King Abdulaziz University
Faculty of Computing and Information Technology
Computer Science Department
Fall 2023
CPCS391 – Computer Graphics
Course Project

Group#3

Name	ID	Section
Rima Algamdi	2005617	B9
Layla Alsulaimani	2005800	
Wajd Alharbi	2007057	
Dima Kanawti	1917074	

Instructor: Prof. Ghada Aldabbagh
Dr. Huda Aljalood

Table of Contents

Table of Figures

1. Introduction

Computer graphics are utilized widely, and it is used to deliver art and picture data to the spectator in a compelling and effective way. Moreover, it processes image data from the outside world, including pictures and videos. as well as computer-aided engineering design and scientific visualization, in which items are sketched and evaluated in software, which employs graphics and colors to mimic complicated processes like electric fields and air currents.

We studied a variety of computer graphics hardware and software tools in the CPCS391 course, and we also identified certain graphics-related algorithms, and we created a project using a new tool that includes what we learned from CPCS391 course. We decided to create the "Kunafa" game, which is our take on the flappy bird game, in which the player controls a bird and tries to fly between columns of green pipes without striking them.

2. Objectives

- Create an updated version of flappy bird game “Kunafa”.
- Implement the game in a way that will help the player improve their finger dexterity and increase reaction speed.
- Construct the outcomes of CPCS391 course in the game.
- Create a fun game to share with people to play.

3. System and Software

3.1 Hardware

- Processor: Intel(R) Core (TM) i7-7700HQ
- CPU: 2.80GHz 2.80 GHz
- System type: 64-bit operating system
- Operating system: Windows 10

3.2 Software

- Unity: Game engine we used to create the game.
- Visual Studio 2019: Used for programming parts of the game (Script).
- Procreate app: Used to draw the cat character.
- Canva: Used to draw the background.

4. Computer Graphics Description

Our game consists of four main scenes: Main menu, Credits, game over screen and the game play, and 4 main objects: Cat, pipes, the scoring system.

4.1 Menu

The first button of the menu takes us to the game, the second one shows the creators of the game, and the third button quits the game.
The buttons were created using unity UI button and to transfer from scene to scene we connected each button by changing the event that connects to the button to the scene we want to go to.

Figure 1: Main menu

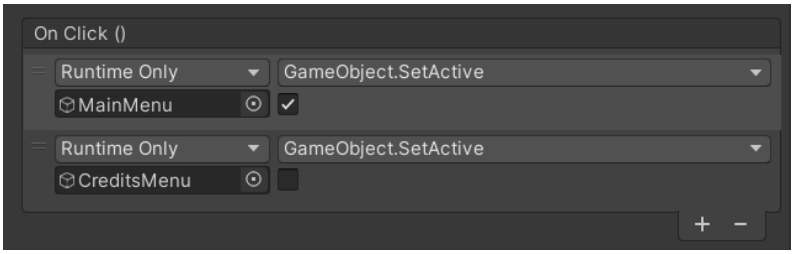

Figure 2: Menu buttons

4.2 Cat (Kunafa)

Who is Kunafa?

Kunafa is the main actor in our game, what makes kunafa special is that it`s designed, drawn, and illustrated by the group.

Figure 3: Kunafa the cat

Features

→ **Type of** Element**:**

Kunafa is a 2d object.

→ **Gravity:**

Kunafa is mainly affected by gravity, without any intervention the cat will keep falling down which considered as a loss.

→ **Movements/Controlling:**

→ Movements: UP.

→ Key pressed: 

space bar key.

In order to not let the cat fall, the cat must keep flying up using the tap key, which will flap the bird. As shown in the following code, as long as the cat is alive, it will be triggered by pressing the space key.

```
if (Input.GetKeyDown(KeyCode.Space) && birdIsAlive)
        myRigidbody.velocity = Vector2.up * flapStrength;
```

→ **Collider:**

The collider in Kunafa is a circle outline, in Kunafa we excluded the ears and the wings from the collider in order to make the game more flexible.

Figure 4: Collider

→ **Collision Triggers:**

The bird is triggered by the pipe's collider shown in figure XX, excluding the red surrounded area which Kunafa can pass the pipe though. As shown in figure XX, when a collision happens, the game will be over and the ***birdIsAlive*** value will be set to "false."

```
private void OnCollisionEnter2D(Collision2D collision)
{
    logic.gameOver();
    birdIsAlive = false;
}
```

Figure 5: Pipe collider

4.3 Pipes

Now although in these types of games it looks like our cat is moving, its actually the pipes that are moving while the cat stays still. To do that, we spawn pipes repetitively and move them along the screen and delete a pipe each time it passes outside of the game parameters to save memory. To import a pipe into our game, we inserted a photoshopped .png pic of a pipe into the project field called "asset." We nested two child objects "top pipe" and "bottom pipe" under the parent object pipe to be able to move them all at once by just moving the parent object. Now, we add a sprite rendered for the pipe image, and add a box collider 2D, because the pipes, unlike Kunafa, will not be affected by physics or gravity.

Figure 6: Pipes

Implementation

PipeMoveScript

To make the parent move across the screen we add a script "**PipeMoveScript**". We initiated a variable called **moveSpeed**, and multiplied that by **Time.deltaTime,** which gives us the speed at which the pipes move along the screen. To delete the spawn pipes when they move out the screen, we picked a certain x position (the variable "**deadZone**") where we want the pipe to be deleted at. Now, each time the pipe's x position moves past the set **deadZone**, the pipe is deleted.

PipeSpawnerScript

Next, we want continually to spawn new pipes. The purpose of the **"PipeSpawnerScript"** is to spawn a new version of the pipes every few seconds. We initiate variables like **spawnRate** (how many seconds it will wait between spawns), and a timer initialized at zero that counts up all the way to the **spawnRate** value. Each time the timer reaches the **spawnRate**, it spawns a new pipe. This is implemented inside the "**void**

spawnPipe()" method and called in the update and start methods. With that done, it will spawn a pipe as soon as the game starts, as well as spawning new pipes each time the timer is maxed out.

To create a variety of positions for the pipes spawned, we also create the variables **lowestPoint** and **heighestPoint** inside the **spawnPipe**() method. The lowest point is equal to the **transform.position.y** value minus the **heightOffset**. we set the y position, **Random.Range(lowestPoint, heighestPoint)** as the min and max values for the y position. Now, pipes will spawn at random y positions each time.

4.4 Score

Logic

We implement the player score system, by creating an invisible collider called triggers, to alarm that two objects have been touched. We placed them in the middle of the prefab pipes. In the "**PipeMiddleScript"** we added a function that will add one point to the score when the cat go through the pipes using the code line: **logic.addScore(1).** And in the in **"LogicScript"** we have a method called **"addScore"** for all scoring related code. And by using the line: **playerScore = playerScore + scoreToAdd** it saves the player's score in player score variable.

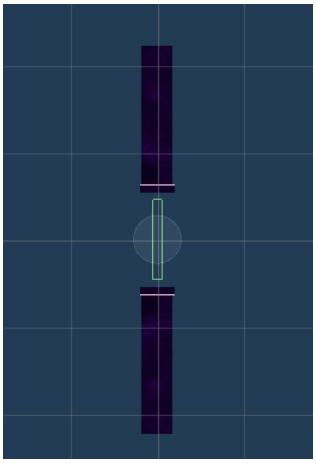

Figure 7: Pipes trigger

UI

The score must add to the UI of the game, for the player to see. We Implemented it by creating a text in the Unity UI and connect it to the logic script as shown in the picture below. In the **"addScore"** method we added the line: **scoreText.text = playerScore.ToString().**

Sound

We added a "ding" sound to activate once the cat scores one point by getting through the pipes. We implemented that by downloading a free sound from unity library then in **"addScore"** method we added the line: **scoreSFX.Play();**

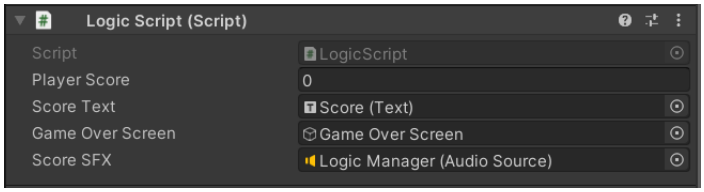

Figure 8: Sounds

4.5 Game Over

The game over screen is activated once the bird collides with the pipes or go above or below the gameplay screen, we implemented the latter using the line:

if (transform.position.y > 15 || transform.position.y < -15).

The game over screen was implemented by disabling the “game over” text and make it active once the “game over” method be activated using the line: **gameOverScreen.SetActive(true).**

The game over screen also includes two buttons: play again and return to Menu. The play again button activates the game once again, as for the return to menu button it takes the player back to the Main menu.

Figure 9: Gameover screen

5. Features from the Course

We were able to make this game thank to what we have learned in CPCS391 course. We learned many important topics like: lights, 2D and 3D objects, texture, rotation, scaling, projection, and events.

5.1 Included

Our game had almost everything we have learned such as:

- 2D Objects: The cat and pipes.
- Camera and lighting.
- Texture: Background picture.
- Rotation: Rotate the pipes.
- Scaling: Scale our objects.
- Events: Buttons to go from scene to scene.
- Interaction: The game interacts with player.
- Gradient colors: Title and button.
- Animation: The cat moves up and down and the pipes moves from right to left.
- More than two Scenes: Menu, Gameplay, Credits, and Game over.
- Scoring system that adds one point to the player each time they get through the pipe without losing.

5.2 Excluded

Our game excluded:

- 3D objects, but every object even if it is 2D, it has depth in Unity, which make it 3D object. But for the player it is still 2D object.
- Saving the player's score. We will add it in the future updates to the game.
- Multiple background pictures. In the future we will allow the player to choose a background from the game store.

6. Conclusion

In conclusion, this project has summed up everything we have learned throughout the semester and put it in nice ending project. We have learned so much while creating this game, even though using Unity was a struggle at the beginning, we managed and made a game we are proud of.

We shared our game on GitHub at the link https://github.com/Dima-Kanawati/Kunafa.