# 基于 Bezier 曲线的三维造型与渲染 实验报告

## 计 52 路橙 2015010137

# Part 1: 功能实现

#### 1. 渐进式光子映射:

- (1) 面向对象实现。底层包括颜色类、数据类和图片类,功能包括物体类、相机类、场景类、光源类、碰撞图类、光子类和光线追踪引擎类。
- (2) 首先实现一遍光线追踪,使用的 BRDF 为可逆 Phong 模型,得到碰撞的点并建立光子图(kd-tree 建树)。接着随机发射光子并使用 kd-tree 寻找邻近光子。

#### 2. Bezier 曲线旋转造型:(对应于 Bezier.cpp)

- (1) 使用 7 个控制点的 6 阶 Bezier 曲线旋转得到半封闭曲面。曲线的求值和求导都手动化简为多项式的表达式,从而减少运算的时间复杂度。
- (2) 曲面与光线求交通过构造长方体包围盒以及牛顿迭代法进行。包围盒求交使用 Woo 算法,若有解,则以此解作为牛顿迭代 t 值的初始值(需要注意的是盒内和盒外不同,为了减小误差,需要先判断光源是否在盒内,是的话直接进行下一步,否则再判断是否与包围盒有交点)。
- (3) 牛顿迭代另外的初值 u 为[0,1]的随机数, v 为[0,2π]的随机数, 进行 20 轮牛顿迭代, 得到一个解(也可能没有解), 并如上循环 10 次, 取最终的 t 值最小(即离光源最近的点)作为解。若每一次循环都不存在解(通过验证最后一次求得的解对应的 F 值是否大于 0.01 来判断),则直接返回失败。

#### 3. 抗锯齿: (对应于 raytracer.cpp 中的 Resampling 函数)

在光线追踪的过程中进行超采样。对点(i, j),相机还发射对应于点 $(i\pm\frac{1}{3}, j\pm\frac{1}{3})$ 的光线,并将这些光线追踪后得到的颜色值平均作为点(i,j)的颜色值。

#### 4. 软阴影: (对应于 arealight.cpp 中的 GetIrradiance)

对于点与面光源的漫反射求颜色,迭代 4 次,每次迭代对面光源上随机选 1 个点,并取

这个点周围的 **15** 个点分别对光线求交,最终将亮度平均作为漫反射的亮度,从而实现 软阴影。

### 5. 纹理贴图:(对应于各个物体类的 GetTexture)

对球体通过球坐标构造映射,对平面通过对长宽取余构造映射,实现 UV 纹理贴图。

### 6. 凹凸贴图 (法线贴图): (对应于 NomalBall 类和 NomalPlane 类)

利用 nomal map 原理,对球体和平面分别通过法向量构造切向量和副法向量,通过导入纹理贴图所对应的法线贴图 (由 Photoshop 手动生成),将其 rgb 归一化后乘以 2 再减 1 得到[-1,1]的值,作为三个正交向量加权的权值,生成新的法向量并作为光线反射、折射所采用的法向量。

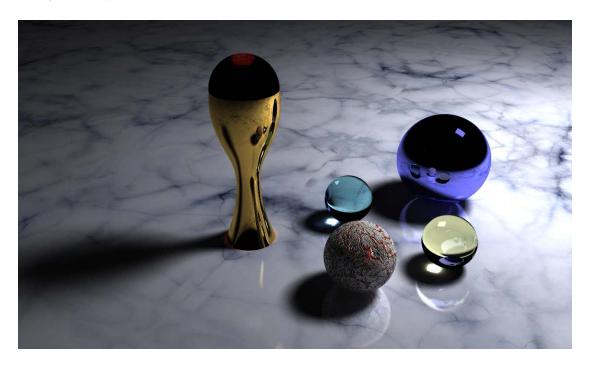
# Part 2: 效果图

## 1. 法线贴图:





# 2. 最终效果图:



Bezier 曲线生成杯子上方放置了一个全折射的吸收颜色的透明球, Path Tracing 过程中并没有红色光圈出现,但 ppm 迭代后出现了逐渐加深的红色光圈,并且在杯子的底部也有红色光透出。青色和淡黄色球都为全折射的玻璃球,

底面为半漫反射半镜面折射,火山裂缝球为法线贴图的全漫反射球,蓝色球为全镜面反射球。

图中可以看出明显的透明球折射产生的焦散,底面地板反射光、金杯曲面的反射光、软阴影。

Part 3: Bezier 曲线旋转曲面网格模型

