In this document this will contain the documentation to how I completed Brief 2:

**Pattern Shuffle**

A puzzle game requires a script to semi-randomly shuffle its game pieces. Each shuffle should mix the components given and be as unpredictable as possible, but should also avoid repetition of the same piece in subsequent shuffles. For example, the first 4 components of a new shuffle shouldn't contain the last 4 components of the previous one. The number of components in the list and the limit on repeating values must be customisable through the Inspector.

You will be provided with a list of integers representing the individual components. You must return a new shuffle each time the Shuffle function is called. You must additionally provide a program to test the shuffle function to ensure that it meets the repetition requirements. This test must also include a demonstration test case to show that it can catch errors.

What I did?

As I have never created a code like this, the first thing I did was to research ways in which this was possible. This is when I came across the Fishers Yates algorithm.

The fisher yates shuffle algorithm works by “to move the "struck" numbers to the end of the list by swapping them with the last unstruck number at each iteration”

<https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle#Fisher_and_Yates'_original_method>

Code:

using System.Collections.Generic;  
using UnityEngine;  
  
public class ShuffleArray : MonoBehaviour {  
  
    public Color[] colors = new Color[3];  
          
    public GameObject[] dots;  
  
  
  
        *// Public so you can fill the array in the inspector*  
        public int[] scenarios;   
  
    void Start ()  
    {  
        colors [0] = Color.red;  
        colors [1] = Color.green; *//changing materials set to the gameobject*  
        colors [2] = Color.blue;  
  
        Shuffle (scenarios);  
    }  
  
  
  
        public void Shuffle(int[] a)  
        {  
            *// Loops through array*  
            for (int i = a.Length-1; i > 0; i--)  
            {  
                *// Randomize a number between 0 and i, so the range consistenly repeats*  
                int rnd = Random.Range(0,i);  
  
                *// Save the value of the current i*  
                int temp = a[i];  
  
                *// Swap the new and old values*  
                a[i] = a[rnd];  
                a[rnd] = temp;  
            }  
  
            *//debug to show working correctly*  
            for (int i = 0; i < a.Length; i++)  
            {  
                Debug.Log (a[i]);  
            }  
        }  
    public void ShuffleButton ()  
    {  
        Shuffle (scenarios);  
        for (int i = 0; i < 3; i++) {  
            dots [i].GetComponent<SpriteRenderer> ().color = colors [scenarios [i]]; *//use the array, take the color from SpriteRenderer and run the scenarios*  
        }  
  
    }  
}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Estimate Start | End | Interruptions | Task |
| 10 april 2019 | 10am | 10.10 | none | Creating Shapes and button |
| 12 april 2019 | 10.30 | 11.30 | 5 minutes | Creating code |
| 20 april 2019 | 11.35 | 12.30 | None | Testing and debugging code |