实验报告

姓名：陈　　诺
学号：516030910199

目录

1. 目的与要求

1.1 实验目的

1. 熟悉游戏中所应具备的基本要素以及游戏开发的基本流程
2. 了解游戏引擎的模型系统、渲染系统、物理系统和脚本系统
3. 学会使用游戏引擎开发游戏的各项基础技术

1.2 实验要求

1. 游戏主菜单：提供能够让用户开始游戏和退出游戏的操作界面；
2. 至少一个游戏关卡/场景：场景至少包含地形（树木）、天空盒和至少一个固定模型（例如房屋）
3. 至少一个游戏结束条件（胜利或者失败）
4. 至少一个外部载入的三维模型
5. 至少一个可由用户控制的物体或角色
6. 至少一个背景音乐或音效
7. 至少一个脚本实现的碰撞响应（例如当两个物体接触时触发一个条件）
8. 计时或计分系统，能在游戏结束时显示用户游戏时长或得分
9. 游戏过程中的二维用户界面，可以用来显示当前得分或者其他游戏状态
10. 形成**实验报告**（Word 或 PDF 格式）：需对以上每个要求的实现过程进行描述，并附对应截图

2. 实验步骤

2.1 游戏设计

游戏目的：在 30s 内用母球将其余 15 个子球撞入洞中，每个球的编号就进洞后能获得的分值。

游戏成功条件：在 30s 内获得 20 分。

游戏失败条件：30s 计时结束时还未获得 20 分，或者在 30s 内母球进洞 2 次。

2.2 模型与场景搭建

2.2.1 游戏场景

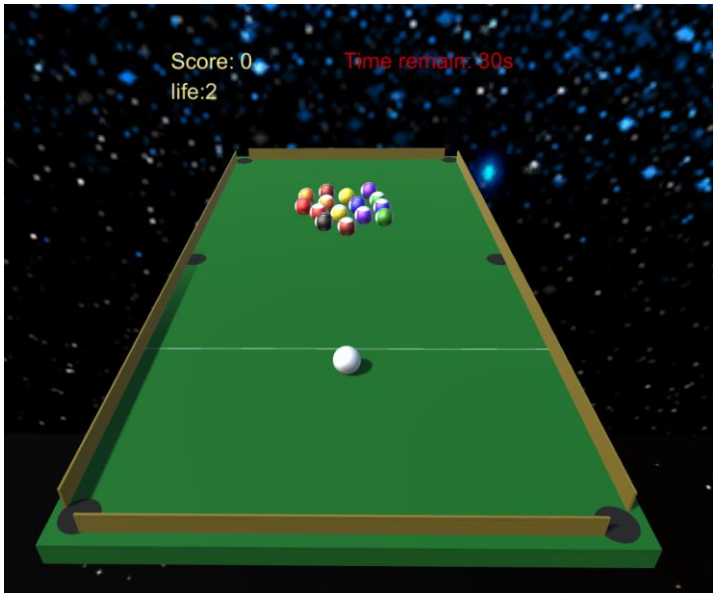

图 1. 基本游戏场景

游戏场景如图：由台球桌（地形）、16 颗台球和天空盒组成，在视线上方有通关条件（当前得分、剩余生命和剩余时间）。

其中：

1. Score 表示当前总得分；
2. Life 表示当前剩余生命；
3. Time remain 表示当前剩余时间。

用户通过鼠标移动及滚轮缩放来改变 Main Camera 的方向。

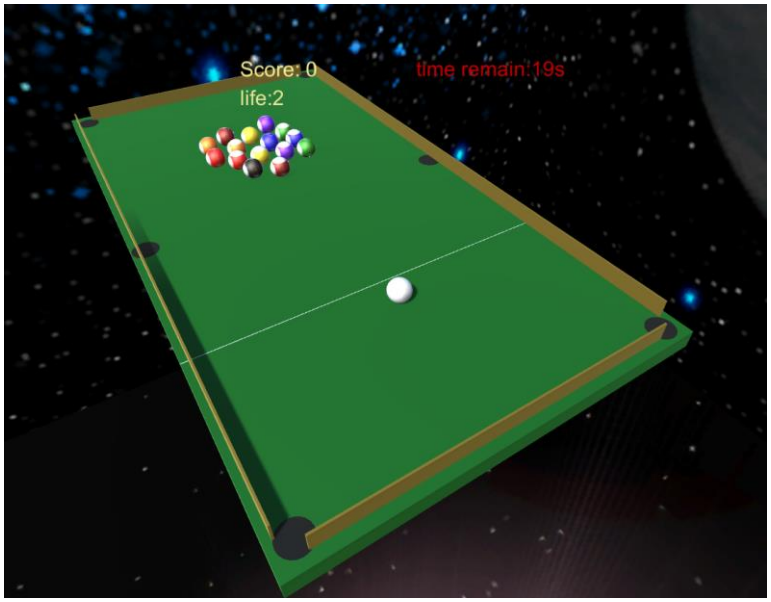

图 2. 旋转后的视角

用户通过点击鼠标左键来弹出白球。白球的速度大小和当前 Main Camera 与白球的距离正相关。白球碰到其他球的时候发出碰撞的声音。

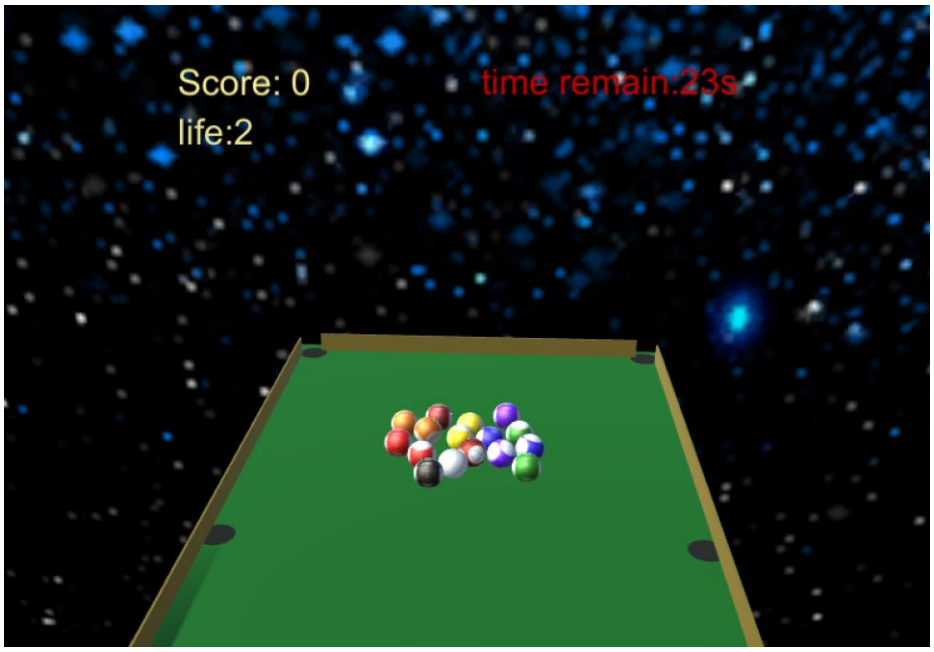

图 3. 碰撞触发声音。

本场景中，台球模型来自于 unity assets store，是外部导入的 3D 模型。

2.2.2 游戏主菜单

游戏主菜单是一个区别于游戏界面的 Scene，命名为 MainMenu。

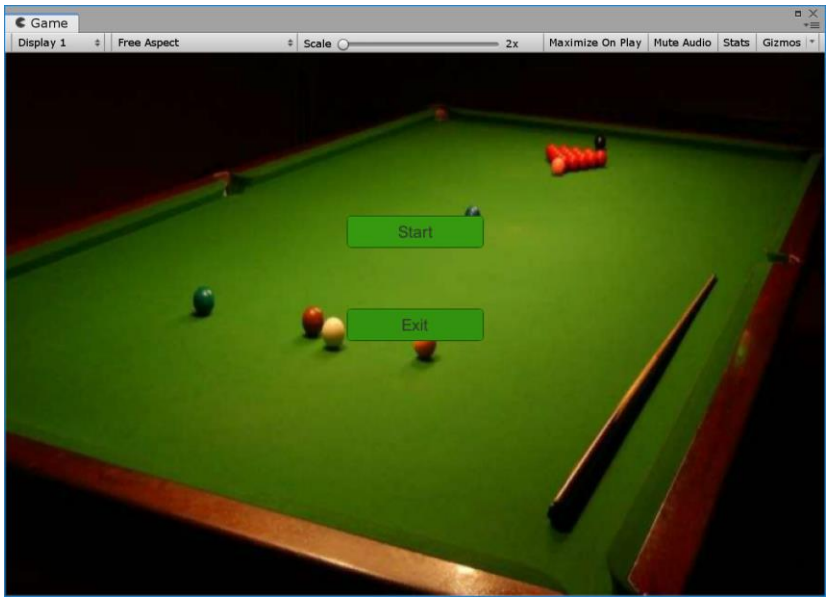

图 4. 游戏主菜单

Start 和 Exit 按钮分别会进入游戏界面和退出程序。

2.2.3 通关界面 & 失败界面

为了使结构更加清晰，所以通关和失败界面也各自设计了一个 Scene。

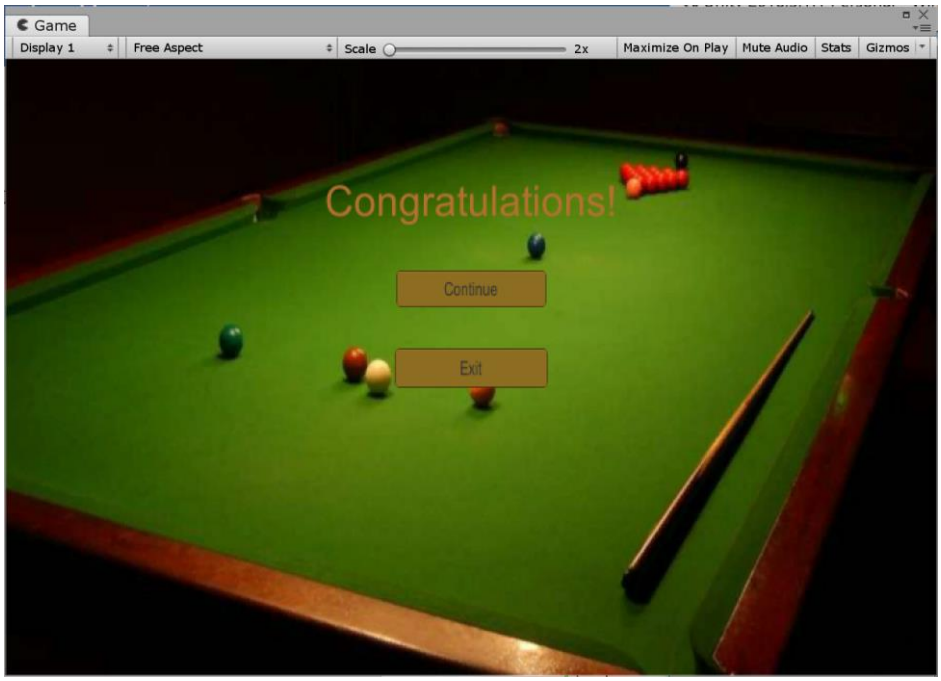

图 5. 通关界面

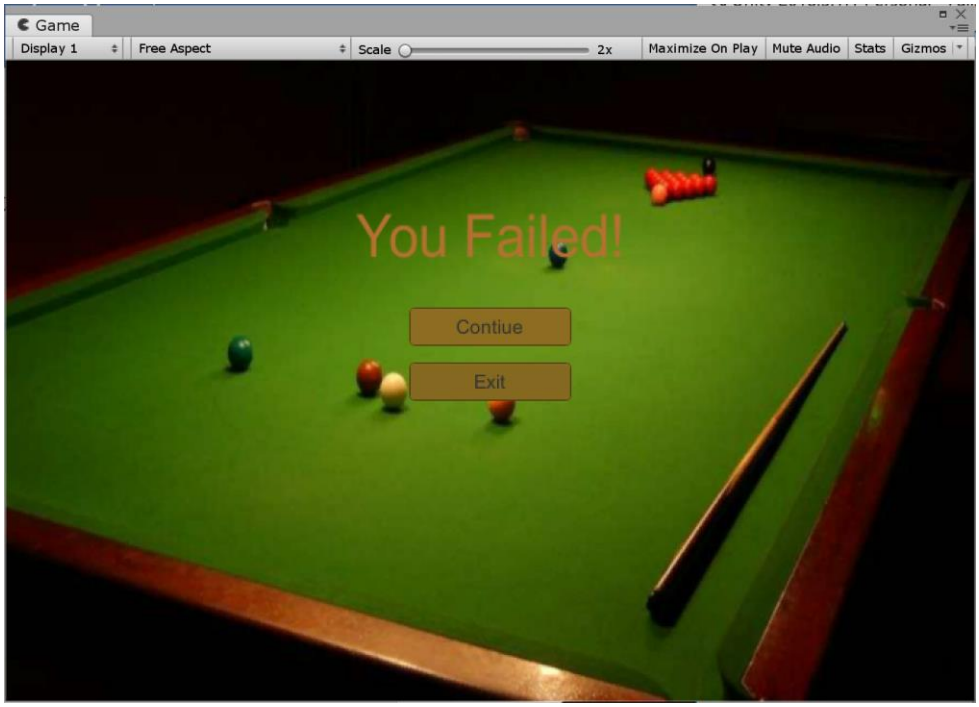

图 6. 失败界面

两个界面中的按钮：

Continue 会开始新的一局游戏；

Exit 会退出游戏。

2.3 项目架构与逻辑代码

2.3.1 项目架构

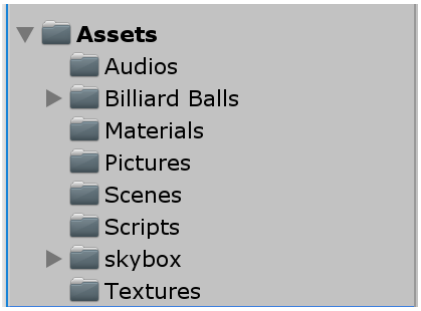

图 7. 项目架构

项目中将不同的资源文件分类放置在不同的文件夹下。

1. Audios: 音频文件；
2. Billiard Balls：外部导入的台球模型
3. Materials：材质文件
4. Pictures：背景图片
5. Scenes：场景文件
6. Scripts：c#代码文件
7. Skybox：天空盒贴图文件
8. Textures：贴图文件

其中 Scenes 构成为：

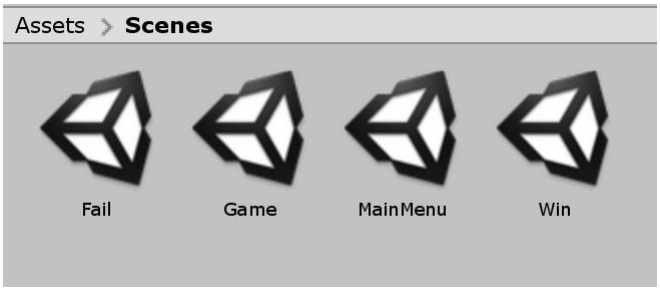

图 8. Scenes

2.3.2 逻辑代码

1. 游戏主菜单

游戏主菜单中使用 Empty Object（重命名为 EventController）来绑定主菜单中的逻辑。

在 EventController 绑定的 Script 文件 EventController.cs 中实现了 Scene 切换、退出游戏的功能代码：

```
1.  public void startGame()
2.  {
3.      Application.LoadLevel("Game");
4.  }
5.
6.
7.  public void exitGame()
8.  {
9.      Application.Quit();
10. }
```

2. 游戏场景

Main Camera 主要的逻辑代码：

```
1.  public GameObject sphere;  // Mother ball
2.  public float moveSpeed;    // controls the move speed of mouse
3.  public float sensitivetyMouseWheel = 10f;  // controls the move speed of mou
    se wheel
4.
5.  void Update()
6.  {
7.      // update remain time
```

```
    GameObject lifeObj = GameObject.Find("Canvas/TimeText");
    Text lifeText = (Text)lifeObj.GetComponent<Text>();
    lifeText.text = "time remain:" + (int)(30- Time.timeSinceLevelLoad) + "s
";
    // if Mother ball falls to ground, fail
    if (sphere.transform.position.y < -0.5)
    {
        Application.LoadLevel("Fail");
    }
    // move Main Camera according to mouse
    float mouseX = Input.GetAxis("Mouse X") * moveSpeed;
    float mouseY = Input.GetAxis("Mouse Y") * moveSpeed;
    if(transform.position.y>0 && transform.position.y<30)
    {
        transform.RotateAround(sphere.transform.position, new Vector3(0, 0,
1), -mouseY * 1.5f);
    }
    transform.RotateAround(sphere.transform.position, new Vector3(0, 1, 0),
mouseX);
    transform.LookAt(sphere.transform.position);

    // move Main Camera according to mouse wheel
    if (Input.GetAxis("Mouse ScrollWheel") != 0)
    {
        this.GetComponent<Camera>().fieldOfView = this.GetComponent<Camera>(
).fieldOfView - Input.GetAxis("Mouse ScrollWheel") * sensitivetyMouseWheel;


    }
}
```

母球的主要功能代码：

```
void Update()
{
    if (Input.GetMouseButtonUp(0))
    {
        // add a force to mother ball when user click the mouse
        rb.AddForce((rb.transform.position - new Vector3(mainCamera.transfor
m.position.x, 1.13f, mainCamera.transform.position.z)) * speed);
    }
}

private void OnCollisionEnter(Collision collision)
{
    if(collision.gameObject.name.Substring(0, 4) == "Ball")
```

```
    {
        // play the audio when mother ball hits other ball(s)
        audio.Play();
    }
}
```

球洞的主要功能代码

```
private void OnTriggerEnter(Collider other)
{
    // if mother ball falls to hole, minus `life` and put it back to the ori
gin position
    if(other.gameObject.name == "Ball_00")
    {
        other.gameObject.transform.position = new Vector3(11.14f, 1.13f, 0);

        // update life
        life--;
        GameObject lifeObj = GameObject.Find("Canvas/LifeText");
        Text lifeText = (Text)lifeObj.GetComponent<Text>();
        lifeText.text = "life:" + life;
        if (life == 0)
        {
            Application.LoadLevel("Fail");
        }
    }
    else
    {
        // add score
        string num = other.gameObject.name.Substring(5, 2);
        int val = int.Parse(num);
        score += val;
        GameObject scoreObj = GameObject.Find("Canvas/ScoreText");
        Text scoreText = (Text)scoreObj.GetComponent<Text>();
        scoreText.text = "score: " + score;
        // if score is bigger than 20, let user win
        if(score >= 20)
        {
            GameObject winObj = GameObject.Find("Canvas/WinText");
            Text winText = (Text)winObj.GetComponent<Text>();
            winText.text = "Congratulations!";
            Application.LoadLevel("Win");
        }
        // other ball should disappear after falls to holes
        Destroy(other.gameObject);
```

```
    }
}
```

3. 通关场景 & 失败场景

采用和游戏主菜单相同的方式为 button 绑定函数。
两个场景的功能函数相同。

```
public void continueGame()
{
    Application.LoadLevel("Game");
}

public void exitGame()
{
    Application.Quit();
}
```

3. 实验结果

3.1 对游戏开发的认知:

1. 基本熟悉了游戏开发流程；
2. 了解了基本的渲染系统、物理系统和脚本系统
3. 尝试了游戏引擎开发游戏的基础技术

3.2 游戏对于功能需求的覆盖度:

1. 游戏提供了主菜单
2. 游戏提供了 4 个场景
3. 游戏拥有结束条件和胜利条件
4. 游戏有外部导入的三维模型
5. 游戏有一个可以由用户控制的物体
6. 游戏拥有碰撞音效
7. 游戏有多个脚本实现了碰撞效应
8. 游戏拥有计时系统和计分系统
9. 游戏有二维界面来显示用户当前得分