成绩:

2019年秋季学期

《游戏引擎与虚拟现实》实验报告

姓名： 王智慧

学号： 2017166106

专业： 数 字 媒 体 技 术

班级： 20171661

目录

[目录 2](#_Toc13352)

[Airplane实验报告 2](#_Toc14836)

[1.1 大作业内容简介 2](#_Toc21607)

[1.2 游戏场景 3](#_Toc20921)

[1.2.1a 游戏背景的创建 3](#_Toc17196)

[1.2.1b UV动画实现背景滚动 4](#_Toc1098)

[1.2.1c 改变观察位置 5](#_Toc19325)

[1.2.2a 键盘控制飞机移动 5](#_Toc13674)

[1.2.2b 子弹的发射 6](#_Toc23426)

[1.2.3a 侧边子弹的发射 7](#_Toc5710)

[1.2.3b 敌机的创建 9](#_Toc5277)

[1.2.4a 碰撞的实现 9](#_Toc11210)

[1.2.4b 敌机的自动销毁 11](#_Toc5475)

[1.2.5a 高级敌人的创建 11](#_Toc23272)

[1.2.5b 高级敌人子弹的创建 12](#_Toc5212)

[1.2.6a 高级敌人子弹设置发射间隔 13](#_Toc22316)

[1.2.6b 玩家子弹声音和爆炸的实现 14](#_Toc22953)

[1.2.6b 敌机爆炸的实现 15](#_Toc26647)

[1.2.7a 敌机生成器 15](#_Toc22283)

[1.2.7b Scene面板中可视化敌机生成器空对象 16](#_Toc26536)

[1.2.7c UI的初步使用 16](#_Toc22742)

[1.2.8a 调整屏幕分辨率 17](#_Toc10725)

[1.2.8b 添加游戏结束canvas 18](#_Toc24541)

[1.2.8c 添加游戏管理器 18](#_Toc15799)

[1.2.9a 实现分数和生命值的动态更新 19](#_Toc29528)

[1.2.9b 开始场景 20](#_Toc14747)

[1.2.9c 场景跳转 22](#_Toc29517)

[1.2.10a 鼠标控制player移动 22](#_Toc6909)

# hello

# Airplane实验报告

### 1.1 大作业内容简介

游戏分为两个场景，开始场景和游戏场景。点击开始场景的开始游戏按钮会切换到游戏场景。在游戏场景中，有一个星空背景。游戏主角是一架飞机，可以通过键盘鼠标控制移动（前进、后退、旋转）。点击鼠标左右键会发射炮弹，炮弹可以打中敌机和敌机子弹，销毁敌机后界面会增加分数。如果被敌机或敌机子弹打中会减少生命值，生命值为0结束游戏，出现游戏结束画面。在游戏结束画面中点击重新开始就会重新回到游戏。

### 1.2 游戏场景

首先需要准备一个游戏的场景。场景中的美术部分实验项目资源文件已经提供。

步骤01：创建一个新的项目，导入AirplaneResources文件夹，在这个文件夹中已经提供了本游戏所需要的模型、动画和音效资源，导入后AirplaneResources目录如图1所示。

步骤02：打开本项目准备好的SimpleScene场景（在Project面板的Assets文件夹下的Scenes目录下）。打开后如图 1所示。

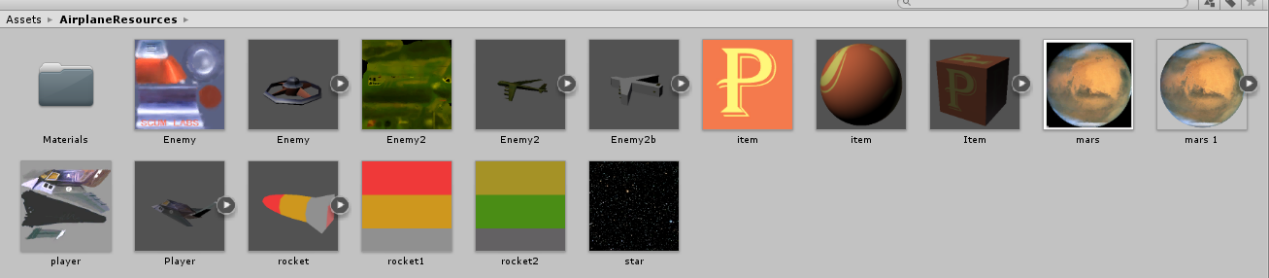


图 1 AirplaneResources目录

### 1.2.1a 游戏背景的创建

步骤01：创建材质。

在Asset目录下创建Materials目录, 进入该目录创建两个Material, 分别命名为backgound和mars。

进入mars的Inspector面板，点击Albedo左侧的小方框，进入图片选择界面，选择mars.png图片，设置好后如图 2所示。

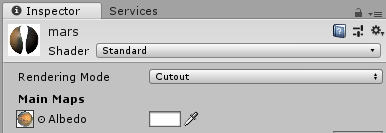


图 2 mars材质的Inspector面板

同养的方式给background材质设置Albedo图片star.png。

步骤 02：背景板的创建。

创建两个plane, 分别命名为Mars,Background，Transform属性如图 3，图 4所示。

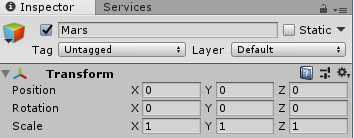


图 3 Mars的Transform属性

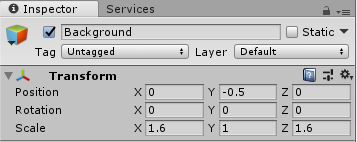


图 4 Background的Transform属性

步骤 03：给Mars、Background对象设置材质。

将mars材质托给Mars, 将background材质拖给Background。你会发现这时mars虽然是个球形但是plane中还是显示了白色边框，而且Background背景颜色也略暗。

步骤 04：修改材质参数。

首先修改mars。在mars材质的Inspector面板中将Rendering Mode设置为Cutout,如图 5所示。这时Mars对象的白色边框不见了，只显示了一个圆形的图案。

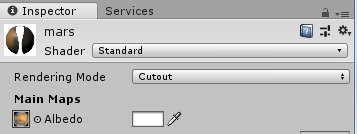


图 5 mars的Inspector面板

然后修改background。在background材质的Inspector面板中，将Shader设置为Unlit->Texture。

这样Background所显示的图案就变回了材质本身的颜色。

设置好后效果如图 6所示。

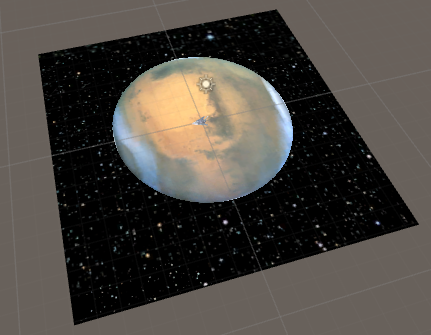


图 6 背景效果图

### 1.2.1b UV动画实现背景滚动

步骤 01：Asset目录下创建一个Animator controller（动画控制器）, 命名为bg。把bg动画控制器拖给Background对象。

步骤 02：添加所要key帧的属性

选中Background对象。

菜单->Window->animation, 打开动画窗口。

首先点击右侧的Create按钮, 此时弹出一个窗口。

输入名称bgMove.anim后点确定，这样就创建了一个动画。

动画窗口中左侧点击Add property, 展开Mesh Render, 选择Meterial.\_\_Main :Tex\_ST, 点击右边的“+”按钮。完成后如图 7所示

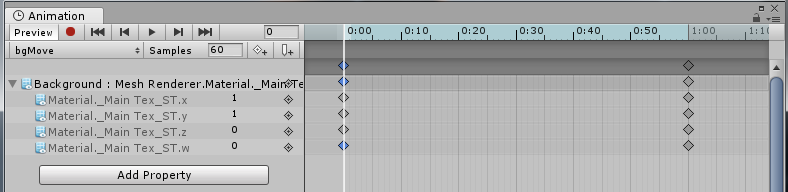


图 7 动画窗口

步骤 03：动画录制。

点击蓝色小菱形可以选中关键帧，在将0秒和1秒处的w值分别设置为0和1。

在动画窗口左上角红圆圈开始录制。

点击三角形播放按钮开始播放动画，播放完后关闭录制。

此时运行游戏可以看到Background的贴图是会循环向后滚动的。

### 1.2.1c 改变观察位置

步骤 01：调整摄像机让scene面板中的画面为俯视Background。

步骤 02：选中main camera, 菜单->gameObject->align with view。

步骤 03：修改 main camera中y轴的值调整摄像机远近。完成后如图 8所示。两个画面都为俯视图。

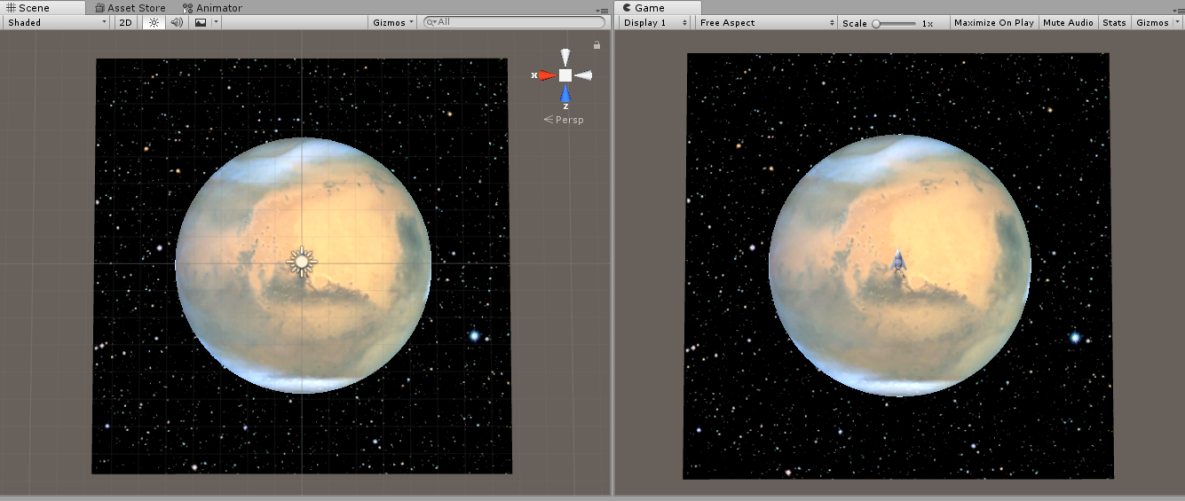


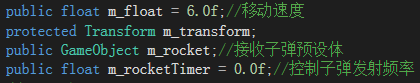
图 8 Scene面板和Game面板

### 1.2.2a 键盘控制飞机移动

步骤 01：在AirplaneResources目录下，将Player.fbx拖到hierarchy面板中。

步骤 02：创建Player.cs脚本

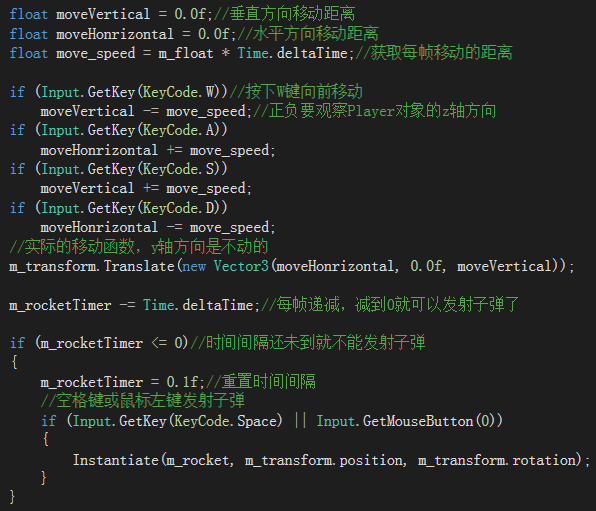
添加属性：



Start函数：



Update函数：



步骤 03：将Player.cs脚本挂载到Player对象上。

### 1.2.2b 子弹的发射

步骤 01：在Asset目录下，创建并进入Prefabs目录，创建一个预设体命名为Rocket。AirplaneResource目录下，将Rocket.fbx拖到hierarchy面板中，然后再将其拖大Rocket预设体中，然后删除hierarchy面板中的Rocket对象。

步骤 02：创建Rocket.cs脚本。

添加属性：



Start函数：



Update函数：



步骤 03：将Rocket.cs脚本挂载到Rocket预设体上。

步骤 04：在Player对象的Inspector面板中的Player（Script）组件中，将Rocket预设体添加到对应位置, 如图 9。



图 9 Player对象的的Player(Scirpt)组件

### 1.2.3a 侧边子弹的发射

#### 方法一：使用空对象确定子弹方向

步骤 01：首先新建两个空对象，分别命名为ShooterLeft,和ShooterRight, 并将它们挂载成Player对象的子对象，位置参数调整成和Player对象一样，并分别旋转45度和-45度，如图 10图 11图 12。



图 10 hierarchy面板

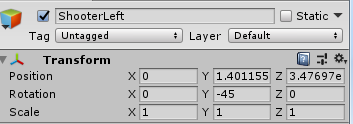


图 11 ShooterLeft的Inspector面板

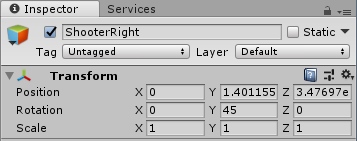


图 12 ShooterRight的Inspector面板

步骤 02：修改Player.cs脚本。

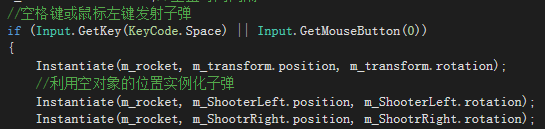
添加属性：



Start函数：



Update函数：

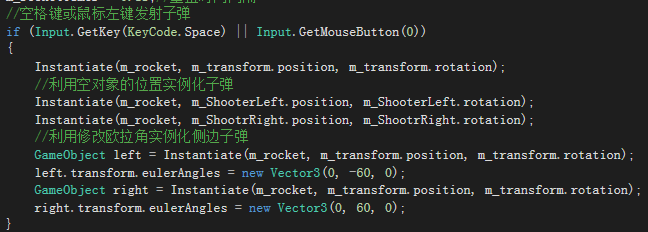


#### 方法二：使用脚本修改子弹方向

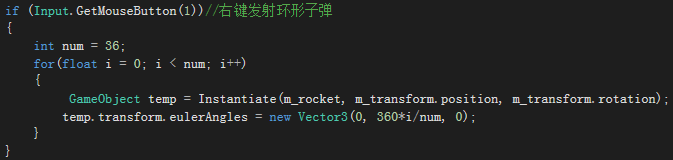
步骤 01：修改Player.cs脚本

Udate函数：

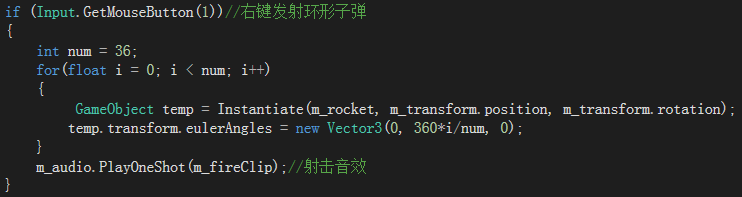
直接修改欧拉角来修改速度。



步骤 02：环形子弹



#### 额外功能：鼠标右键发射环形子弹

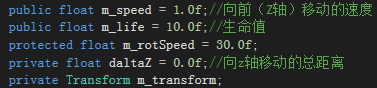


### 1.2.3b 敌机的创建

步骤 01：将airplaneResources目录下的Enemy.fbx拖入hierarchy面板。修改其位置使其一开始载游戏画面的顶部。

步骤 02：创建Enemy.cs脚本，并挂载到Enemy对象上

添加属性：

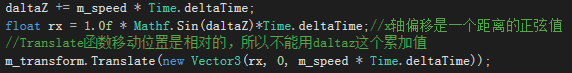


Start函数：



Update函数：控制敌机的移动。

**方法一：使用敌机移动的距离作为sin的自变量**



**方法二：使用时间作为sin的自变量**



步骤 03：在Prefabs目录下创建Enemy预设体，将Enemy对象拖到其中。删除Enemy对象。

步骤 04：修改Player对象和Enemy预设体的y轴位置为0.5，方便以后做碰撞检测。Player的Z轴位置修改为5.0。

### 1.2.4a 碰撞的实现

步骤 01：为主角飞机对象，敌机预设体，主角飞机子弹预设体都添加rigbody,组件，勾选is kinematic。

添加box collider组件，勾选is trigger，设置好后如图 13。

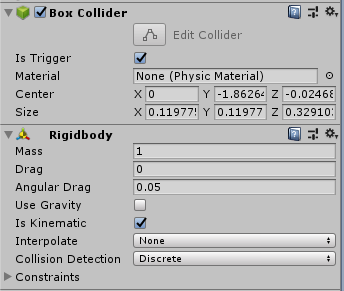
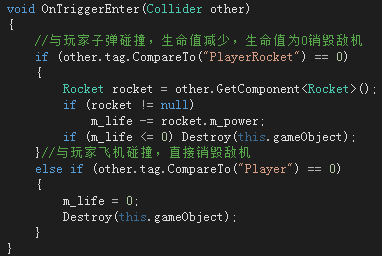


图 13 Inspetor面板中Rigbody和boxCollider组件的设置

步骤 02：为主角飞机对象，敌机预设体，主角飞机子弹预设体设置标签，分别为Player, Enemy, PlayerRocket。

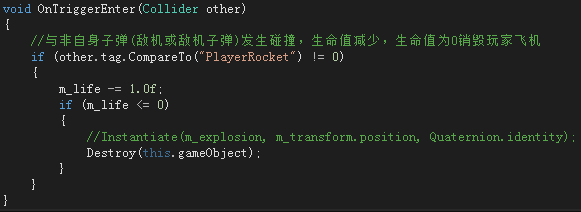
步骤 03：敌机的碰撞检测。Enemy.cs脚本。

OnTriggerEnter函数：



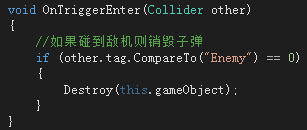
步骤 04：玩家飞机的碰撞检测。Player.cs脚本。

OnTriggerEnter函数：



步骤 05：子弹的碰撞检测。Rocket.cs脚本。

OnTriggerEnter函数：



### 1.2.4b 敌机的自动销毁

步骤 01：销毁敌机。当敌机走出屏幕外时就销毁敌机。

修改Enemy.cs脚本。

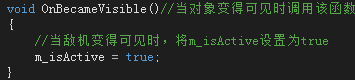
添加属性：



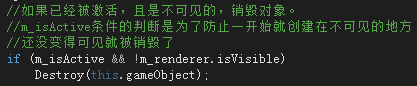
Start函数：



OnBecameVisible函数：



Update函数：



### 1.2.5a 高级敌人的创建

步骤 01：将AirplneResoues目录下的Enemy2拖到hierarchy面板，设置其位置参数和旋转参数，使其方向朝前，y轴和主角飞机一样。

步骤 02：创建一个预设体SuperEnemy,将enemy2对象拖到SuperEnemy预设体上，删除Enemy2对象，SuperEnemy的Inspector面板如图 14。

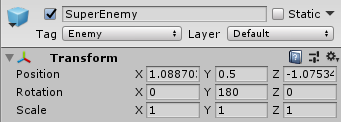
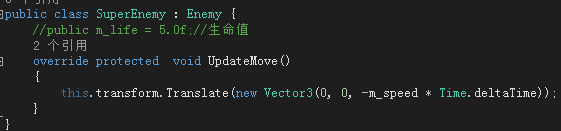


图 14 SuperEnemy对象的Inspector面板

步骤 02：创建SuperEnemy.cs脚本

继承Enemy类，重载其UpdateMove函数，使其走直线。



### 1.2.5b 高级敌人子弹的创建

步骤 01：用airpaneResources目录下的bullet.fbx创建预设体EnemyBullet。

步骤 02：修改子弹的材质使其与主角飞机的子弹颜色不同。Inspector面板->mesh render->materials->element0,进行设置。

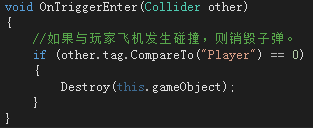
步骤 03：为EnemyBullet预设体添加rigbody,组件，勾选is kinematic。添加box collider组件，勾选is trigger。Tag设置为”Enemy”。

步骤 04：创建EnemyRocket.cs脚本。

继承于Rocket:



重载OnTriggerEnter函数：

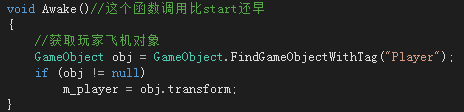


步骤 05：修改SuperEnemy.cs脚本。

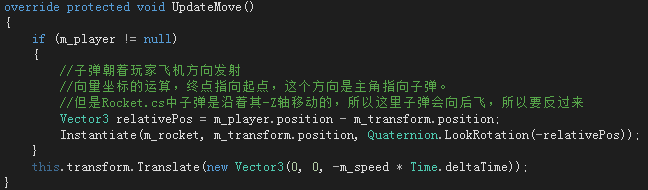
添加属性：



添加Awake函数：



修改UpdateMove函数：



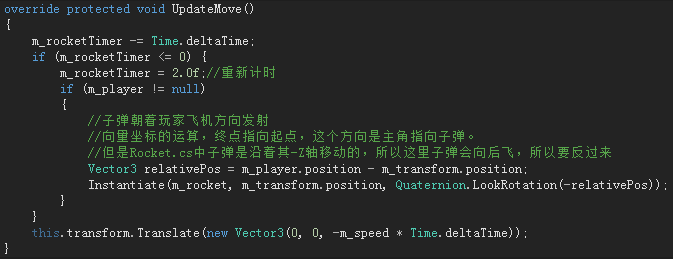
### 1.2.6a 高级敌人子弹设置发射间隔

步骤 01：修改SuperEnemy.cs脚本。

添加属性：



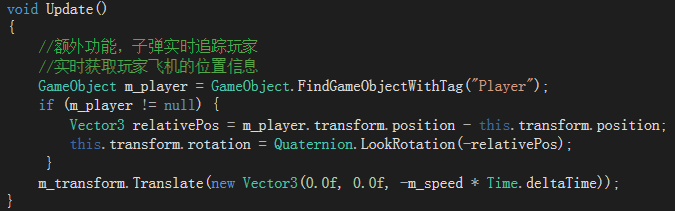
修改UpdateMove函数：



#### 额外功能1：敌机子弹实时自动追踪主角飞机

修改EnemyRocket.cs脚本。

重载Update函数：

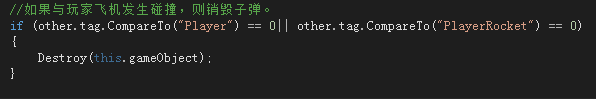


#### 额外功能2：玩家子弹与敌人子弹之间的碰撞检测

实现子弹销毁子弹

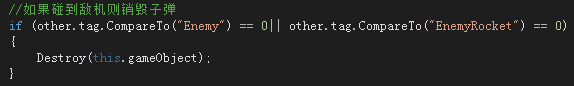
修改EnemyRocket.cs脚本。

OnTriggerEnter函数



修改Rocket.cs脚本。

OnTriggerEnter函数



### 1.2.6b 玩家子弹声音和爆炸的实现

步骤 01：在Asset目录下导入资源包ShootingFX.unitypage。

步骤 02：修改Player.cs脚本。

添加属性：



修改Start函数：两种添加生源组件的方式。

第一种是在Inpetcor面板中添加，然后获取该组件。

第二种是直接在脚本中添加组件。

这里采用第二种。



修改Udate函数：子弹实例化后播放音效。



修改OnTriggerEnter函数：玩家飞机销毁前实例化爆炸特效。



步骤 03：界面变量赋值。

点击Player对象，然后将FX目录下的shoot.wav, explosion.prefab拖给Player对象Inspector面板中Player.cs组件模块的对应变量。

**备注**：这里的exposion预设体是自带音效的，实例化时会播放音效一次。

### 1.2.6b 敌机爆炸的实现

步骤 01:修改Enemy.cs脚本。

添加属性：



修改OnTriggerEnter函数：在销毁敌机前实例化爆炸特效。



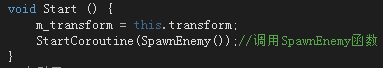
### 1.2.7a 敌机生成器

步骤 01：创建EnemyGenerator.cs脚本。

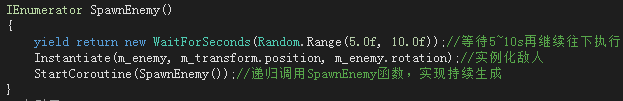
添加属性：



Start函数：



SpawnEnemy函数：



步骤 02：创建敌人生成器预设体。

创建一个空对象EnemyGenerator，将其y轴的值设置为0.5，再将EnemyGenerator.cs脚本挂载到EnemyGenerator对象上。

创建一个预设体EnemyGenerator，将EnemyGenerator对象拖到其上，然后删除EnemyGenerator对象。

将Enemy预设体拖到EnemyGenerator预设体的Inspector面板上EnemyGenerator.cs组件的对应位置，如图 15。

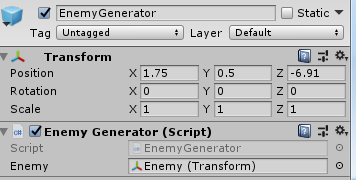


图 15 EnemyGenerator预设体的Inspector面板

步骤 03：创建高级敌人生成器预设体。

同步骤02，不同点是挂载到EnemyGenerator.cs组件的预设体是SuperEnemy预设体。

步骤 04：创建与主角飞行方向一致的敌人生成器预设体。

同步骤02，不同点是，先用SuperEnemy预设体的实例创建一个S2Enemy预设体，将y轴的旋转值调整180度。然后挂载到EnemyGenerator.cs组件的预设体是S2Enemy预设体。

步骤 04：将三种敌人生成器拖到hierarchy面板中想要的位置。

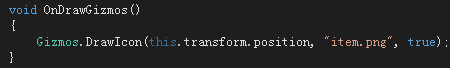
### 1.2.7b Scene面板中可视化敌机生成器空对象

步骤 01: 在Asset目录下，新建文件夹Gimos文件夹，在资源管理器这个将AirplaneResources目录下的item.png拷贝大Gmos目录下。

步骤 02：修改EnemyGenerator.cs脚本。

添加onDrawGizmos函数：

这里不能用m\_transform,因为它是在Start函数中初始化的，而Start函数是要在游戏运行开始时调用。



效果如图 16。

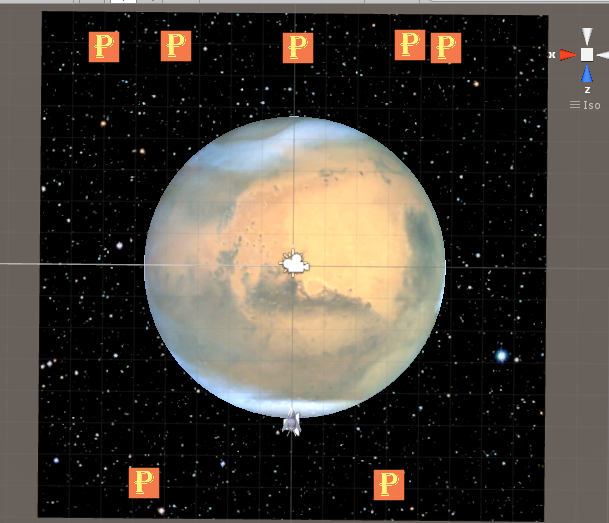


图 16 Gizmos效果图

### 1.2.7c UI的初步使用

步骤 01：hierarchy面板->右键->UI->Canvas创建画布，双击画布可以看到其在Scene面板中的位置。

步骤 02：选中hierarchy面板中的Canvas对象，右键->UI->Text，创建3个文本UI, 在其Inspector面板中可以修改文本的值，和文本颜色，字体大小等属性，如图 17。

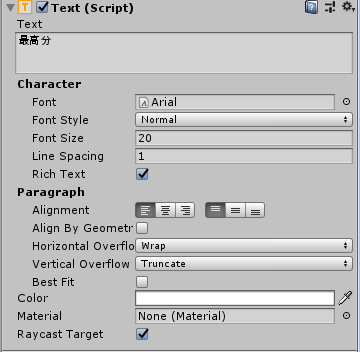


图 17 Inspector面板中的Text组件的

调整好位置后效果如图 18。

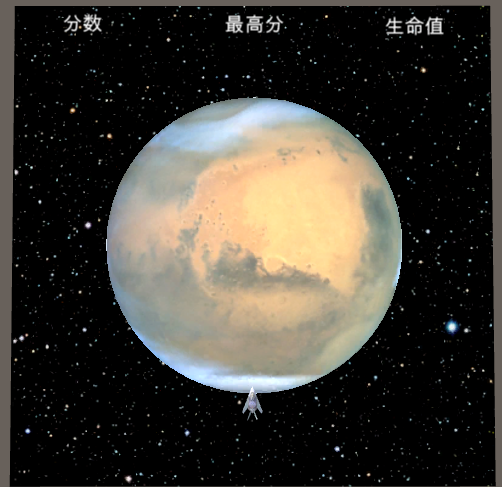


图 18 UI效果图

### 1.2.8a 调整屏幕分辨率

步骤 01：调整game面板的屏幕分辨率为800x800，如图 19。



图 19 game面板的分辨率

步骤 02：调整canvas的分辨率为800x800, 使UI与游戏画面匹配，如图 20。

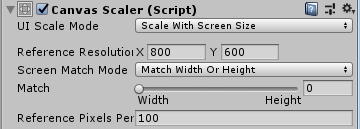


图 20 canvas的canvas scaler组件

### 1.2.8b 添加游戏结束canvas

步骤 01：在hierarchy面板添加canvas, 命名为Canvas\_gameover。

步骤 02：给Canvas\_gameover对象添加子节点Text:Text\_gameover，修改其文本为“游戏结束”。

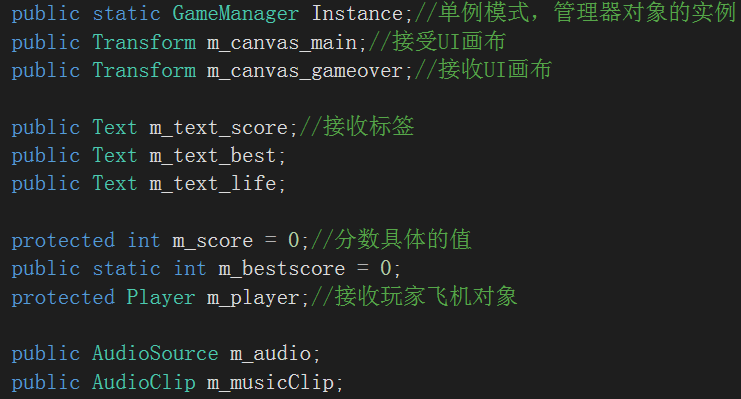
步骤 03：给Canvas\_gameover对象添加节点Button:Button, 修改其子节点Text的内容为“重新开始。”

### 1.2.8c 添加游戏管理器

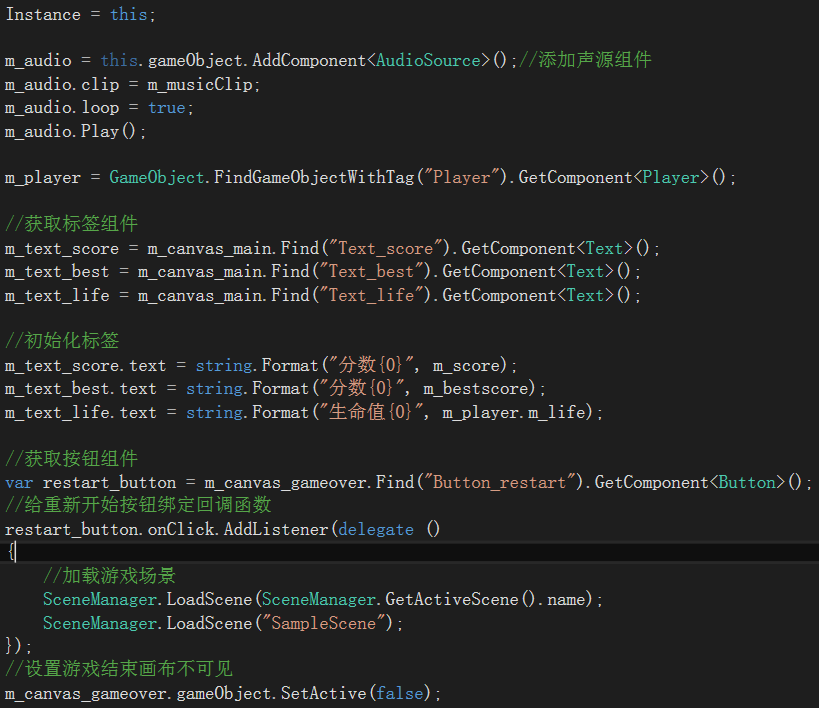
步骤 01：新建一个空对象，命名为GameManager。

步骤 02：创建GameManager.cs脚本。

添加属性：



Start函数：



步骤 03：将GameManager.cs脚本挂载大GameManager.cs对象上，并将画布、背景音乐拖动到其Inspector面板的GameManager.cs组件对应的位置，如图 21 GameManager对象的Inspector面板。

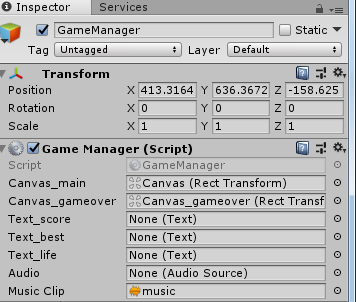
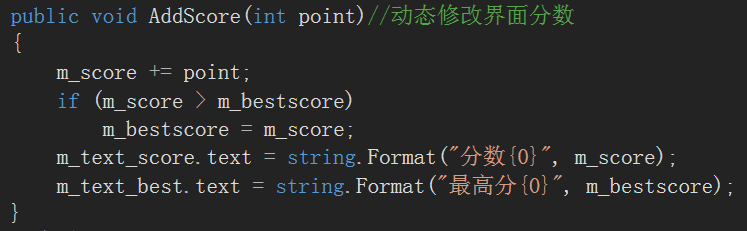


图 21 GameManager对象的Inspector面板

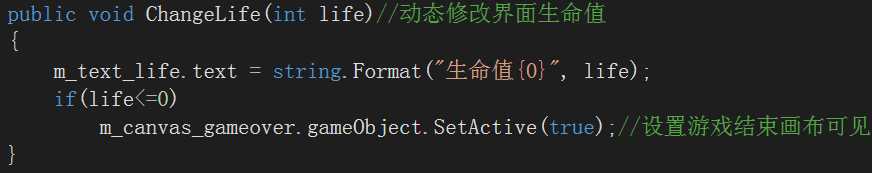
### 1.2.9a 实现分数和生命值的动态更新

步骤 01：修改GameManager.cs脚本。

添加AddScore函数：



添加ChangerLife函数：

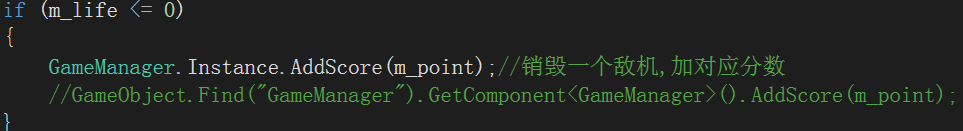


步骤 02：修改Enemy.cs脚本。

添加属性：



在OnTriggerEnter函数最后添加下面代码来调用AddScore函数：



步骤 03：修改player.cs脚本。

修改OnTriggerEnter函数，生命值减少后调用ChangerLife函数：



步骤 04：修改SuperEnemy.cs脚本。

添加属性，使用new操作符隐藏父类变量：



#### 遇到的问题：unity重新加载场景后光线变暗了。

解决：Windows->lighting->settings, 把Light设置下面的auto generator勾去掉，点击generate lighting烘焙光照，关闭。

### 1.2.9b 开始场景

步骤 01：菜单->file->new scene, 新建一个场景Start，保存场景到Scenes目录下。

步骤 02：创建画布，画布上创建一个Image,修改图片大小为界面大小800x800，如图 22。

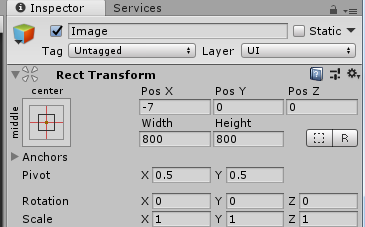


图 22 Image的inspector面板

步骤 03：为Image设置图片。进入AirplaneResources目录，点击mars.png图片，在inspector面板上将其Texture Type设置为Sprite(2D and UI), 如图 23。（图中的是已经设置好的mars 1.png, 以和mars.png做区分）

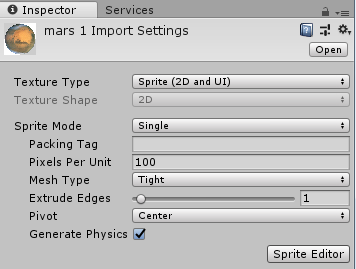


图 23 mars.png 的inspector面板

步骤 04：在画布上添加Text对象。

为其Text组件设置内容“星球大战”，字体设置为红色，大小48。

设置好字体大小后，需要设置Text对象的width和height的值，否则内容可能会不显示。

步骤 05：在画布上添加button按钮。

其子对象Text的内容设置为“开始游戏”，字体大小36。

同样可以设置buton的width和height。

以上UI对象都添加好后效果如图 24。

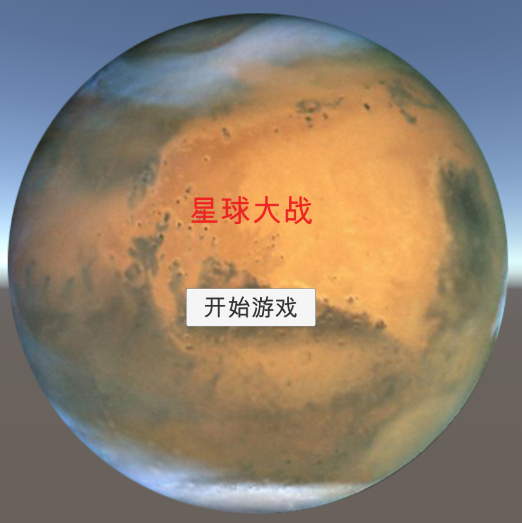
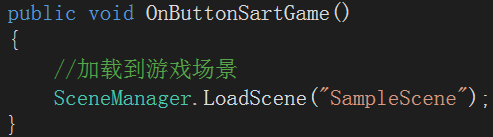


图 24 开始界面的效果图

### 1.2.9c 场景跳转

步骤 01：创建TitleScene.cs脚本。将其挂载到Main Camera上。

添加自定义OnButtonSartGame函数：



步骤 02：给button绑定OnButtonSartGame回调函数。

进入button的inspector面板，在On Click()的下面，将Main Camera对象拖动到小圆圈对应的框中，然后选择其TitleScene组件的OnButtonSartGame函数，完成后如图 25。

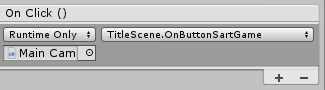


图 25 button的回调函数绑定

### 1.2.10a 鼠标控制player移动

步骤 01：SimpleScene中，在hierarchy面板上添加一个quad（gameObject->3DObject->Quad）, 将其Layer设置为Reference Layer(下拉后可新建一个再添加)，然后调整其位置大小，如图 26。

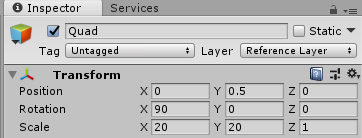


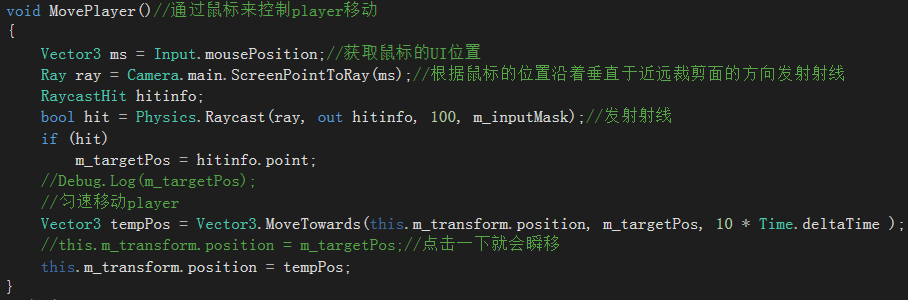
图 26 Quad的inspector面

步骤 02：修改Player.cs脚本。

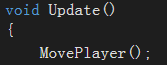
添加属性：



添加MovePlayer函数：



在Update函数中调用MovePlayer函数：



Start函数中设置鼠标不可见：



OnTriggerEnter函数中在Player死亡后设置鼠标可见：



步骤 03：在Player的Inspector面板中将Inputmask设置为Reference Layer，如图 26。

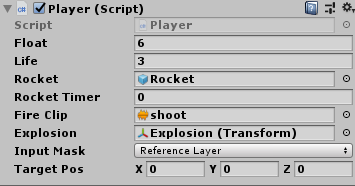


图 27 Player对象的Inspector面板中的脚本模块