FINAL REPORT
3D Greedy Snake

Wenbo Du, Jiading Li, Xinyuan Zhang

• *Introduction.*

In our project, we decided to create a classical game, “Greedy Snake”. Because this game was a famous 2D game in the early 20th century, and it is a good memory of our childhood. We think it is pretty cool to design a 3D greedy snake game.

All details and implementation of our project are based on the Unity and C# script. Our plan is to follow agile project development methods. We broke the whole project into a few sub-projects to implement, such as the environment, snake body design, movement of the snake and player control, game object design (fruit, obstacle...). We currently focus on the design of the environment, the snake's body and its movement. In this mid-term report, we will show something about these tasks. The idea to achieve these tasks is building the object in Unity, and designing its functions or operations in C# script. It’s like when using HTML and javascript for web design, HTML design is used for the web body, and javascript is used to achieve functions. Additionally, the project also uses MVC framework, so it is hard to fully match the MVC framework in Unity. But we still follow the idea of it.

We created a level file that includes all components of the environment, and a file called script includes all functions and operations of objects. And all single objects in a file are called models. It means, for example, the environment includes both walls and grounds. We didn’t design them into one big object, instead we split them into single walls and grounds then connected them to a big environment object.

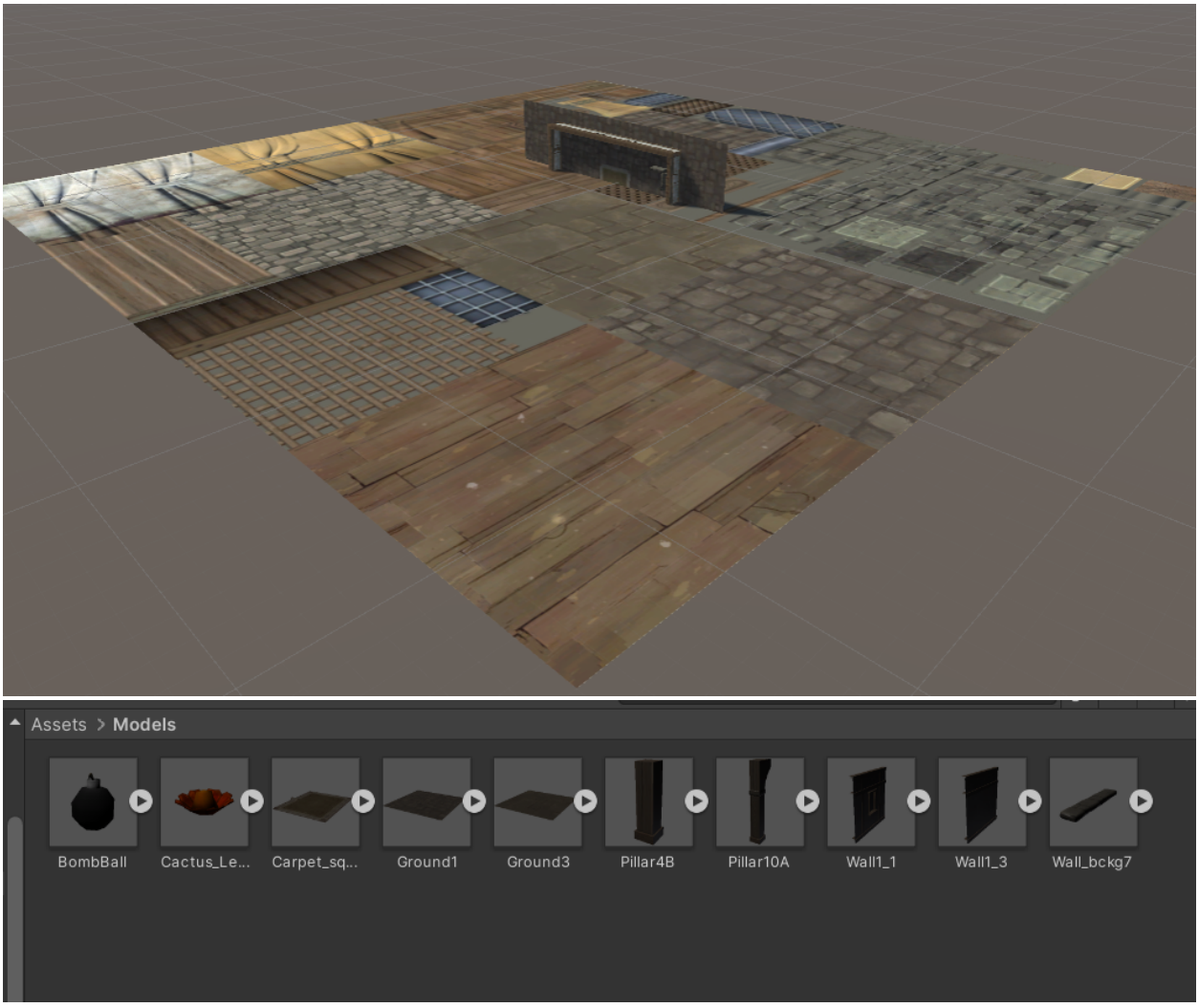
The VR equipment we used is Oculus, which is a very popular platform at present. And we have realized the function of observing the game graphics in the VR headset.

The link to the demo is included in the PPT presentation.

• *Technical Approach.*

Scene design:

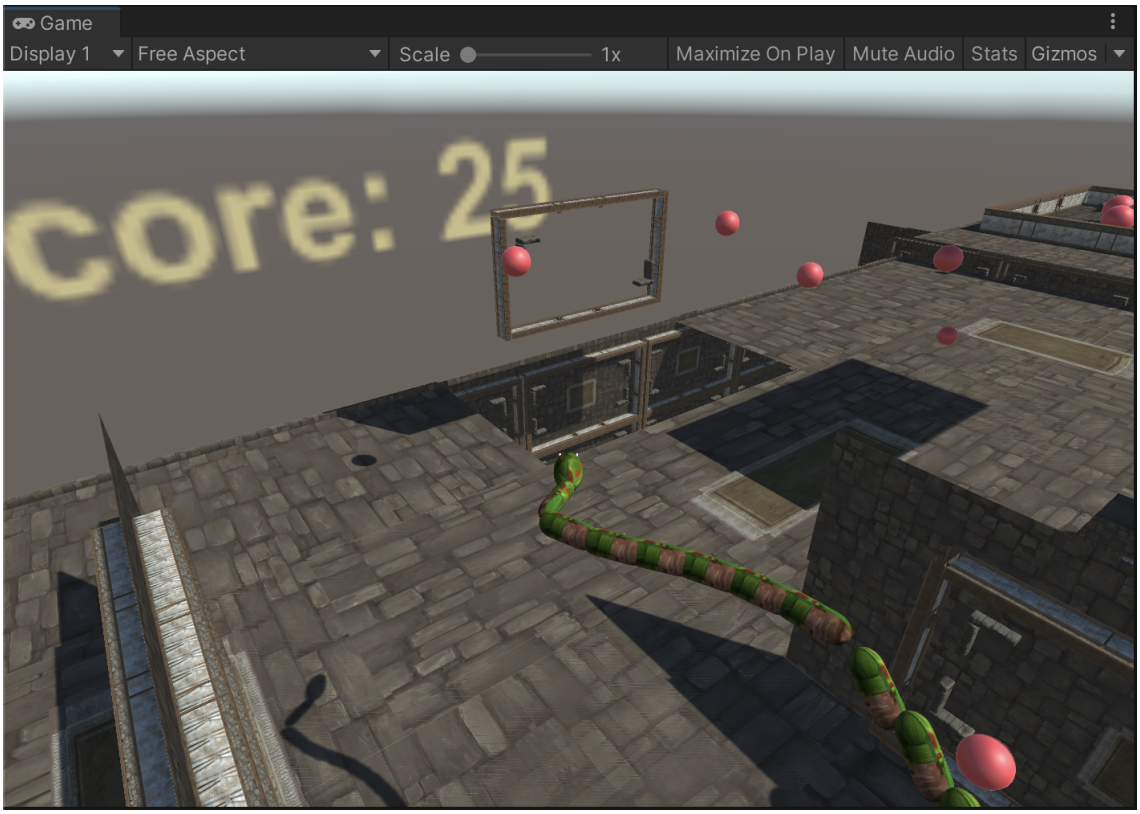
We created an environment file from some objects that were easy to make and edit with Unity built-in objects (inside a file called model). This step didn't involve any C# scripts because these objects do not have any interactions with players. So, we just edited some Unity built-in objects and connected them to build the environment.



The scene we finally built includes two layers, the first layer is enclosed space and the second layer is open space. Besides, there is a scoreboard on the second layer, recording how many scores the player has got so far:

Scene examples on the first floor

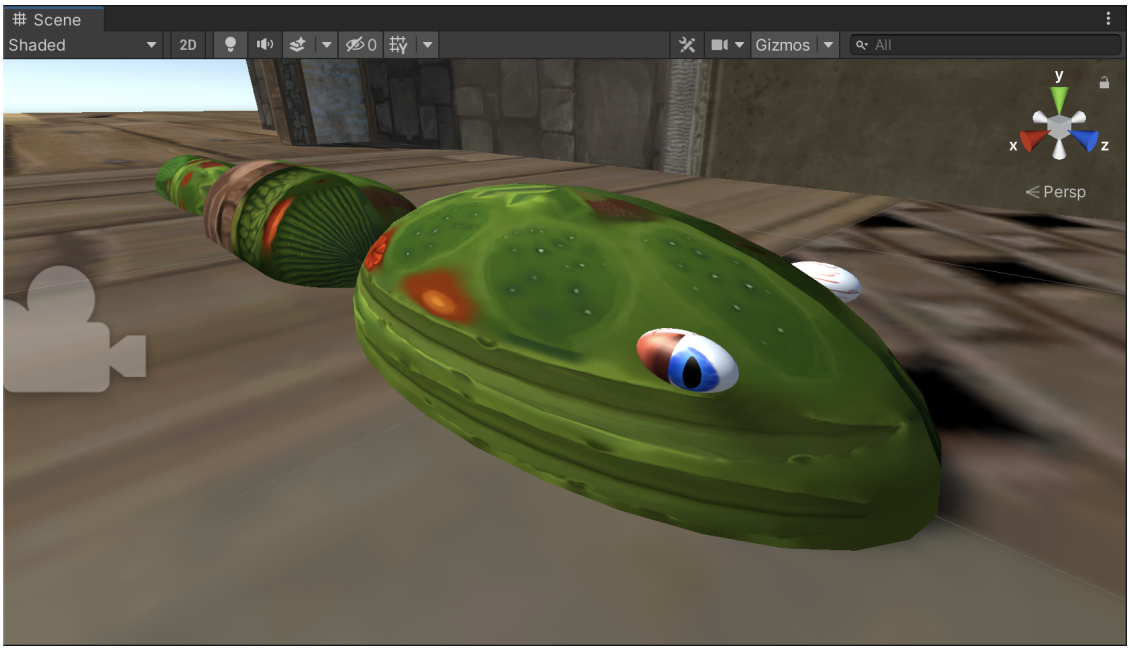
Scene examples of exits between the first and second floors

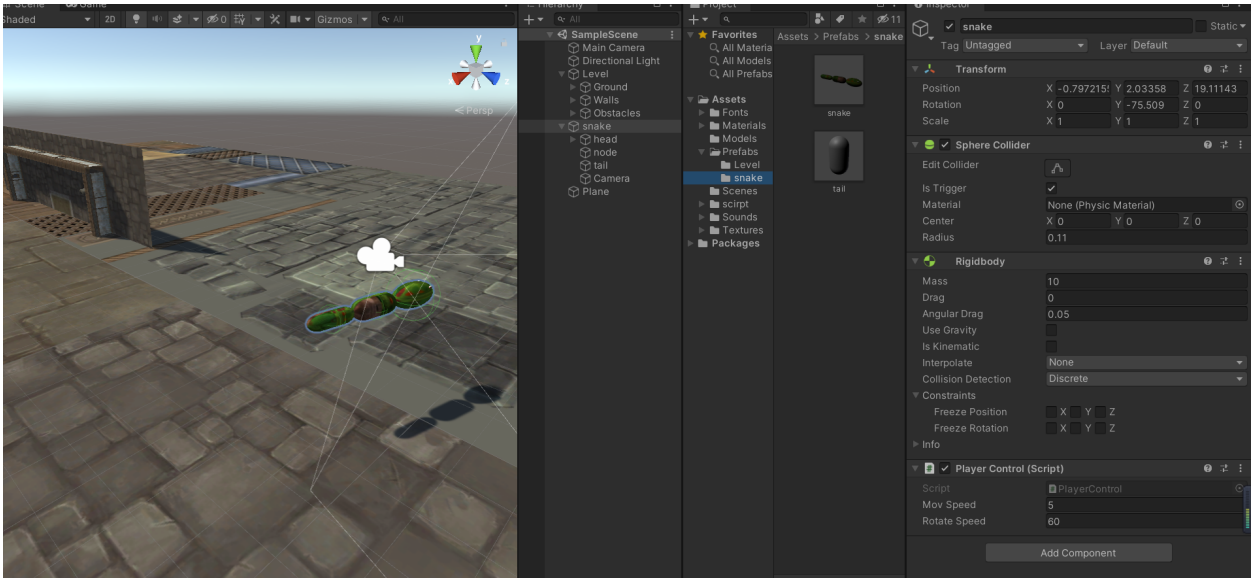


Scene examples on the second floor

Snake body design:
We designed the snake. And we chose to use nodes to make the snake. The nodes were just white balls initially, but we finally made it more like a real snake. The method we used to build the snake was, first of all, creating a white ball for the snake's head. Then we added another white ball called "node" next to the head. This operation would be iterated further to extend the snake's length. And finally, we added a white ball called "Tail." And for each node, we added the components named 'Rigidbody' and 'Sphere Collider.'

After that, we refined the model and scenes to make the game more realistic. We created a Prefabs containing two parts: one includes the snake's head and tail, which is also the initial state of the snake, and another is the node, which is used to represent the body part of the snake. We also added the green texture to the nodes to make the snake more realistic.





Snake control:

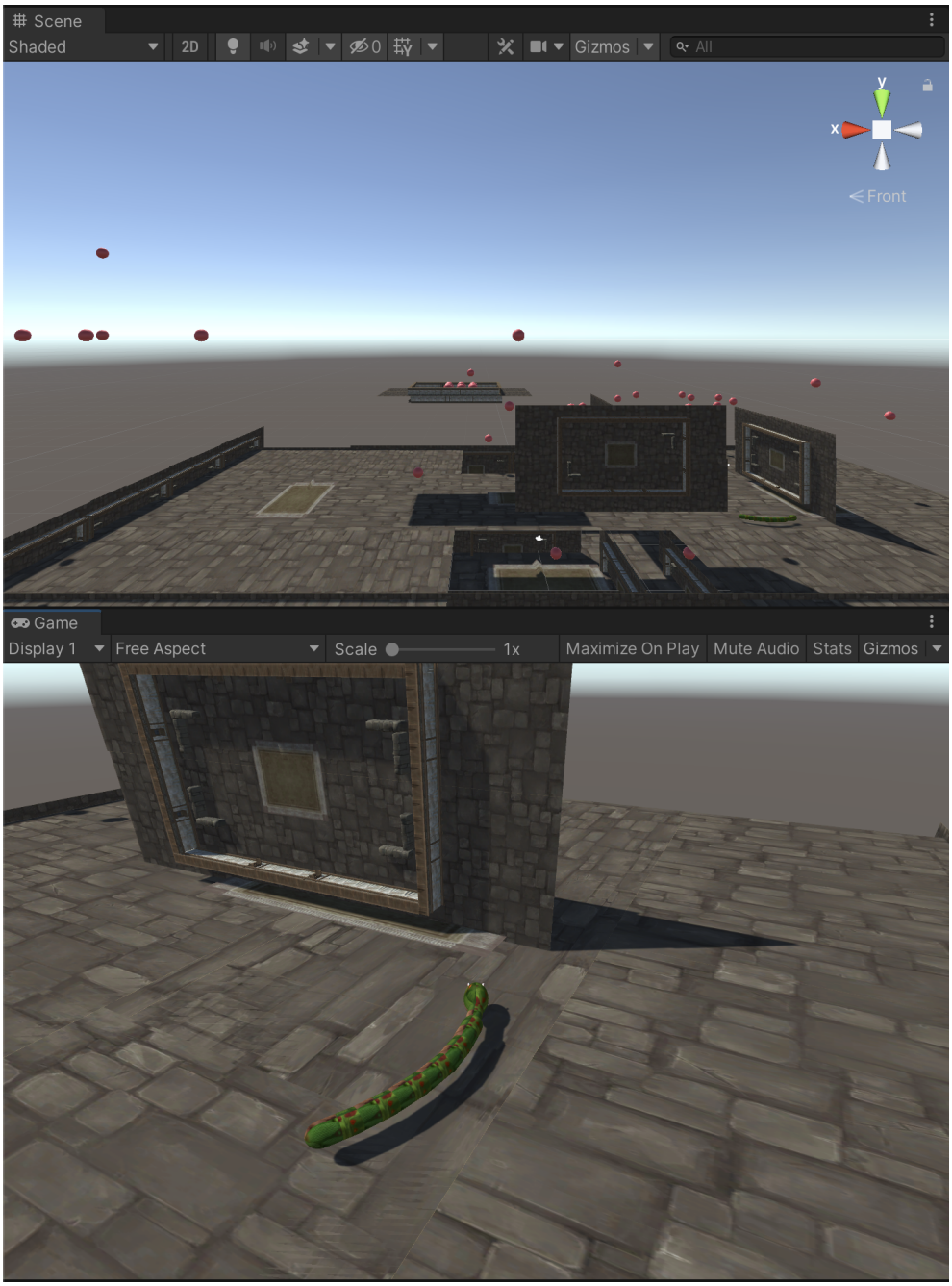
The “W”, “A”, “S” and “D” keys control the horizontal movement direction, and use the “SPACE” key to speed up the snake. Besides, the "O" and "P" keys control the vertical movement of the snake. The following link is a gif to simply show the operation: https://drive.google.com/file/d/1oHSmIDOL24ZnhXxsdJck9uq9ctWHd0i0/view?usp=sharing

We also tried to control the snake’s motion with the handles of the Oculus. For example, we tried to use the right thumbstick to control the horizontal movement. And use the left

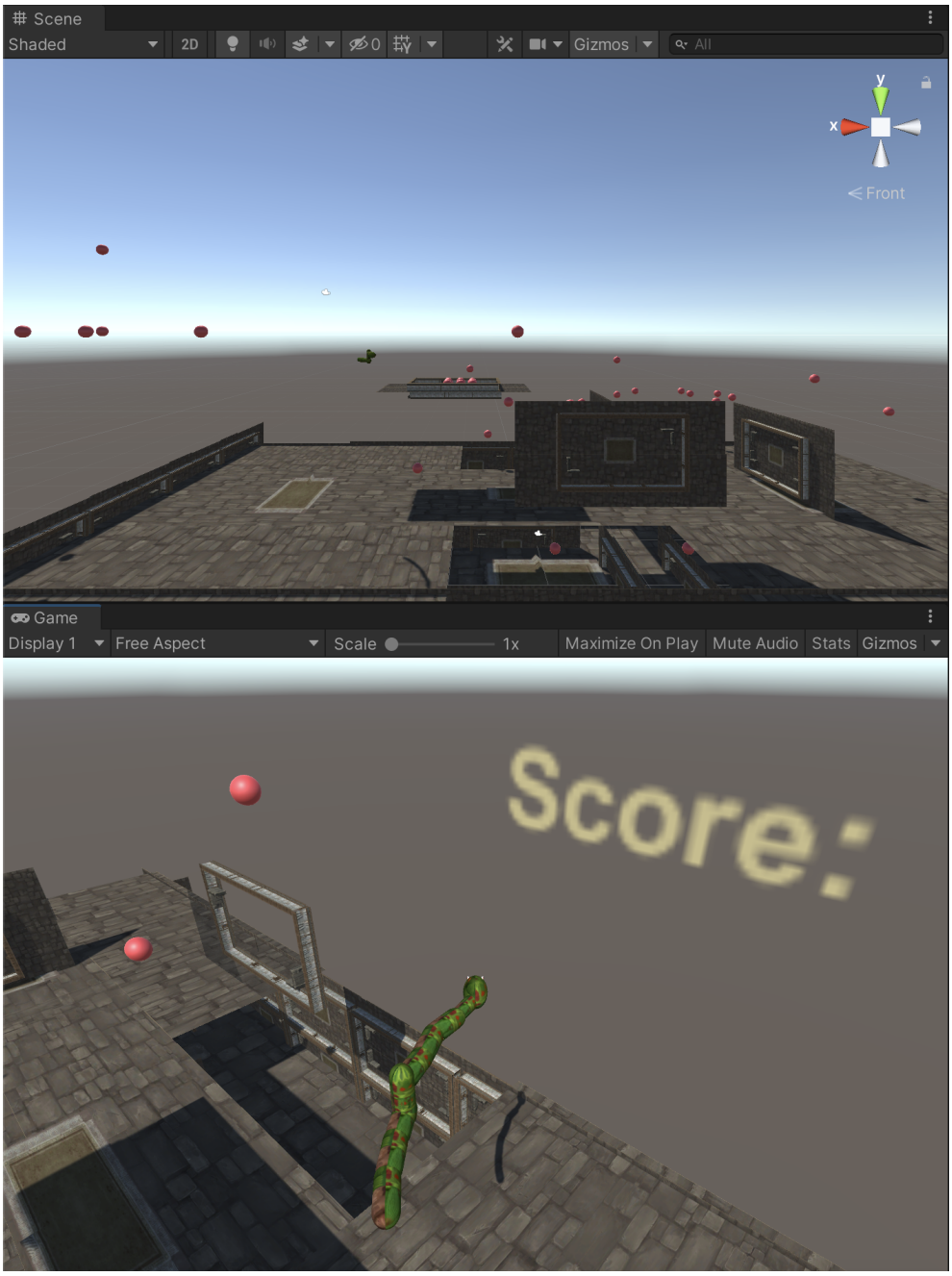
thumbstick to control the vertical movement and the angle of view. But due to some problems with device configuration, we didn't implement that function in the end.

First-person viewpoint:

Instead of a normal third-person perspective, we added another first-person perspective to the game. In this mode, the view's perspective would keep changing with the movement of the snake's head. As the figure shown below:



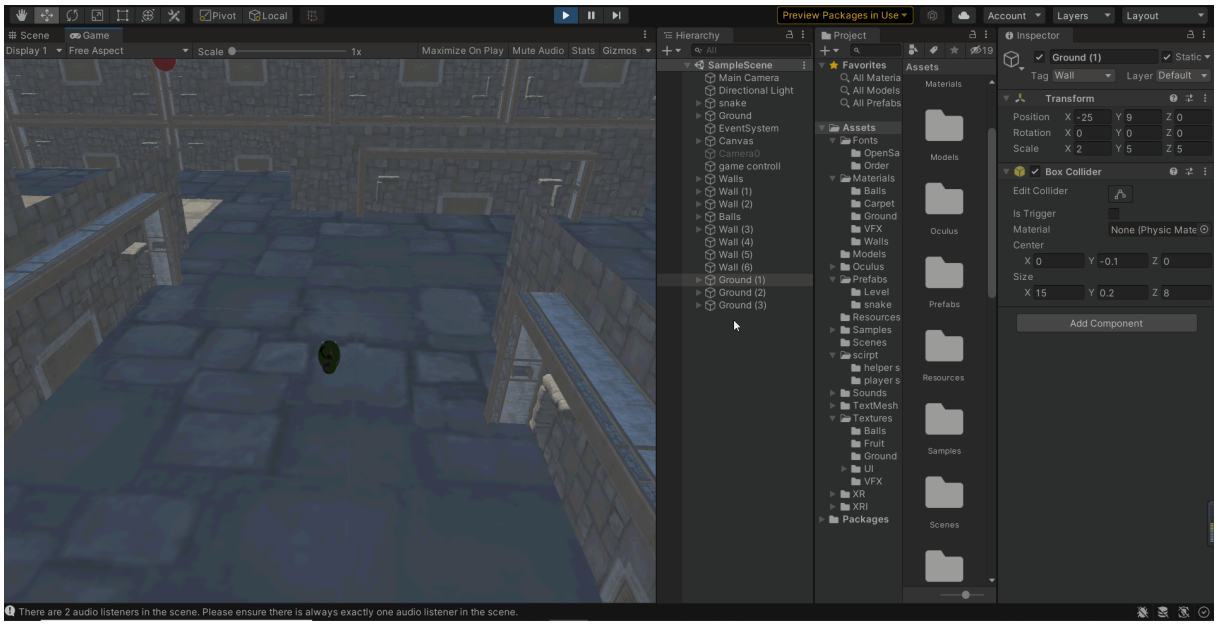
Horizontal motion



Vertical motion

Collision detection:

When the snake bumped into any wall, it couldn't cross the border and would end the game. The strategy we used is that we first set a collision trigger to detect if the current motion causes the collision. Once the collision process is triggered, the position and rotation will not be updated and will go back to the previous state.




```
 if(other.tag=="Wall")
 {
  m_transform.GetChild(0).Rotate(Vector3.up, -movement_ang,Space.Self);
  m_transform.GetChild(0).Translate(-2 * movement_dis,Space.Self);
  //m_transform.GetChild(0).position = FollowTransform_po[0];
  //m_transform.GetChild(0).eulerAngles = FollowTransform_ro[0];
  hitwall = 1;
  //m_movSpeed = 0;
  //m_rotateSpeed = 0;
 }
}

private void OnTriggerStay(Collider other)
{
 if(other.tag=="Wall")
 {
  //rd.AddForce(new Vector3(1, 0, 0) * force * 100 * Time.deltaTime);//对物体施加力
  //m_movSpeed = 0;
  //m_rotateSpeed = 0;
 }
}

private void OnTriggerExit(Collider other)
{
 if(other.tag=="Wall")
 {
  m_movSpeed = 5;
  m_rotateSpeed = 60;
  hitwall = 1;
 }
}
}
```

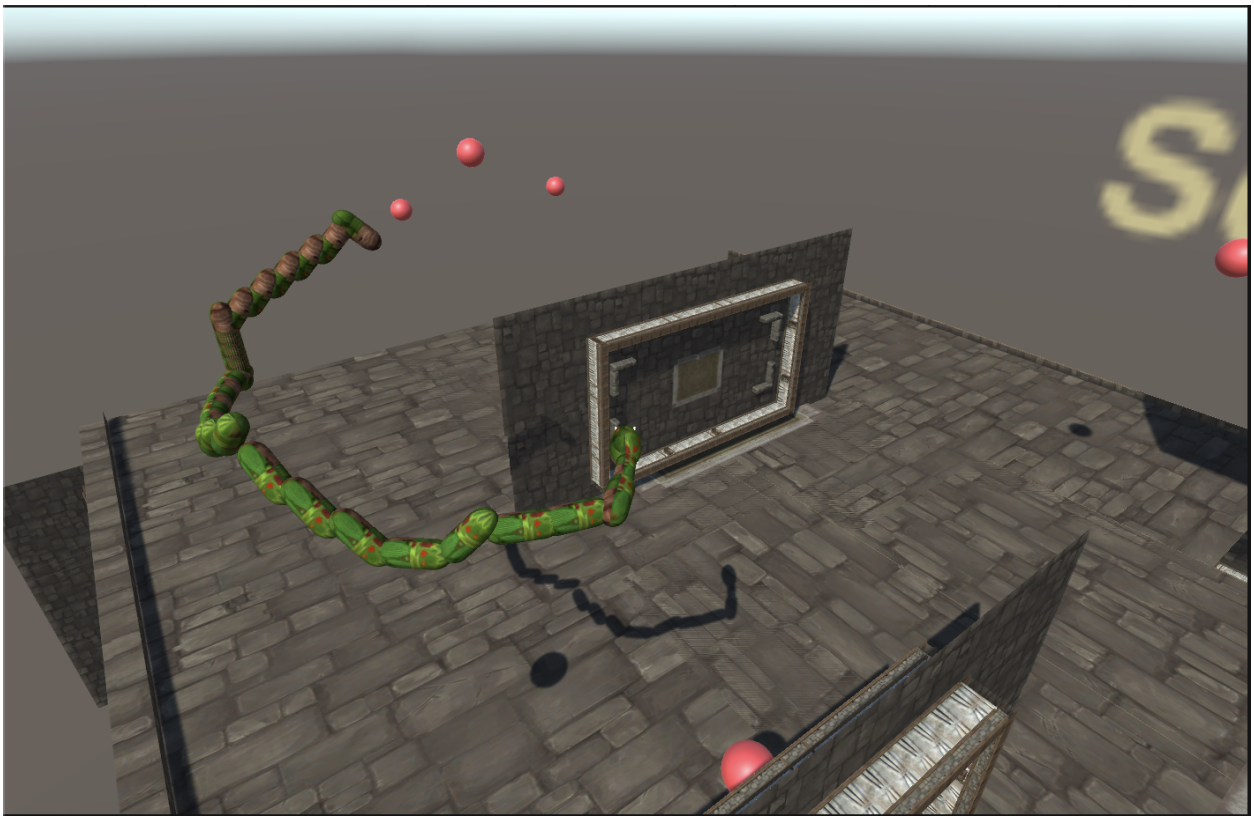
Snake-like motion:

To make the object operated by the player move like a snake, we used the following methods to implement that function:

First, we created two lists to record the motion trajectory of the head of the snake, one was used to record its position, and another was used to record its rotation. When the position and rotation of the head were updated at each frame, the program stored the latest data into the two lists and deleted the oldest data if the list was full. The remaining parts of the body simply followed the motion data recorded in lists. Besides, we also considered the angle change between the body part and the Z-axis when the snake moved in the vertical direction. If the head was moving up, then when the body node moved to that current place, its front end would be higher than the rear. Similarly, if the head was moving down, the front end of the node would be lower than its rear when it reached this position.

The program fragment is shown below:

```
private void OnTriggerEnter(Collider other)
{
 if(other.tag=="Ball")
 {
  gamecontroller.instance.IncreaseScore();
  Destroy(other.gameObject);
  //GameObject pfb = Resources.Load("Assets/Prefabs/snake") as GameObject;
  //GameObject prefabInstance = Instantiate(pfb);
  hit = 1;
 }
 if(other.tag=="Wall")
 {
  m_transform.GetChild(0).Rotate(Vector3.up, -movement_ang,Space.Self);
  m_transform.GetChild(0).Translate(-2 * movement_dis,Space.Self);
  //m_transform.GetChild(0).position = FollowTransform_po[0];
  //m_transform.GetChild(0).eulerAngles = FollowTransform_ro[0];
  hitwall = 1;
  //m_movSpeed = 0;
  //m_rotateSpeed = 0;
 }
}
```

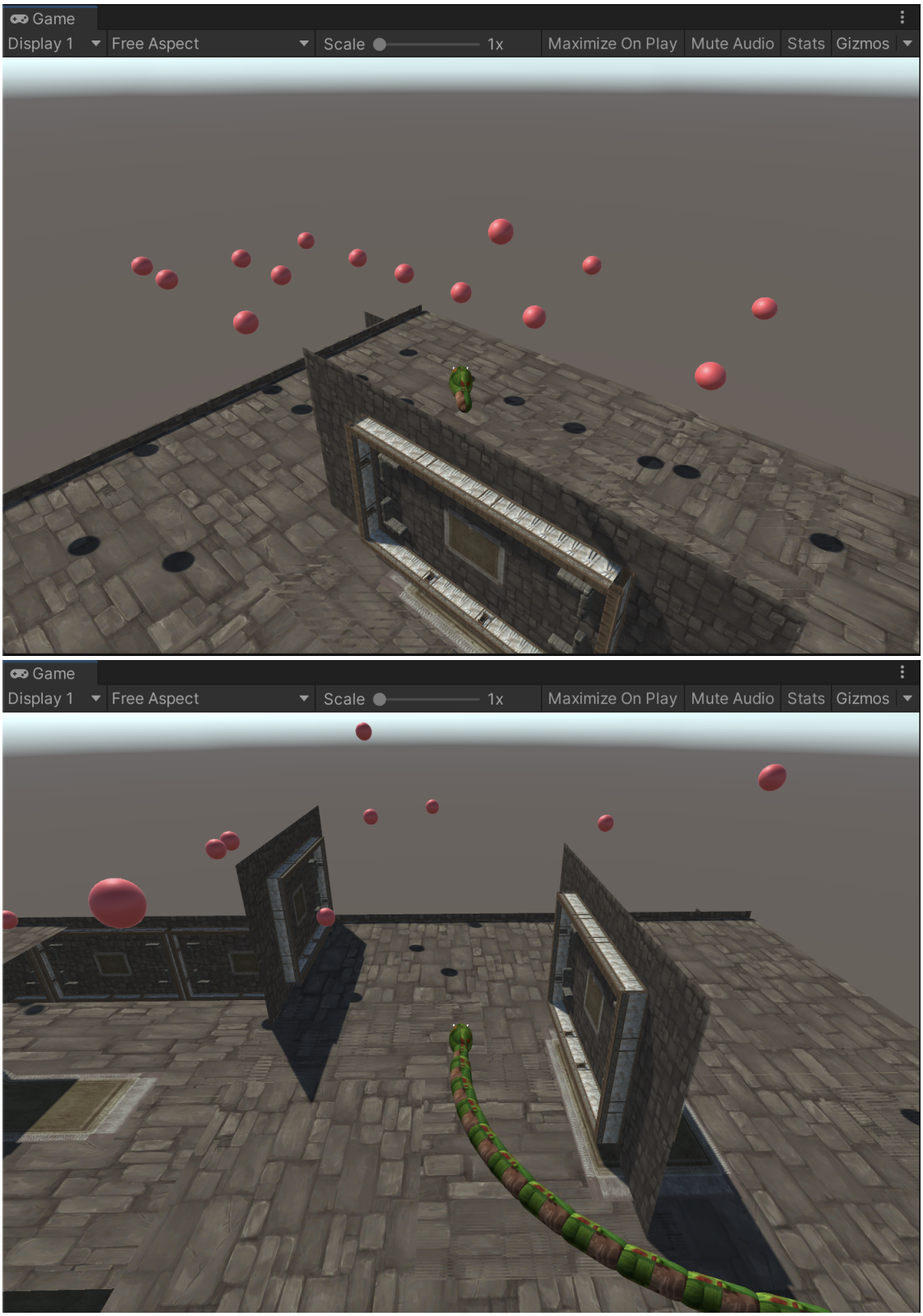


Example of snake-like motion

The growth of the body:

The method we used to realize this function is that when the snake's head hits one apple, the program will add a node stored in the Prefabs to the body. The program fragment is shown below:

```
if(other.tag=="Ball")
{
 gamecontroller.instance.IncreaseScore();
 Destroy(other.gameObject);
 //GameObject pfb = Resources.Load("Assets/Prefabs/snake") as GameObject;
 //GameObject prefabInstance = Instantiate(pfb);
 hit = 1;
}
```



Example of the snake in initial state and snake after body growth

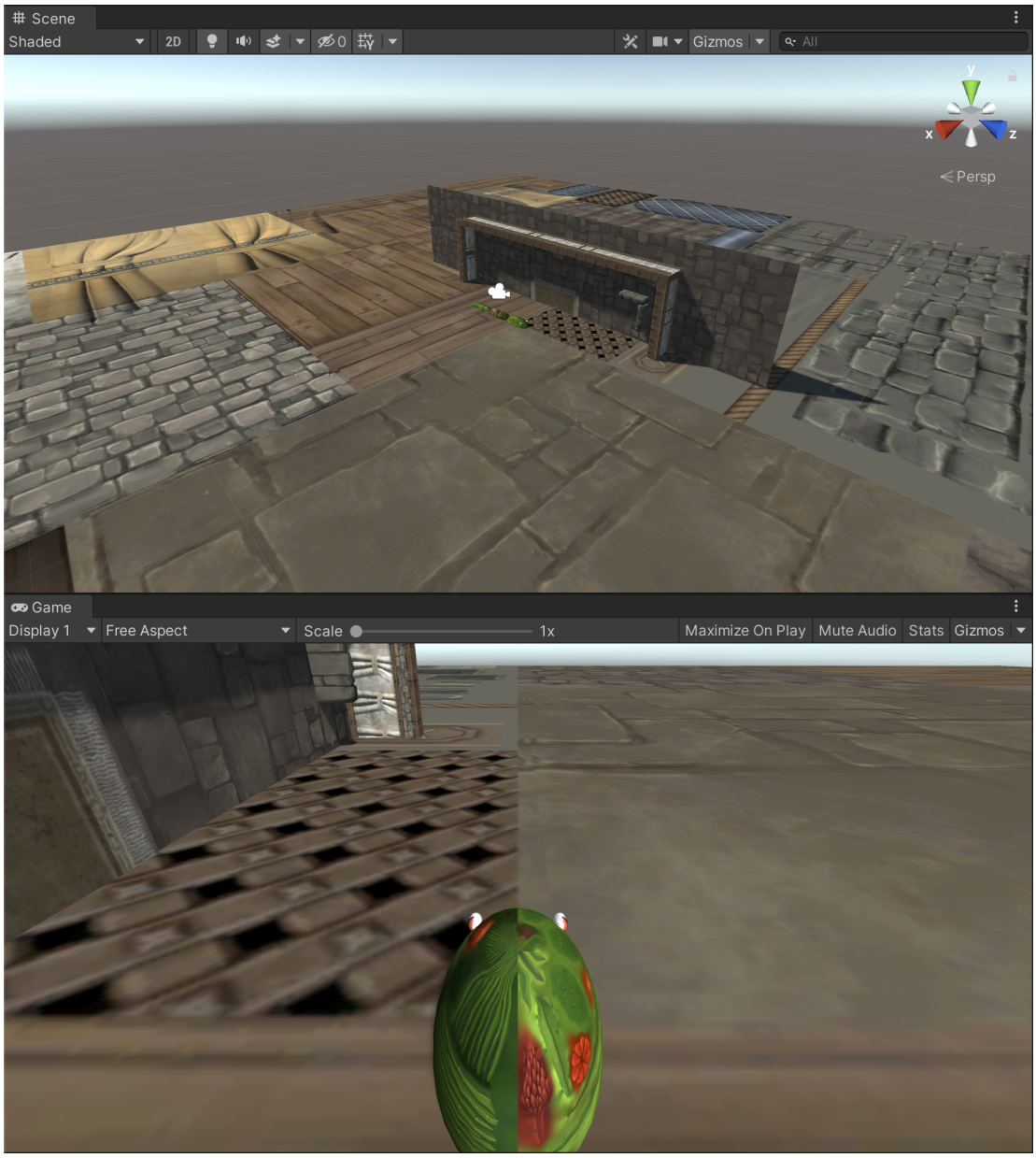
Scoreboard design:

Separate two components: UI text and C# script. The UI text shows the score for the user. The C# script applies the score increase function to the text.

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

public class gamecontroller : MonoBehaviour
{
    public static gamecontroller instance;
    private Text score_Text;
    private int scoreCount;
    // Start is called before the first frame update
    void Awake()
    {
        MakeInstance();
    }
    void Start()
    {
        score_Text = GameObject.Find("Score").GetComponent<Text>();
    }
    void MakeInstance()
    {
        if (instance == null)
        {
            instance = this;
        }
    }

    public void IncreaseScore()
    {
        scoreCount++;
        score_Text.text = "Score: " + scoreCount;
    }

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

    }
}
```

• *Experiments.*

We successfully created the environment and the snake mode as mentioned in the method part.



We successfully controlled the movement of the snake. It’s achieved based on the C# script, part of code of control of snake's movement shown below:

```
 if (Input.GetKey(KeyCode.Space)) //press space to accrelate
 {
  m_movSpeed = 15;
 }
 if (Input.GetKeyUp(KeyCode.Space)) //relase space to normal speed
 {
 m_movSpeed = 5;
 }

 if (Input.GetKey(KeyCode.W)) //press w to move forword
 {
  zm += m_movSpeed * Time.deltaTime;
 }
 else if (Input.GetKey(KeyCode.S))//press s to move backword
 {
  zm -= m_movSpeed * Time.deltaTime;
 }

 if (Input.GetKey(KeyCode.A))//press a to move view to left
 {
  xm -= m_movSpeed * Time.deltaTime / 2;
  xa -= m_rotateSpeed * Time.deltaTime;
 }
 else if (Input.GetKey(KeyCode.D))// press d to move view to right
 {
  xm += m_movSpeed * Time.deltaTime / 2;
  xa = m_rotateSpeed * Time.deltaTime;
 }

 if (Input.GetKey(KeyCode.O))//press o move up
 {
  ym += m_movSpeed * Time.deltaTime;
 }
 else if (Input.GetKey(KeyCode.P))//press p move down
 {
  ym -= m_movSpeed * Time.deltaTime;
 }
 m_transform.Rotate(Vector3.up, xa);
 m_transform.Translate(new Vector3(xm,ym,zm),Space.Self);
}
}
```

The following code is to achieve snake body increment by eating fruit.

```
    if (hit == 1)
    {
        GameObject prefabInstance = Instantiate(prefab);
        prefabInstance.transform.parent = m_transform;
        m_transform.GetChild(m_transform.childCount - 2).position = m_transform.GetChild(m_transform.childCount - 1).position;
        m_transform.GetChild(m_transform.childCount - 2).eulerAngles = m_transform.GetChild(m_transform.childCount - 1).eulerAngles;
        hit = 0;
    }

    if (movement_dis != Vector3.zero && hitwall == 0)
    {
        for (int i = 0; i < 14999; i++)
        {
            if (FollowTransform_po[14998 - i] != Vector3.zero)
            {
                FollowTransform_po[14999 - i] = FollowTransform_po[14998 - i];
            }
            if (FollowTransform_ro[14998 - i] != Vector3.zero)
            {
                FollowTransform_ro[14999 - i] = FollowTransform_ro[14998 - i];
            }
        }
        FollowTransform_po[0] = m_transform.GetChild(0).position;
        FollowTransform_ro[0] = m_transform.GetChild(0).eulerAngles + v;
        if (movement_dis.y > 0)
        {
            FollowTransform_ro[0] = FollowTransform_ro[0] + v1;
            if (movement_dis.x == 0 && movement_dis.z == 0)
            {
                FollowTransform_ro[0] = FollowTransform_ro[0] + v1;
            }
        }
        else if (movement_dis.y < 0)
        {
            FollowTransform_ro[0] = FollowTransform_ro[0] + v2;
            if (movement_dis.x == 0 && movement_dis.z == 0)
            {
                FollowTransform_ro[0] = FollowTransform_ro[0] + v2;
            }
        }
        for (int j = 1; j < m_transform.childCount; j++)
        {
            if (FollowTransform_po[speed_count * j] != Vector3.zero)
            {
                m_transform.GetChild(j).position = FollowTransform_po[speed_count * j];
                m_transform.GetChild(j).eulerAngles = FollowTransform_ro[speed_count * j];
            }
        }
    }
    else if (hitwall == 1)
    {
        hitwall = 0;
    }
}
```

• *Conclusion*

The project was designed to learn and practice some knowledge of computer vision and virtual reality. We designed and completed the project to cover as many topics mentioned in the class as possible. For instance, camera viewpoint, rigid body design, rendering, pose tracking, integration with the user, etc. Most of those features are directly callable in Unity. Some of them need some C# script to achieve, such as body movement, score updating…

The main operation of this project is the snake body and its movement and tracking design. Some functions were mentioned in the experiment and method part. We set the main camera in the top of the snake's head, and it's also designed for the user's viewpoint when they wear the headset. The snake body includes three parts: head, tail, and middle nodes(which will be incremented after eating fruits). The snake's body is set as a rigid body to achieve collision detection.

Finally, we develop the integration between VR devices and our project. All tests are based on the oculus queue 2. User /Player can wear the headset and see how the snake moves. This is the part that is most difficult for us. We had to use a few packages from Unity and build an android version to run on a VR device.

Overall, the project we designed focuses on using any model or concept we learned from the class to apply to the project. We learned a lot from the project and enjoyed it very much.