Mobile Applications Development 3 Developer Diary

Matthew Sloyan – G00348036

Game: Colour Switch

Designed by: Marian Ziacik

# Introduction

I have developed a Colour Switch clone and tweak game which was designed by Marian Ziacik. Below you’ll will find an outline of the work undertaken, my thought process, my design decisions and the contact between the designer along with testing efforts, a conclusion and the developer & designer review.

# Work Outline

I initially read through the design document thoroughly to get a grasp of what was required and how I could go about implemented it. I then had a quick chat in person with Marian which he told me more about the game, his ideas and what he wants to include. He told me to download the Colour Switch game from the PlayStore to get a feel for it, this is the game his design document is based on. From this I found it is a very addictive, fun and easy to play game which is the three core elements I wanted to enforce. <https://play.google.com/store/apps/details?id=com.colorswitch.switch2&hl=en_IE>

## 4th October – Start of Development

Today I began by setting up a blank project and some basic sprites. I felt the best place to start was the player ball as it’s the key element of the game. To do this I set up script that tracks the input of the keyboard (Space, up arrow and left mouse click) which changes the velocity of the ball using a variable jump speed. This created a simple jump mechanic for the player.

I then implemented the first of the obstacles (basic spinner). I imported the supplied artwork for each segment, added colliders and wrote a script to handle this using tags. From research of referencing objects in Unity I found that tags were the best way to distinguish between each segment, so the player colour only pass through their current colour. <https://docs.unity3d.com/ScriptReference/GameObject-tag.html>

The next thing I worked one was the random colour generation of the player. I created a colourManager script that I will use to manage all colour changes throughout the game, such as when the player hits a colour changer ball.

Current state – The player is randomly assigned a colour when starting and can jump up the level to get through the spinner.

## 6th October

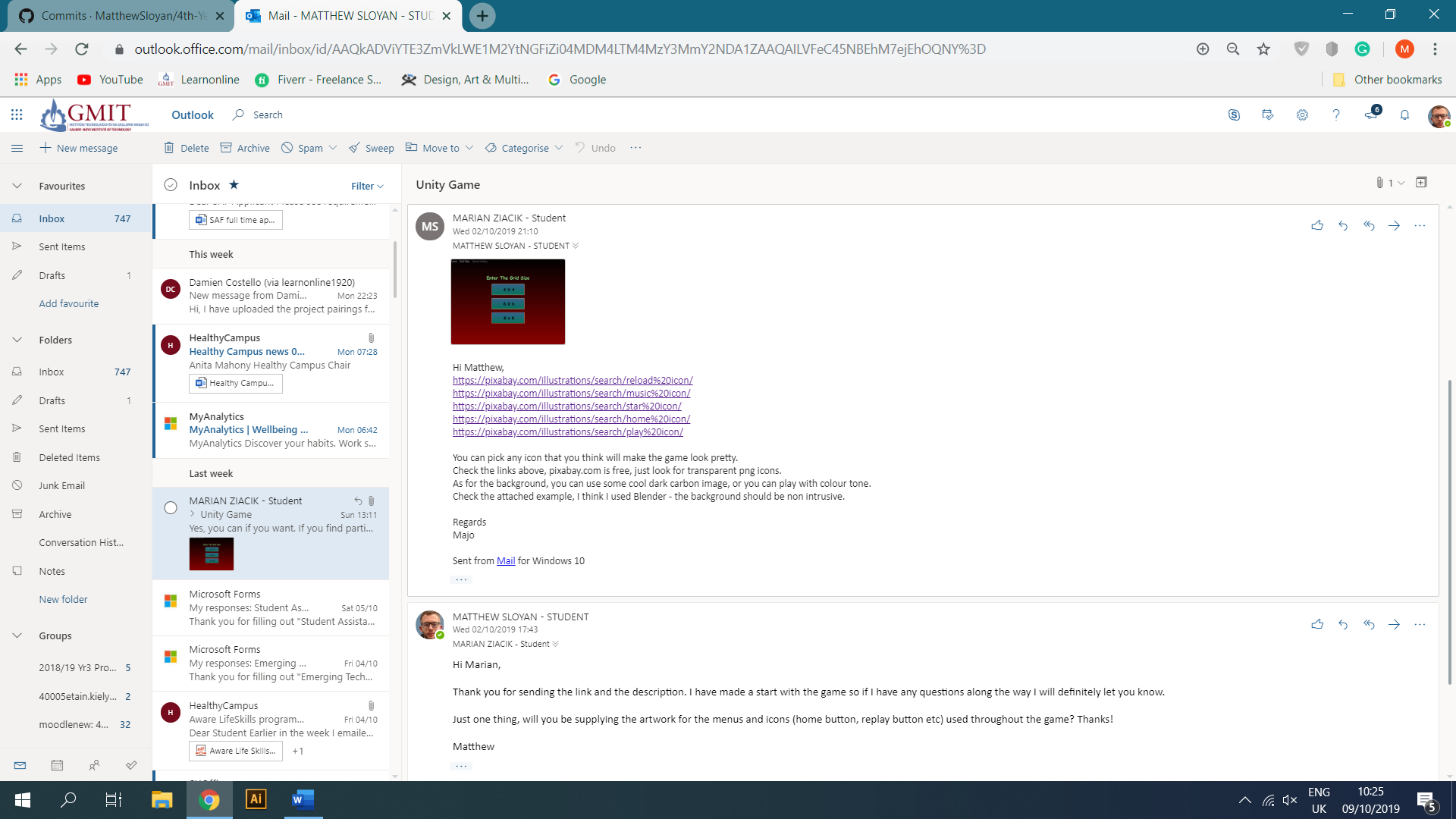
Today I worked on some features to improve gameplay so that the player doesn’t fall right off the screen when starting. To achieve this I researched ways on how to change the gravity of the player programmatically and found that the body type of the ridgidbody could be set to static and then to dynamic when the player plays the game (E.g presses a button). Later in development a tutorial will be laid over this explaining how to play the game. <https://docs.unity3d.com/ScriptReference/Rigidbody.html>

I also created a script that would follow the player by checking if their position is greater than half the screen, if so the cameras postion is moved to the current player position.

## 7th October

I wanted to get a start on the menus to get the overall game functional, I started by asking Marian to supply the in game menu graphics which he did (Images were sourced from Pixabay and FlatIcon as seen in the below image.) From this I imported all the icons and created a canvas, panel and UI elements to lay out the menus. I created a pause menu and game over menu exactly like the design document supplied, however I may contact Marian about some suggestions to improve user experience for things such as the game over screen possibly displaying “Game Over” rather than “Colour Switch” so the player know they have failed the level.

With the pause menu, game over menu and a separate scene main menu created and designed I then plugged them all in so that the player could navigate between all menus. There was one problem I found which was trying to access the instance of the menu manager class in the collider class. I researched some possible solutions and found that a singleton would suit best for simplicity. I implemented this using the work we had done in the lab. It worked perfect for displaying the game over menu from the collider and player class.



## 8th October

To continue from yesterday, I fully connected the main menu so that the user could go to main menu, start the game or restart it at any time which satisfies the initial design document. I achieved using scene loading when an onclick is called. I also added in another end game condition where if the player falls off screen then the game will end. This is only applicable at the start of the game.

## 13th October

The current game isn’t very dynamic as each spinner gameObject must be added manually. I wanted to fix this so the game could be implemented as an endless play type. So, what I needed is the ability to instantiate new objects as the player progresses through the game. Firstly, I researched ways that objects could be instantiated and found the “Instantiate(object, newPos, Quaternion.identity);” method from the Unity documentation. <https://docs.unity3d.com/ScriptReference/Object.Instantiate.html>

I also had to solve the problem of them of getting the new position, as from initial tests they would spawn on top of each other. To solve this, I passed in the initial spinner which would be used to get the previous position which I added 5f to. This worked perfectly. I then set this spinner to the previous position for the next call. I used the same way to instantiate a new colour changer and collectable star. Lastly, I added a call for this method when a player collects a star, this allows the game to load the next set of items without the user seeing it. Later I will also implement deletion of objects when a user has passed them, so it doesn’t slow down the game.

With this implemented an infinite number of spinners, stars and colour changers are created as the player progresses. I will have to update it later with different types of obstacles, levels and difficulty as in the design document.

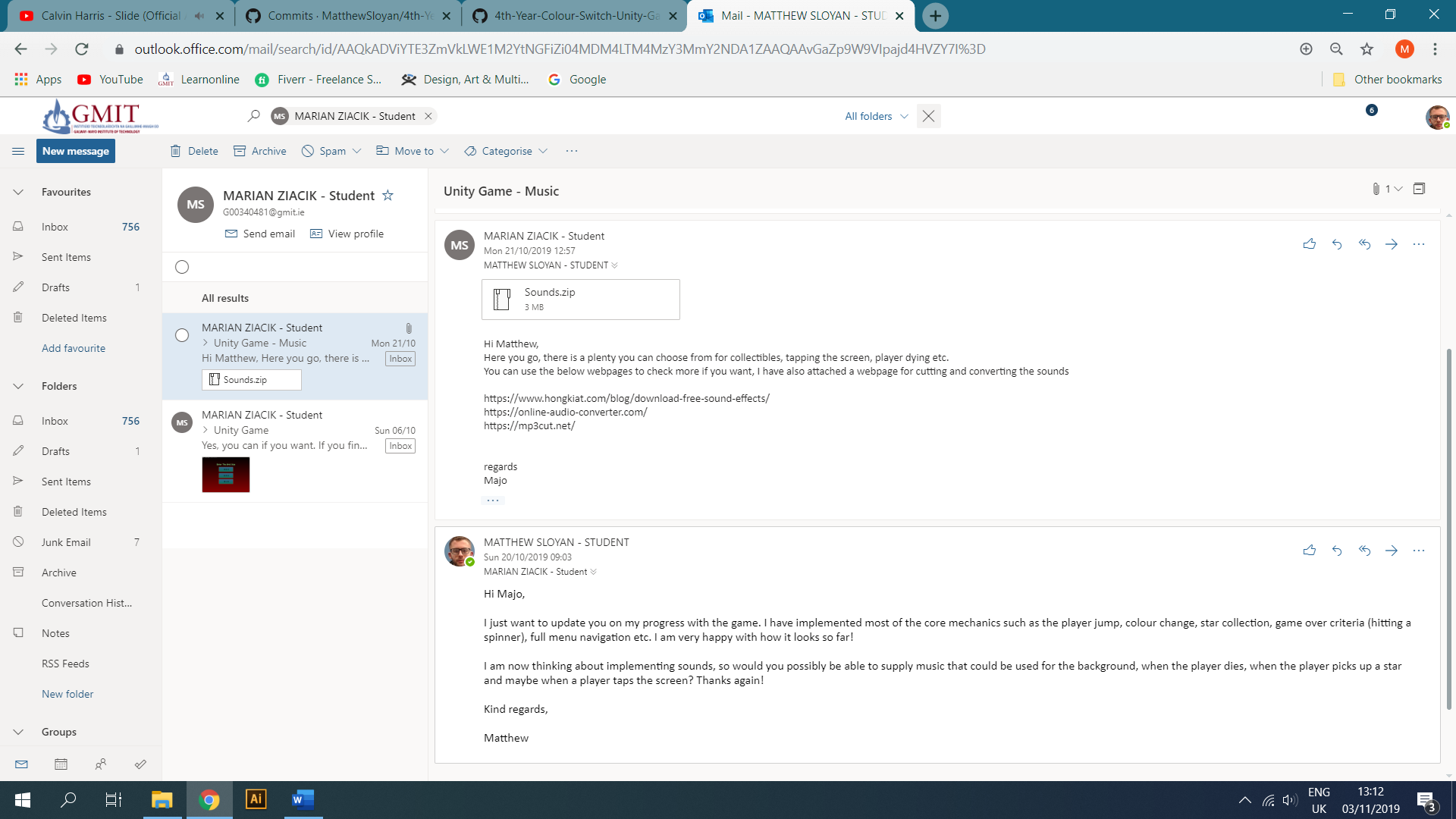
## 14th October

Today I wanted to fix the colour swapper, as it wasn’t working correctly. I did various tests but couldn’t get it working correctly with the tags as it would always end the game. I think the problem was that I was making it too complicated, so started again and updated the setPlayerColour() method in the colourManager script to handle all colour changes. Using a randomly generated colour I set the spriteRender’s colour and tag so that it would work with the spinners, and then called it when the player collides with a colour changer.

However, this posed a problem what if the randomly assigned colour was the players current colour? To fix this I first checked if the new colour is equal to the current colour, and if so then call the method again using recursion which will ensure that a new colour is always created.

## 20th October

The next thing I thought I would implement is music and sound effects, so I contacted Marian (below) who sent me all the required sound effects such as the background music, player death, screen tap and star collection.



## 2nd November

With job interviews and other projects, I didn’t get around to the music elements till today. I started by created a simple AudioController script and created individual methods to play each sound. I again used a singleton pattern so that other classes could get the instance and play the sounds. For example when a player collects a star it calls the playCollectStarClip() method from the collider script. Also from testing playing the sounds would stop the background music so to fix this I researched possible solutions and found source.PlayOneShot(clip); from the Unity documentation. <https://docs.unity3d.com/ScriptReference/AudioSource.PlayOneShot.html>. This allows the other sounds to be played once on top of the background music. I still need to implement the ability to turn on and off sound and have it working across multiple scenes.

I also worked on the scoring systems today. I thought about the various way to do this and I remembered how we used events in the labs so I thought that would work best as it creates no dependencies and makes it’s easy to change. When a player collects a star, an event is fired which adds to the current user score.

I also looked up ways to save the high score and found PlayPrefs which would work across all platforms easily. The score is checked against the saved high score and if it’s greater then a new high score is saved. I then read these values in on the game over menu so the player can see their current and high score. <https://answers.unity.com/questions/1325056/how-to-use-playerprefs-2.html>

## 17th November

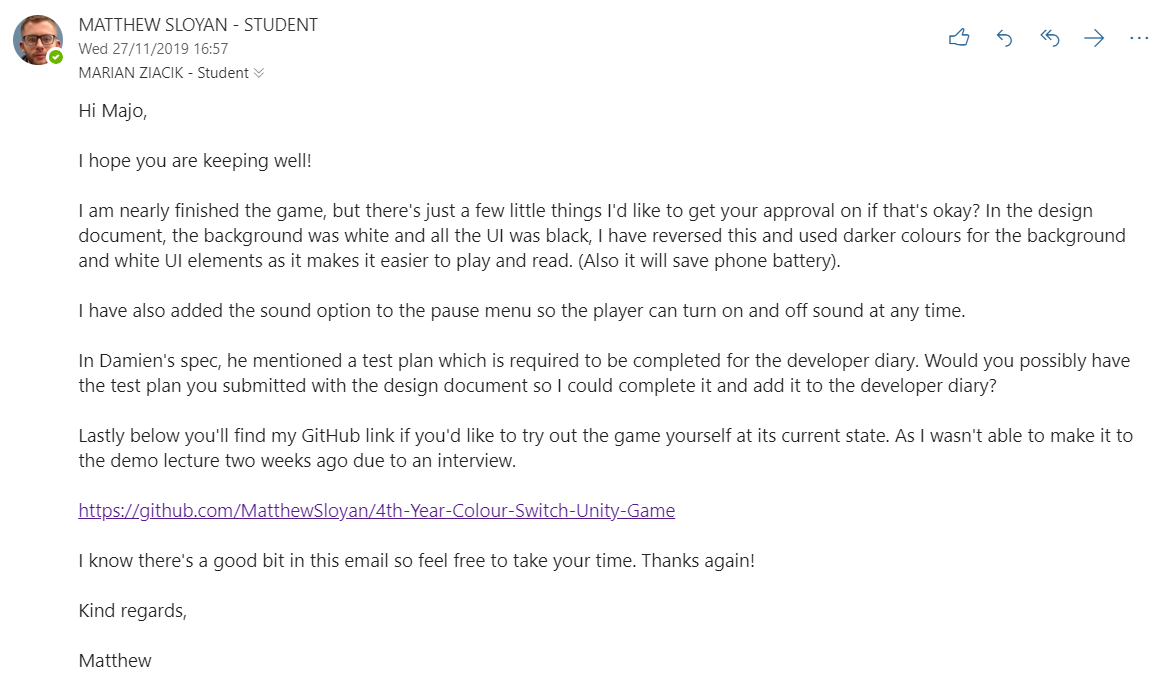
Over the last two weeks and I had a quick start at implementing the second type of level which is a horizontal slider that comes from the left or right. I just wanted to get it working one way for now so to achieve this I tested it in the Unity inspector to find the x positions where it should start and where it would be out of bounds. In the update method I then checked if the x position is greater than the bounds E.g. 8.45. If so, set x position to initial position again and start moving.

## 27th November

I wanted to get the option to turn the sound on an off. What I had already was an image button that would switch on click, but it was handled through the UI. I tried to recreate this at runtime but found it hard to swap the image, and keep it swapped so I removed it. From a read of the UI documentation I found a toggle switch button which would work better. I adapted code from the documentation on how to access and turn the toggle on and off by adding an onValueChanged listener. (<https://docs.unity3d.com/2019.1/Documentation/ScriptReference/UI.Toggle-onValueChanged.html>).

With this implemented I had to allow the button activation to be saved if the user closes the game. As I had used PlayPrefs before I thought that would be a good solution. So, on change the toggle value (true/false) is saved, and when the game loads it checks the PlayPrefs and sets the toggle accordingly.

The next step was to incorporated muting the sound. I initially tried it with source.Stop() but it didn’t work as expected so from further research I found .mute = true/false, which would mute all sounds and restart it from the current point. I implemented this in the update method of the Audio controller, which checks the PlayerPrefs to see if sound is on or off. <https://docs.unity3d.com/ScriptReference/AudioSource-mute.html>



I also asked Marian about some improvements to the design document which will be described in the “Improvements & Additions” section.

## 28th November

Today I wanted to get full level implementation working, as it was just spawning infinite spinners. I had the slider code working but it just had to be added to the GameController script. I started by drawing out the way the game objects would load from the design document which would look like the below.

* **Level 1** - spinner with star -> spinner with star -> colour swapper.
* **Level 2** - slider -> star -> slider -> star -> slider -> colour swapper.
* Repeat level 1 & 2 with increasing difficulty.

How I got this working is using the previous game object position to instantiate new game objects. For example, using the previous spinner position I would instantiate a new colour swapper 2 points up on the y axis. This was an easy way to be able to progressively create new game objects that don’t overlap. I initially started by using two methods, one to set up the initial spinners as they don’t have previous position and another method to create the following spinners, sliders etc. As this contained a lot of repeated code (DRY) I merged it into one method and if the previous position was null, then I would set the position to 0 rather than “prevPositon”.

To swap between levels I again used PlayPrefs to save a 0 or a 1. (0 = spinners, 1 = sliders). Depending on the option it would dynamically create the level at runtime when a player hits the colour swapper from the previous level. This took a lot of time of testing to get it right. To improve the design even further I added a loop to create two spinner and three sliders with stars, which cut down duplicate code and made it easier to read.

Lastly, I wanted to clean up the Unity inspector when instantiating objects and they all go into the root. To do this I used code from the labs to set each object as a child of an empty game object parent.

## 29th November

I found when playing it would be nice to know what level the current player is on, so I asked Marian in person if that was okay to add which he agreed and thought it was a great idea. So, I began researching how to add text to the scene rather than the UI and found TextMeshPro. (<https://docs.unity3d.com/ScriptReference/TextMesh-fontStyle.html>). I incorporated this at runtime which would then get the level count from PlayerPrefs.

One of the last main features was to add difficulty to the game, so as you progress the speed of the game objects increases. To do this I added a static get/set for each movement speed which I updated slightly when the level loads.

## 30th November

As I had the gets/sets for the difficulty in two separate scripts duplicated it made sense to make one DifficultyController script to handle it all. I also added methods to reset speeds when the game resets.

From testing I noticed the game would get quite laggy the longer you played due to the number of objects being created. To clean this up I did some research and found the OnBecameInvisible method which is called when the game object goes off screen. (<https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnBecameInvisible.html>). When attached a prefab with a renderer it handles all the clean up easily.

Lastly, I updated the background to move with the camera by adding it as a child to the MainCamera. I then did some playtests to fully test the game. At its current state the game has satisfied all the conditions of the design document and made some improvements with approval.

## 7th December

Some of the last things I wanted to implement was just general improvements and tweaks to the difficulty from testing etc. I also wanted to make the game more user friendly so I added a tutorial to the main menu which will allow the player to view the controls and what to do before they start playing. I also added a splash screen to the application to the again improve user experience.

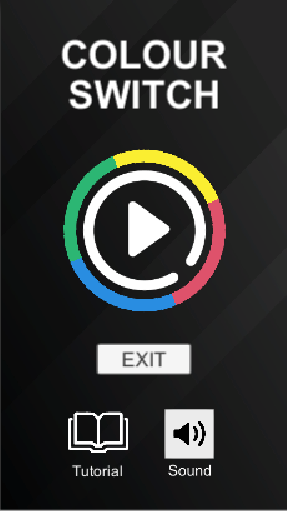
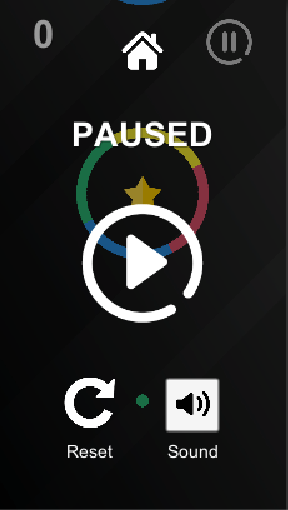
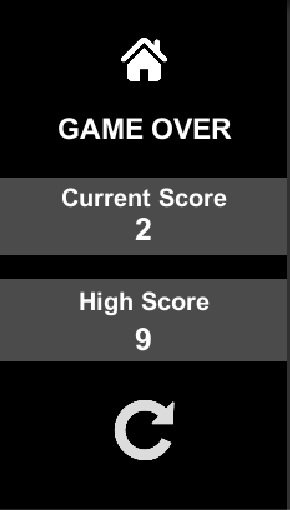
Another area I looked at was building the game to Android so it could be run on a mobile device. On the first build however the scaling of all the UI was tiny, to fix this I researched online and found the canvas setting had to be set to “Scale with screen size” which fixed a lot of the issues with some tweaks. “<https://gamedev.stackexchange.com/questions/115910/unity-ui-not-scaling-correctly-on-android>”

# Additions & Improvements

All improvements and additions were first given approval from the designer before implementation.

* Improved layout of some elements such as reset and home button to make it as easier to access on mobile devices.
* Added level numbers to each level to show the player their progress through the game.
* Inverted the UI and background colours to create a sharper contrast. This made it easier on the eyes, as the solid white background proposed was quite strong and the colour obstacles got lose on the screen. Also, the darker colours would help with battery optimization.
* I added a tutorial screen to the main menu which would increase the user experience.
* I added a second spinner obstacle to the first level as from testing it didn’t pose much of a challenge and could easily be jumped through with just one.

Below you can see the final game implementation with some of the extras included.

# Testing

Throughout the process I completed multiple user tests, and then it was tested with a wider audience in the presentation session. I also completed a test plan which covers all the main aspects of the game. All tests were completed and passed and can be found in the separate Excel document.

# Conclusion

Overall, I am extremely happy with how the game turned out, it meets all the requirements and the designer is extremely happy with the implementation. I also added extras to improve the overall game experience. From testing no bugs were found, and the game was thought to be addictive, fun and challenging during the playtest in class. I also learned a lot from this experience, especially regarding working in a customer/developer environment but also, I have learned much more about Unity game development, user experience, C# programming amongst many other things.

# Designer Review

Marian Ziacik designed this Colour Switch game. He was extremely helpful, replied to every query I had quickly and supplied everything needed. It was a great experience overall.

# Developer Review

Unfortunately, the developer that was assigned the game I designed (Space Escape) didn’t develop the game due to unknown reasons. I supplied all the assets, graphics, sounds etc within a few days of being assigned. However, I was only told towards the end the game would not be developed despite multiple attempts to ask about the progress.