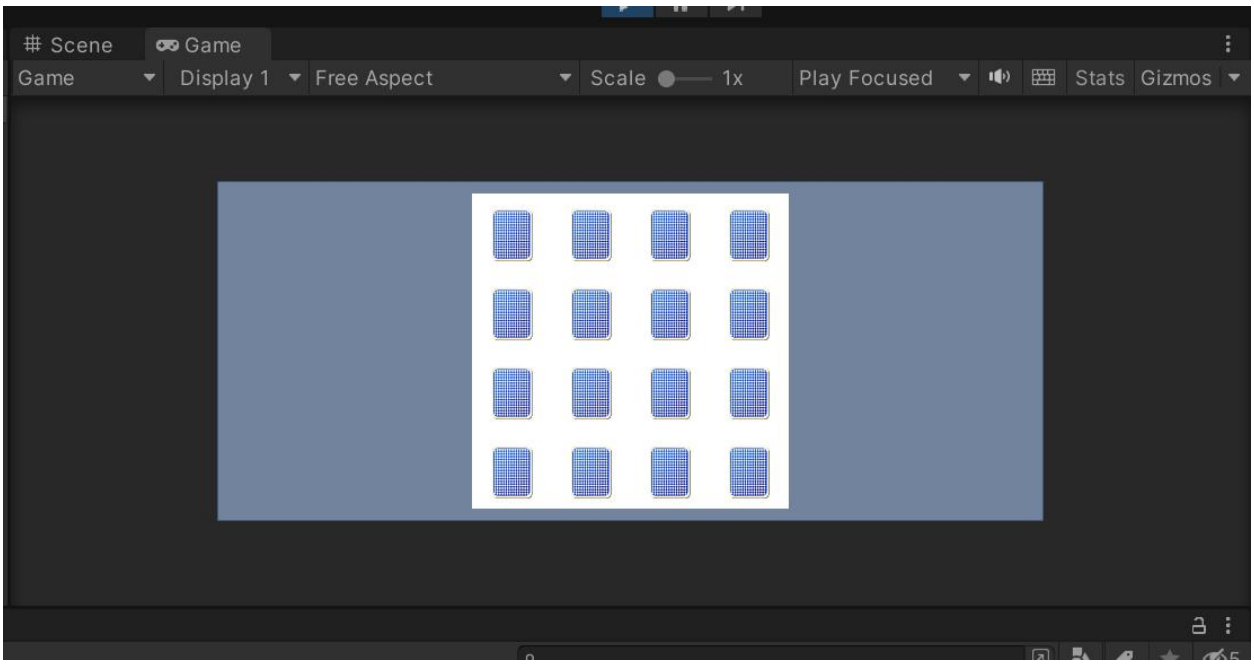
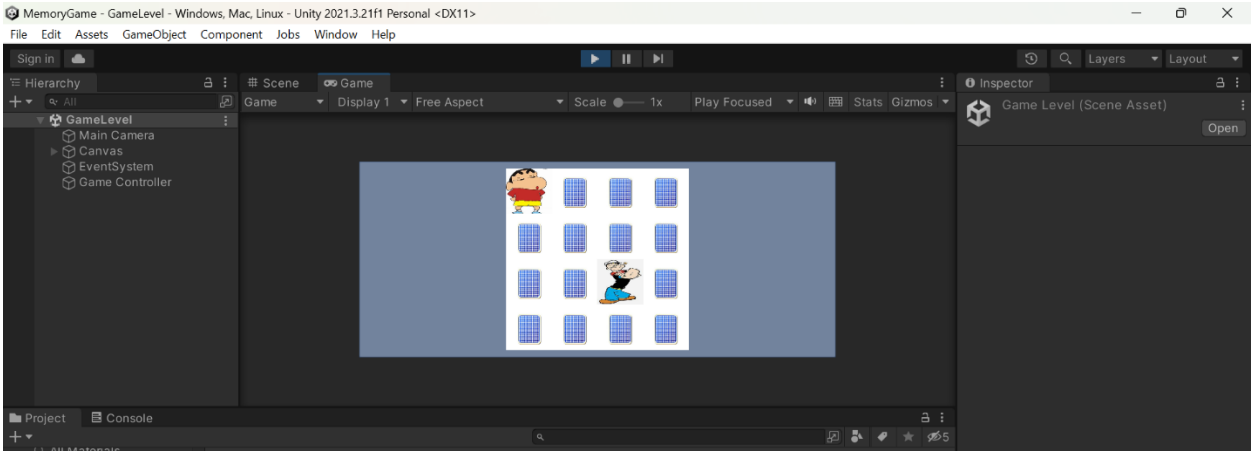
CS4478
Game Programming
Dr. Sabah Mohammed
Department of Computer Science
Assignment 2: Memory Game In Unity

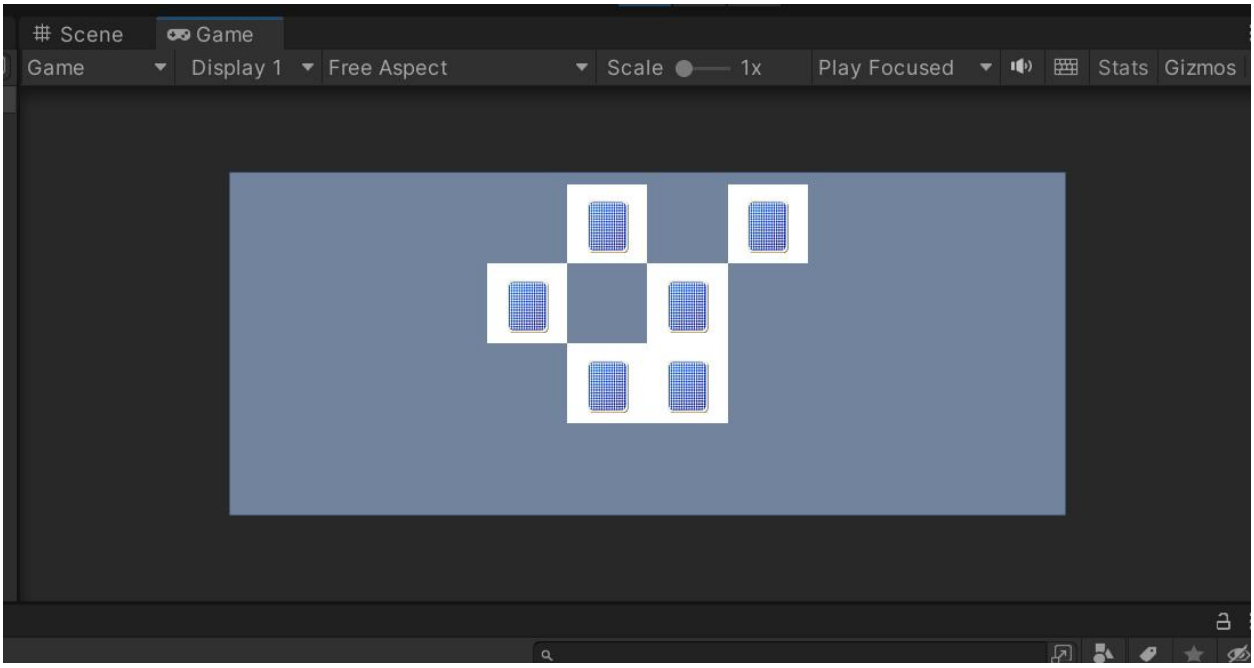
Here I made 2D unity memory game with blank cards that are laid in a 4X4 manner on the screen. If you flip two unmatched cards these cards will flip again, and you will play another round. Once you identified all the characters it will should produce a Game over Screen with option to play again where the characters will have different positions.



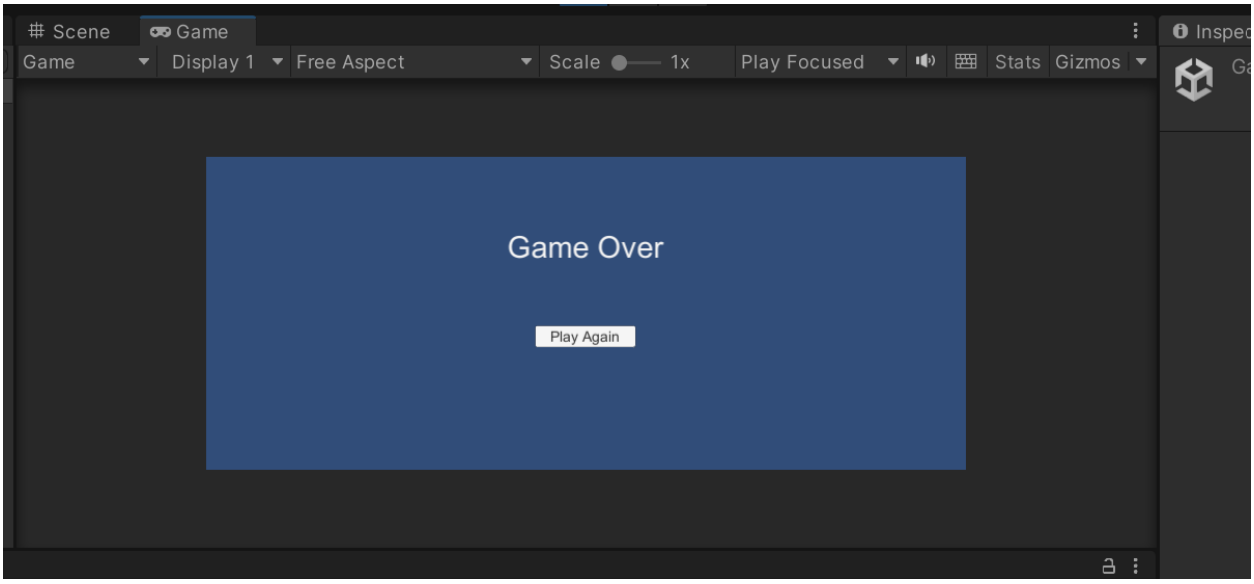
In the above screenshot we can see blank cards.



In the above picture we can see the blank cards being flipped, since it does not matches it will flip over again.



In the above screenshot we can see once the cards match-up, those cards will disappear, and it will only show the remaining ones.



Once all cards match-up, it will show you the game over screen with an option to play again.

#Csharp Scripts:

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class AddButtons : MonoBehaviour
6   {
7       [SerializeField]
8       private Transform puzzleField;
9
10      [SerializeField]
11      private GameObject btn;
12
13      void Awake()
14      {
15          for(int i=0; i < 16; i++)
16          {
17              GameObject button = Instantiate(btn);
18              button.name = "" + i;
19              button.transform.SetParent(puzzleField, false);
20          }
21
22      }
23  }
24
```

In the above screenshot, I have made a #csharp script called AddButtons in which generates a set of 16 buttons in a puzzle field by instantiating a prefab button and setting its parent to a specified transform. The buttons are named with a unique integer value, starting from 0.

In the Game Controller script:

```
1
2   using UnityEngine;
3   using UnityEngine.UI;
4   using System.Collections;
5   using System.Collections.Generic;
6   using UnityEngine.SceneManagement;
7
8   public class GameController : MonoBehaviour
9   {
10      [SerializeField]
11      private Sprite bgImage;
12
13      public Sprite[] puzzles;
14
15      public List<Sprite> gamePuzzles = new List<Sprite>();
16
17      public List<Button> btns = new List<Button>();
18
19      private bool firstGuess, secondGuess;
20
21      private int countGuesses;
22      private int countCorrectGuesses;
23      private int gameGuesses;
24      private int firstGuessIndex, secondGuessIndex;
25
26      private string firstGuessPuzzle, secondGuessPuzzle;
27
```

Here I have imported the required libraries and declared some variables, including sprites, lists, integers and booleans.

```
    void Awake()
    {
        puzzles = Resources.LoadAll<Sprite>("Sprites");
    }
```

Here in the Awake() method, images used are loaded from the Resources folder.

```
void Start()
{
    GetButtons();
    AddListeners();
    AddGamePuzzles();
    Shuffle(gamePuzzles);
    gameGuesses = gamePuzzles.Count / 2;
}
```

Here in the in the Start() method, the script gets the puzzle buttons, add listeners and adds the game puzzles. The game puzzles are generated by assigning each button with a puzzle sprite, which is taken from the shuffled list of puzzle sprites.

Miscellaneous Files | GameController | GetButtons()

```
43    void GetButtons()
44    {
45        GameObject[] objects = GameObject.FindGameObjectsWithTag("PuzzleButton");
46
47        for(int i=0; i < objects.Length; i++)
48        {
49            btns.Add(objects[i].GetComponent<Button>());
50            btns[i].image.sprite = bgImage;
51        }
52    }
53
54    void AddGamePuzzles()
55    {
56        int looper = btns.Count;
57        int index = 0;
58
59        for (int i=0; i < looper; i++)
60        {
61            if (index == looper / 2)
62            {
63                index = 0;
64
65            }
66
67            gamePuzzles.Add(puzzles[index]);
68
69            index++;
70
71        }
72    }
```

Miscellaneous Files | GameController | gameGuesses

```
73
74      void AddListeners()
75      {
76          foreach (Button btn in btns)
77          {
78              btn.onClick.AddListener(() => PickAPuzzle());
79          }
80      }
81
82      public void PickAPuzzle()
83      {
84          string name = UnityEngine.EventSystems.EventSystem.current.currentSelectedGameObject.name;
85
86
87          if (!firstGuess)
88          {
89              firstGuess = true;
90
91              firstGuessIndex = int.Parse(UnityEngine.EventSystems.EventSystem.current.currentSelectedGameObject.name);
92
93              firstGuessPuzzle = gamePuzzles[firstGuessIndex].name;
94
95              btns[firstGuessIndex].image.sprite = gamePuzzles[firstGuessIndex];
96          }
97          else if (!secondGuess)
98          {
99              secondGuess = true;
100
101             secondGuessIndex = int.Parse(UnityEngine.EventSystems.EventSystem.current.currentSelectedGameObject.name);
102
103             secondGuessPuzzle = gamePuzzles[secondGuessIndex].name;
104
105             btns[secondGuessIndex].image.sprite = gamePuzzles[secondGuessIndex];
106
107             StartCoroutine(CheckIfThePuzzlesMatch());
108
109         }
110     }
111
```

```
IEnumerator CheckIfThePuzzlesMatch()
{
    yield return new WaitForSeconds(1f);

    if (firstGuessPuzzle == secondGuessPuzzle)
    {
        yield return new WaitForSeconds(.5f);

        btns[firstGuessIndex].interactable = false;
        btns[secondGuessIndex].interactable = false;

        btns[firstGuessIndex].image.color = new Color (0, 0, 0, 0);
        btns[secondGuessIndex].image.color = new Color(0, 0, 0, 0);

        CheckIfTheGameIsFinished();
    }
    else
    {
        yield return new WaitForSeconds(.5f);

        btns[firstGuessIndex].image.sprite = bgImage;
        btns[secondGuessIndex].image.sprite = bgImage;
    }
    yield return new WaitForSeconds(.5f);

    firstGuess = secondGuess = false;

}
```

Here when we click on a button, PickAPuzzle() method is called. This method sets the first or second guess and updates the puzzle button's image sprite.

Then CheckIfThePuzzlesMatch() method is called after the second guess is made and this method checks whether the two sleleted sprites match and if they do, the corresponding button are made non-interactable, and their images are set to transparent. If they don't match, the button images are set back to the default image sprite.

```
    void CheckIfTheGameIsFinished()
    {

            countCorrectGuesses++;

            if (countCorrectGuesses == gameGuesses)
            {
                SceneManager.LoadScene("GameOver");
                Debug.Log("GameFinished");
                Debug.Log("It took you" + countGuesses + "many guess(es) to finish the game");
            }
        }

    void Shuffle(List<Sprite> list)
    {
        for(int i=0; i < list.Count; i++)
        {
            Sprite temp = list[i];
            int randomIndex = Random.Range(i, list.Count);
            list[i] = list[randomIndex];
            list[randomIndex] = temp;
        }
    }
}
```

CheckIfTheGameIsFinished() method is called when a pair of matching puzzle sprites is found. It increments the count of correct guesses and checks if the game is over by comparing the count of correct guesses with the total number of guesses and loads the GameOver scene.

And the Shuffle() method takes a list of sprites and shuffles it randomly.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.SceneManagement;
5
6   public class PlayAgain : MonoBehaviour
7   {
8       // Start is called before the first frame update
9       void Start()
10      {
11
12      }
13
14      // Update is called once per frame
15      void Update()
16      {
17
18      }
19
20      public void LoadGame()
21      {
22          SceneManager.LoadScene("GameLevel");
23      }
24  }
25
```

LoadGame() method is called when the Play again button is clicked. This method uses SceneManager.LoadScene() to load the GameLevel scene. When the scene is loaded, all the components to game object with the PlayAgain script will reset.

GitHub link: https://github.com/jaiminsojitra/MemoryGameInUnity.git

Unity Version: 2021.3.21f1