《计算机图形学》实验报告

《Unity 3D 模拟城市搭建》

姓名 孙洋

学号 3018216046

专业 计科

班级 二班

天津大学智能与计算学部

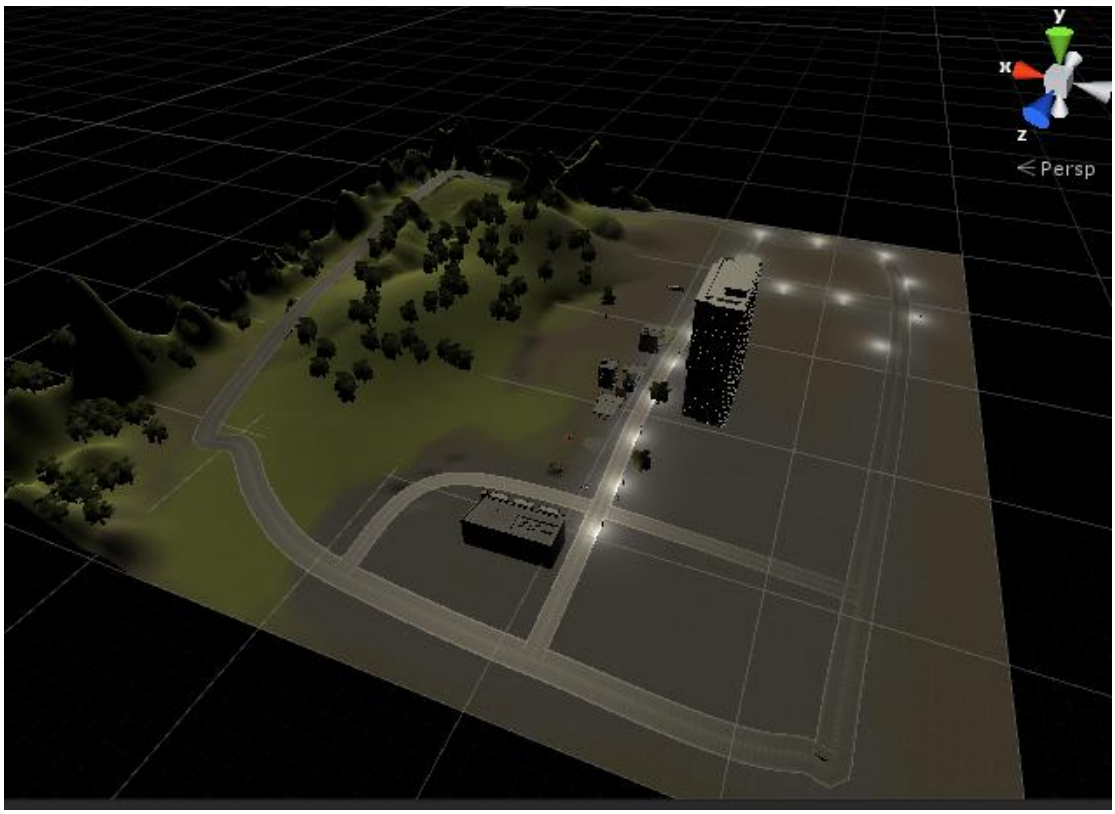
2020 年 10 月 12 日

1. 实验内容

1.1 第三方资源列表

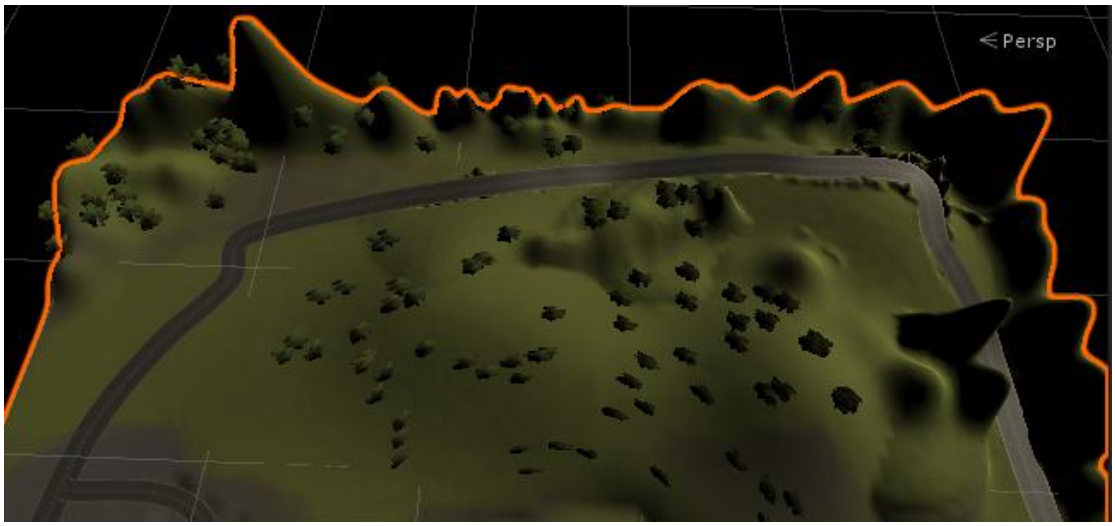
Standard Assets/Sound.cs	给赛车增加启动时声音代码
EasyRoad3D Free	建设道路插件
HQ Racing Car Model No.1203	赛车模型
Nature Manufacture Assets	自然人造物模型资源
Simple city plain	简单建筑物资源
Standard Assets	标准资源库
Traffic Essentials Asset Pack	交通信号灯模型资源
Windridge City Assets	城市模型资源
ResourceChecker	帮助检查现在已用资源的插件
StreetLamp	路灯模型资源

1.2场景搭建



如上图所示，在一块 500 乘 500 的地形上搭建场景，主要分为两个部分：山区和街区。本次作业为了突出灯光的效果，特地使用了夜晚主题，将平行光亮度调小，模拟月亮的亮度，天空盒赋予黑夜天空盒。

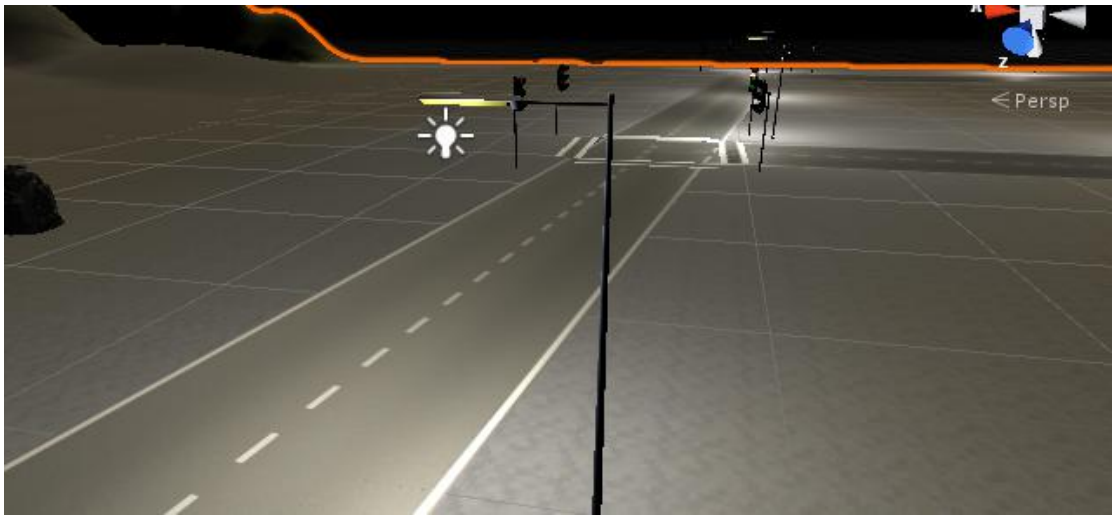
1.2.1 山区



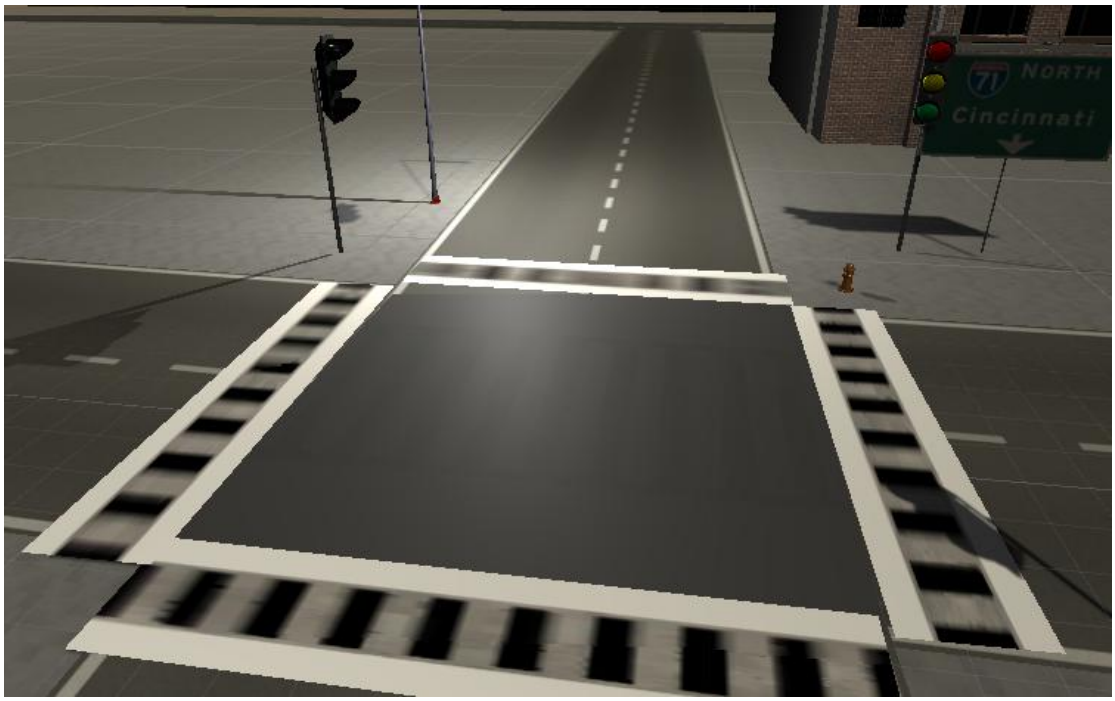
其中山区使用地形编辑画刷隆起一座山；使用随机生成物体的笔刷刷上树木；使用贴图笔刷刷上地板草地绿色。最后使用一条环城道路穿越山区。

1.2.2 街区

主要构建了一条街作为样例：包含路灯、红绿灯、十字路口和建筑物等杂物。



路灯的是使用一个点光源+一个路灯模型实现的，其中灯光需要从 Auto 设置为 Important，否则会出现灯光的错误计算。



因为 EasyRoad Free 插件的免费版本不支持十字路口的创建，所以自己做了个十字路口贴图。

加入红绿灯。

1.3 车辆运动控制实现

车辆控制代码本作业是仔细阅读 Standard Assets 中的车辆控制代码，并将其理解学会。然后自己在该代码基础上保留必要功能，加入车辆控制函数、倒车等代码写成。故可以使用 Sound.cs 的接口。

本代码通过 Unity3d 组件 wheel collider 的自带函数实现动力阻力系统、车辆转弯系统。

1.3.1 控制函数

```
private void FixedUpdate()
{
    //提供前后左右走的常数，传入Move
    float h=0;
    float v=0;
    if (Input.GetKey(KeyCode.W))
    {
        v = 1;
    }
    else if (Input.GetKey(KeyCode.S))
    {
        v = -1;
    }
    if (Input.GetKey(KeyCode.A))
    {
        h = -1;
    }
    else if (Input.GetKey(KeyCode.D))
    {
        h = 1;
    }
    Move(h, v, v);
}
```

FixedUpdate()：指每在现实中流逝单位时间进行一次更新。

其中 WS 键盘控制速度的增减、AD 控制左右转。

v 代表前后走的变量控制，h 是左右的变量控制。

```
public void Move(float steering, float accel, float footbrake)
{

    //clamp input values
    steering = Mathf.Clamp(steering, -1, 1); //限制steering范围 方向盘
    AccelInput = accel = Mathf.Clamp(accel, 0, 1); //加速
    BrakeInput = footbrake = -1 * Mathf.Clamp(footbrake, -1, 0); //刹车
    //handbrake = Mathf.Clamp(handbrake, 0, 1); //手闸


    m_SteerAngle = steering * m_MaximumSteerAngle;
    m_WheelColliders[0].steerAngle = m_SteerAngle;
    m_WheelColliders[1].steerAngle = m_SteerAngle;



    ApplyDrive(accel, footbrake); //前进后退速度变化。
    CapSpeed(); //限制速度不能超过最大值



    CalculateRevs();//计算转速
    GearChanging(); //计算档位

    AddDownForce(); //添加抓地力

    TractionControl(); //控制牵引力
}
```

加速和减速的参数传入。

通过对 WheelCollider 组件的函数调用，赋给轮胎旋转角度。

通过 ApplyDrive 函数根据当前不同速度计算扭力的正向反向。

也有转速和档位的函数，方便 Sound.cs 接入。

其中转速、档位、和控制牵引力函数为非必要函数。

1.3.2 速度函数

```
//加速和减速
private void ApplyDrive(float accel, float footbrake)
{

    float thrustTorque;

    thrustTorque = accel * (m_CurrentTorque / 4f);

if (accel == 1.0 && footbrake == 0)//前进
{
    for (int i = 0; i < 4; i++)
    {
        m_WheelColliders[i].motorTorque = thrustTorque;
    }

}
else if (footbrake == 1.0 && accel == 0 && CurrentSpeed <= 0.1f)
{//倒车
    for (int i = 0; i < 4; i++)
    {
        m_WheelColliders[i].motorTorque = -5000f/4f;

    }
}
else if (footbrake == 0 && accel == 0)
{//停车
    for (int i = 0; i < 4; i++)
    {
        m_WheelColliders[i].motorTorque = 0f;
        m_WheelColliders[i].brakeTorque = 0f;
    }
}

    for (int i = 0; i < 4; i++)
    {
        if (CurrentSpeed > 5 && Vector3.Angle(transform.forward, m_Rigidbody.velocity) < 50f)
        {
            m_WheelColliders[i].brakeTorque = 20000f * footbrake;
        }
        //把刹车力去掉，方便再启动
        if (CurrentSpeed <= 0.1 && CurrentSpeed > -0.001f)
        {
            m_WheelColliders[i].brakeTorque = 0;
        }

}
}
```

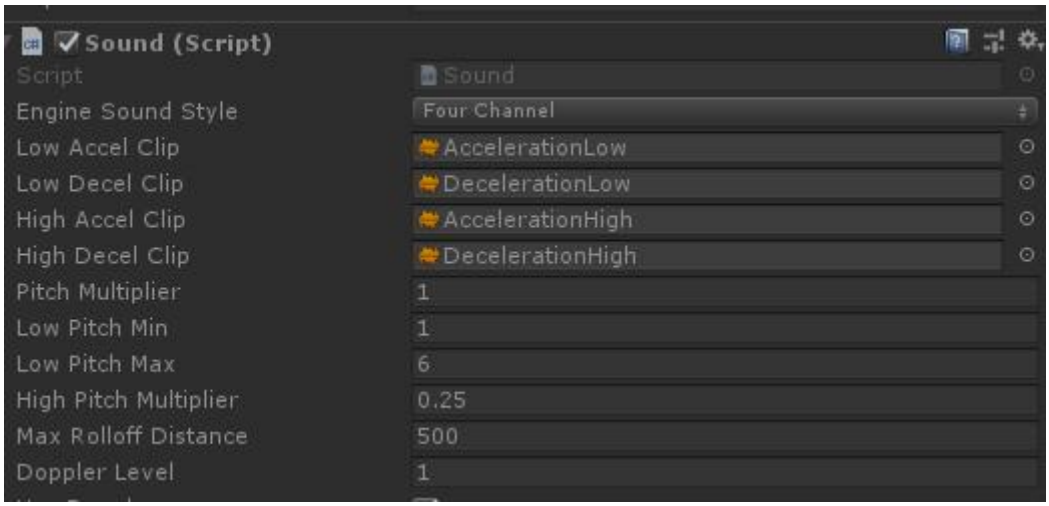
通过调整轮子组件的牵引力和刹车力来实现车辆的加速、减速、倒车、停车、再启动。

1.4额外效果/功能

车灯系统：两个射线车前灯，两个点光源车后灯。

车灯系统：实际运行效果如图。

借用 Sound.cs 和档位转速函数，实现了车声音的分阶段播放。



2. 问题与解决

2.1 如何实现汽车运动代码

网上的汽车控制代码难以找到且质量不高，在阅读理解标准控制代码后，掌握了 WhellCollider 的使用方法，就可以较为容易的实现汽车的左右转弯。

2.2 如何给路灯增加灯光

在路灯组件下新建一个子组件点光源/面光源，调整适度亮度。

2.3 开始运行之后路灯一闪一闪总是消失

设置所有路灯的灯光类型从 Auto 到 Important。

2.4 如何搭建地形

新建一个 Terrain 组件即可，不要使用 plain 或者 3Dcube。

2.5 如何添加树林和地面贴图

使用 Tree 画刷添加树林，使用 Texture 画刷添加地面贴图。

2.6 如何画出道路

使用 EasyRoad3D Free 插件可以创建道路，免费版不能创建十字路口，需要自己用组件画出来。

2.7 如何给房子添加碰撞

网上添加 Box Collider，然后点击 Edit Collider 将绿色方块拉扯到覆盖建筑边缘。

2.8 车辆过于容易侧翻

重心太高，是 WeelCollider 组件的碰撞盒子太大了，需要调低半径。

2.8 相机所能拍摄角度过小，易眩晕

俯视角太高，看不到前面的路况，需要调整摄像头位置

3.总结

通过这次的 Unity3D 模拟城市实验，我掌握了以下内容：

1. Unity3D 的基础用法
2. 更形象地体验到了相机的取景和光线在场景里的状态。
3. 初步掌握 C#的使用方法。
4. 初步了解 Unity3D 的组件架构、代码使用方式。
5. 初步掌握汽车运动代码的编写。
6. 体验到了计算机图形学讲到的镜头、渲染和光线追踪的实际应用。