Introduction

I started my adventures with game design and making computer games last year when I came across the opportunity to help two other students as an artist/animator on the Global Game Jam. Since then we have made a few games, but I was always the artist and never really had the opportunity to explore more technical side of making computer games like coding and actually using the game engine. I consider that knowledge would be very helpful to understand the whole process better, to reduce mistakes in the art part that will make the programmers life easier and the process of creating the game quicker so for example my rigs will not be too complicated, and models will be right sizes etc.

This project provided me with the opportunity to explore the areas I normally do not work on when creating computer games, but I am very interested in, so the game mechanics and game logic, scripting, gameplay design and getting comfortable with the workflow of Unity 5 game engine.

I decided that I want to take something relatively simple and focus on its programming side and exploring game design to understand why games are so fascinating and addictive time consumers for the society and why a simple game can be so successful.

I had a look at mobile games and simple computer games like for example 2D and 3D platformers, Doodle jump, Rider, Flappy bird and Color Switch.

I decided that a game that will let me to focus mainly on programming but will not be too complicated might be colour switch as it has quite simple mechanics. You just have to click or tap to jump through the same colour as your player. I quite like the games where you can die easily, and the aim is to get more points and survive not to win so I decided that something like that might work.

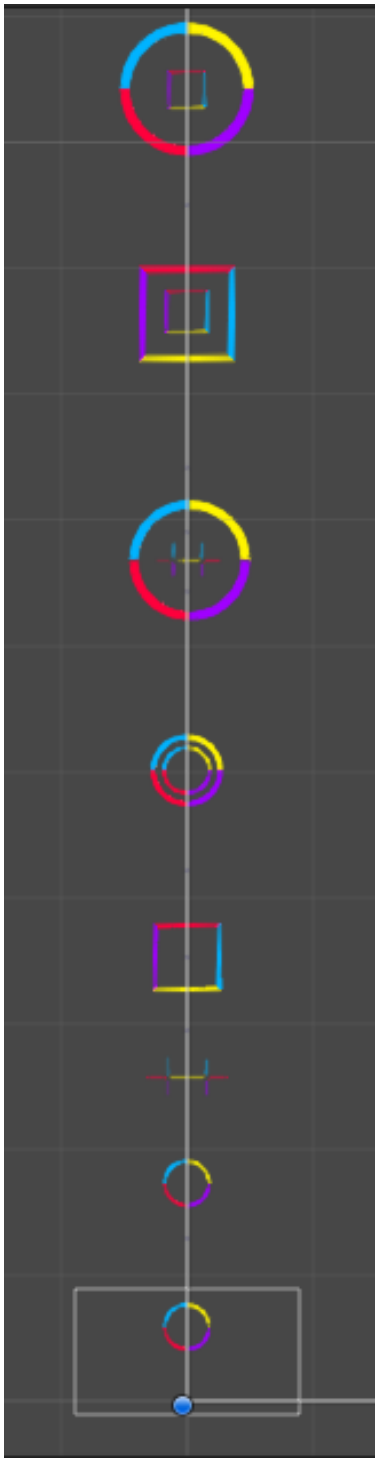
My goal when it comes to the artefact was to create a full playable, yet simple 2D game in Unity. I have never been really programming much before and I have never used C# which is Unity's programming language, so it was quite a challenge.

I began by researching the game mechanics so how it works and how it looks as I did not really remember much from playing it years ago. I watched some gameplay videos and read an interview with its creator. That allowed me to explore the idea of game design and why people like simple games. I found out that basic colours and simple shapes as well as not too complicated mechanics and logic is what made people enjoy this mobile game so much. It is also somehow similar in its mechanics to Flappy Bird. I decided that what would make this game even more interesting would be making it endless, kind of like Flappy Bird with progressively generated circles and different obstacles, gradually increasing difficulty/speed level but first as I have never done anything like this before I had to make the game mechanics work.

My starting point was a short YouTube tutorial about how to make a Color Switch replica.

First problems loading unity and following the tutorial

It seemed very easy at the beginning when I was following the tutorial 1 to 1 to help myself with the understanding of C# scripting. I learned for example that using velocity function gives quite predictable player movement and that when you forget the semicolon or brackets your game might not run.



I was very happy when I saw that my little dot started jumping just like I wanted it to.

I created some basic sprites in Photoshop starting with a 4-coloured circle. I have done this before while working on some 2D animations, so I knew how to bring it to Unity and how to slice it. Positioning everything in the Scene viewport and creating a perfect circle positioned in the exact centre was quite challenging but I managed to do that.

At this stage I had some problems with rotation of the game object as it was not right, I mean the pivot point was in the wrong place but I figured out I have made a mistake when I forgot to first reset the empty game object position to 0 before parenting all circle parts below it.

While following the tutorial I finally understood what is debug in the console in Unity and why it is so helpful and what it does, as I saw my friends using it when we were making games but never really understood what it does.

I made another classic Unity mistake: I forgot to exit gameplay mode when I was doing some changes, so I had to redo the tags on objects.

I became more confident with coding and started adding my own lines and changing some parameters or names to make them different than in the tutorial as I finally understood what each thing does. For example when checking if the colours on the circle match the player I added Debug.Log (“Hurray”); so it started displaying things in the console and I was very happy that it worked because without it I could not see if everything worked fine as the game was not restarting yet every time you missed and it would be very hard to code more if it would do that from the beginning.

After I have done some scripts for the player and for the rotations of the coloured circles, I created some more sprites in Photoshop and combined them in different ways in my scene to create different levels with different difficulty level that depends on obstacles allocation and speed of their rotations.

I started thinking about the design of my game and how I can change things to make it more aesthetically pleasing. I considered changing the ball player to a rocket and making the game look as if it was happening in space. But First I downloaded some fonts as the only font Unity has by default is Arial.

Further development of the game

When my tutorial help ended and I had to figure out my own ways of developing the game more, the hard part started. I felt quite confident with using Unity now, not so much with scripting, but I understood it better now. I researched some other videos and tried to make a scoring system for the

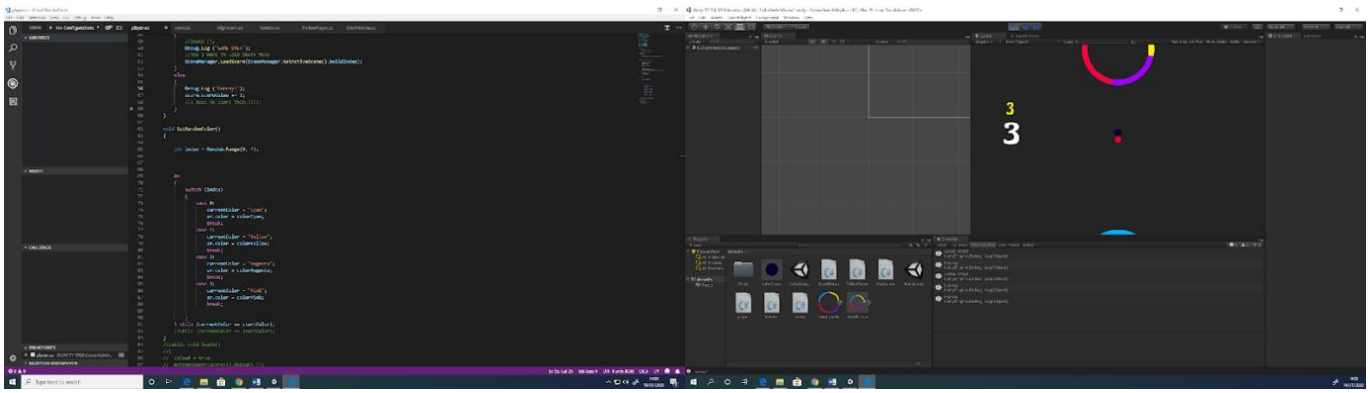
game. I ended up plugging the score to the distance of the player in y as it was the easiest solution at the given moment to see the scoring system working. I considered whether it might be better to score as the camera moves as then I considered making it to score when it goes through colours too hard to code in, but I knew that I should code something in in the same place where I have my debug Hurray.

I tried to make the high scores work But as I did not have a clue how to change my thoughts into C# code, I tried first to do the high score system and to make it get the value from score. I did not end up making it work exactly how I wanted it to so for a time being I made the high score script get current values of score script so both texts displayed the current score.

I tried to make the death screen so my "Game Over" text with some other elements to appear, but it did not really work. I tried to create like a death function/condition but later resigned from that idea. I tried switching scenes with the scene manager code but something was not right there either and for some reason I have an impression that before doing the death menu scene switch the game worked better as it reloaded the scene again after you lost and now it stopped reloading the screen again so I tried to focus first on fixing that.

Solving issues with ColorChanger

At the moment my ColorChanger so the dark blue dot was changing the colour of the player into random one but sometimes it was changing it to the same colour as it previously was, and I wanted to fix that to always be different. I have done it with do while loop and was struggling with some definitions and semicolons but finally made it work. When testing the game, it started sometimes freezing on the ColorChanger either the first one or on some of the next ones and that was crashing the whole Unity



so after having to relaunch the engine a few times I figured out that something is wrong with my logic and it crashes when the player changes the colour to the same as it was before. I got some help from the demonstrators and we fixed the code by adding public int curr colour idx and public int prev colour idx which allowed later to make the condition for the ColorChanger to not to change to the same color and to get rid of an infinite loop that was created by me earlier. So now the colour is checking if it is the same number value/index as it was, before it enters the loop and that also allows to change the loop from do while to a simple while loop that is not infinite any more.

Scores

I managed to make the score to change on the collision with the same colour instead of the position of the player, but it was counting points every time you go through colours even on the same circle so after some consultations with a friend we set up a boolean value touched so when it is set to true you can score but it sets back to false on the ColorChanger and that allows you to score only once on the level and also gets rid of the problem of scoring when you lost and you are falling down through circles or you just jump close to previous obstacle.

I had some issues with the points not zeroing out after you lose and the game reloads but the lecturer helped to fix it. The issue was mainly in instancing the values at the beginning.

Player's death & the DeathMenu

There were some issues that seemed minor but took a lot of time and a lot of trial and error to fix. One of them was making the player to start jumping with the first click as it was just starting to fall the moment the game started. I got this finally fixed by talking to a friend on the Games course and looking at Unity Documentation. It was the issue of turning off the rigid body of the player so the function simulated finally turned it off.

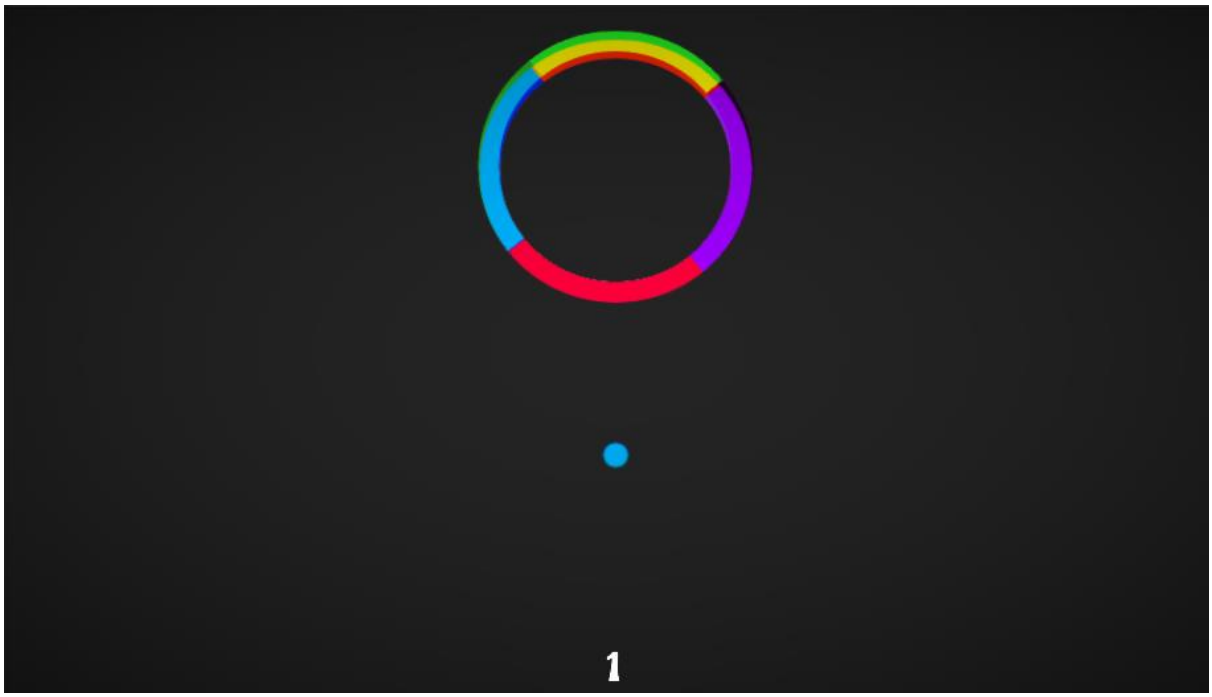
Another issue that got fixed much later with some help of another friend was making the player to die so the scene to reload and death screen to appear also when the player has fallen down below it's starting point. We made it so you lose and the game reloads when player's position is below -4.

For a very long time I could not get the Death menu to work and I even got a little bit lost with different methods of showing it when you die so I had some kind of a mixture of just displaying it on top and changing scenes, but finally managed to get it to display on top with some help from my friend.

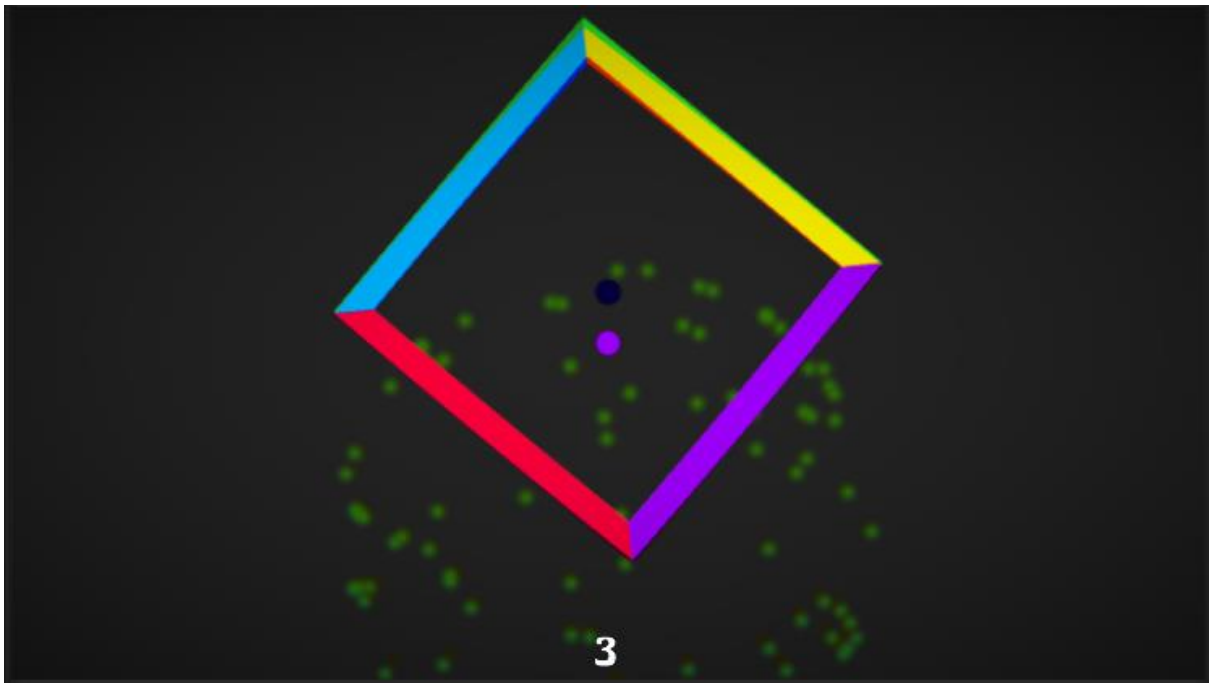
The menu contains also a button that allows to restart the game after you lose so that required a little bit of research to make it work on click and some scripting.

Effects & Making the game look better

Now was the time to make the game look a little bit better so the placing of the score was changed and some font colours too. After talking to a friend, I tried to make the game blurred below the UI when the deathText appears, but the downloaded package did not work on this version of Unity so some shadows under the UI elements were added by creating some grey semi-transparent gradients in Photoshop. I tried to use the same gradient to blur the elements at the top and bottom of the screen and make them follow the camera but later after looking through the Unity Assets Store I found a Post Processing package that made my game look much better – it allowed to add a vignette and chromatic aberration and some other camera effects. It also was supposed to solve the issues with anti-aliasing as at some point the quality of my game suddenly decreased very much, but it did not really change much.



Other thing that I added was the explosion particle effect that turns on every time you score. It is also done in a way that it changes colours for the ones that are used in the game.



People response to my game

A lot of people in labs were playing my game and they quite enjoyed it. They said it's addictive. Some of the early play tests they have done allowed me to implement some changes in the difficulty of the obstacles as a lot of people could not get through the first obstacle and it seemed weird for me as I have played it a lot, so everything looked easy. Their opinion helped me to improve the game a lot.

What can be improved or was not achieved

The only thing that was not made in the game and is really annoying me is the global High Score. I tried many times and many ways to make it work but it finally turned out that it is very hard as the score value is a string not a float or integer so it would involve a lot of work to make the high score to work. The sound effects are missing, the start menu could have been added and I did not make the levels progressively generated plus some more levels could have been designed. As well there might still be some bugs in the game and the project and scripts are quite messy and could be cleaner and kept in folders etc, but I am overall very proud of my first game.

ANNOTATED BIBLIOGRAPHY

IGN (2014). *Flappy Bird Gameplay*. [video] Available at: https://www.youtube.com/watch?v=fQoJZuBwrkU [Accessed 23 Jan. 2020]. – The video of a Flappy bird gameplay that was helpful when I was choosing a game to make and it gave me some ideas about game mechanics and progressive generation of things.

Takahashi, D. (2018). *After 40 mobile games, David Reichelt hit gold with Color Switch*. [online] VentureBeat. Available at: https://venturebeat.com/2018/03/31/after-40-mobile-games-david-reichelt-hit-gold-with-color-switch/view-all/ [Accessed 23 Jan. 2020].- An interview with the creator of Color Switch game which replica I created. It says a lot about the design of the game and how it works. It also contains some information about the possible reasons for its popularity and why a simpler game turned out to be more successful than some previous more complicated games created by David Reichelt. It helped me a lot with thinking about what can I change and what should I keep simple so for example when I finally decided to add some camera effects like chromatic aberration I decided to keep the player as a simple dot and to not to change it for a rocket etc just because it could have been too much. This interview was very informative and thought provoking.

Nogueira, E. (2016). *Color Switch Gameplay!*. [video] Available at: https://www.youtube.com/watch?v=9L6E9L2QRZM [Accessed 23 Jan. 2020]. – This Color Switch gameplay video was very helpful for me to remind myself the mechanics and the look of the original mobile game.

Brackeys (2017). *How to make a COLOR SWITCH Replica in Unity (Livestream Tutorial)*. [video] Available at: https://www.youtube.com/watch?v=gE7gc1sblUA&t=145s [Accessed 23 Jan. 2020]. – That was the main tutorial I used for creating the actual game. At the beginning I was following it nearly 1 to 1 but later started changing things and adding my own code, developing the levels and creating new shapes of sprites. So I was basically taking what I learned from this tutorial and implementing that later and developing it further. Without this really concise tutorial it would be much harder to find out what scripts do I need to create etc. I found this video extremely helpful and as it's creator is explaining everything step by step and talks about how it works I learned a lot about C# coding and working with Unity especially how to create places to drag in objects that the script is using or acting on. I still do not know how it is called but I know how it works. I think making a first game using a tutorial is just easier to understand than starting straight just with the documentation. I could understand other people's scripts and other tutorials and Unity documentation later much better. I learned much more from actually following the tutorial and doing things myself than by observing other people coding. It also help me to understand what were my friends doing when I was watching them coding during Game Jams and similar occasions.

Typo-Graf (2016). *Excalibur Nouveau Font*. [online] Dafont.com. Available at: https://www.dafont.com/excalibur-nouveau.font?l%5b%5d=10&text=250 [Accessed 23 Jan. 2020]. – A font I downloaded and used to make the UI prettier, as Unity contains only Arial as default and it does not look appealing for the user in my opinion.

Brackeys (2017). *How to make a HIGH SCORE in Unity*. [video] Available at: https://www.youtube.com/watch?v=vZU51tbgMXk&list=PLPV2KyIb3jR5xTUU7fitudKE6hCDA4Hso&index=9 [Accessed 23 Jan. 2020]. – That is one of the highscore videos, but it turned out to be not helpful after all as my score is a string so there is a lot to change to make it all work so I just decided to not to code high scores, but the video was helpful in explaining how it all works and I actually tried some things after watching it but were just not so applicable to my game, so I resigned from doing the high scores at the end.

Brackeys (2017). *SCORE & UI - How to make a Video Game in Unity (E07)*. [video] Available at: https://www.youtube.com/watch?v=TAGZxRMloyU [Accessed 23 Jan. 2020]. – Another very helpful video tutorial about making score to work. I ended up combining some of the different score

videos elements and making the scores to work how I wanted them and some fragments from this particular video were really helpful.

N3K EN (2016). ***Unity Endless Tutorial • 8 • Game UI, Score & Difficulty [Tutorial][C#].*** **[video] Available at: https://www.youtube.com/watch?v=G9jHLRqdWyU&list=PLLH3mUGkfFCXps_IYvtPcE9vcvqmGMpRK&index=9 [Accessed 23 Jan. 2020].** – Another score video about scores etc. I think that was the one that made me notice that I should disable the player when you die even though I did not realise the fact until much later.

N3K EN (2016). ***Unity Endless Tutorial • 9 • Death Condition [Tutorial][C#].*** **[video] Available at: https://www.youtube.com/watch?v=yOqjGo4XfcU&list=PLLH3mUGkfFCXps_IYvtPcE9vcvqmGMpRK&index=10 [Accessed 23 Jan. 2020].** – When I was trying to toggle the end menu and make it to appear after you lose I tried to use some of the ideas for creating a death condition/function that is shown in that tutorials, finally have done things the other way but it was still quite helpful to learn and understand the coding process.

N3K EN (2016). ***Unity Endless Tutorial • 10 • Death Menu [Tutorial][C#].*** **[video] Available at: https://www.youtube.com/watch?v=Wy8IsxtD224&list=PLLH3mUGkfFCXps_IYvtPcE9vcvqmGMpRK&index=11 [Accessed 23 Jan. 2020].**

Zotov, A. (2017). ***How to add a score counter into your Unity 2D game| Easy Unity 2D Tutorial.*** **[video] Available at: https://www.youtube.com/watch?v=QbqnDbexrCw [Accessed 23 Jan. 2020].** – This simple video helped me a lot with finally making my player to score when you collide with certain colours/tags. It is a completely different idea of a game but I could use one of the elements demonstrated there to help me with creating the scores in the game and in making them work correctly not on player's position but on collision with colors.

Takano, M. (2018). ***How to make an explosion effect by Particle System in Unity ~ "Karaage" Style.*** **[online] STYLY | VR Creative platform. Available at: https://styly.cc/tips/explosion01/ [Accessed 23 Jan. 2020].** – This explosion tutorial is very easy to follow and very clear and it helped a lot to get the basic particle effect to look a little bit like confetti explosion or firework of colourful dots. After using the tutorial I was later able to tweak some more things like the color gradient of particles so they have different colors throughout the lifespan.

Unity Technologies (2016). ***Unity - Scripting API: UI.Button.onClick.*** **[online] Docs.unity3d.com. Available at: https://docs.unity3d.com/530/Documentation/ScriptReference/UI.Button-onClick.html [Accessed 23 Jan. 2020].**

Answers.unity.com. (2017). ***Restarting the scene when button is pressed - Unity Answers.*** **[online] Available at: https://answers.unity.com/questions/1301342/restarting-the-scene-when-button-is-pressed.html [Accessed 23 Jan. 2020].** – To get the button for reloading the game to work I had a look at that two websites and code examples presented there. As it is actually nearly the same as what my button has to do it was a great source.

Unity Technologies (2019). ***Unity - Scripting API: SceneManagement.SceneManager.LoadScene.*** **[online] Docs.unity3d.com. Available at: https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.LoadScene.html [Accessed 23 Jan. 2020].** – This is one of the examples of unity documentation I used a lot for various functions. I did not include all of the links as they are all from the same website which is this one, but the documentation with all the descriptions of the functions and code examples was very helpful.

Brackeys (2017). ***How to BUILD / EXPORT your Game in Unity (Windows | Mac | WebGL)*****. [video] Available at: https://www.youtube.com/watch?v=7nxKAtxGSn8 [Accessed 23 Jan. 2020].** – A tutorial on exporting and building the whole game so it works not only on this one computer. This tutorial also showed me how to create my own icon for the game application which was really nice and makes everything look more professional. I had some issues with my builds and this video helped me to realise that I should first create a folder and then I will have all the build elements in one place that after zipping will work on another computers.

Todor, A. (2013). ***Blur shader (Pro Only) - Asset Store*****. [online] Assetstore.unity.com. Available at: https://assetstore.unity.com/packages/vfx/shaders/blur-shader-pro-only-6296 [Accessed 23 Jan. 2020]. -** A blur package/script that I ended up not using as it was for the wrong version of Unity and even though the idea of making the game blurred when the deathtext appears was nice I think the way around it so creating some gradients and using another post processing package made the game look better than the blur would do.

Unity Technologies (2017). Assetstore.unity.com. ***Legacy Cinematic Image Effects - Asset Store*****. [online] Available at: https://assetstore.unity.com/packages/essentials/beta-projects/legacy-cinematic-image-effects-51515 [Accessed 23 Jan. 2020].** – A Post Processing package that allows to add some very interesting effects to the main camera of my scene which makes the whole game to look much better. I used among others the chromatic aberration that this package contains and the vignette which both made a huge difference to the look of my game so it looks interesting and like nothing I have seen before. It makes it look like an actual proper full game. This package was also supposed to fix the issues with anti-aliasing that is worse in Unity than in Unreal and it partially did, but it could have been better. I was also experimenting with adding more motion blur but it did not look so aesthetically pleasing.

Mantas Ruigys Software Engineering Student [online communication] – One of my friends I usually make games with. He gave me a lot of ideas about how to fix some of the code, not many of them worked but it was still helpful as I was learning about different functions when I was trying them.

Philip Rudd Game Software Engineering Student [online communication] – Probably the only person that really knows well Unity and C# that I know. He pointed me in the right direction when I could not get the rigid body of the player to be not active at the beginning of the game, before the first click.

Oleg Frazinov lecturer, unit leader [personal communication] – Some help with making the scores to zero out at the beginning of the game play when you reload the scene

Constantinos Glynos demonstrator [personal communication] – He helped me a lot with making the colour changer to work again and not to freeze unity. We discovered together that it was because of the do while loop that was infinite there. We finally got the ColorChanger to always change the colour to a different one. He also first suggested instancing some of the things that I had not instanced.

Teodor Sava, Software Engineering Student [personal communication] – Teo is one of my friends that I usually make games with, he usually does the programming and is like the boss of the whole team so usually designs the levels etc. I asked him for some help with the high scores and death screen loading, so he came to help with that but ended up finding some more issues with my game that we managed to fix together like the issue I had with double scoring on one obstacle and many more. He also forced me to make my game look better as he plays a lot of games so he knew better

than I do what appeals more to the player. He also showed me how to download packages from the asset store. His help was priceless in finishing the game and giving it a final look. He explained a lot of things to me regarding coding and game design, and as it was much better and easier to have someone explain things on a particular example and in person. I think I learned much more from that conversation than from some of the documentation and tutorials.

APPENDIX – Scripts

The code should be clearer, and I should get rid of the things that I am not using but I wanted to show the process so it is left there for now so it is visible how many trials I have done with different code lines before getting it right. There were also other scripts that did not work but they got deleted during the game development process.

player.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class player : MonoBehaviour {


    public float jumpForce = 10f;
    public Rigidbody2D rb;
    public SpriteRenderer sr;
    public string currentColor;
    public string startColor;
    public GameObject endScore;
    public GameObject deathText;
    public GameObject deathGrey;
    public GameObject cameraID;
    public bool touched;
    public int curr_colour_idx;
    public int prev_colour_idx;
    public Color colorCyan;
    public Color colorYellow;
    public Color colorMagenta;
    public Color colorPink;
    public bool isDead = false;
    public int currentScore;
    public Text scoreText;
    public Text scoreText2;
    public ParticleSystem explosion;

    void Start ()
    {

        curr_colour_idx = prev_colour_idx = 0;
        SetRandomColor();
        currentScore= 0;
        startColor = currentColor;
        //GetComponent<Rigidbody2D> ().enabled = false;
        //rb.enabled = false;
        //use rigidbody.isKinematic = false;
        //rb.SetActive(false);
```

```
        //player.SetActive(false);
        //rb.enabled = !rb.enabled;
        //GetComponent<RigidBody>().Sleep()
        rb.simulated = false;
    }

    // Update is called once per frame
    void Update () {
        scoreText.text= currentScore.ToString();
        scoreText2.text= currentScore.ToString();
        //if (isDead)
        //return;

        if (gameObject.transform.position.y < -4f) ///IF ITS BELOW -4 DIE
        {
            gameObject.SetActive(false);
            Debug.Log ("GAME OVER");
            deathText.SetActive (true);
            endScore.SetActive (false);
            deathGrey.SetActive (true);
        }

        if (Input.GetButtonDown("Jump") || Input.GetMouseButtonDown(0))
        {
            rb.simulated = true;
            rb.velocity = Vector2.up * jumpForce;
        }
    }

    void OnTriggerEnter2D (Collider2D col)
    {
        if (col.tag == "ColorChanger")
        {
            SetRandomColor();
            touched= false;
            Destroy(col.gameObject);
            return;
        }

        if (col.tag != currentColor )
        {
            //Death ();
            gameObject.SetActive(false);
            Debug.Log ("GAME OVER");
            deathText.SetActive (true);
            endScore.SetActive (false);
            deathGrey.SetActive (true);
            //NOW I HAVE TO LOAD DEATH MENU
```

```
        }
        else if ( col.tag == currentColor && !touched)
        {
            Debug.Log ("Hurray!");
            currentScore++;
            explosion.transform.position= new
Vector3(explosion.transform.position.x,gameObject.transform.position.y, 0f);
            explosion.Play();
            touched=  true;
            //I need to score then!!!!!!
        }
    }

    void SetRandomColor()
    {
        prev_colour_idx = curr_colour_idx;
        curr_colour_idx = Random.Range(0, 4);
        //Random.seed = 4;

        while (curr_colour_idx == prev_colour_idx)
            curr_colour_idx = Random.Range(0, 4);

        switch (curr_colour_idx)
        {
            case 0:
                currentColor = "Cyan";
                sr.color = colorCyan;
                break;
            case 1:
                currentColor = "Yellow";
                sr.color = colorYellow;
                break;
            case 2:
                currentColor = "Magenta";
                sr.color = colorMagenta;
                break;
            case 3:
                currentColor = "Pink";
                sr.color = colorPink;
                break;
        }

    }

    //public void Death()
    //{
    //  isDead = true;
```

```
    //  GetComponent<score>().OnDeath ();
    //  //Debug.Log ("DEAD");
    //}
}
```

highScoreScripts.cs

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

public class highScoreScript : MonoBehaviour {

    public int highscore;
    // Use this for initialization
    void Start () {
        GameObject theplayer= GameObject.Find("player");
        player player= theplayer.GetComponent<player>();
        highscore= player.currentScore;
    }

    // Update is called once per frame
    void Update () {
        Debug.Log(highscore);
    }
}
```

clickScript.cs

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

public class clickScript : MonoBehaviour {


    public Button yourButton;

    // Use this for initialization
    void Start () {
        Button btn= yourButton.GetComponent<Button>();
        btn.onClick.AddListener(TaskOnClick);
    }

    // Update is called once per frame
    void TaskOnClick(){
```

```
        SceneManager.LoadScene(0);
    }
}
```

DeathMenu.cs

```
using UnityEngine;

public class DeathMenu : MonoBehaviour {

    public DeathMenu deathMenu;

    // Use this for initialization
    void Start () {
    }

    // Update is called once per frame
    void Update () {

    }
    public void ToggleEndMenu (string scoreText)
    {
        return;
    }
}
```

FollowPlayer.cs

```
using UnityEngine;

public class FollowPlayer : MonoBehaviour {

    public Transform player;

    void Update()
    {
        if (player.position.y > transform.position.y)
        {
            transform.position = new Vector3 (transform.position.x,
player.position.y, transform.position.z);
        }
    }
}
```

Rotator.cs

```
using UnityEngine;

public class Rotator : MonoBehaviour {

    public float speed = 100f;

    // Update is called once per frame
    void Update () {
        transform.Rotate(0f, 0f, speed * Time.deltaTime);
    }
}
```