COMP222 Game Principle

Plane VS. Asteroids (Unity Engine)

Report

1 Introduction

In this 3D game, the player needs to control a plane to destroy all the asteroids within three lives. The plane can be operated as follows: “A” and “D” for rotations in left and right; “W” and “S” for move forwards and backwards; “C” and “V” for descend and ascend; “Space” or “Left click” for shooting. When an asteroid is shot, it will divide into two smaller pieces or be totally destroyed and player will get 100 scores accordingly. Once the score reaches the 1200, the difficulty level will be set from 1 to 2, where the speed of the bullet will be half slower than level 1, which is more difficult for player to shoot the asteroid from the long distance (Closer to the asteroid means higher risk of collision and thus means higher difficulty). When reaching the score of 2000, the level will be set to three where the speed of the bullet will be half slower as the level 2.

Each time when player is hit by the asteroid, the player will die and lives will be subtracted by one and player will reborn after three second as long as lives is still more than zero. If all three lives are used, “game over” UI will be shown and if all the asteroids are destroyed, “victory” UI will be shown.

To make the game fancier, there will be a small explosion particle effect once the asteroids are hit or player is collided. Lives, number of left asteroids, difficulty level and score will be shown at the left corner of game. And player can choose to quit or restart when game over or victory.

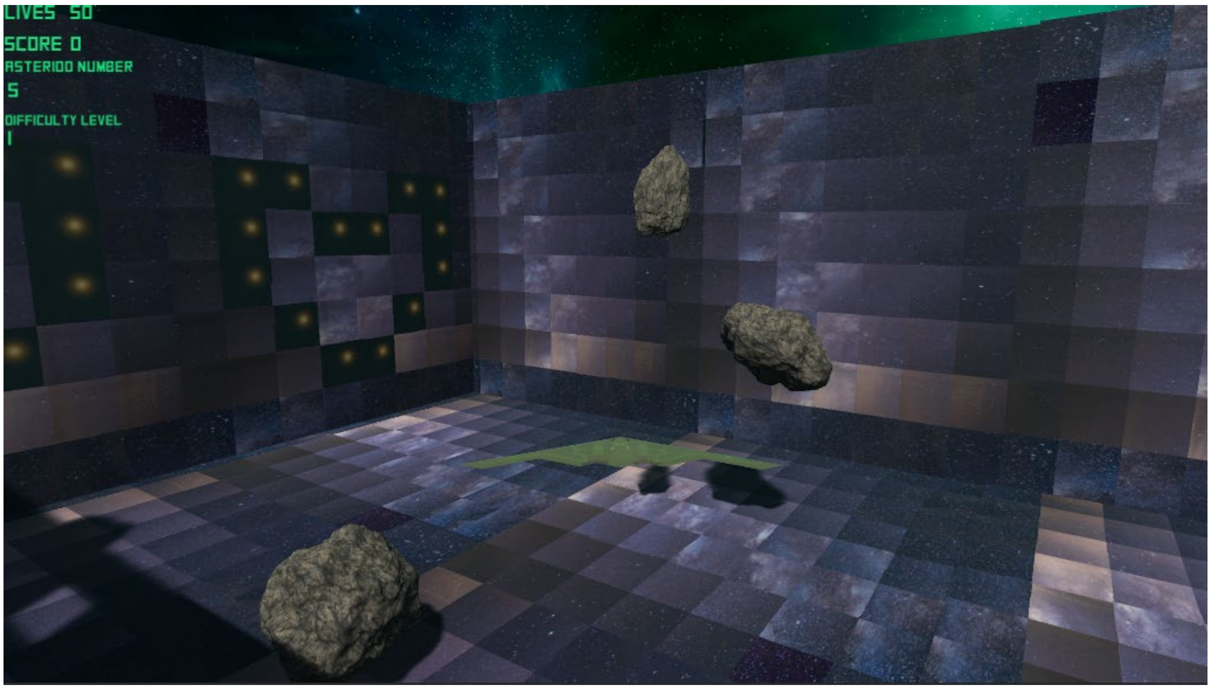


Figure 1. Basic game interface

2 Check List

2.1 List one: Modelling and environment (10%)

The game has two main environments: the star sky with light and the box in which the asteroids fly.

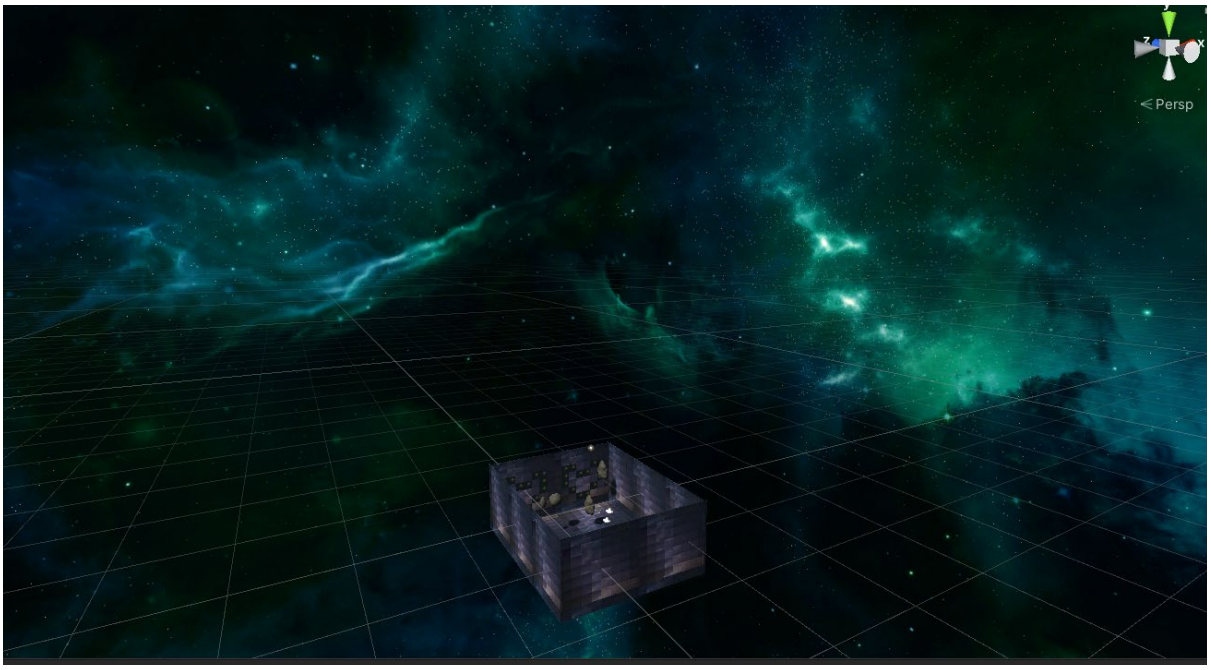


Figure 2. The environment in the game

The asteroids cannot fly outside the purple box but to make the game easier to play, the plane can fly through the wall to avoid some sudden collision with asteroids (fixed view camera in 3D scene makes it difficult to see some asteroids from the back and above). What is more, the light from the sky will cast asteroids' shadow on the ground which can helps player to find out whether there is an asteroid flying above when ascending.

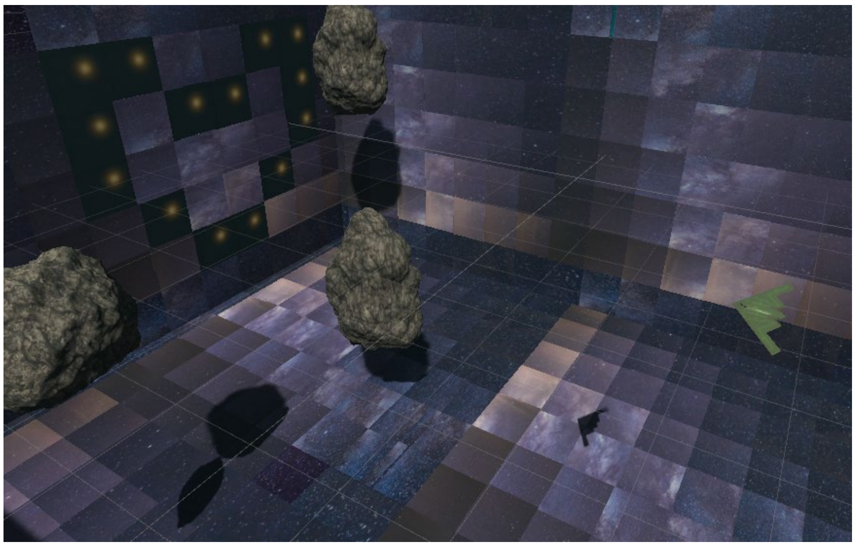


Figure 3. Shadow of the asteroids and plane

2.2 List two: Keyboard interaction (20%)

As is mentioned in the introduction, “A” and “D” for rotations in left and right; “W” and “S” for move forwards and backwards; “C” and “V” for descend and ascend; “Space” or “Left click” for shooting.

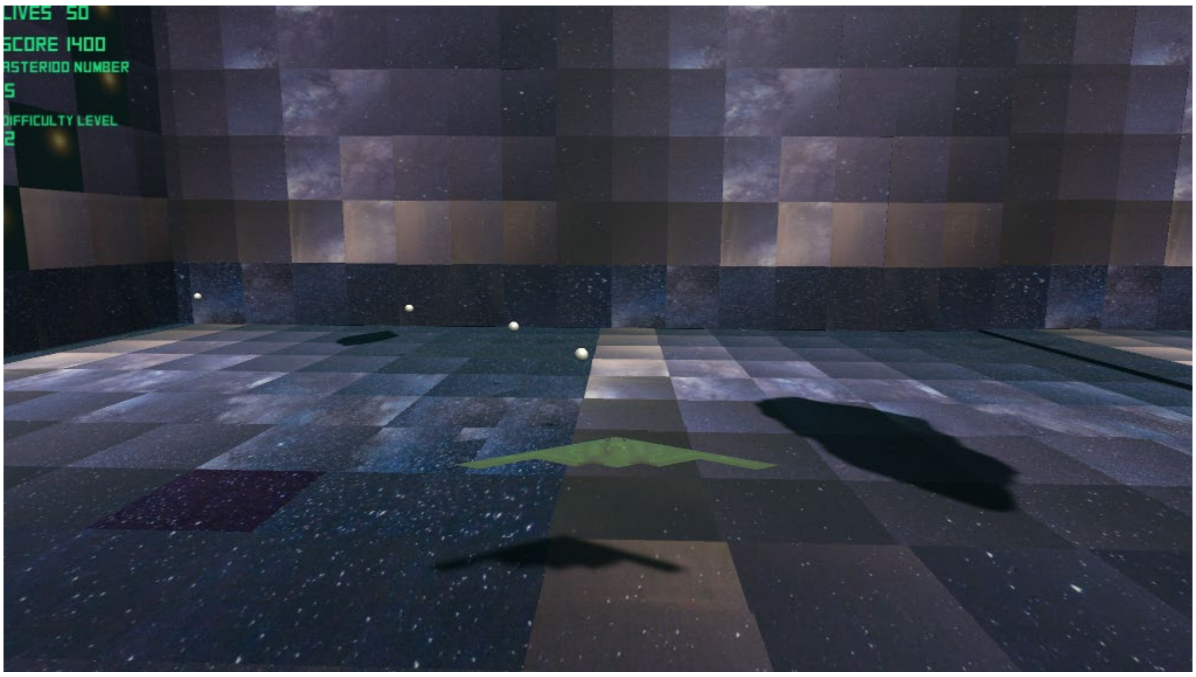


Figure 4. Fly and shooting

2.3 List 3: Physics (30%)

The asteroids have zero gravity and floating in the space and it has “Bounce” reflect once it collides with the walls or other asteroids.

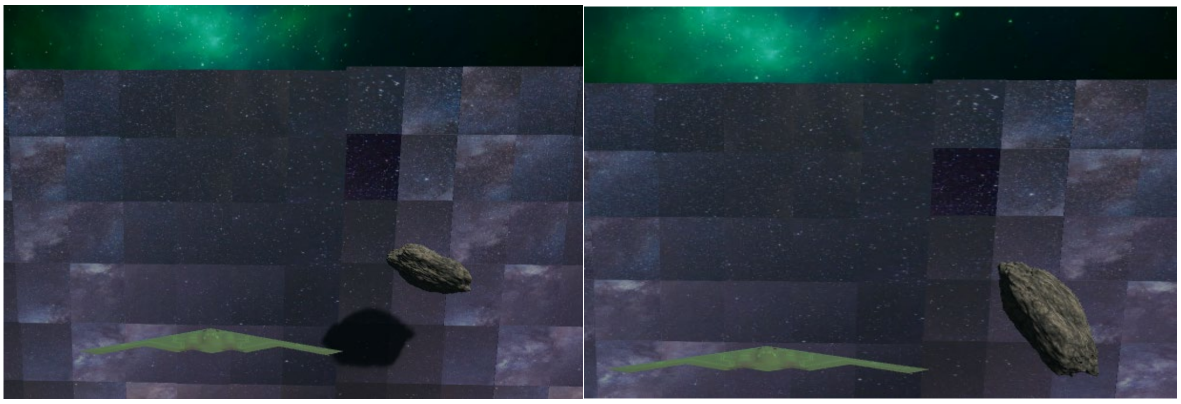


Figure 5. The bounce before hitting and after hitting the walls

When the asteroids hit by the bullets, it will “break” into two smaller pieces or be destroyed directly.

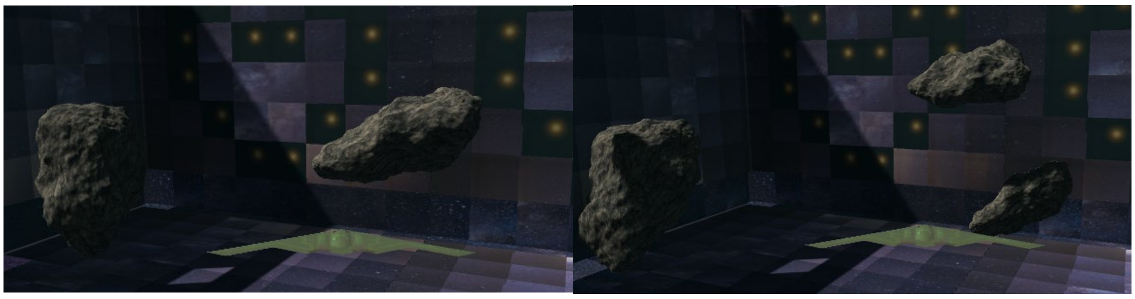
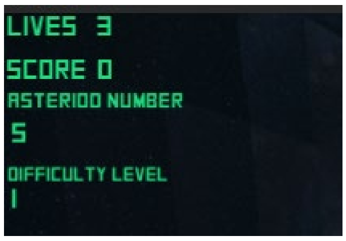


Figure 6. Asteroid breaks when hit by bullets

2.4 List 4: Gameplay (20%)

Lives, number of left asteroids, difficulty level and score will be shown at the left corner of game.



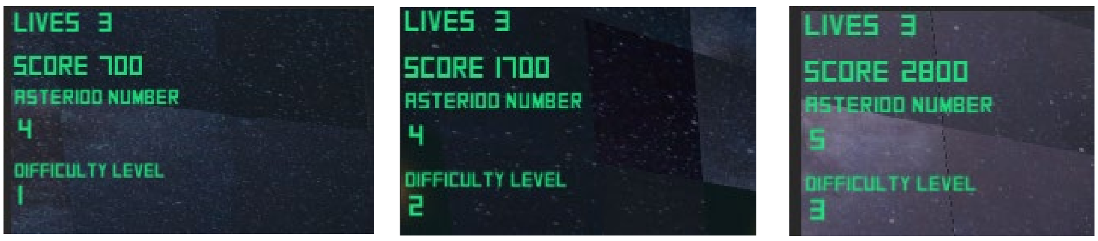
```
LIVES 3
SCORE 0
ASTERIOD NUMBER
5
DIFFICULTY LEVEL
1
```

Figure 7. Display of information

And when the difficulty increases, the speed of bullet will be slower, means player must be closer to shoot the asteroid and will take higher collision risk and thus means higher difficulty.

(ie. It is difficult to capture the speed change in the picture, you need to play the game by yourself! 😊)

And there are three difficulty level in total.



```
LIVES 3
SCORE 700
ASTERIOD NUMBER
4
DIFFICULTY LEVEL
1
```

```
LIVES 3
SCORE 1700
ASTERIOD NUMBER
4
DIFFICULTY LEVEL
2
```

```
LIVES 3
SCORE 2800
ASTERIOD NUMBER
5
DIFFICULTY LEVEL
3
```

Figure 8. Three different difficulty levels

2.5 List 5: Creativity (20%)

First, the game has a light source and a beautiful scene.

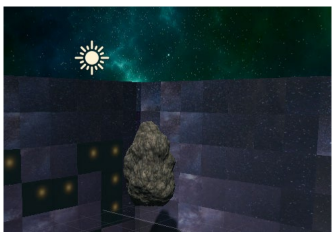


Figure 9. The light source

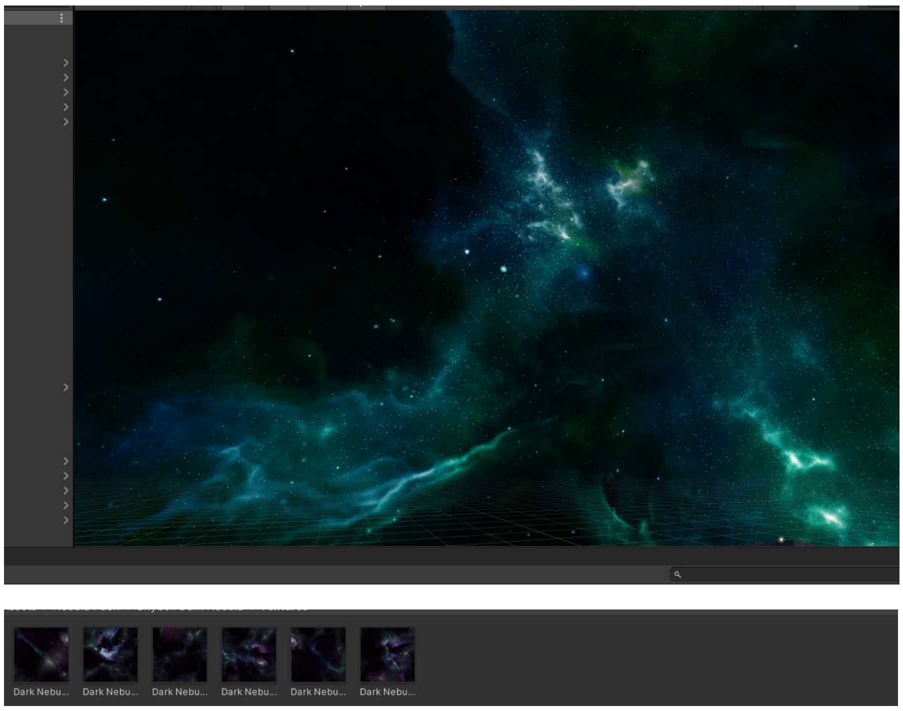


Figure 10. The scene with textures

What is more, the plane itself also has detailed textures and material reflections.

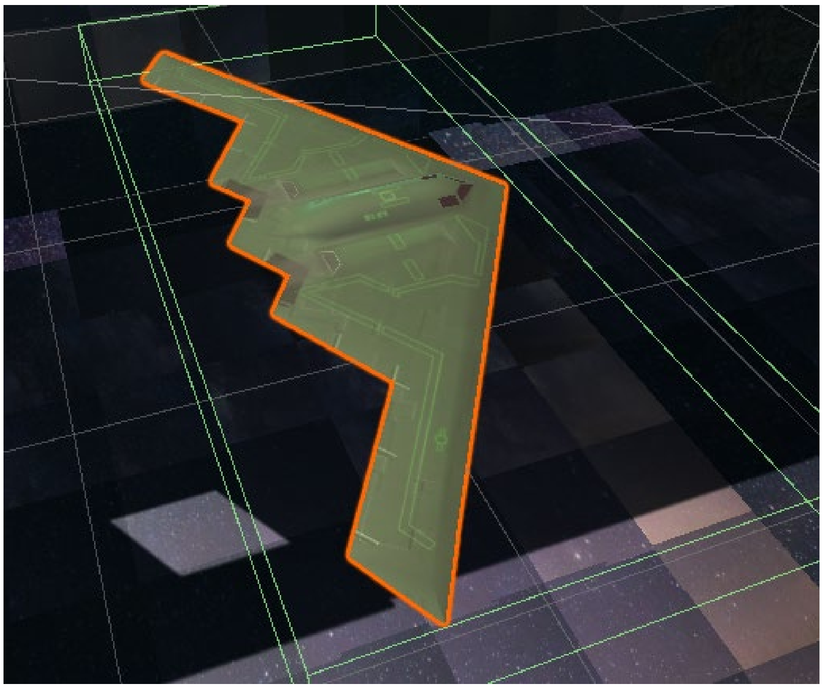


Figure 11. Plane texture

And when the asteroids and plane destroyed, there will be particle effects at the shooting point.

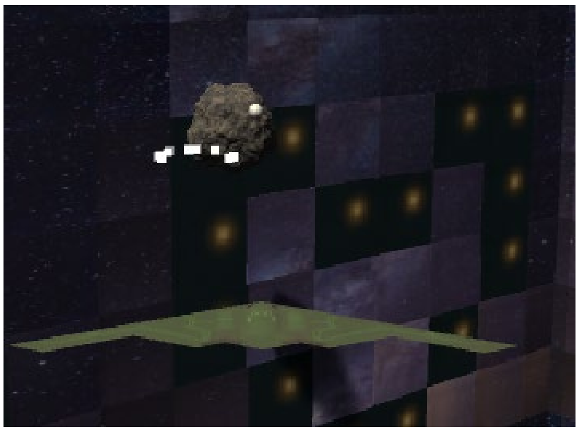
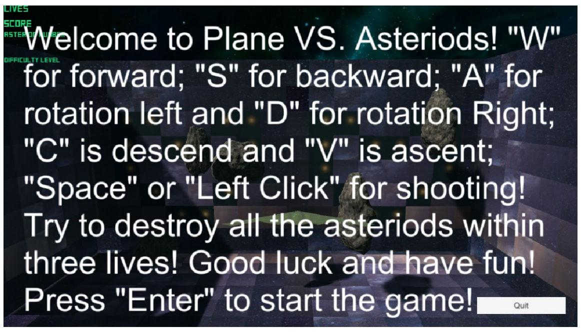
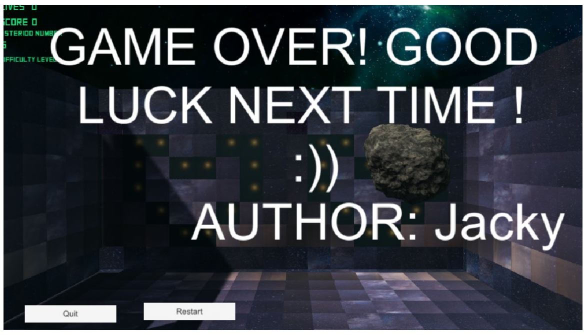
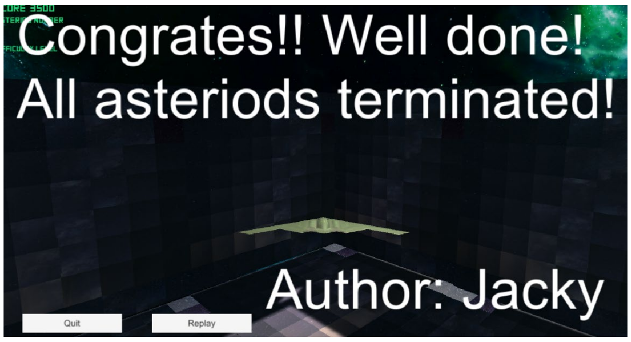


Figure 12. Particle effect

For the user interface, there are buttons and instructions for user to follow. Player can choose quit or restart when dying or victory.


Welcome to Plane VS. Asteriods! "W" for forward; "S" for backward; "A" for rotation left and "D" for rotation Right; "C" is descend and "V" is ascent; "Space" or "Left Click" for shooting! Try to destroy all the asteriods within three lives! Good luck and have fun! Press "Enter" to start the game!


GAME OVER! GOOD LUCK NEXT TIME ! :))
AUTHOR: Jacky

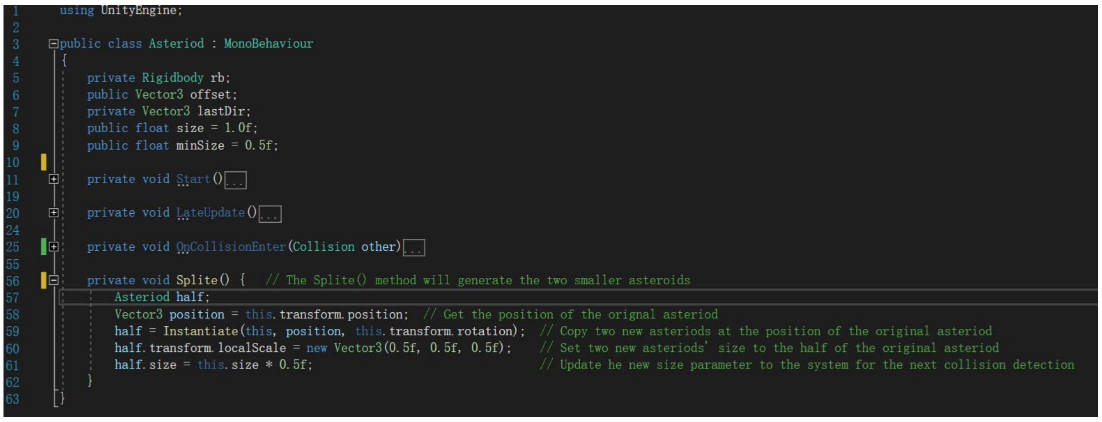

Congrates!! Well done! All asteriods terminated!
Author: Jacky

Figures 13. Different user interfaces and buttons

3 Codes and methods

The logic and control of the whole game is consisted five scripts: Asteroid, Bullet, ExitGame, GameManager, Players, RestartGame.

The Asteroid script defines the asteroid and the collision detection and split methods for “Break”.

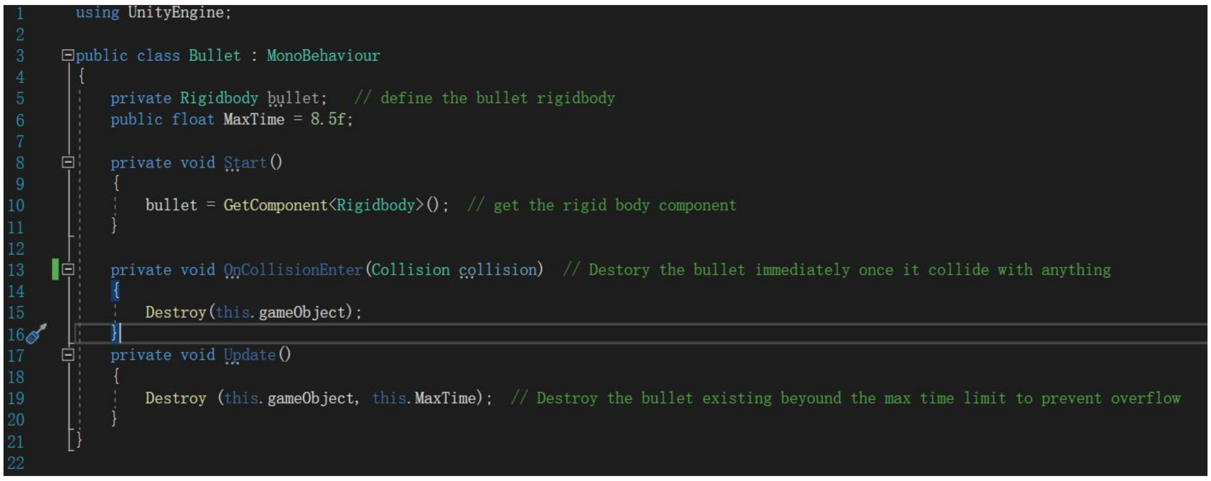


```
using UnityEngine;

public class Asteriod : MonoBehaviour
{
    private Rigidbody rb;
    public Vector3 offset;
    private Vector3 lastDir;
    public float size = 1.0f;
    public float minSize = 0.5f;

    private void Start()[...]

    private void LateUpdate()[...]

    private void OnCollisionEnter(Collision other)[...]

    private void Splite() {   // The Splite() method will generate the two smaller asteroids
        Asteriod half;
        Vector3 position = this.transform.position;  // Get the position of the orignal asteriod
        half = Instantiate(this, position, this.transform.rotation);  // Copy two new asteriods at the position of the original asteriod
        half.transform.localScale = new Vector3(0.5f, 0.5f, 0.5f);    // Set two new asteriods' size to the half of the original asteriod
        half.size = this.size * 0.5f;                                 // Update he new size parameter to the system for the next collision detection
    }
}
```

Figure 14. The skeleton of Asteroid script (60 lines)

(ie. See the script for comments and detail)

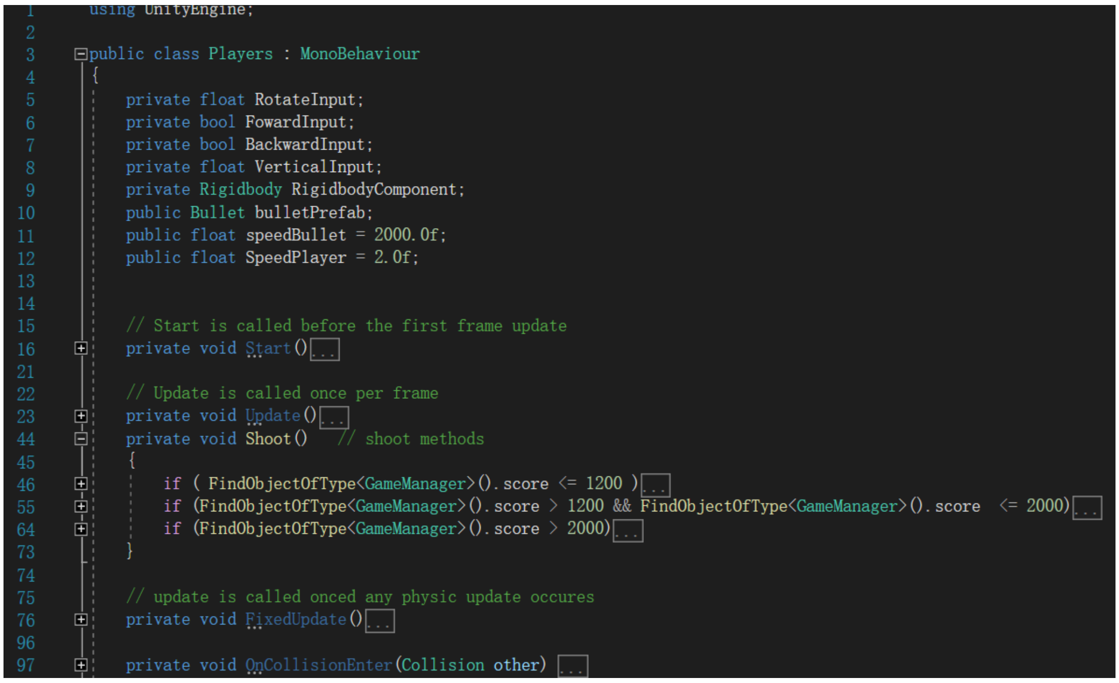
The Bullet script defined the bullet and destroy method.



```
using UnityEngine;

public class Bullet : MonoBehaviour
{
    private Rigidbody bullet;   // define the bullet rigidbody
    public float MaxTime = 8.5f;

    private void Start()
    {
        bullet = GetComponent<Rigidbody>();  // get the rigid body component
    }

    private void OnCollisionEnter(Collision collision)  // Destory the bullet immediately once it collide with anything
    {
        Destroy(this.gameObject);
    }
    private void Update()
    {
        Destroy (this.gameObject, this.MaxTime);  // Destroy the bullet existing beyound the max time limit to prevent overflow
    }
}
```

Figure 15. Complete code for Bullet script (20 lines)

The Player script is two of the core scripts. It records all the behaviour of the player and defines the methods for shooting, collision detection and score updating.

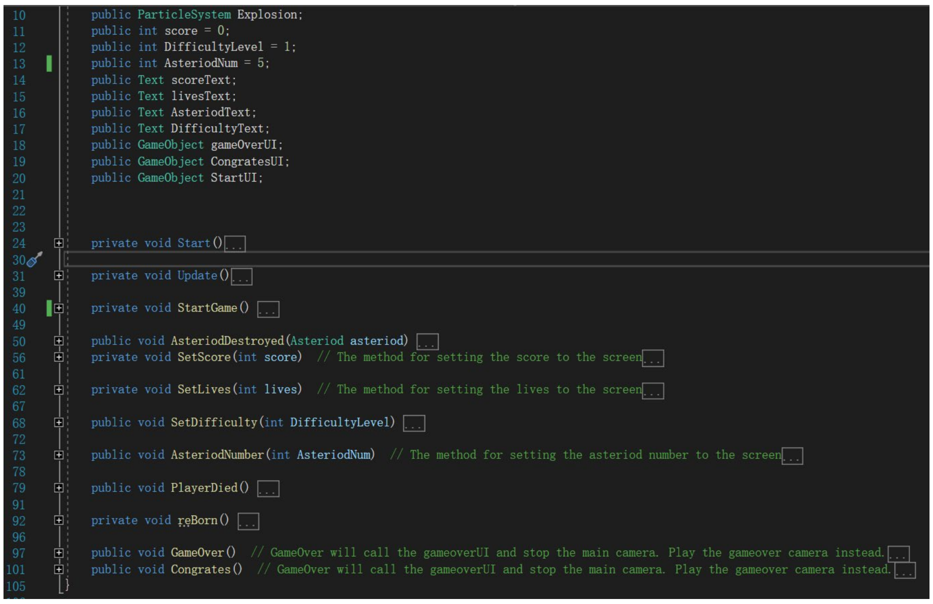


```
using UnityEngine;

public class Players : MonoBehaviour
{
    private float RotateInput;
    private bool FowardInput;
    private bool BackwardInput;
    private float VerticalInput;
    private Rigidbody RigidbodyComponent;
    public Bullet bulletPrefab;
    public float speedBullet = 2000.0f;
    public float SpeedPlayer = 2.0f;


    // Start is called before the first frame update
    private void Start()[...]

    // Update is called once per frame
    private void Update()[...]
    private void Shoot()    // shoot methods
    {
        if ( FindObjectOfType<GameManager>().score <= 1200 )[...]
        if (FindObjectOfType<GameManager>().score > 1200 && FindObjectOfType<GameManager>().score  <= 2000)[...]
        if (FindObjectOfType<GameManager>().score > 2000)[...]
    }

    // update is called onced any physic update occures
    private void FixedUpdate()[...]

    private void OnCollisionEnter(Collision other) [...]
```

Figure 16. The skeleton of Player script (90 lines)

(ie. See the script for comments and detail)

The GameManager script controls all the behaviour of the game including recording and displaying scores, lives, number of asteroids and difficulty level and methods for reborn and die. And it controls the different user interface under different conditions. What is more, it controls the particle effects in the game. It is another main core script of the game

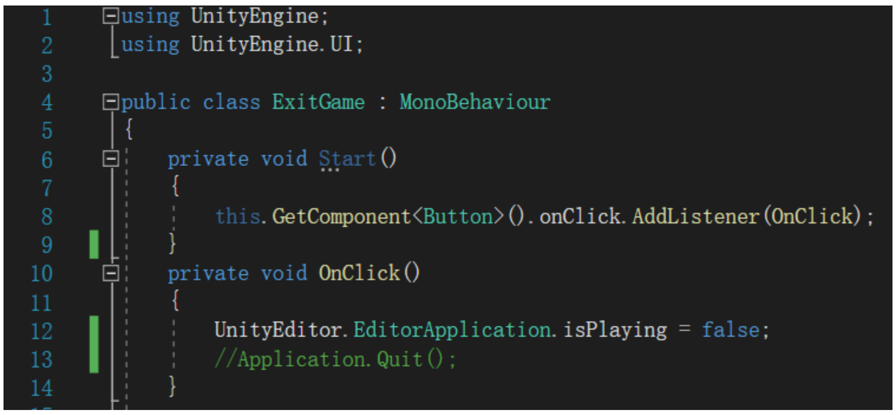


```
    public ParticleSystem Explosion;
    public int score = 0;
    public int DifficultyLevel = 1;
    public int AsteriodNum = 5;
    public Text scoreText;
    public Text livesText;
    public Text AsteriodText;
    public Text DifficultyText;
    public GameObject gameOverUI;
    public GameObject CongratesUI;
    public GameObject StartUI;



    private void Start()[...]

    private void Update()[...]

    private void StartGame() [...]

    public void AsteriodDestroyed(Asteriod asteriod) [...]
    private void SetScore(int score)  // The method for setting the score to the screen[...]

    private void SetLives(int lives)  // The method for setting the lives to the screen[...]

    public void SetDifficulty(int DifficultyLevel) [...]

    public void AsteriodNumber(int AsteriodNum)  // The method for setting the asteriod number to the screen[...]

    public void PlayerDied() [...]

    private void reBorn() [...]

    public void GameOver()  // GameOver will call the gameoverUI and stop the main camera. Play the gameover camera instead.[...]
    public void Congrates()  // GameOver will call the gameoverUI and stop the main camera. Play the gameover camera instead.[...]
}
```

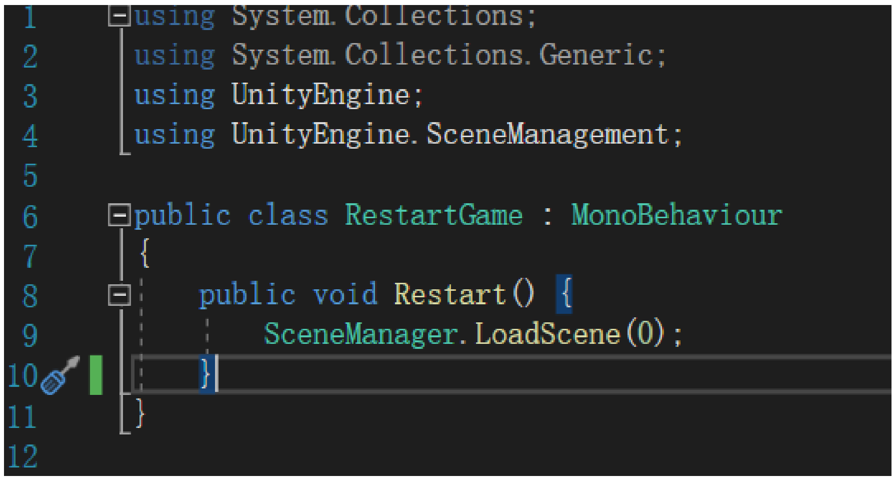
Figure 17. The skeleton of GameManager script (100 lines)

(ie. See the script for comments and detail)

The last two scripts are ExitGame and RestartGame are two buttons for quit and restart.



```
using UnityEngine;
using UnityEngine.UI;

public class ExitGame : MonoBehaviour
{
    private void Start()
    {
        this.GetComponent<Button>().onClick.AddListener(OnClick);
    }
    private void OnClick()
    {
        UnityEditor.EditorApplication.isPlaying = false;
        //Application.Quit();
    }
```



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

public class RestartGame : MonoBehaviour
{
    public void Restart() {
        SceneManager.LoadScene(0);
    }
}
```

Figure 18. Complete code for ExitGame and RestartGame scripts

4 Conclusion

In this game design assignment, we know that the rigid body collision detection in Unity Engine. And we know how to use the Start, Update, Fixupdate and Awake class properly. Moreover, we know how to use the tag and layer system in the Unity Engine. Lastly, we get know how to use the Vector3 to represent all the position, velocity and all the movements. This is the very first time that I designing the 3D game and hope to do better and faster with Unity Engine in the next time.