# 真实感图像渲染

石景宜 2016011395

## 一、 完成的功能

- [x] RT
  - [x] 折射、反射、漫反射等基本功能
  - [x] 模糊反射，软阴影
- [x] 贴图
- [x] 超采样抗锯齿
- [x] 贝塞尔曲线
  - [x] 曲面方程求交
- [x] 加速
  - [x] 包围盒加速
  - [x] `openmp`

## 二、软阴影、模糊反射、纹理贴图、抗锯齿

**软阴影**

采用多个点光源模拟面光源。

**模糊反射**

反射时将原反射光线变成多条有扰动的光线。

**纹理贴图**

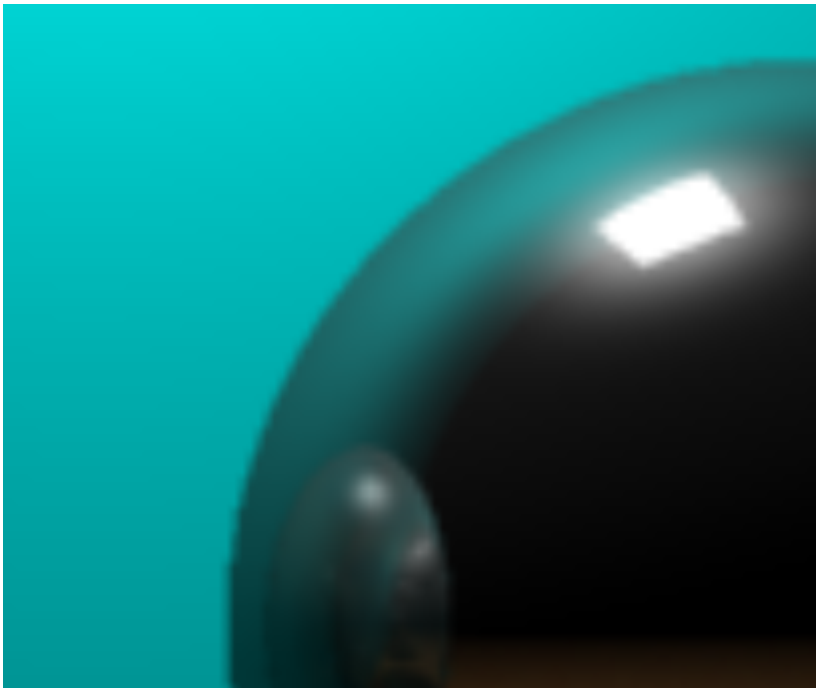
将点坐标映射到纹理贴图的坐标上，为了让贴图更加精准，使用目标点周围四个点颜色的加权求和作为目标点的颜色。

**抗锯齿**

在第一轮渲染后，进行第二轮超采样来达到抗锯齿的效果，超采样对和周围颜色不同的点进行四个点加权和的采样。

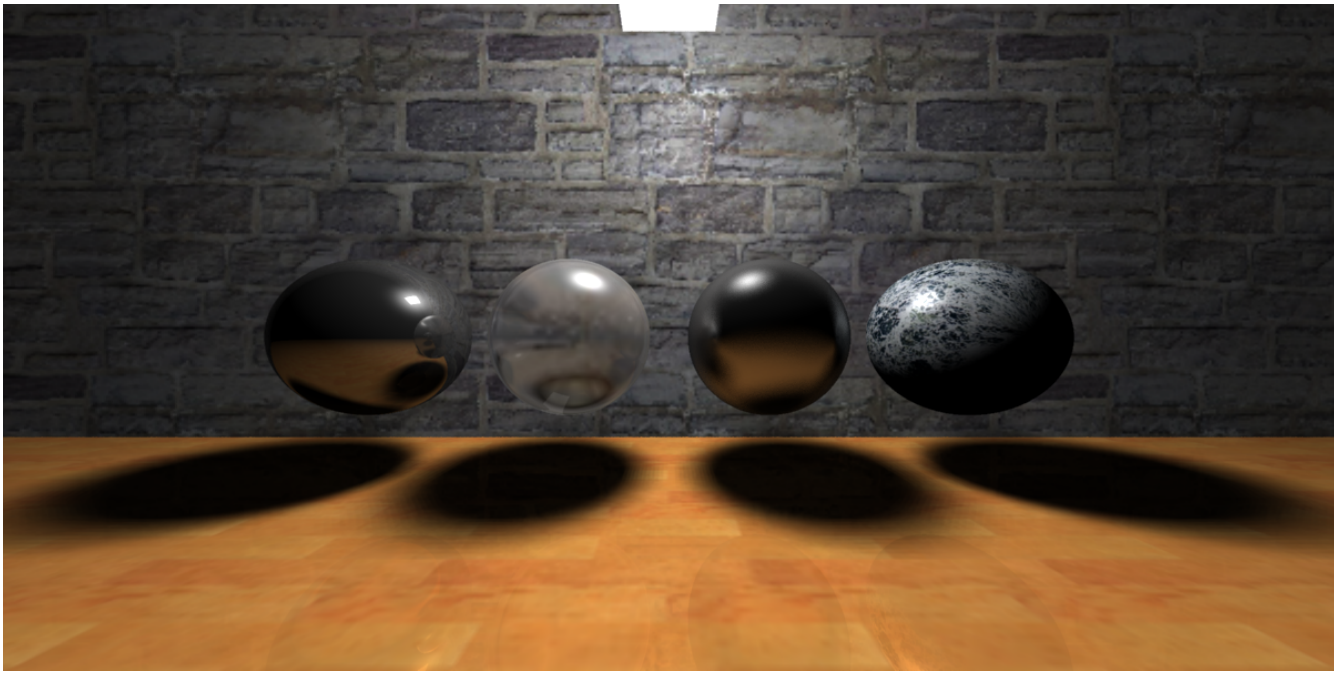
抗锯齿前：

抗锯齿后：



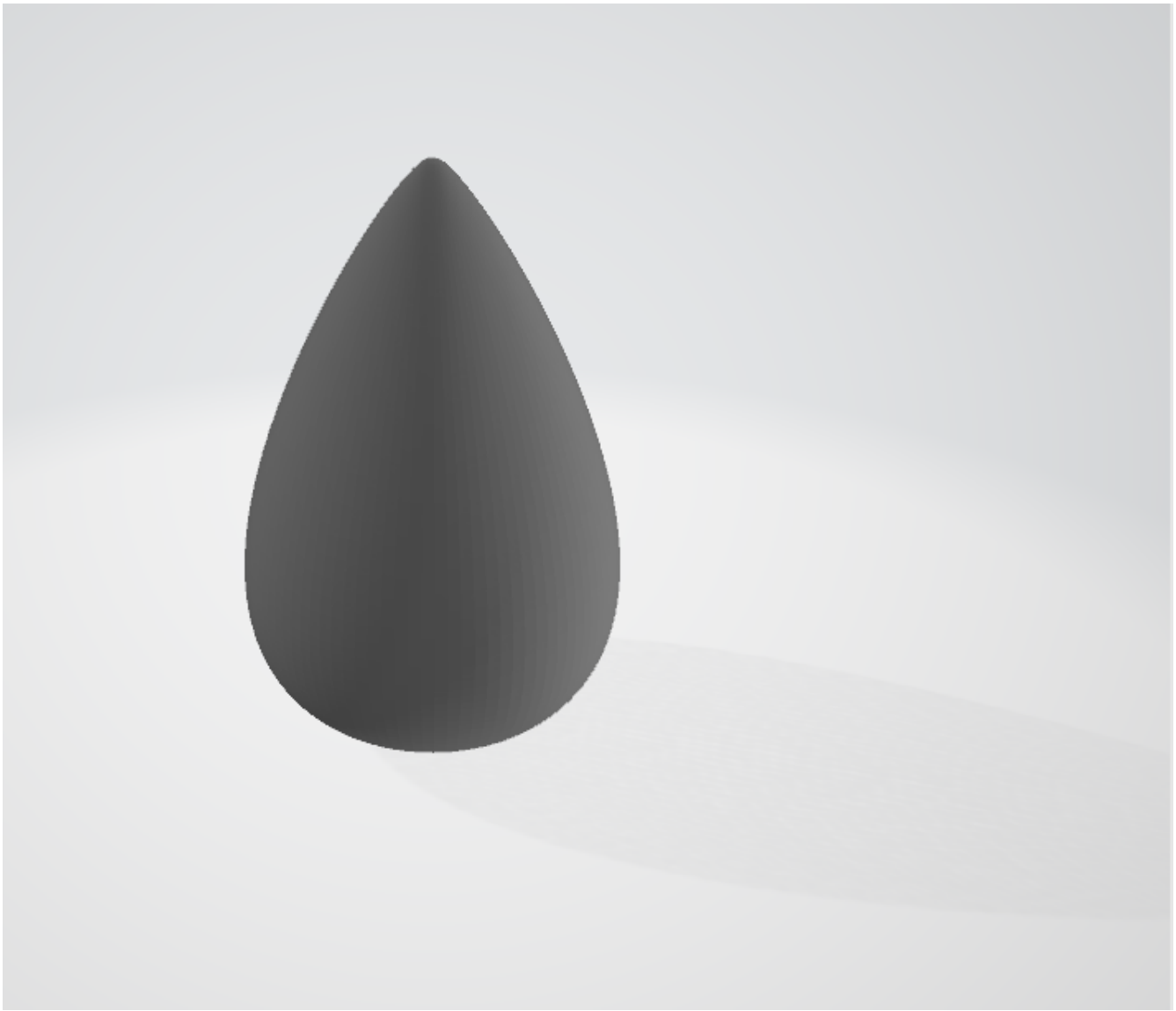
图像质量有明显提高。

**效果图：**

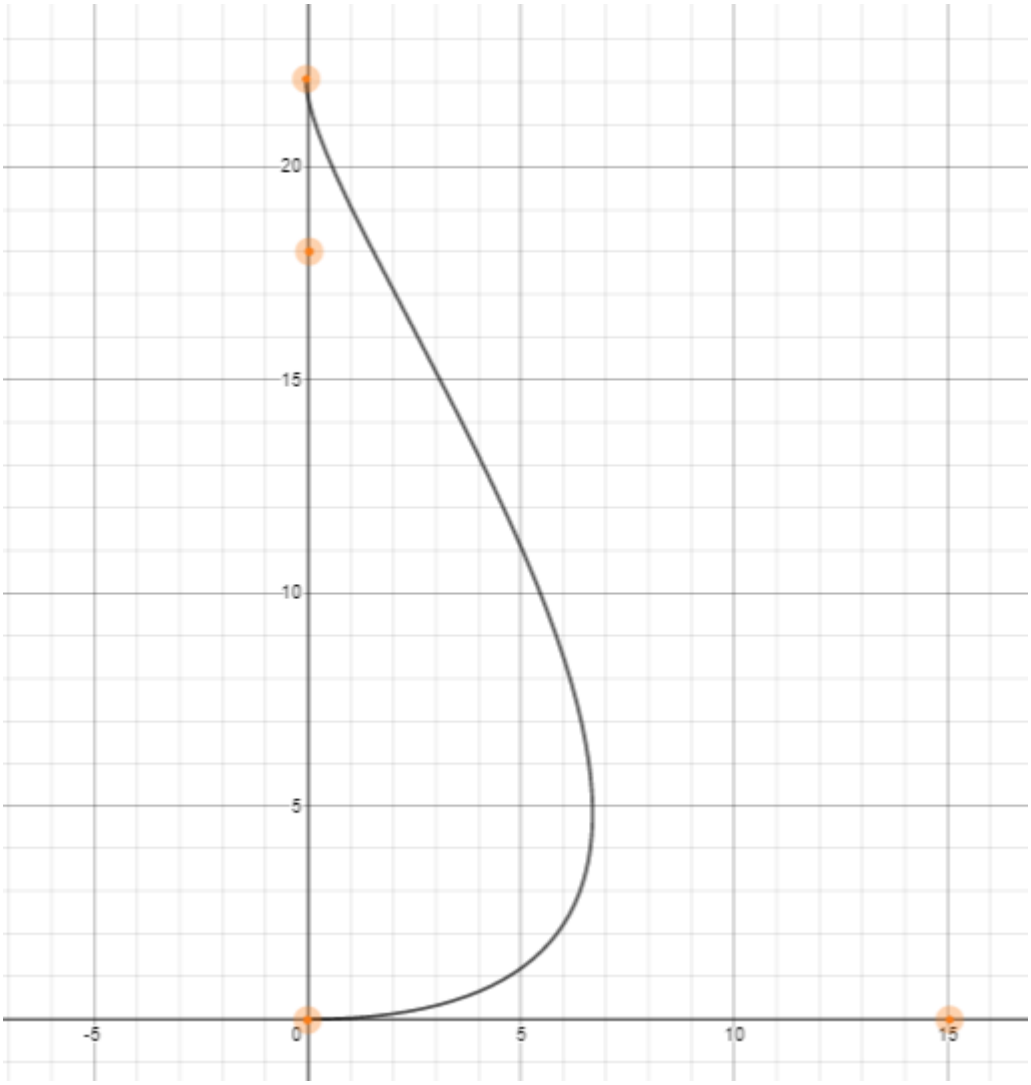


## 三、 贝塞尔曲线

贝塞尔旋转体如下：



旋转曲线如下：




**阻尼牛顿法迭代求交过程**

贝塞尔曲线参数形式：

$$\mathrm{P(t)} = \sum_{i=0}^{n} P_i B_{i,n}(t), t \in [0,1]$$

$P_i$是控制点的位置，$B_{i,n}(t)$形式如下：

$$B_{i,n}(t) = C_n^i t^i (1-t)^{n-i}, \mathrm{i} = 0,1,\dots,\mathrm{n}$$

在三维坐标系中，贝塞尔曲面可表示如下，其中O是贝塞尔曲面所在位置：

$$\begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} * P(u) + O$$

光线与贝塞尔曲面求交，相当于求解如下方程：

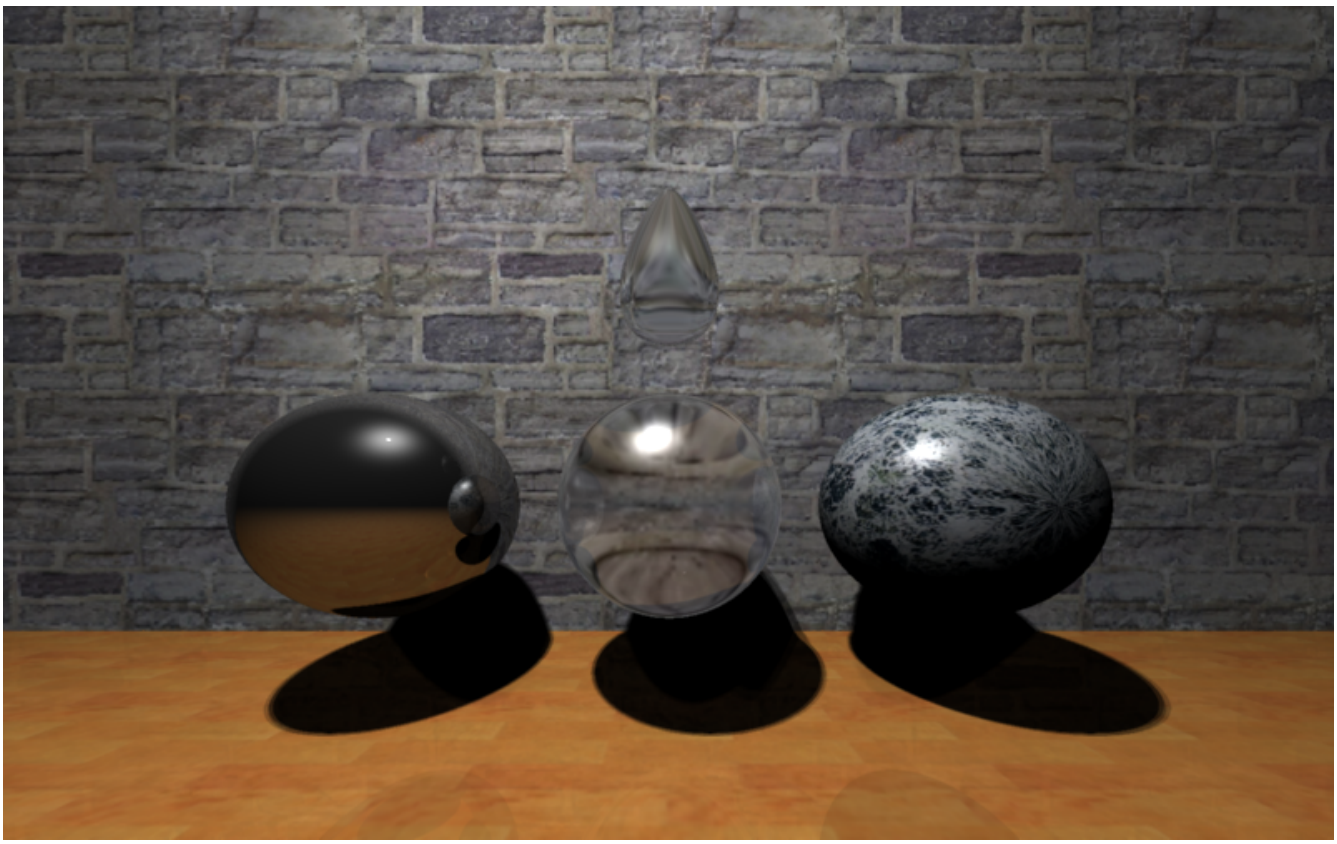
$$\begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} * P(u) + O - (rayO + rayD * t) = 0$$

其中，`rayO`表示光线的起点，`rayD`表示光线的方向,`t`为光线到交点的距离。未知量为u,t,θ，t可以用u和θ以及光线的信息求出。记`x`为一个三维向量，表示`(u,θ,t)`，即方程的解，上述方程左边记为为`F(x)`,则可列出牛顿迭代公式：

$$x_{i+1} = x_i + Jacobi(F(x_i))^{-1} * F(x_i)$$

在选取初始点时，在曲面上随机取一个点，即随机初始化一个u和θ，然后算出光线到该点的距离t，迭代三十轮在绝大部分情况下能够收敛。

**效果图：**



# 四、实验总结

谢谢老师、助教以及向我推荐这门课的同学。在这次实验中我对游戏、动画中的图像渲染原理有了深入的了解。同时也很遗憾没有坚持写完PPM。

# 五、运行说明

**windows**

vs打开项目文件即可。

**linux**

在代码目录中使用如下命令编译，然后运行`./main`：

```
cmake .
make
```

**注意**

在windows下，在`RayTracer.cpp`加入：

```
#define WIN
```

在linux下，加入：

```
#define UBUNTU
```