# 天津大学

# 《计算机图形学实验报告》


姓名 松良 （留学生）

学号 6318000433

专业 计算机科学与技术

班级 2018

2021 年 1 月 10 日

# 计算机图形学实验报告

## 1. 实验内容

- 查找网络资源，选取合适的现代城市模型，城市模型需包含城市建筑、城市道路、路灯及道路口、红绿灯等具体设施，更多细节可自行选择添加，并在 Unity 3D 中进行模拟城市搭建。
- 查找车辆模型，编写控制车辆运动的脚本代码，使车辆可以在上述城市场景中正常驾驶。

### 1.1. 第三方资源的使用

本次实验使用的第三方资源库包括：low ploy city，这是在 turbosquid 网站中检索到的城市模型资源，包括 fbx 格式的 3d 模型和装扮城市的贴纸．网址为 https://www.turbosquid.com/3d-models/3d-model-used-1283609．同样在这个网站上找到了实验使用的汽车模型．使用的脚本为网络上搜集的 unity 照相机平滑跟随汽车脚本．它的基本思路是，紧跟汽车，根据汽车的运动而运动，根据汽车的旋转而旋转．

### 1.2. 场景搭建

本次实验直接使用场景自带的 Directional Light 作为光源．这种类型的灯光可以被放置在无穷远处，影响场景中的一切游戏对象，类似于自然界中太阳光的照明效果．Directional Light 的 Shadow 属性，用于生成阴影．Shadow 分为 Hard Shadow 和 Soft Shadow．区别是 Soft Shadow 的阴影边缘比较平滑，接近真实，但是性能消耗大于 Hard Shadow．选择 Soft Shadow，可以看出，场景中所有物体都有光照产生的阴影．

本实验使用的素材都是简单的多边形构成的立体图形，搭建场景的工作比较简单．先铺设一张巨大的平面作为地面，在地面上搭建各类建筑，建筑之间搭设道路,摆设红绿灯，最后为模型贴上合适的图片即可．

为了小车能在城市中正常运动，要给各类物体添加碰撞属性．地面添加 mesh collider，楼房添加 box collider.

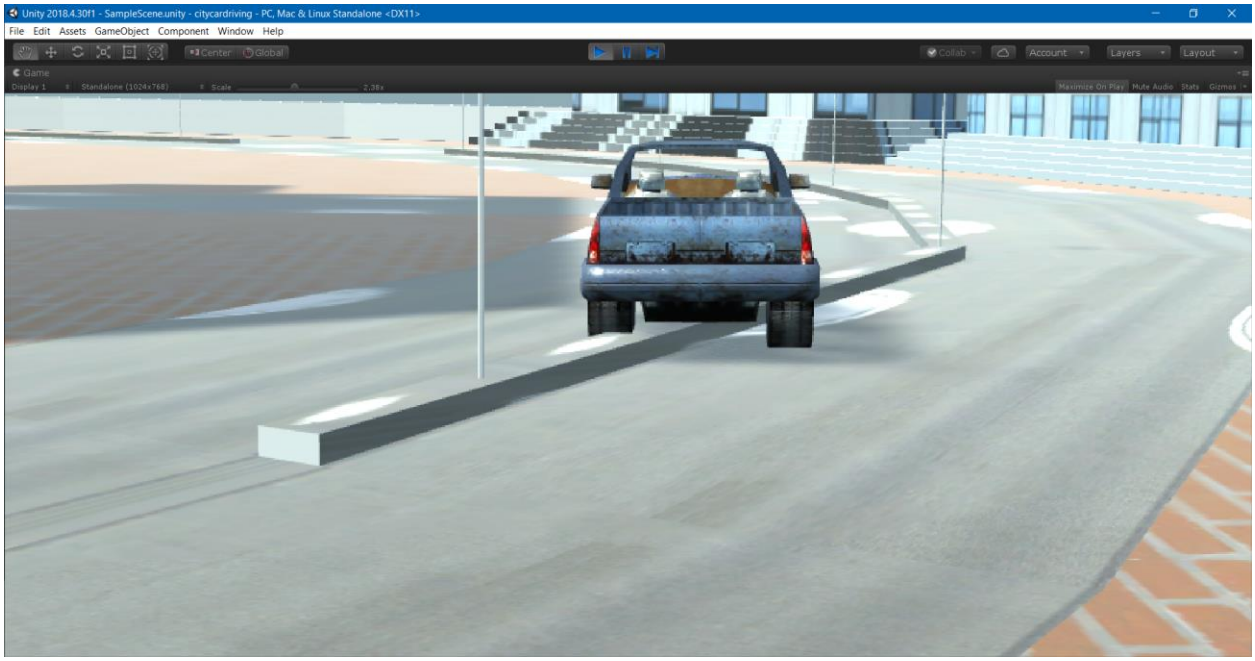


图 1.1 场景搭建

## 1.3. 车辆运动控制

### ➢ Car control 代码实现

```
using UnityEngine;
using System.Collections;

public class CarCtrl: MonoBehaviour {

    public WheelCollider WheelFL;
    public WheelCollider WheelFR;
    public WheelCollider WheelRL;
    public WheelCollider WheelRR;
    public Transform WheelFLtrans;
    public Transform WheelFRtrans;
    public Transform WheelRLtrans;
    public Transform WheelRRtrans;
```

```
    public AudioClip AC;
    public Vector3 com;
    float gravity = 9.8f;
    private bool braked = false;
    private float maxBrakeTorque = 500;
    private Rigidbody rb;
    private float maxTorque = 1000;

    void Start ()
    {
        rb = GetComponent<Rigidbody>();
        rb.centerOfMass = com; // Set the car's center of
gravity
    }

    // collision sound
    public void OnCollisionEnter(Collision collision)
    {
        //The origin of the hit object makes a sound (the second
parameter is used to set the world coordinates of the sound)
        AudioSource.PlayClipAtPoint(AC,
transform.localPosition);
    }

    void FixedUpdate () {
        if(!braked){
            WheelFL.brakeTorque = 0;
            WheelFR.brakeTorque = 0;
            WheelRL.brakeTorque = 0;
            WheelRR.brakeTorque = 0;
        }
        // Car forward torque
        WheelRR.motorTorque = maxTorque *
Input.GetAxis("Vertical");
        WheelRL.motorTorque = maxTorque *
Input.GetAxis("Vertical");

        // Car rotation angle
        WheelFL.steerAngle = 30 * (Input.GetAxis("Horizontal"));
        WheelFR.steerAngle = 30 * Input.GetAxis("Horizontal");

    }

    void Update()
    {
        Brake(); // brake

        // Wheel rotation

WheelFLtrans.Rotate(WheelFL.rpm/60*360*Time.deltaTime ,0,0);

WheelFRtrans.Rotate(WheelFR.rpm/60*360*Time.deltaTime ,0,0);
```

```
WheelRLtrans.Rotate(WheelRL.rpm/60*360*Time.deltaTime ,0,0);

WheelRRtrans.Rotate(WheelRL.rpm/60*360*Time.deltaTime ,0,0);

        // Wheels change direction
        Vector3 localeuler = WheelFLtrans.localEulerAngles;
        Vector3 localeuler2 = WheelFRtrans.localEulerAngles;
        localeuler.y = WheelFL.steerAngle;
        WheelFLtrans.localEulerAngles = localeuler;
        localeuler2.y = WheelFR.steerAngle;
        WheelFRtrans.localEulerAngles = localeuler2;
    }

    // brake
    void Brake()
    {
        if(Input.GetButton("Jump"))
        {
            braked = true;
        }
        else
        {
            braked = false;
        }
        if(braked){

            WheelRL.brakeTorque = maxBrakeTorque * 20;
            WheelRR.brakeTorque = maxBrakeTorque * 20;
            WheelRL.motorTorque = 0;
            WheelRR.motorTorque = 0;
        }
    }
}
```

表 1.1 Car control 代码

## ➢ 代码解析

首先，为小车整体添加刚体属性，车轮添加 WheelCollider 碰撞属性．在小车控制脚本中定义四个 WheelCollider 型变量，控制小车的四个车轮．这四个变量分别是 WheelFL，WheelFR，WheelRL，WheelRR．另有四个 Transform 型变量，控制小车车轮的旋转．

小车的后轮提供动力：WheelCollider 变量中的 motorTorque 属性，模拟在轮轴上的电机力矩，根据方向正或负．控制小车向前．要想让小车前进，必须先修改 motorTorque 这个变量：

```
WheelRR.motorTorque = maxTorque * Input.GetAxis("Vertical");
```

同样的，steerAngle 控制旋转的角度，通过旋转小车的前轮，控制小车转向．

```
WheelFL.steerAngle = 30 * Input.GetAxis("Horizontal");
```

以上控制小车前轮与后轮的代码都写在 FixedUpdate 函数中，他与 Update 的区别是，Update 跟当前平台的帧数有关，而 FixedUpdate 是真实时间，所以处理物理逻辑的时候要把代码放在 FixedUpdate 而不是 Update．

在 Update 函数中，实现了小车的前进与转向．通过变 wheelcollider.rpm，可以获取车轮的转速．控制车辆前进，实际就是控制车轮模型的旋转．算出每秒转动的度数：wheelcollider.rpm360/60，使用车轮模型.Rotate() 进行控制，车轮模型.Rotate(wheelcollider.rpm*360/60*Time.deltaTime)，即可让车轮转动，小车前进．

```
    WheelFLtrans.Rotate(WheelFL.rpm/60*360*Time.deltaTime ,0,0);
    WheelFRtrans.Rotate(WheelFR.rpm/60*360*Time.deltaTime ,0,0);
    WheelRLtrans.Rotate(WheelRL.rpm/60*360*Time.deltaTime ,0,0);
    WheelRRtrans.Rotate(WheelRL.rpm/60*360*Time.deltaTime ,0,0);
```

控制车轮模型的转向转动，实际是控制模型在竖直向上方向的转动，也就是 Y 轴方向. 可以通过 wheelcollider.steerAngle 改变转向，修改 Transform 型变量的 localEulerAngles:Vector localeuler = 轮子型.localEulerAngle, localeuler.y = wheelcollider.steerAngle 再把 localeuler 赋值给轮子模型.localEulerAngle 就行了.代码如下：

```
Vector3 localeuler = WheelFLtrans.localEulerAngles;
Vector3 localeuler2 = WheelFRtrans.localEulerAngles;
localeuler.y = WheelFL.steerAngle;
WheelFLtrans.localEulerAngles = localeuler;
localeuler2.y = WheelFR.steerAngle;
WheelFRtrans.localEulerAngles = localeuler2;
```

控制小车刹车的功能也很简单，brakeTorque 是刹车的力矩. 只要写个刹车函数,赋予 brakeTorque 一个值，再将动力距 motorTorque 设为 0，Update 函数调用一下即可. 详情可见表 1.1.

## 1.4. 额外实现功能- audio 添加音效

游戏在运行时没有声音，如果加入声音，会大大的增强沉浸感，提高游戏体验.unity 提供了强大的声音组件来帮助播放音效，即 AudioSystem，使用 AudioSystem 为小车运行时提供背景音乐.

本次实验我添加了碰撞音效：OnCollisionEnter 可以监测碰撞事件，在这个事件函数中添加播放的音效：

```
// 碰撞音效
public void OnCollisionEnter(Collision collision)
{
//被撞得物体原点发出声音（第二个参数用来设置发出声音的世界坐标）
AudioSource.PlayClipAtPoint(AC, transform.localPosition);
}
```

类似的可以实现包括了加速、减速、刹车、碰撞等等的音效. 思路就是根据汽车的当前的行驶速度和驱动力来为汽车划分为多个行驶状态, 然后对于每个行驶状态,根据行驶速度和驱动力的大小来完善声音的细节. 例如: 驱动力增大以后, 声音的音量加大. 就是在 update 函数里, 得到汽车当前的状态, 并判断汽车当前的状态, 然后为 AudioSource 组件指定相应的 clip, 指定完成后, 还可以对各个片段的播放细节进行处理, 以达到更加真实的效果.时间关系此部分功能未能完全实现.

## 2. 问题与解决

实验当中中遇到所有的问题有:

- 车轮旋转但是小车不前进.经过查阅资料得知,是碰撞器之间发生相互冲突,才使小车前进失败.删除了多余的碰撞器后,问题得以解决.
- 小车落地后碰撞飞走.从网上搜集资料后发现,必须为小车的车体添加box-collider,小车才能正常落地.
- 小车从地面坠落.因为小车轮胎的弹力Spring 设置的太小,才使小车坠落,把这个值调大即可.

## 3. 总结

此次实验, 我在 unity 中搭建了一个城市, 并在城市中运行一个汽车. 实验帮助我熟悉了 unity 的基本操作, 如 3d 模型的导入, 物理属性的添加等. 在 unity 中, 一切物体和运动都在模仿真实的物理过程, 包括光照阴影的计算, 小车的重力和重心,甚至还考虑到了小车轮胎的弹力和摩擦力.

在为小车添加控制脚本时, 我通过控制小车车轮的旋转与转向, 成功实现了汽车的转向与前进, 而不是单纯地坐标改变. 之后, 我还搜集声音素材, 通过 audio, 为小车的加速刹车等操作添加了音效.

实验帮助我进一步理解了计算机图形学课上的内容, 物体的运动实际上是坐标的改变, 物体的旋转可以直接在局部坐标系中进行.