COURSE: COMP 4478

ASSIGNMENT 2 – POKEMON MEMORY GAME IN UNITY

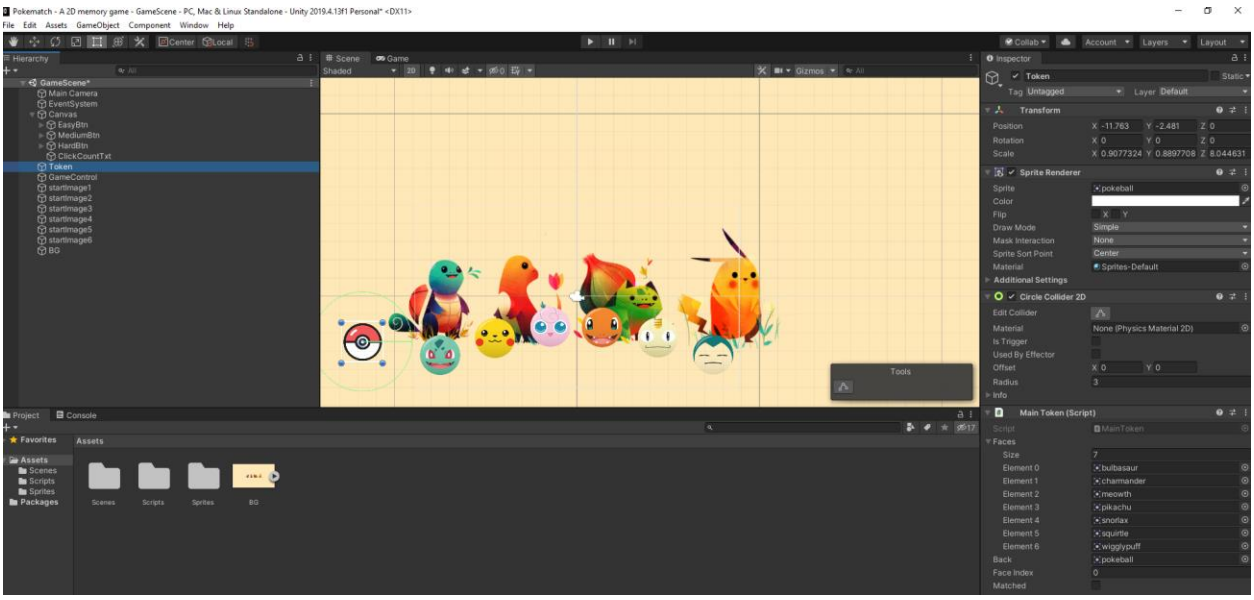
NAME: DHAVAL THANKI

STUDENT NUMBER: 0871347

Objective

In this assignment we a re tasked with created a card matching game that is programmed using unity2D. The game is simple, there are three levels, each level will have an increasingly higher difficulty by adding the number of card pairs to match. Here are links the C# scripts used within this work. GameControls.cs, MainToken.cs.

Unity GUI



Here is a screenshot of the unity GUI, I followed the tutorial and created numerous game objects, for the longest time I was stuck trying to figure out how to show these sprites on the camera, finally figured out that I had to move the Z to a negative value. On the bottom right you can see each of the elements I use to populate my faces array.

Script for main functionality in the game

```
void StartGame()
{
    int startTokenCount = numofTokens;
    float xPosition = -6.5f;
    float yPosition = yStart;
    int row = 1;

    float ortho = Camera.main.orthographicSize / 2.0f;
    for (int i = 1; i < startTokenCount + 1; i++)
    {
        shuffleNum = rnd.Next(0, (numofTokens));
        var temp = Instantiate(token, new Vector3(
            xPosition, yPosition, 0),
            Quaternion.identity);
        temp.GetComponent<MainToken>().faceIndex = faceIndexes[shuffleNum];
        temp.transform.localScale =
        new Vector3(ortho / tokenScale, ortho / tokenScale, 0);
        faceIndexes.Remove(faceIndexes[shuffleNum]);
        numofTokens--;
        xPosition = xPosition + 4f;
        if (i % 4 < 1)
        {
            yPosition = yPosition + yChange;
            xPosition = -6.5f;
            row++;
        }
    }
    token.SetActive(false);
}
```

Here is a screenshot of the StartGame function, it is one of the most important in the GameControl.cs file. The function uses the positioning set globally in the script to clone the initial token.

The initial token is cloned as many times as numOfTokens will allow it, that variable is set accordingly for each game difficulty level, i.e easy, medium or hard.

The for loop at the bottom is to set the positions when moving down rows so we can have a crisp and even looking board of cards at the end of the day.

The screenshots on the side and below represent the tokens and how they will be matched, they're matched (checked) using another function called checkTokens but the TokenUp and TokenDown functions allow me to store the immediate value into a temporary token to make it easy to compare essentially.

```
public void TokenDown(MainToken tempToken)
{
    if (tokenUp1 == tempToken)
    {
        tokenUp1 = null;
    }
    else if (tokenUp2 == tempToken)
    {
        tokenUp2 = null;
    }
}
```

```
public bool TokenUp(MainToken tempToken)
{
    bool flipCard = true;
    if (tokenUp1 == null)
    {
        tokenUp1 = tempToken;
    }
    else if (tokenUp2 == null)
    {
        tokenUp2 = tempToken;
    }
    else
    {
        flipCard = false;
    }
    return flipCard;
}
```

The TokenUp function will return a Boolean, the reason for this is to check if the card is already flipped, this is useful because then we can limit how many other cards can be flipped, this is seen in the MainToken.cs file where we don't allow the player to flip more cards if the matched bool is false.

```
public void CheckTokens()
{
    clickCount++;
    clickCountTxt.text = clickCount.ToString();
    if (tokenUp1 != null && tokenUp2 != null &&
        tokenUp1.faceIndex == tokenUp2.faceIndex)
    {
        tokenUp1.matched = true;
        tokenUp2.matched = true;
        tokenUp1 = null;
        tokenUp2 = null;
    }
}
```

Here is a screenshot of CheckTokens, this is also where the score is incremented based on how many cards the player flips.

```
public void HardSetup()
{
    HideButtons();
    tokenScale = 12;
    yStart = 3.8f;
    numofTokens = 24;
    yChange = -1.5f;
    StartGame();
}

public void MediumSetup()
{
    HideButtons();
    tokenScale = 8;
    yStart = 3.4f;
    numofTokens = 16;
    yChange = -2.2f;
    StartGame();
}

public void EasySetup()
{
    HideButtons();
    StartGame();
}
```

```
void onEnable()
{
    easyBtn.onClick.AddListener(() => EasySetup());
    mediumBtn.onClick.AddListener(() => MediumSetup());
    hardBtn.onClick.AddListener(() => HardSetup());
}
```

These screenshots represent how I set up different levels by adding onClick triggers to start a certain scenario based on the button that the player clicks. Obviously, these are also connected within the GUI to GameControl.cs.

A huge problem with my setup is that there is an error when I start a hard or medium game, the error is an IndexOutOfBounds error with the faces array, this randomly breaks the game and disables one to properly play/finish it, I still don't understand why it does that.

Script for creating main card token

MainToken.cs | GameControl.cs

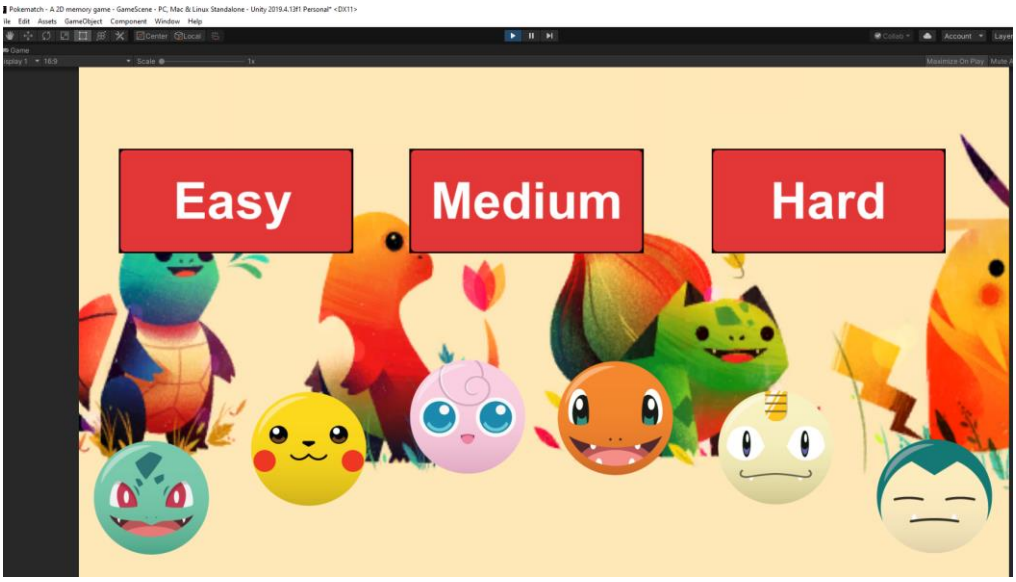
Assets > Scripts > MainToken.cs

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

public class MainToken : MonoBehaviour
{
    GameObject gameControl;
    SpriteRenderer spriteRenderer;
    public Sprite[] faces;
    public Sprite back;
    public int faceIndex = 100;
    public bool matched = false;

    public void OnMouseDown()
    {
        if (matched == false)
        {
            GameControl controlScript = gameControl.GetComponent<GameControl>();
            if (spriteRenderer.sprite == back)
            {
                if (controlScript.TokenUp(this))
                {
                    spriteRenderer.sprite = faces[faceIndex];
                    controlScript.CheckTokens();
                }
            }
            else
            {
                spriteRenderer.sprite = back;
                controlScript.TokenDown(this);
            }
        }
    }

    private void Awake()
    {
        gameControl = GameObject.Find("GameControl");
        spriteRenderer = GetComponent<SpriteRenderer>();
    }
}

```

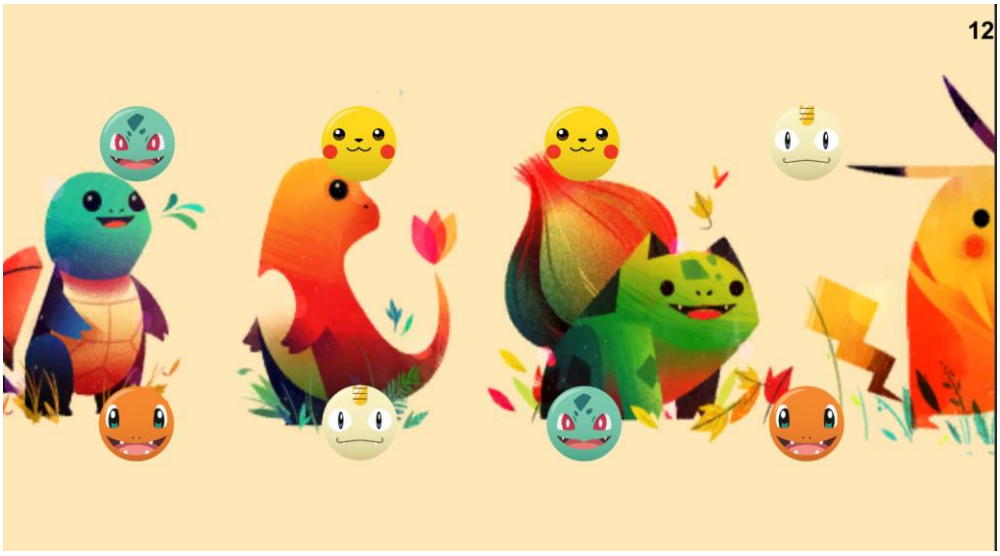
In the MainToken.cs we control the flow of events when a player flips a card, since each of the cards are clones of the initial token we can check if there are two cards that are already flipped or if there is only one card that is flipped, this will limit the player from looking at all the cards at once.

Results

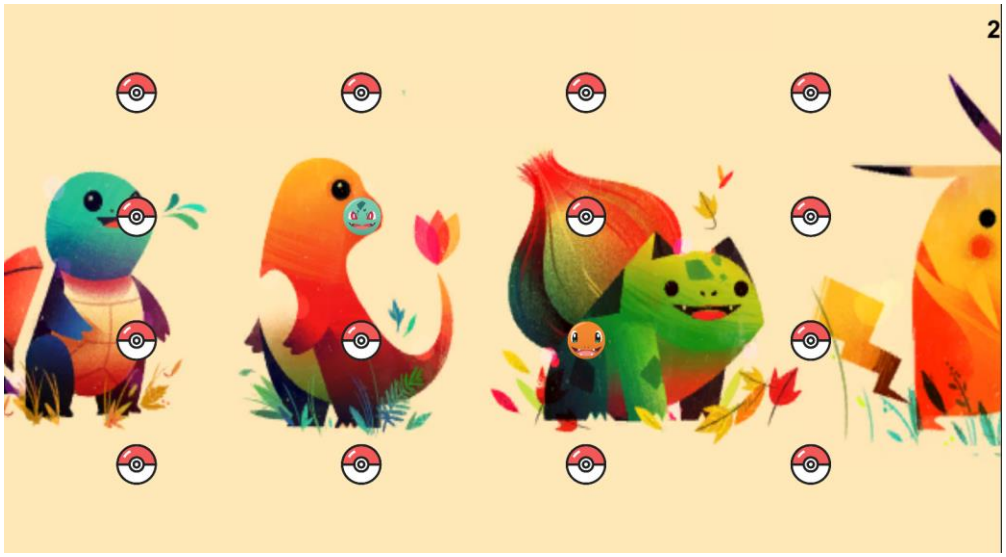
Initial screen



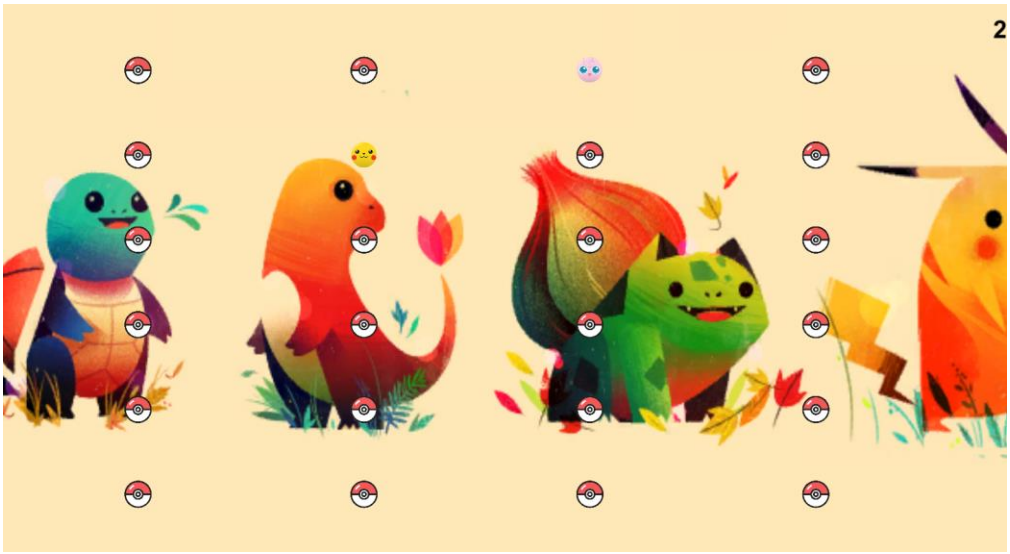
Selecting Easy difficulty



Selecting Medium difficulty



Selecting Hard difficulty



Resources Used

Tutorial on the memory game itself P1 – https://youtu.be/1vVdCXjXja4

Tutorial on the memory game itself P2 – https://youtu.be/bdOeOvvOKl8

Background graphic – https://wallup.net/wp-content/uploads/2016/01/10017-minimalism-Pikachu-Bulbasaur-Charmander-Squirtle-sketches-Pokemon.png