Politehnica University of Bucharest

Game and interactive simulation systems
Report

Final Project

Student :
Charly Ginevra

Professor :
Bujorel PAVALOIU

February 7, 2023

Contents

1 Presentation of the project **2**
1.1 The game . . . 2
1.2 Inspiration . . . 2
1.3 Techonology used . . . 2

2 Game development **3**
2.1 Creating the characters . . . 3
2.1.1 The chicken . . . 3
2.1.2 The cars . . . 3
2.2 Creating the environment . . . 4
2.3 Animating the objects . . . 5
2.3.1 The cars . . . 5
2.3.2 The chicken . . . 5
2.4 Collisions with the environment . . . 6
2.5 The fireballs . . . 6

3 Musics **7**
3.1 The menu . . . 7
3.1.1 Main menu . . . 7
3.1.2 Settings menu . . . 8

4 Conclusion **9**

5 Bibliography **10**

6 Annex **11**
6.1 Car class . . . 11
6.2 Fireball class . . . 11
6.3 Spell class . . . 12

1 Presentation of the project

1.1 The game

My game name Crossy Road Legacy and it is a third person shooter. The player controls a chicken on a map crossed by three roads. The chicken can through fireballs in order to destroy a car. I a car hit the player then he lose a life and re-appear at the starting point. If the player has no more life available, it's the game over. A game last between (duration) and (duration). If the player still alive at the end of the game, he wins. The score depends on the number of car destroyed all along the game.

1.2 Inspiration

This game is grandly inspired by Crossy Road. A mobile game in which the user also controls the chicken. The objective was to cross as most road as possible without being hit by a car.

Figure 1: Banner of the game Crossy Road

I have try to be as much as possible of the esthetic of the orginal game. That is why the cars, the chicken and the environment looks the same. I've added the ideas of the fireball and now the objective is to destroy as much car as possible without begin hit by a car.

1.3 Techonology used

Unity is a game engine to develop games in multiple platforms such as Windows, Linux and MacOS. In the editor you can create you own 3D objects. A lot of predefined properties are available such as collision, gravity or also controllers. You also have the possibility to connect your 3D objects to C# script, which is basically an object that will be attached to the object.

I used this technology for convenience. We have used it in class and the teacher recommended us the book Unit In Action, by Joseph Hocking. There was very useful chapter talking about on to create a First Person Shooter which needed some adjustments to correspond to my use case.

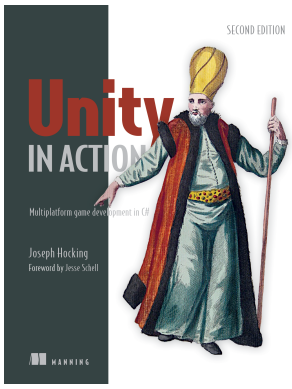

Figure 2: Unity In Action

2 Game development

2.1 Creating the characters

In order the keep as much authenticity as possible, I decided to make everything myself and not using Unity Asset Store at all (also because the game objects are not really hard to make).

2.1.1 The chicken

Like a puzzle game, the character are only composed of blocks that I have assembled.

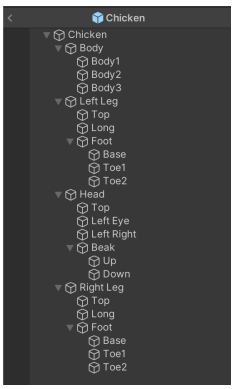

Figure 3: Tree view of the chicken

I have decided to turn the player into a prefabs, a reusable component. I think it is a good practice because we never know what will be the next feature to be implement. For example, if I want to create a cooperative mode, I have the prefabs for the second available for player. Also, if I want to apply a property to all the players, it will be easier this way.

2.1.2 The cars

I used the same philosophy in order to create the cars.

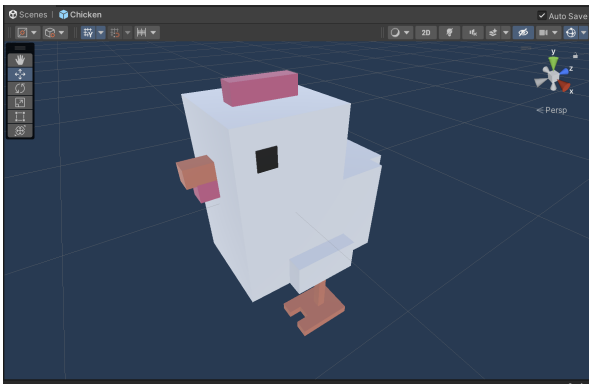

Figure 4: Chicken in the editor

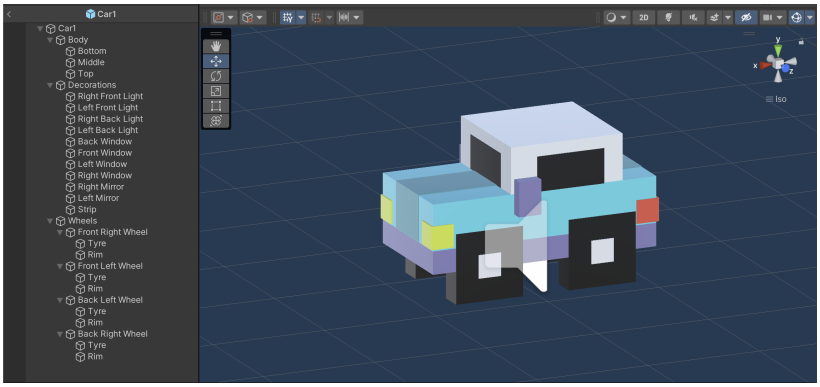

Figure 5: Car in the editor

2.2 Creating the environment

In the original Crossy Road the game is infinite. The map is aut generated as long as the player moves into the level.

In my game, I wanted the play ground to be small but also with a lot of interactions. The map is crossed by three road in which cars are coming from both sides. The game zone is also delimited by transparent walls which can't be crossed by the player.

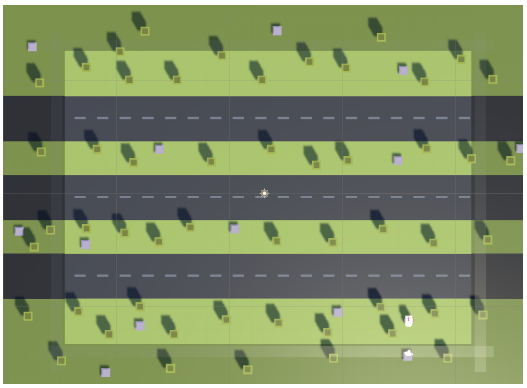

Figure 6: Play ground upper view

In terms of dimension the map, in width it corresponds to thirty times is size of the player and in height forty times.

2.3 Animating the objects

2.3.1 The cars

The first idea was to keep a certain amount of car, defined by user, on the map. Each car has for objective to cross the map form left to the right or right to left according to the starting position. When a car reach the other side of the map, it will be destroy.

This tricky thing put aside, the code of a car object is very simple because the object goes always forward.

Listing 1: Car script

```
public class Car : MonoBehaviour
{
    public float speed = 9.0f;

    void Update()
    {
        transform.Translate(speed * Time.deltaTime, 0, 0);
    }
}
```

2.3.2 The chicken

In order to have on abstraction of input settings and make the development easier and faster, I have decided to follow the advice from the book Unity In Action and use the Input Manager. In the code below I also use the Character Controller component to perform the movement.

Listing 2: Car script

```
public class Chicken : MonoBehaviour
{
    public float sensitivityHor = 9.0f;
    public float speed = 9.0f;

    ...

    private CharacterController charController;

    void Start()
    {
        charController = GetComponent<CharacterController>();
    }

    void Update()
    {
        ...

        // Translate
        float deltaX = Input.GetAxis("Vertical") * speed;
```

```
        Vector3 movement = new Vector3(0, 0, -deltaX);

        movement = Vector3.ClampMagnitude(movement, speed);
        movement.y = gravity;
        movement *= Time.deltaTime;
        movement = transform.TransformDirection(movement);

        charController.Move(movement);

        // Rotation
        transform.Rotate(0, Input.GetAxis("Horizontal") *
            sensitivityHor * Time.deltaTime, 0);

        ...
    }

    ...
}
```

2.4 Collisions with the environment

As I said in the previous section, the player object use the Character Controller component to perform the movement. An other advantage of this component is to take care of the collision with the environment. Unfortunately, the only thing that I have to implement myself is the gravity. To do that, I apply a constant force which push the player down.

2.5 The fireballs

The Chicken is able to throw fireball in order to destroy cars. The goals were to make the direction linear (not reached) and react on collision.

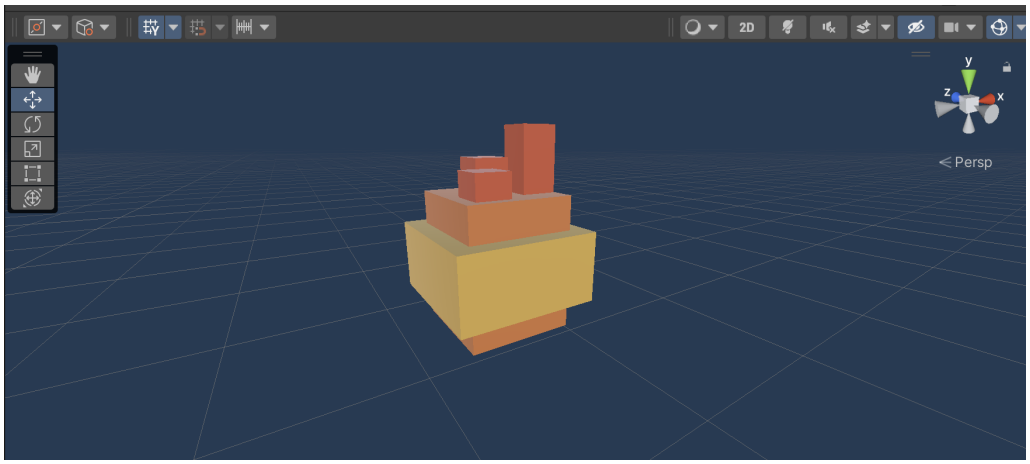

Figure 7: Fireball prefab

To be thrown, the Chicken have a script *Spell* which react on a click. A Fireball is created in front of the player and a force is added to move in the player direction.

In order to react to a collision a box collider has been added to the prefab. The prefab also comes with a script *Fireball*, in annex. In this script, the method *OnTriggerEnter* is implemented and is executed when a collision happens. If the collide object contains a *ReactiveTarget*, it means it was a car, then the method *ReactToHit* is executed. In any case, the object is destroyed after a collision.

3 Musics

During the game, a music is played. To do so, I used an Audio Source component on the SceneController object. This component has a property *Play On Awake*, which allow to play the sound when the object is created.

In order to play a sound as reaction to an event, for example the roar of Godzilla, I used the same component but with *Play On Awake* disabled. In the Chicken class, each time the user press the right click the audio is played.

Listing 3: Play sound in Chicken script

```
public class Chicken : MonoBehaviour
{
    ...
    public AudioSource Roar;
    ...

    void Update()
    {
        ...

        if (Input.GetMouseButtonDown(1) && Roar != null)
            Roar.Play();
    }

    ...
}
```

3.1 The menu

3.1.1 Main menu

The first thing was to create a new scene for the menu.

The main component for the menu is the canvas. It contains, the title and the three button, *Play* to run the game, *Settings* open the settings menu and *Exit* to quit the game.

In order to switch between the menu and the game scenes, I used the SceneManager. Bascilly, in is a list that you create at build. You can use the SceneManger package to switch the scene by indicating the index of the scene in the list.

Listing 4: SceneManager in MainMenu class

Figure 8: Main Menu

```
private const int GAME_SCENE = 1;

public void PlayGame()
{
    SceneManager.LoadScene(GAME_SCENE);
}
```

To keep the project simple, I decided to not create a third scene for the setting menu. I preferred to create a new component which will contains all the elements of the settings menu and them appear on click.

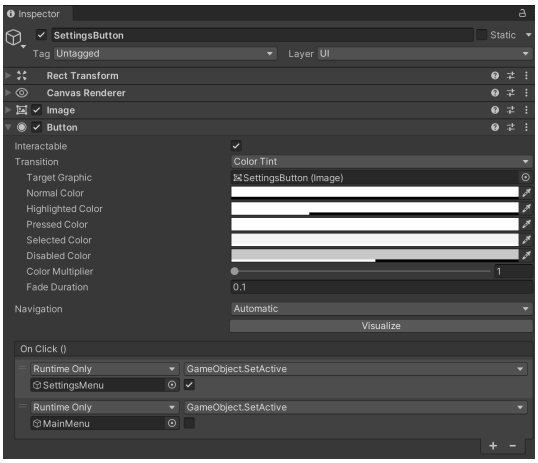

Figure 9: Settings Menu Button Properties

3.1.2 Settings menu

In propose of the settings menu is to give the user the ability to customize the game. Axes of customization are, the number of life, the game duration and the number of enemies (car). Those three variables are modifiable via sliders.

In order to send the information to the other scenes, I used the PlayerPref package. It can be easily compared as a manager for environment variables inside Unity.

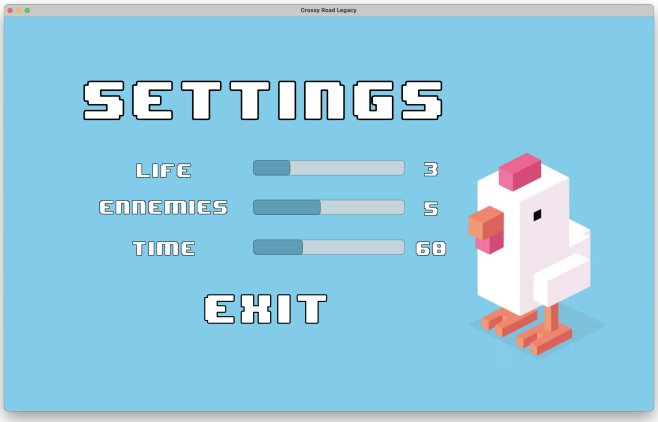

Figure 10: Settings Menu

4 Conclusion

One of the uncertainty of this project was about the build. I use the application Unity on a MacBook M1, and I assume the teacher (or great majority of the player) use Windows 10 or 11. That means the game is developed on another operating system with an other processors architecture (ARM for MacOS and AMD64 for Windows). The problem was solved pretty easily thanks to unity build system which provides the ability to build for multiple platforms.

As a general conclusion, this project allowed my to discover and to make an idea about game development. Before this module, this universe was an unknown for me. Due to his format (two homeworks and one project), this module combine learning new concepts and allow us to express our creative and made it fun to follow.

5 Bibliography

References

[1] Hocking, Joe, *Unity in Action: Multiplatform game development in C#*, 2022.

6 Annex

6.1 Car class

```
public class Car : MonoBehaviour
{
    public float speed = 9.0f;

    void Update()
    {
        transform.Translate(speed * Time.deltaTime, 0, 0);
    }
}
```

6.2 Fireball class

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


public class FireBall : MonoBehaviour
{
    public float speed = 10.0f;

    private CallBack cb;

    void Update()
    {
        //transform.Translate(0, 0, speed * Time.deltaTime);
    }

    void OnTriggerEnter(Collider other)
    {
        ReactiveTarget player = other.GetComponent<ReactiveTarget>();
        if (player != null)
        {
            player.ReactToHit();

            if (cb != null)
            {
                cb.Call();
                cb = null;
            }
        }
        Destroy(this.gameObject);
    }
```

```
    public void SetOnHit(CallBack cb)
    {
        this.cb = cb;
    }
}
```

6.3 Spell class

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

public class Spell : MonoBehaviour
{
    [SerializeField] GameObject fireballPrefab;

    [SerializeField] GameObject gameController;

    public AudioSource sound;

    // Update is called once per frame
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            GameObject fireball = Instantiate(fireballPrefab, transform.pos
            fireball.GetComponent<FireBall>().SetOnHit(new FireBallCallback

            Rigidbody fireballRigibody = fireball.GetComponent<Rigidbody>()

            Vector3 forceToAdd = (transform.forward*-1) * 10f + transform.u

            fireballRigibody.AddForce(forceToAdd, ForceMode.Impulse);

            if (sound != null)
                sound.Play();
        }
    }

    private class FireBallCallback: CallBack
    {
        GameObject gameController;

        public FireBallCallback(GameObject gameController)
        {
            this.gameController = gameController;
```

```
        }

        public void Call()
        {
            Score score = this.gameController.GetComponent<Score>();
            score.AddScore(100);
        }
    }
}
```