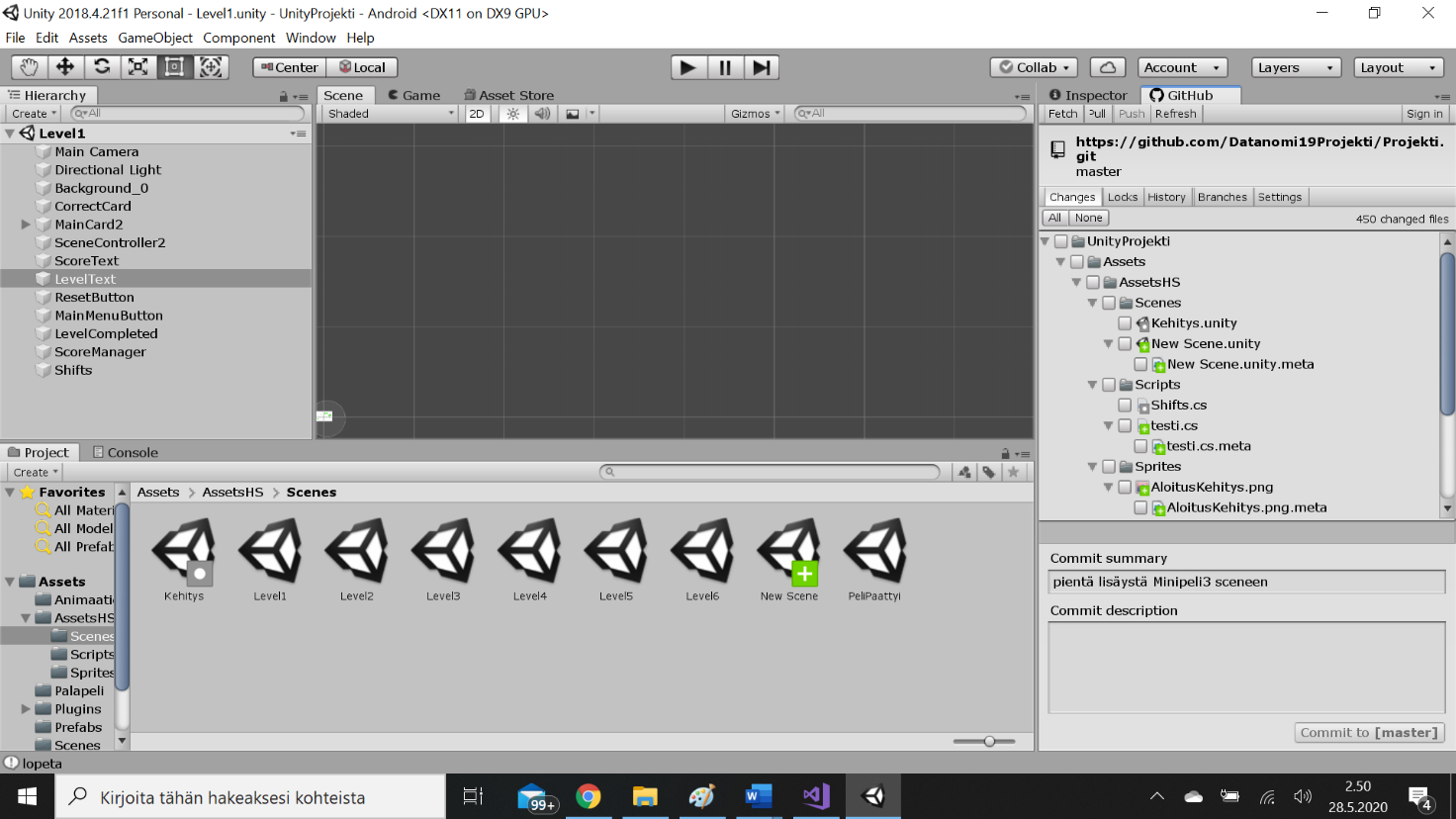
TYÖSKENTELY UNITYSSÄ

ASENNETUT OHJELMAT

Unity, versio 2018.4.21f1

Unity Hub

<https://docs.unity3d.com/Manual/GettingStartedUnityHub.html>

Unityyn tulee oma välilehti, jossa alhaalla Commit.

Microsoft Visual Studio, jolla tehtiin C# scriptit

Android Studio, jonka asensimme vain, jotta peli saadaan Androidille. Ohjelman SDK-tiedostojen sijainti piti asettaa Unityn asetuksiin mutta muuten ohjelmaa ei käytetty. Android Studiolla voi tehdä Android sovelluksia javalla C:llä sekä C++:lla. <https://developer.android.com/studio/index.html>

GitHub Desktop: Unityn Hub ei useinkaan ottanut toimiakseen. Käytin Unityä läihinnä Desktopin kautta. <https://desktop.github.com/>

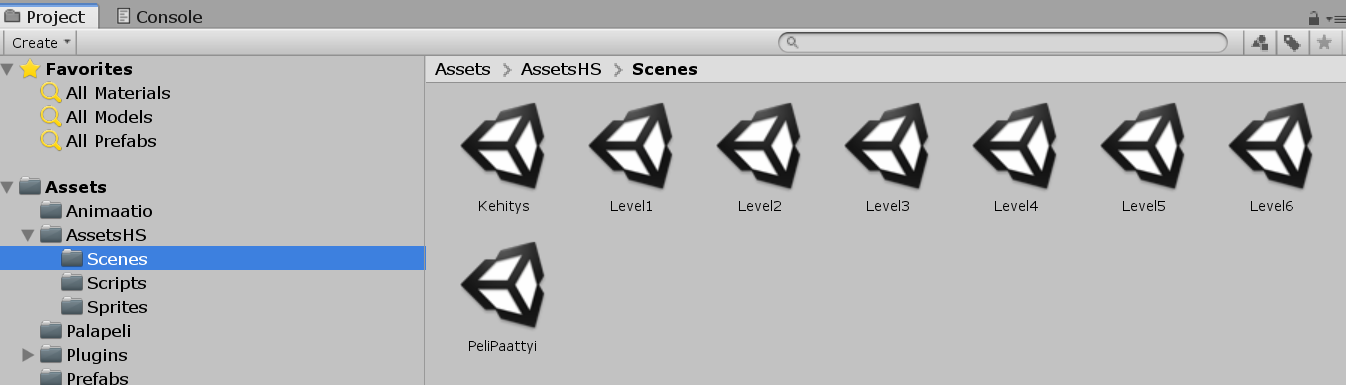
UNITYN JA C# OPISKELU

Aloitimme projektin opiskelemalla Unityä. Lopussa on erilaisia linkkejä. Varsinkin YouTube videot olivat hyödyllisiä. Minulla oli käytössä kaksi kannettavaa. Aluksi laajensin näytön televisioon mutta sitten huomasin, että paljon helpompaa, kun voi toisesta katsoa opetusvideot ja toisella tehdä. Ikkunasta toiseen pomppiminen oli muuten melko työlästä, koska opetusvideoita täytyy pysäyttää koko ajan.

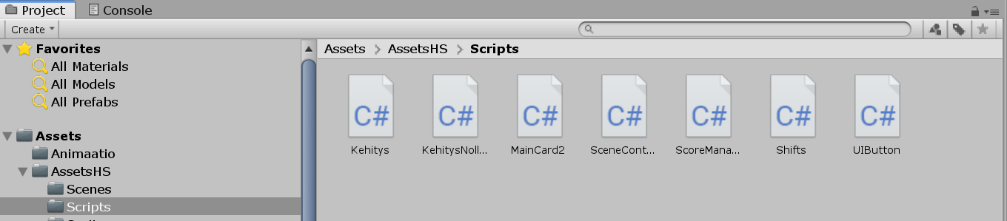
UNITY PROJEKTIN TEKEMINEN

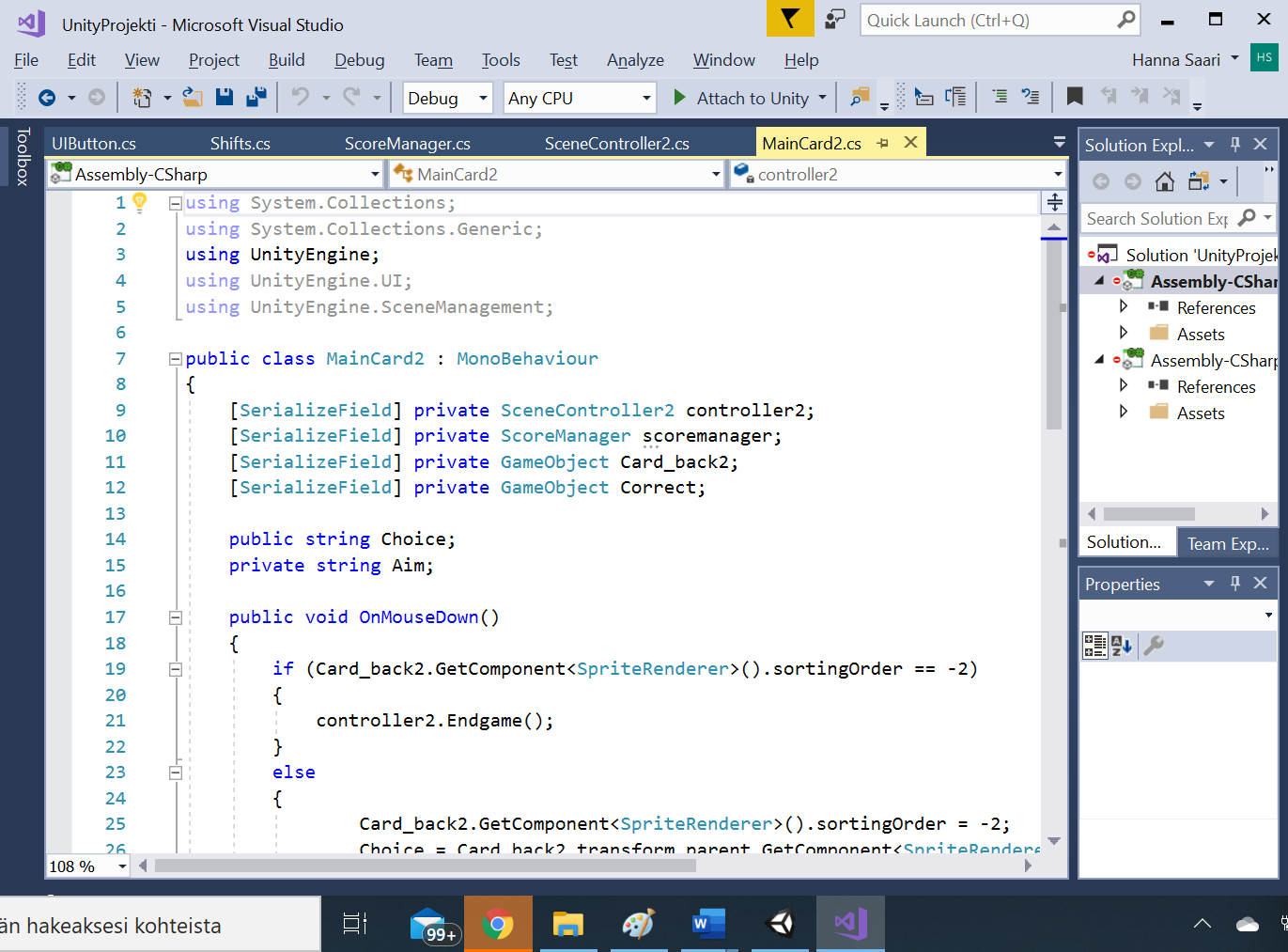
Projektin osat tallennetaan ja järjestetään Unityn Assets-kansioon. Assets-kansiolle voi tehdä alikansioita (kannattaa tehdä vähintään omat kansiot sceneille, scripteille ja kuville (tässä Sprites, koska Unityssä kuvat tuodaan Spriteinä). Teimme Assetsiin omat kansiot kummallekin pelille. Muistipeli on AssetsHS kansiossa.

Pelin eri ikkunat tallennetaan sceneiksi. Muistipelissä on kuusi eri tasoa, joista jokainen tehty omaan Sceneen. Lisäksi on Kehitys, johon tallentuu 6 parasta tulosta ja PeliPaattyi, joka ladataan, kun peli päättyy. Scenessä luodaan ulkoasu ja elementit kuten painikkeet ja tekstikentät.



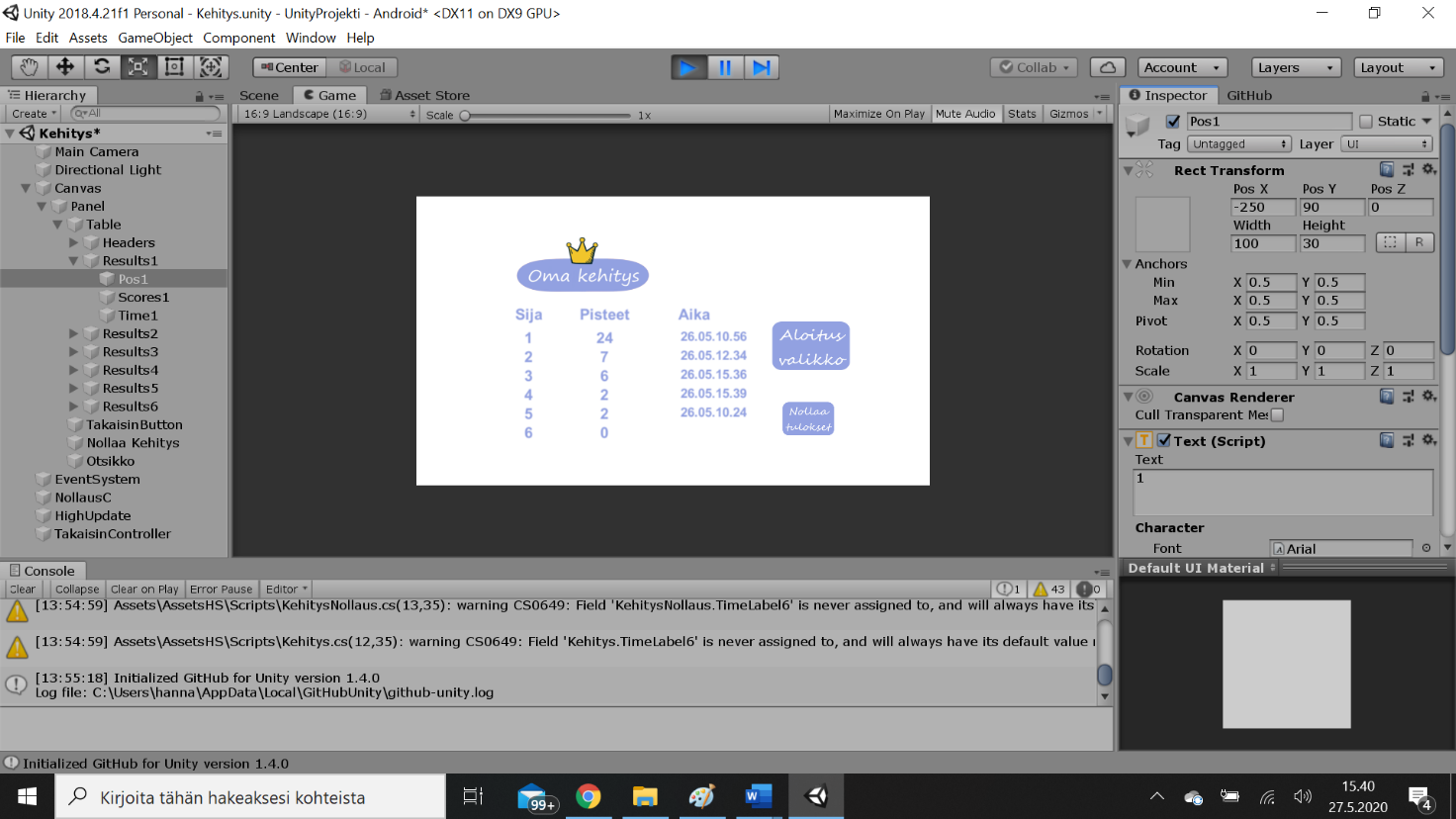
Scriptit tehdään esim. Visual Studio 2017 ohjelmalla (Kun Unity asennetaan, valitaan ohjelma). Asetettu Unityyn siten, että aukeaa automaattisesti, kun scriptejä luodaan tai muokataan. Muistipeliin liittyy 7 scriptiä. Siihen, että scriptit on suunniteltu loogisesti, metodit järkevästi omissa luokissaan, kannattaa käyttää aluksi käyttää aikaa ja tehdä hyvät suunnitelmat.





Visual Studio 2017

Sceneen lisätään ja järjestetään eri elementit Hierarchyssa (oletuksena ohjelman vasemmassa reunassa). Elementtien ominaisuudet asetetaan Inspectorissa (oletuksena ohjelman oikeassa reunassa).

OMA KEHITYS SCENE

Hierarchy

Main Camera Kamera kuvantaa pelin. Unity luo sen automaattisesti, koska ilman sitä peli ei näy. Cameraa voi säätää eri tavoilla. <https://docs.unity3d.com/Manual/CamerasOverview.html>

Canvas: Kaikki elementit asetetaan canvasin päälle. Canvas ei ole pakollinen (esimerkiksi Level1, Level2 jne. on tehty ilman Canvasia) mutta esimerkiksi tekstiruudut vaativat sen. Kun lisäät Sceneen UI (User Interface) elementin kuten tekstin, kuvan, painikkeen tai panelin., Unity tekee Canvasin automaattisesti

Panel Taustakuva on määritelty Panelissa. Taustakuvan voi asettaa myös pelkkänä kuvana ilman Panelia niin kuin Level-sceneissä on tehty.

Table Sija, pisteeet ja aika on asetettu Tablen sisään. Näin niitä voi käsitellä ryhmänä. Esimerkiksi nyt taulukon koko ja sijaintia on helppo muuttaa.

Headers Taulukon otsikot

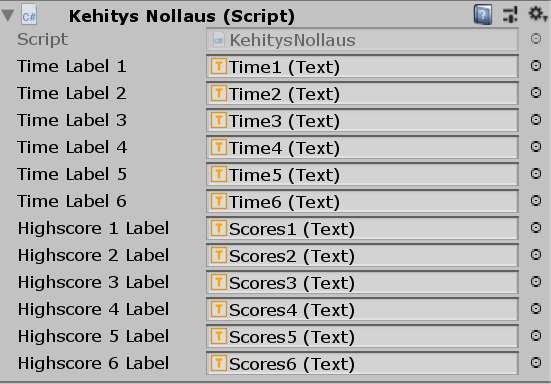
Results1 – 6 Jokainen rivi ryhmitelty omakseen. Helpottaa siirtämistä ja koon muuttamista.

TakaisinButton Painike, jolla pääsee aloitusvalikkoon

Nolla Kehitys Painike, jolla voi nollata tulokset

Otsikko Otsikko kuva

Event System Sisältää elementtien perusominaisuuksia. Unity luo automaattisesti, kun lisätään jokin UI elementti kuten teksti, kuva, painike, canvas tai panel. <https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/EventSystem.html>

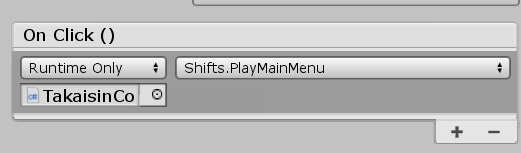
NollausC GameObject, joka sisältää scriptin tulosten nollaamiseen. Scripti yhdistetään tähän ja nyt voidaan määritellä mm. julkiset muuttujat Unityn Inspectorissa. Kehitys taulussa jokaisen pisteen ja ajan näyttämiseen tehty tekstikenttä on määritelty Inspectorissa: Scriptit löydät tiedoston lopusta.

Scriptissä on määritelty kentät TimeLabel1 – TimeLabel2 aikojen näyttämiseen sekä kentät Highscore1Label – Highscore2Label pisteiden näyttämiseen. Unityssä tehdyt tekstikentät yhdistetään näihin kenttiin. Nyt Unity tietää, mikä tekstikenttä liittyy mihinkin koodin muuttujaan ja osaa näyttää tiedot oikein oikeissa paikoissa.

HighUpdate GameObject, joka sisältää scriptin highscorerien päivittämiseen. Tähän on liitetty samalla tavalla pisteiden ja aikojen näyttämiseen tehdyt kentät kuin NollausC:ssä.

TakaisinController

GameObject, joka sisältää scriptin aloitusvalikkoon siirtymiseen. Kun TakaisinController yhdistetään (eli raahataan painikkeen päälle Hierarchyssä) TakaisinButtoniin, Inspectorissa voidaan määritellä On Click (mitä tapahtuu, kun painiketta painetaan):

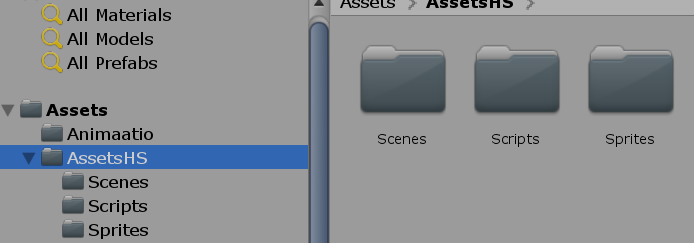


Ensin määritetetään GameObject, joka sisältää vaadittavan scriptin raahaamalla se kenttään (alempi nuoli). Sitten valitaan, mitä scriptissä olevaa metodia käytetään.

OHJE TASOJEN MUOKKAAMISESTA PELIN YLLÄPITÄJÄLLE

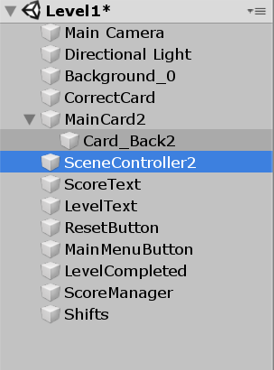
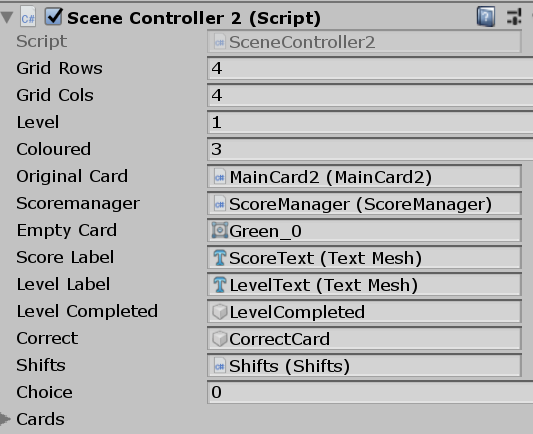
MUISTIPELIN MUOKKAAMINEN

Muistipeliin liittyvät tiedostot löytyvät Unityn kansiosta Assets / AssetsHS



Jokaisen tason scenessä voi muokata seuraavat asiat ilman scriptin avaamista: rivien ja sarakkeiden määrä (Grid Rows ja Grid Cols), Tason numero (Level), muistettavien korttien määrä (Coloured), ”tyhjien” korttien kuva (Empty Card) ja muistettavien korttien kuva (Correct).

Valitse tason Hierarchyssa SceneController2, => Inspector:

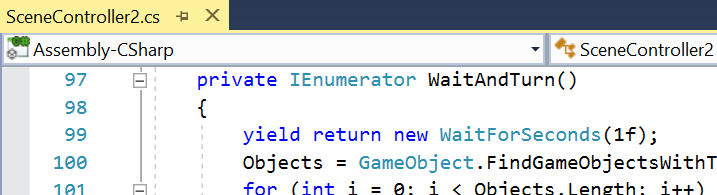


Tasot on tehty kopioimalla. Jos tasojen nimet ovat muotoa Level + tason numero ja tasojen numerot ovat peräkkäisiä (Esim. Level1, Level2, Level3 jne.), siirtyminen tasosta toiseen on hoidettu SceneController2 scriptin lopulla:

SceneManager.LoadScene("Level" + Level);

Taso täytyy luomisen jälkeen lisätä Build Settings-ikkunan Scenes In Build -kenttään (File / build Settings…)

Jos kortit vaihtuvat liian hitaasti tai nopeasti, sen voi muuttaa SceneControl2:ssa WaitAndTurn metodissa. Nyt siinä on 1 sekuntti (f tarkoittaa, että arvo annetaan liukulukuna, joten esim. 1.5 on myös mahdollinen)



Tasossa on oltava seuraavat elementit:



Aloitusvalikko: painike aloitusvalikkoon

Lintu: CorrectCard

Aloita uudestaan: painike, jolla voi aloittaa pelin uudestaan (siirtyy tasolle 1)

Taso suoritettu: tulee esille, kun taso on suoritettu

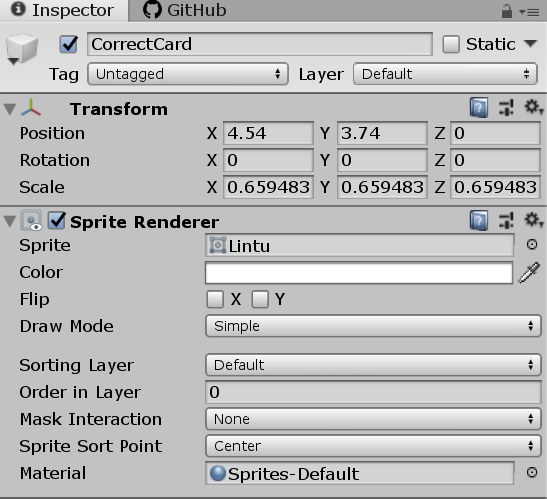
Score: pisteet näkyvät. Muutettu sanaksii Pisteet

Level: taso näkyy, TÄYTYY MUUTTAA SANAKSI Taso

Kortit vasemmassa alalaidassa: kortit jotka kloonataan, kun taso ladataan. Level1 scenessä näet, kuinka iso tila korteille tarvitaan maksimissaan (4x4 ruutu)

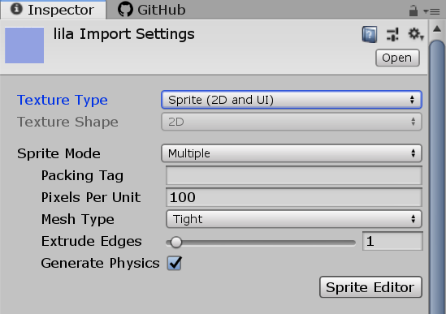
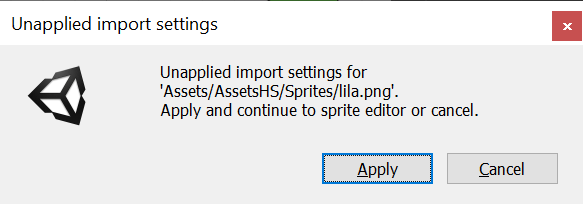
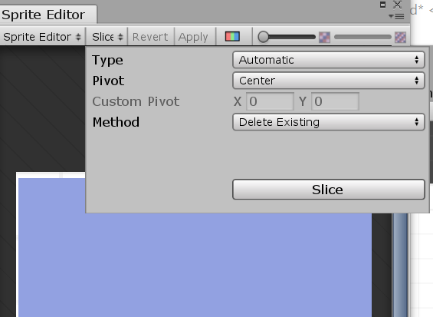
Jos haluat muokata esimerkiksi aloitusvalikkoon- tai aloita uudestaan -painikkeita, ota huomioon, että painikkeita ei ole luotu Prefabseina <https://docs.unity3d.com/Manual/Prefabs.html> ja muutokset täytyy tehdä jokaiselle tasolle erikseen. Peli on luotu siten, että ensin on tehty Level1 valmiiksi ja sitten sitä on kopioitu. Kopioiminen käy nopeasti, koska voit luodessa antaa tason muutettavat tiedot Inspectorissa.

Kortti, joka on tarkoitus muistaa on tallennettu sceneen. Jos haluat muuttaa tason kortin erilaiseksi, vaihda se Inspectorissa: Valitse kuva ensin Hierarchyssä (CorrectCard) ja raahaa haluamasi kuva Sprite Rendererin kenttään Sprite kenttään. <https://docs.unity3d.com/Manual/Sprites.html>



KUVIEN LISÄÄMINEN

Kuvat voi tehdä Paintilläkin ja ne voi tallentaa esim. png muodossa. Kuvat tulee tallentaa Unity projektin Assets-kansioon tai sen alikansioon (Assets hierarkia näkyy Unityssä.). Ne tulevat heti näkyviin ohjelmassa. Kuvatiedostot tarvitsee muokata ennen käyttöä. Valitse kuva ja sitten Inspectorissa Texture Type: Sprite (2D and UI) ja Sprite Mode; Multiple. Siirry sitten Sprite Editoriin Sprite Editor -painikkeella. Hyväksy muutokset. Valitse Sprite Editorissa ylhäältä Slice, paina Slice-painiketta. Hyväksy muutokset ylhäältä Apply-painikkeella. Sulje ikkuna.

Linkkejä:

YOUTUBE VIDEOT

Making an IOS/Android game in UNITY - Beginner Tutorial - #1

<https://www.youtube.com/watch?v=CGleQZVgdN4>

START MENU in Unity (Brackeysillä useita hyviä Unity ja C# videoita. Kotisivut: <http://brackeys.com/>)

<https://www.youtube.com/watch?v=zc8ac_qUXQY>

Unityn käyttöliittymä | Näin teet pelin (osa 1) (Suomenkielinen, hitaasti selitetty, osia yhteensä 10. Holotnan kotisivut: <https://holotna.fi/>)

<https://www.youtube.com/watch?v=lmxTKiDSw8M>

MUUT LINKIT

UNITYN KÄYNNISTYS, NÄKYMÄT, PERUSTEET (SUOMEKSI). Sivustolla myös C# alkeita:

<https://sepeliry.github.io/unity1.html>

Unity UI:lla toteutetun käyttöliittymän skaalauksen ongelmia ja ratkaisuja:

<https://www.theseus.fi/bitstream/handle/10024/168665/Lahtinen_Emilia.pdf?sequence=2>

MOBIILIPELIOHJELMOINTI JA KÄYTTÖLIITTYMÄN TOTEUTUS UNITYLLÄ:

<https://www.theseus.fi/bitstream/handle/10024/149438/Jylha_Samuli.pdf?sequence=1>

ANDROID SOVELLUKSEN KEHITYS:

<https://www.theseus.fi/bitstream/handle/10024/190898/Opinn%E4ytety%F6_Aleksi_J%E4rvenp%E4%E4.pdf?sequence=2>

LEVEL SCRIPTIT

Unityn kolmen ylimmän peruskirjaston lisäksi tarvitaan:

UnityEngine.UI tekstikenttiä varten

UnityEngine.SceneManagement siirtymiä varten

Serializefield tekee private muuttujan näkyväksi. Nyt muuttujaa voidaan käsitellä Unityn Inspectorissa. Näihin muuttujiin yhdistetään niitä vastaavat elementit Unityssä.

SceneController2 on toinen luokka, johon scriptissä viitataan. Se täytyy alustaa.

MAINCARD2

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class MainCard2 : MonoBehaviour

{

[SerializeField] private SceneController2 controller2;

[SerializeField] private ScoreManager scoremanager;

[SerializeField] private GameObject Card\_back2;

[SerializeField] private GameObject Correct;

public string Choice;

private string Aim;

public void OnMouseDown() // Suoritetaan, kun GameObjektia klikataan

{

//Jos kortti on jo asetettu taakse, peli lopetetaan. Siihen liittyvä scripti sijaitsee eri luokassa

if (Card\_back2.GetComponent<SpriteRenderer>().sortingOrder == -2)

{

controller2.Endgame();

}

else

{

/\*kun korttia klikataan, tulee näkyviin kortin sisäpuoli ja takapuoli eli Card\_back2 asetetaan taakse. Jos kortin sisäpuolen kuvan nimi on sama kuin CardCorrectin, pelaaja saa pisteen. Piste tallennetaan eri luokkaan choice-muuttujaan. Tämän jälkeen siirrytään eri luokan Scores metodiin.\*/

Card\_back2.GetComponent<SpriteRenderer>().sortingOrder = -2;

Choice = Card\_back2.transform.parent.GetComponent<SpriteRenderer>().sprite.name;

Aim = Correct.GetComponentInChildren<SpriteRenderer>().sprite.name;

if (Choice == Aim)

{

controller2.choice = 1;

controller2.Scores();

}

else

{

controller2.Endgame();

}

}

}

}

SCENECONTROLLER2

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class SceneController2 : MonoBehaviour

gridRows = montako riviä kortteja tehdään

gridCols = montako saraketta kortteja tehdään

Level = Taso

coloured = ”värillisten” eli muistettavien korttien määrä

Asetetaan originalCard (liitetään aluksi tehty kortti Unityssä tähän kenttään. Kortissa MainCard2 luokan scripti liitettynä)

Liitetään ScoreManager nimellä scoremanager

{

[SerializeField] private int gridRows = 2;

[SerializeField] private int gridCols = 2;

[SerializeField] private int Level;

[SerializeField] private int coloured;

[SerializeField] private MainCard2 originalCard;

[SerializeField] private ScoreManager scoremanager;

[SerializeField] private Sprite EmptyCard;

[SerializeField] private TextMesh ScoreLabel;

[SerializeField] private TextMesh LevelLabel;

[SerializeField] private GameObject LevelCompleted;

[SerializeField] private GameObject Correct;

[SerializeField] private Shifts shifts;

int[] numbers;

float offsetX = 2f;

float offsetY = 2f;

int score;

int HighScore;

int beginscore;

public int choice;

public List<GameObject> cards = new List<GameObject>();

GameObject[] Objects;

//void Start() sisältämät toiminnot suoritetaan ohjelman alussa siinä järjestyksessä kuin ne on kirjoitettu

private void Start()

{

//Haetaan PlayerPrefseihin tallennettu pistemäärä nimellä GameScore (jos kyseessä on muu taso kuin taso 1, edelliseltä tasolta saadut pisteet säilyvät). Asetetaan alkupisteet beginscoreksi. Tätä tarvitaan, jotta peli osaa siirtyä seuraavalle tasolle. Kun pisteet = beginscore + coloured (eli muistettavien korttien määrä), seuraava taso ladataan.

score = PlayerPrefs.GetInt("GameScore");

beginscore = score;

Päivitetään pisteet. Sijaitsevat ScoreLabelin tekstissä. Päivitetään taso.

”Taso suoritettu” teksti asetetaan passiiviseksi.

Suoritetaan CreateTable metodi jonka jälkeen WaitAndTurn (metodi on coroutine: sisältää yield komennon, joka pysäyttää pelin kulun hetkeksi)

Create Table: Korttien lisääminen määrätysti

ScoreLabel.text = "Pisteet: " + score;

LevelLabel.text = "Taso: " + Level;

LevelCompleted.SetActive(false);

CreateTable();

StartCoroutine(WaitAndTurn());

}

private void CreateTable()

{

int CardAmount = gridCols \* gridRows;

int[] numbers = new int[CardAmount];

numbers = ShuffleArray(numbers);

// valitaan tehdyn kortin sijainti aloitussijainniksi

Vector3 startPos = originalCard.transform.position;

//Luodaan sarakkeet (ulompi for) ja rivit, lisätään kortit

for (int i = 0; i < gridCols; i++)

{

for (int j = 0; j < gridRows; j++)

{

MainCard2 card;

if (i == 0 && j == 0)

{

card = originalCard;

}

//Kloonataan kortti (jos se ei ole ensimmäinen kortti, joka on luotu valmiiksi

else

{

card = Instantiate(originalCard) as MainCard2;

}

//joka toiseen kortin indeksiin vaikuttaa gridCols, joka toiseen vain i: Jos ruudukko 4x2 kokoinen tulokset ovat: 0, 4, 1, 5, 2, 6, 3, 7

int index = j \* gridCols + i;

int id = numbers[index];

if (id < coloured)

{

//vaihdetana annettu määrä kortteja

card.GetComponent<SpriteRenderer>().sprite = Correct.GetComponentInChildren<SpriteRenderer>().sprite;

}

else

{

card.GetComponent<SpriteRenderer>().sprite = EmptyCard;

}

//Asetetaan korttien välit

float posX = (offsetX \* i) + startPos.x;

float posY = (offsetY \* j) + startPos.y;

//Asetetaan kortin uusi sijainti

card.transform.position = new Vector3(posX, posY, startPos.z);

}

}

}

//sekoitetaan numbers-listan numerot eli korttien indeksien järjestys

private int[] ShuffleArray(int[] numbers)

{

int CardAmount = gridCols \* gridRows;

for (int n = 0; n < CardAmount; n++)

{

numbers[n] = n;

}

//tehdään uusi lista, johon kopioidaan numbers-listan numerot.

int[] newArray = numbers.Clone() as int[];

//silmukka suoritetaan niin monta kertaa kuin alkioita

for (int i = 0; i < newArray.Length; i++)

{

//tmp tarvitaan, jotta sama alkio tulee vain kerran

int tmp = newArray[i];

//r saa satunnaiset arvot väliltä 0 – listan pituus

int r = Random.Range(i, newArray.Length);

//newArrayn alkio i vaihtaa paikkaa newArrayn alkion r kanssa

newArray[i] = newArray[r];

newArray[r] = tmp;

}

return newArray;

}

//odotetaan hetki ja käännetään kortit

private IEnumerator WaitAndTurn()

{

//odotetaan 1 sekuntia (f tarkoittaa, että tyyppi on float)

yield return new WaitForSeconds(1f);

//haetaan GameObjectit, joiden Tag = card

Objects = GameObject.FindGameObjectsWithTag("card");

//lisätään haetut GameObjectit listaan

for (int i = 0; i < Objects.Length; i++)

{

cards.Add(Objects[i]);

}

//Vaihdetaan korttien etupuolen Order In Layer -1 eli Card\_Back tulee näkyviin

foreach (GameObject card in cards)

{

card.GetComponentInChildren<SpriteRenderer>().sortingOrder = -1;

}

}

//Pisteiden lisäys ja lataus & tallennus Unityn PlayerPrefs luokkaa käyttäen.

public void Scores()

{

if (choice == 1)

{

score = PlayerPrefs.GetInt("GameScore");

score++;

ScoreLabel.text = "Score: " + score;

PlayerPrefs.SetInt("GameScore", score);

}

//jos pisteitä on tarpeeksi, siirrytään seuraavalle tasolle

if (score == beginscore+coloured)

{

StartCoroutine(NextLevel());

}

}

//coroutine metodit voidaan pysäyttää yield komennon avulla määrätyksi hetkeksi.

private IEnumerator NextLevel()

{

foreach (GameObject card in cards)

{

card.SetActive(false); //asetetaan kortit passiivisiksi

}

LevelCompleted.SetActive(true); //asetetaan teksti ”Taso suoritettu” aktiiviseksi

yield return new WaitForSeconds(4f); //odotetaan 4 sekuntia

Level++; //kasvatetaan tason numeroa

SceneManager.LoadScene("Level" + Level); //siirrytään tasolle

}

}

public void Endgame()

{

SceneManager.LoadScene("PeliPaattyi");

scoremanager.UpdateHighScore();

scoremanager.ResetScore();

}

}

UIBUTTON

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class UIButton : MonoBehaviour

{

//Voidaan Unityn Inspectorissa asettaa GameObject, jossa koodi sijaitsee

[SerializeField] private GameObject targetObject;

//Voidaan Unityn Inspectorissa asettaa metodin nimi

[SerializeField] private string targetMessage;

public Color highlightColor = Color.cyan;

public void OnMouseOver()

{

SpriteRenderer sprite = GetComponent<SpriteRenderer>();

if(sprite != null)

{

sprite.color = highlightColor;

}

}

public void OnMouseExit()

{

SpriteRenderer sprite = GetComponent<SpriteRenderer>();

if (sprite != null)

{

sprite.color = Color.white;

}

}

public void OnMouseDown()

{

transform.localScale = new Vector3(1.2f, 1.2f, 1.0f);

}

public void OnMouseUp()

{

transform.localScale = new Vector3(1f, 1f, 1f);

if(targetObject != null)

{

targetObject.SendMessage(targetMessage);

}

}

}

KEHITYKSEN SCRIPTIT

Kehitys: Näytetään 6 parasta tulosta ja aika, jolloin ne on saavutettu. Tulokset ja ajat luodaan SceneController2 scriptissä.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

UnityEngine.Ui tarvitaan aina, kun käsitellään tekstikenttiä.

Awake suoritetaan heti, kun Kehitys ladataan.

Pisteet ja ajat tallennetaan ja ladataan PlayerPrefs luokan avulla.

public class Kehitys : MonoBehaviour

{

[SerializeField] private Text TimeLabel1;

[SerializeField] private Text TimeLabel2;

[SerializeField] private Text TimeLabel3;

[SerializeField] private Text TimeLabel4;

[SerializeField] private Text TimeLabel5;

[SerializeField] private Text TimeLabel6;

[SerializeField] private Text Highscore1Label;

[SerializeField] private Text Highscore2Label;

[SerializeField] private Text Highscore3Label;

[SerializeField] private Text Highscore4Label;

[SerializeField] private Text Highscore5Label;

[SerializeField] private Text Highscore6Label;

int highscore1, highscore2, highscore3, highscore4, highscore5, highscore6;

string time1, time2, time3, time4, time5, time6;

private void Awake()

{

highscore1 = PlayerPrefs.GetInt("HighScore1");

Highscore1Label.text = highscore1.ToString();

highscore2 = PlayerPrefs.GetInt("HighScore2");

Highscore2Label.text = highscore2.ToString();

highscore3 = PlayerPrefs.GetInt("HighScore3");

Highscore3Label.text = highscore3.ToString();

highscore4 = PlayerPrefs.GetInt("HighScore4");

Highscore4Label.text = highscore4.ToString();

highscore5 = PlayerPrefs.GetInt("HighScore5");

Highscore5Label.text = highscore5.ToString();

highscore6 = PlayerPrefs.GetInt("HighScore6");

Highscore6Label.text = highscore6.ToString();

time1 = PlayerPrefs.GetString("Time1");

TimeLabel1.text = time1;

time2 = PlayerPrefs.GetString("Time2");

TimeLabel2.text = time2;

time3 = PlayerPrefs.GetString("Time3");

TimeLabel3.text = time3;

time4 = PlayerPrefs.GetString("Time4");

TimeLabel4.text = time4;

time5 = PlayerPrefs.GetString("Time5");

TimeLabel5.text = time5;

time6 = PlayerPrefs.GetString("Time6");

TimeLabel6.text = time6;

}

}

KehitysNollaus: Asetetaan pisteiden ja aikojen arvoksi 0.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class KehitysNollaus : MonoBehaviour

{

[SerializeField] private Text TimeLabel1;

[SerializeField] private Text TimeLabel2;

[SerializeField] private Text TimeLabel3;

[SerializeField] private Text TimeLabel4;

[SerializeField] private Text TimeLabel5;

[SerializeField] private Text TimeLabel6;

[SerializeField] private Text Highscore1Label;

[SerializeField] private Text Highscore2Label;

[SerializeField] private Text Highscore3Label;

[SerializeField] private Text Highscore4Label;

[SerializeField] private Text Highscore5Label;

[SerializeField] private Text Highscore6Label;

int highscore1, highscore2, highscore3, highscore4, highscore5, highscore6;

string time1, time2, time3, time4, time5, time6;

public void ClearHighScore()

{

PlayerPrefs.Get hakee tiedon. PlayerPrefs.Set asettaa tiedon. PlayerPrefsiin voi tallentaa int, string ja float tyyppistä tietoa.

PlayerPrefs.SetInt("HighScore1", 0);

highscore1 = PlayerPrefs.GetInt("HighScore1");

Highscore1Label.text = highscore1.ToString();

PlayerPrefs.SetInt("HighScore2", 0);

highscore2 = PlayerPrefs.GetInt("HighScore2");

Highscore2Label.text = highscore2.ToString();

PlayerPrefs.SetInt("HighScore3", 0);

highscore3 = PlayerPrefs.GetInt("HighScore3");

Highscore3Label.text = highscore3.ToString();

PlayerPrefs.SetInt("HighScore4", 0);

highscore4 = PlayerPrefs.GetInt("HighScore4");

Highscore4Label.text = highscore4.ToString();

PlayerPrefs.SetInt("HighScore5", 0);

highscore5 = PlayerPrefs.GetInt("HighScore5");

Highscore5Label.text = highscore5.ToString();

PlayerPrefs.SetInt("HighScore6", 0);

highscore6 = PlayerPrefs.GetInt("HighScore6");

Highscore6Label.text = highscore6.ToString();

PlayerPrefs.SetString("Time1", " ");

time1 = PlayerPrefs.GetString("Time1");

TimeLabel1.text = time1;

PlayerPrefs.SetString("Time2", " ");

time2 = PlayerPrefs.GetString("Time2");

TimeLabel2.text = time2;

PlayerPrefs.SetString("Time3", " ");

time3 = PlayerPrefs.GetString("Time3");

TimeLabel3.text = time3;

PlayerPrefs.SetString("Time4", " ");

time4 = PlayerPrefs.GetString("Time4");

TimeLabel4.text = time4;

PlayerPrefs.SetString("Time5", " ");

time5 = PlayerPrefs.GetString("Time5");

TimeLabel5.text = time5;

PlayerPrefs.SetString("Time6", " ");

time6 = PlayerPrefs.GetString("Time6");

TimeLabel6.text = time6;

}

}

Shifts: Siirtymät

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

UnityEngine.SceneManagement tarvitaan, koska siirrytään SceneManagerin kautta toisiin sceneihin. Scenet täytyy olla listattuna Unityn Build Settingsissä (File / Build Settings)

Debug.Log antaa tekstin Consoleen. Tätä käytin peliä tehdessä, koska näin tiesin, että koodi kulkeutuu tähän asti ja komennot on suoritettu.

public class Shifts : MonoBehaviour

{

public void PlayRestart()

{

SceneManager.LoadScene("Level1");

}

public void PlayKehitys()

{

SceneManager.LoadScene("Kehitys");

}

public void PlayMainMenu()

{

SceneManager.LoadScene("MainMenu");

}

public void QuitGame()

{

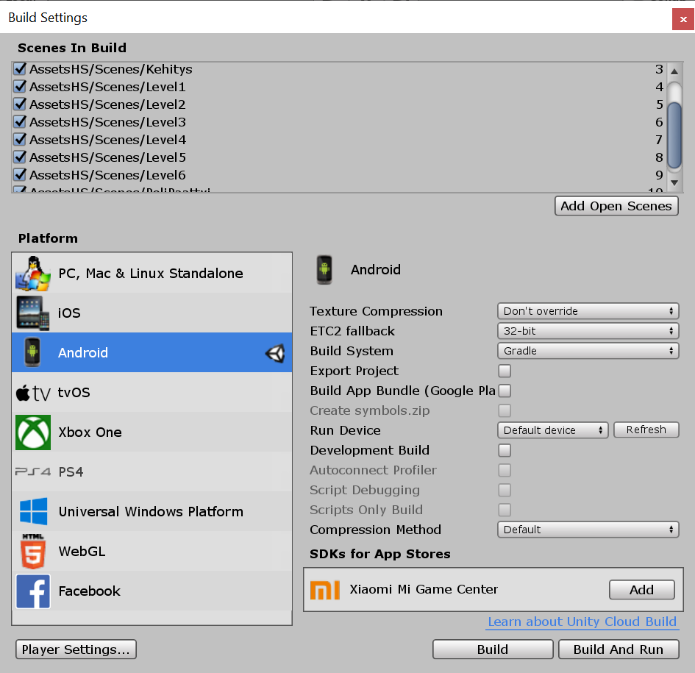
Debug.Log("lopeta");

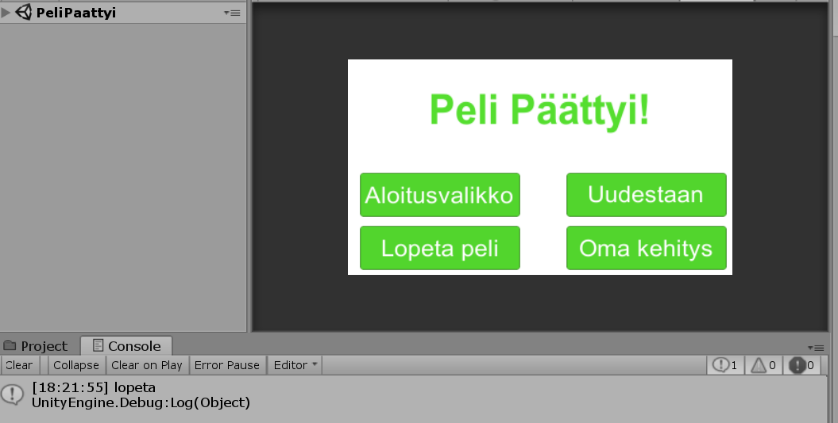
Application.Quit();

}

}

Unityn Build Settings (File / Build Settings): Debug.Logissa oleva teksti tulee Consoleen:





SCOREMANAGER

ResetScore: nollataan pelissä saavutetut pisteet (suoritetaan, kun peli loppuu) ja tallennetaan pisteet PlayerPrefsiin nimellä Gamescore

UpdateHighScore: katsotaan, kuinka hyvä tulos on, tallennetaan tarvittaessa PlayerPrefsiin pisteet ja aika, jolloin tulos saavutettu. Jos tulos tallennetaan, tulosten järjestys muuttuu: esimerkiksi, jos uusi tulos on 3. paras niin entinen 3. sija on nykyinen 4. sija, entinen 4. sija nykyinen 5. jne.

public class ScoreManager : MonoBehaviour

{

[SerializeField] private SceneController2 controller2;

string time;

int highscore1, highscore2, highscore3, highscore4, highscore5, highscore6;

string time1, time2, time3, time4, time5, time6;

int score;

public void ResetScore()

{

score = 0;

PlayerPrefs.SetInt("GameScore", score);

}

public void UpdateHighScore()

{

score = PlayerPrefs.GetInt("GameScore");

highscore1 = PlayerPrefs.GetInt("HighScore1");

highscore2 = PlayerPrefs.GetInt("HighScore2");

highscore3 = PlayerPrefs.GetInt("HighScore3");

highscore4 = PlayerPrefs.GetInt("HighScore4");

highscore5 = PlayerPrefs.GetInt("HighScore5");

highscore6 = PlayerPrefs.GetInt("HighScore6");

time1 = PlayerPrefs.GetString("Time1");

time2 = PlayerPrefs.GetString("Time2");

time3 = PlayerPrefs.GetString("Time3");

time4 = PlayerPrefs.GetString("Time4");

time5 = PlayerPrefs.GetString("Time5");

time6 = PlayerPrefs.GetString("Time6");

Sisäkkäisten if-lauseiden avulla katsotaan, kuinka hyvä tulos on ja tallennetaanko se PlayerPrefseihin. Ensin katsotaan, onko tulos pienempi kuin PlayerPrefseissa tallennettu 6. sija. Sitten verrataan 5:een ja jos näin on, asetetaan HighScrore6:n ja Time6:n arvo.

if (score < highscore6)

{

int lastscore = score;

}

else

{

if (score < highscore5)

{

highscore6 = score;

PlayerPrefs.SetInt("HighScore6", score);

time6 = System.DateTime.UtcNow.ToLocalTime().ToString("dd.MM. HH:mm");

PlayerPrefs.SetString("Time6", time6);

}

else

{

if (score < highscore4)

{

highscore6 = highscore5;

PlayerPrefs.SetInt("HighScore6", highscore6);

highscore5 = score;

PlayerPrefs.SetInt("HighScore5", score);

time6 = time5;

PlayerPrefs.SetString("Time6", time6);

time5 = System.DateTime.UtcNow.ToLocalTime().ToString("dd.MM.HH:mm");

PlayerPrefs.SetString("Time5", time5);

}……..jne….