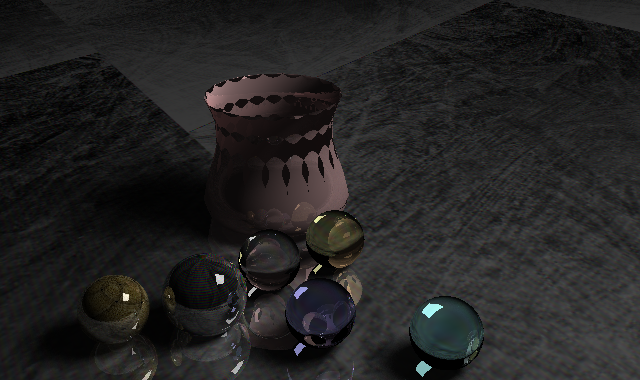
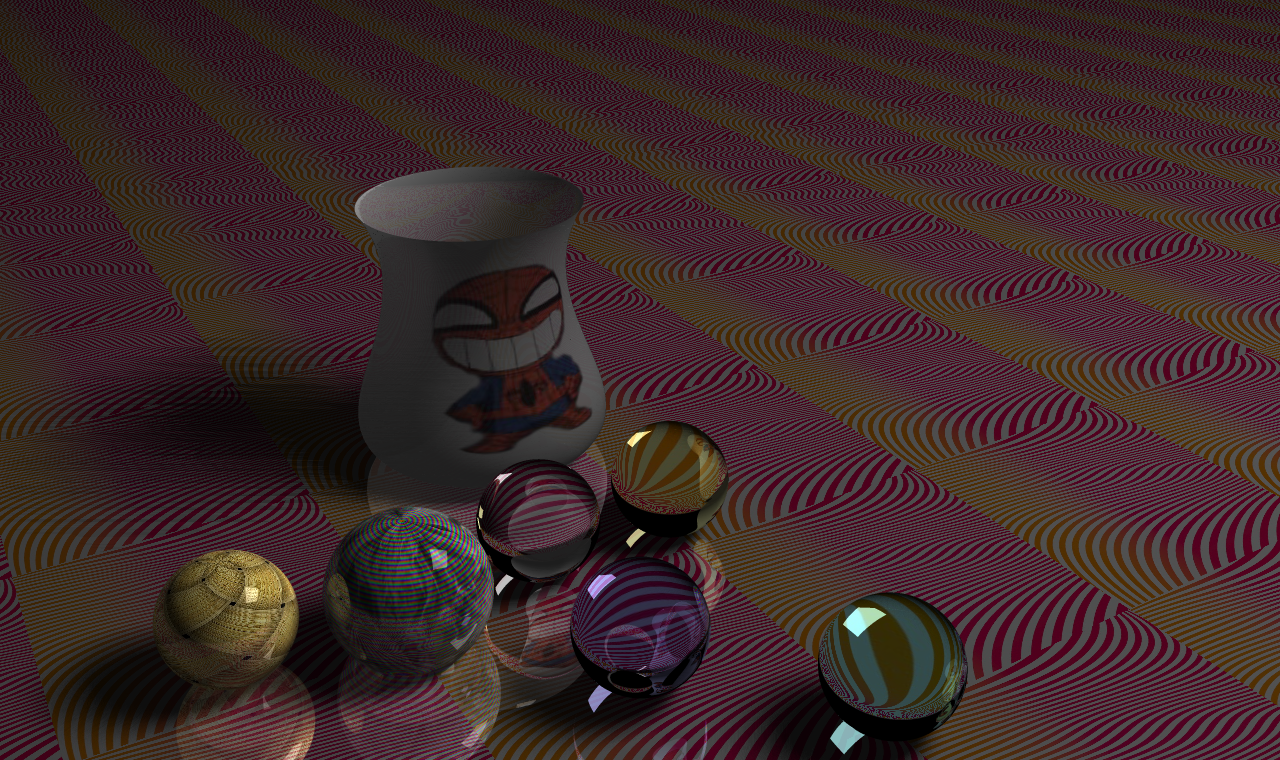
基于Bézier曲线的三维造型与渲染

---计55 赵嘉霖 2015011344

1. 实现算法：
   1. 光线跟踪：从视角与视平面上的像素点的连线发出射线，与接触到的物体进行折射、反射递归，漫反射则对于所有光源判断是否有物体遮挡。代码段：raytracer.cpp
   2. Bézier曲面求交：三次曲线求交，先估算初值，再利用牛顿迭代，如果收敛则算出结果。这里我试验了两种方法，第一种是先将Bézier曲面转化为面片，然后通过面片求交，得出处置点进行牛顿迭代(代码段：obj.cpp和square.cpp)；第二种是利用包围盒方法，如果与包围盒相交则在包围盒中随机一组（20个）初值，分别进行牛顿迭代，收敛且离光源最近的点就是光线的交点（代码段obj.cpp）。
   3. 贴图：将曲面（或平面）映射为平面，然后进行贴图，在手动调整一下贴图的位置。(代码段IO.cpp)
   4. 超采样抗锯齿：计算每条光线经过的物体，对于相邻两条经过不同物体的光线进行超采样。（代码段raytracer.cpp）
   5. 软阴影：通过蒙特卡洛方法对面光源采样。（代码段light.cpp）
   6. 包围盒：加速Bezier曲面求交运算。（代码段box.cpp）
2. 最终效果图：
   1. 面片求交再牛顿迭代：



* 1. 包围盒求交再牛顿迭代：



* 1. 网格模型截图：

