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

# Tank实验报告

### 1.1 大作业内容简介

在游戏场景中，有若干排砖块。游戏主角是一辆坦克，可以自由移动（前进、后退、旋转），可以向任意方向发射炮弹。炮弹的轨迹是抛物线，并且带有爆炸特效。当子弹打中砖块时二者都会消失。

### 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：打开项目工程目录Tank\_Step1\_Practice，在这个工程中已经提供了本游戏所需要的模型、动画和音效资源。

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）。打开后场景如图1所示，层次结构图如图2所示。

图1 坦克模型 图2 对象层次结构图

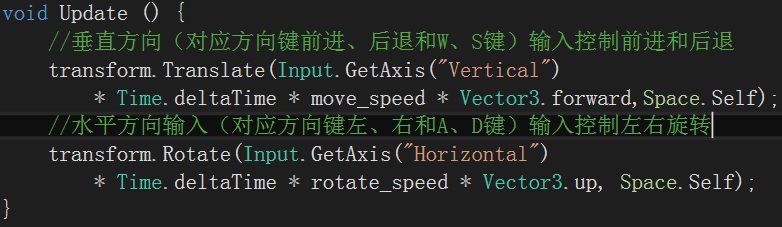
### 1.2.1a Tank的移动、旋转

接下来，我们将在脚本ControlWholeCar.cs中添加一些代码，该脚本用来控制Tank的移动和旋转。

步骤01：打开ControlWholeCar.cs，添加用于控制移动旋转的属性：



步骤 02：在Update函数中添加以下代码，用于接受键盘输入，利用输入信息来移动和旋转Tank。



步骤 03：将ControlWholeCar.cs脚本挂载到Tank（对应场景中的Car）上, 直接将脚本拖动到层次结构图中的Car对象上，结果如图3。

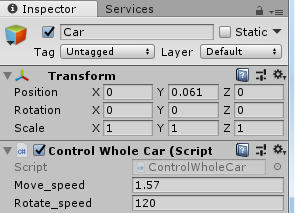


图3 Car的Inspector面板

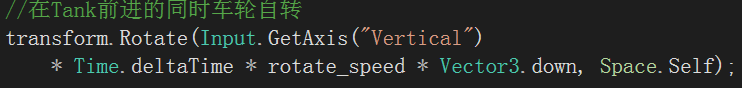
### 1.2.1b Tank车轮的自转

在RotateWheel.cs中添加一些代码，用来控制车轮的自转。

步骤 01：添加控制车轮旋转速度的属性



步骤 02：在Update函数中添加以下代码，接收垂直方向的键盘输入，用来控制车轮自转。



步骤 03：将RotateWheel.cs脚本挂载到四个车轮对象

（BackLeftWheel,BackRightWheel,FrontRightWheel,FrontLeftWheel）上，方法同1.2.1a步骤 03。

### 1.2.2a 子弹的生成

步骤 01：创建一个球体（菜单->GameObject->3D Object->Sphere），命名为Bullet。在球体的Inspector面板中设置缩放属性，如图4。

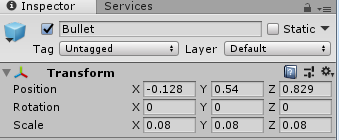
、

图4 Bullet的Inspector面板

步骤 02：为子弹设置材质。进入Materials目录，右键->Create->Material, 新建了一个材质球，进入该材质球的Inspector面板，点击Albedo右边的颜色块设置材质球的颜色，如图5。最后将材质球拖到Hierarchy面板中的Bullet对象上。子弹的颜色就被设置成了材质球的颜色。

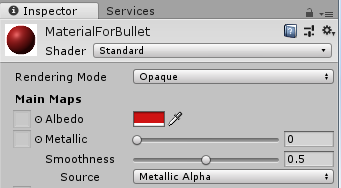


图5 材质球的Inspector面板

步骤 03：创建BulletFly.cs脚本，在Upate函数中添加以下代码，用于控制子弹的飞行。

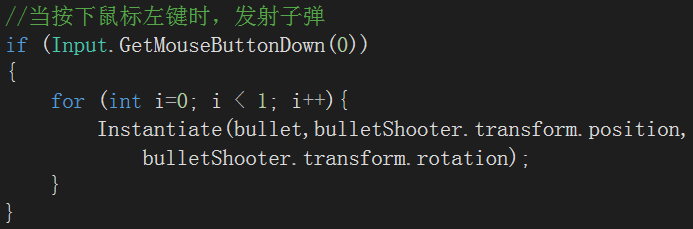


在Start函数中添加以下代码，用于在一定时间后销毁子弹。



步骤 03：创建子弹预设体。在Asset目录下新建文件夹Prefabs, 进入Prefabs目录，右键->Create->Prefab, 将预设体命名为Bullet。将hierarchy面板中的Bullet对象拖到预设体Bullet中，这样子弹预设体就创建完成了。最后删除hierarchy面板中的Bullet对象。

步骤 04： 创建CreateBullet.cs脚本，添加两个public属性。

Update函数中添加以下脚本，用来接收鼠标参数并在bulletShooter的位置实例化子弹。

步骤 05：将CreateBullet.cs脚本挂载大hierarchy面板中的Gun对象上。将Bullet预设体、BulletShooter对象拖到Gun对象的Inspector面板中CreateBullet(Script)组件中的对应位置，如图6。

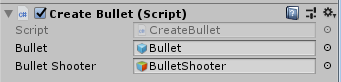


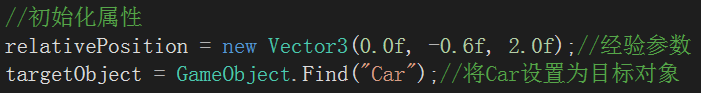
图6 Gun组件的Inspector面板

### 1.2.2b 摄像机跟随

步骤 01：创建CameraFllow.cs脚本，添加下面两个属性。



Start函数中添加以下代码，初始化属性



Update函数中添加以下代码，实时获取目标对象的位置并且更新摄像机的位置。使摄像机和目标对象任何时候都保持在一个相对的位置。



最后将CameraFllow.cs脚本挂载到MainCamera上。

### 1.2.3a 砖块的生成

步骤 01：创建一个cube对象，命名为Brick，参数如图7所示，并为其添加刚体组件（Inspector面板最下面Add Component->Physics->Rigbody）,添加好后如图7所示。

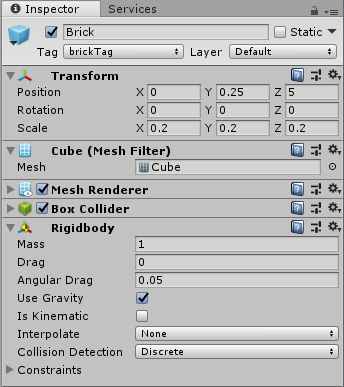


图7 Brick对象的Inspector面板

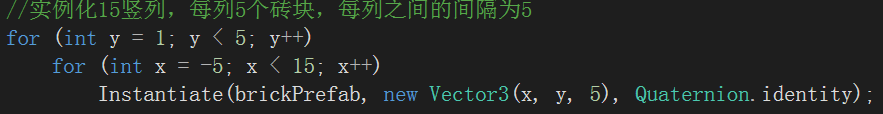
步骤 02：创建砖块预设体Brick, 将Brick对象拖到Brick预设体中，删除Brick对象。

步骤 03：创建CreateBrick.cs脚本。

添加属性。



Start函数



步骤 04：创建createBrick空对象。

将createBrick对象的位置设置在离Car不远处，把CreateBrick.cs脚本挂载到createBrick上。在游戏开始时便会生成砖块了。

### 1.2.3b 射线碰撞检测销毁砖块

步骤 01：**给Bullet添加刚体**（Rigbody）组件，这样才能使用碰撞检测。

步骤 02：给Brick对象添加标签（Tag）。

在Brick预设体的Inspector面板中，点击Tag右侧的下拉选项菜单->Add Tag…进入如图8所示界面，点击“+”按钮，输入标签名“brickTag”->Save, 然后返回Brick对象的Inspector面板->Tag的下拉选项菜单->brickTag。这样就将Brick的标签设置为”brickTag”.。

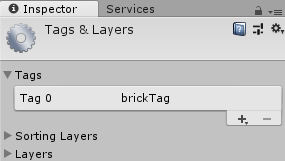
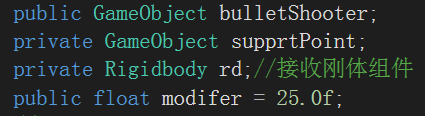


图8 Tag界面

步骤 03：修改BulletFly.cs脚本。

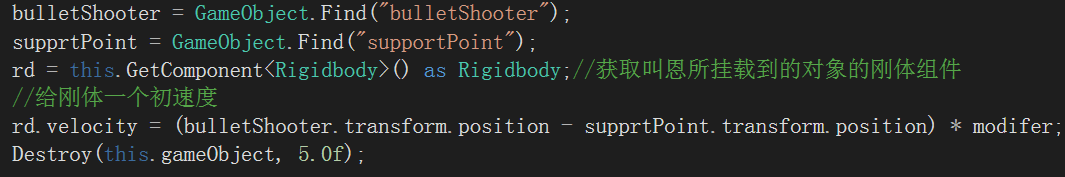
添加属性



Start函数

用脚本给Bullet对象添加刚体组件。

设置刚体的初始速度。

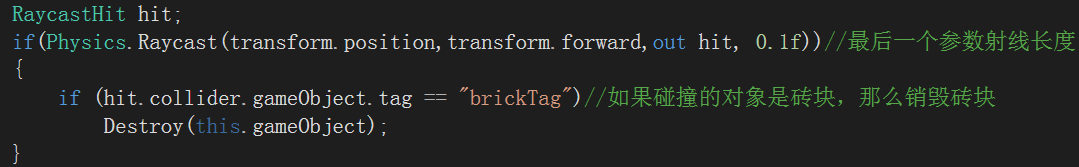


Update函数

将原来Update函数中下面这行用来控制子弹移动的代码删除。（因为已经通过刚体获得初速度了。）



添加以下代码，用来检测子弹是否碰撞到了砖块。



### 1.2.4 枪管的旋转

步骤 01：设置枪管颜色。

在Materials目录下添加一个材质球，设置材质球颜色，将材质球挂到Gun对象上。

步骤 02：创建旋转支撑点。

创建一个空对象supportPoint, 参数如图9，将其设置为Car对象的子对象（在hierarchy面板中将supportPoint对象拖到Car对象上）。

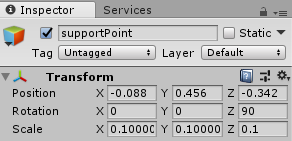
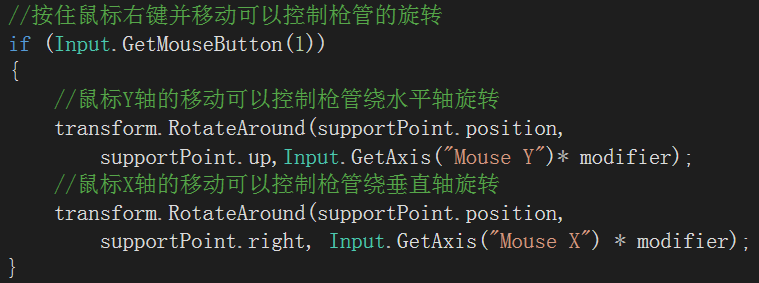


图9 supportPoint对象的Inspector面板

步骤 03：创建RotateGun.cs脚本，用来控制枪管的旋转。

添加属性



Update函数

绕水平轴的旋转可以指向一个垂直的面所有方向，在加上绕垂直轴的旋转，相当于一个垂直面可以水平旋转360度。所以理论上将，这两个轴的旋转可以让枪管指向三维空间的任何一个方向。

步骤 03：将RotateGun.cs脚本挂载到Gun对象上，并将supportPoint对象拖到它的Inspector面板找那个的RotateGun(Script)组件的对应位置，如图10。

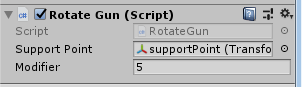


图10 supportPoint对象的Inspector面板

### 1.2.5 子弹的声音和特效

步骤 01：导入特效文件。

进入Asset目录，右键->Import Package->Custom Package…, 然后选择WildFire文件夹，点击确定。导入后如图11所示。

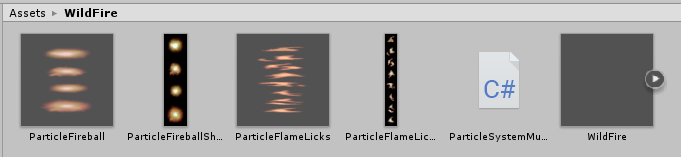


图11 WildFire目录

步骤 02：导入声音文件。

在Asset目录下，创建并进入Sound目录，右键->Import New Asset->选中声音文件->添加。导入后如图12。

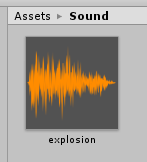
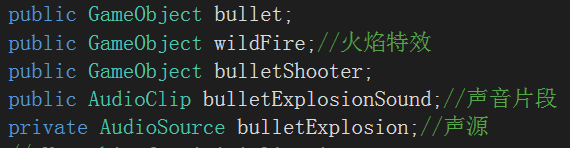


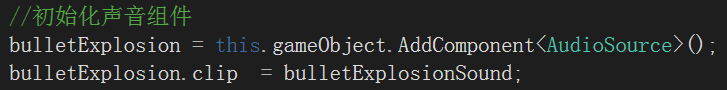
图12 Sound目录

步骤 03：修改CreateBullet.cs脚本。

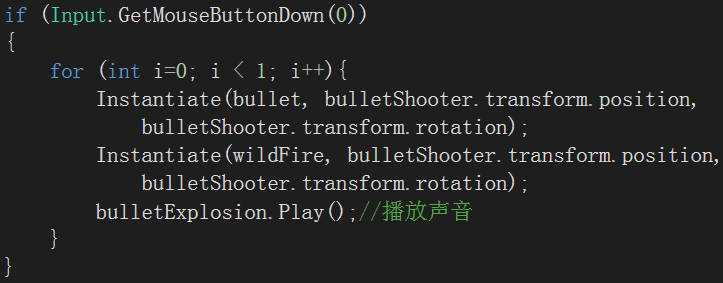
添加属性



Start函数



Update函数



步骤 04：设置WildFire预设体。

给WildFire预设体添加刚体组件和BulletFly.cs脚本。

设置WildFire预设体的粒子系统，如图13。Start LifeTime设置为0.5,1。

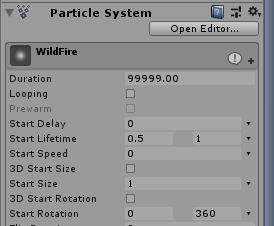


图13 WildFire预设体的粒子系统

步骤 05：在Gun的Inspector面板中，将WildFire预设体和explosion声音文件拖到WildFirCreate Bullet(Script)组件中的对应位置，如图14。

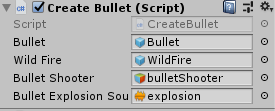
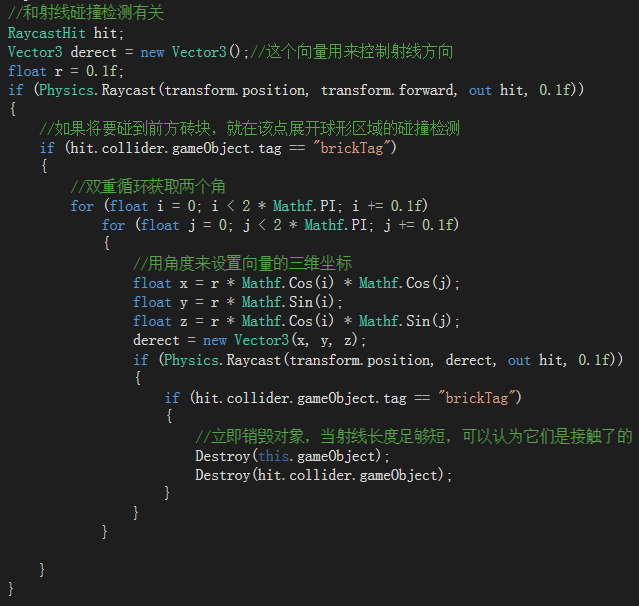


图14 Gun对象的Inspector面板

### 1.2 .6 子弹的爆炸效果

步骤 01：修改BulletFly.cs脚本

修改Update函数。



### 1.2.7球形碰撞检测方法对比

**方法1：**

我一开始写的方法。

先创建一个Vec3向量，然后每一帧都用三重循环遍历向量的三个轴向（x,y,z）来设置向量方向，再进行碰撞检测。

缺点：子弹刚开始飞行的很长一段时间是不会碰到物体的，这部分时间就做球形碰撞检测会浪费很多计算资源。

**方法2：**

老师的方法。

用三角函数的知识，只遍历两个角度就能计算处向量的三个轴向。这里较方法1从做球形碰撞检测来说，时间复杂度从O(n^3)降低为O(n^2)，大大减少了计算量。

此外，该方法，只在即将要碰到物体时才进行球形碰撞检测。相比较方法1，不会做很多无用功。

**方法3：**

杨鑫同学的方法。

首先，它的对象是在检测成功后的零点几秒后才销毁的(因为射线碰撞检测的射线是有长度的，相当于是在实际碰撞之前做检测)，所以当第一次检测到将要碰撞后的零点几秒内，子弹每一帧都会执行球形碰撞检测。

在这里，第一次检测当碰撞后的每一帧都没有比较在进行球形碰撞检测了。杨鑫同学给类添加了一个属性用来标识是否已经检测到碰撞，在第一次检测到碰撞就就将这个属性的值设置为true。在进入球形碰撞检测之前，都要先判断一下这个属性，如果是true就不进行球形碰撞检测。

**方法4：**

基于杨鑫同学发现的问题，结合老师的方法，我又做了进一步的修改。

首先，当检测到碰撞时，是立即销毁对象，而不是延迟零点几秒。这样就不会发生第一次检测到后面的零点几秒还会继续进行球形检测的问题。

但是这样会引入一个新问题，就是射线是有长度的，相当于是在实际发生碰撞之前就检测到了。

我的做法：将射线设置为足够小，这样只要在一定精度内就可以认为它们是实际发生碰撞的了。比如将射线长度从1改为0.1，实际多出的碰撞检测次数是就是在这缩短的0.9范围内发生的碰撞检测次数。只要速度不是太小，那么这个多出的次数是不会很大的。