CG Project：区间扫描线 Z 缓冲器算法实现

1. 编程环境

- C/C++ & Win10 x64 & Visual Studio 2013 x86

2. 计算机配置

- 处理器：Inter(R) Core(TM) i5-4590 CPU @ 3.30GHz 3.30GHz
- 已安装的内存(RAM)：RAM 8.00GB

3. 用户界面使用说明

1) 鼠标交互

- 按下鼠标左键模型持续单方向旋转，释放左键则旋转停止。
- 按下鼠标右键可选择 models 文件夹下的*.obj 模型文件进行绘制。
- 鼠标滚轮可调整模型尺度大小，上滚放大，下滚缩小。

2) 控制台交互

- 控制台显示当前绘制的*.obj 文件信息、文件加载时间和区间扫描线算法用时等信息。

4. 数据结构说明

1) vec3f

- 三元浮点数，可表示点，向量，颜色 (RGB) 等。

2) struct_edge

- x: 当前扫描线与此边交点的 x 值。
- dx: 边的 x 变化率 (-1/k)。
- y_max: 边的上端点的 y 值。
- dy: 此边跨越的扫描线数。
- polygon_id: 此边所在的多边形的 id。

3) struct_polygon

- polygon_id: 多边形的 id。
- color: 多边形绘制后的颜色。
- normal: 多边形法向量，也是多边形所在平面 (ax + by + cz + d = 0) 中的 (a, b, c) 。
- d: 多边形所在平面 (ax + by + cz + d = 0) 中的 d。
- is_in: 扫描线运行过程中此多边形是否还在活化队列中。
- edges: 此多边形中除与 xOz 平面平行的边外的所有边。
- points: 此多边形中的所有点

4) struct_line

- x0, y0: 需要绘制的线段的左端点的二维坐标。
- x1, y1: 需要绘制的线段的右端点的二维坐标。

5) 全局变量

- lines: 需要绘制的线段
- total_polygons: 总多边形表
- total_edges: 总边表
- total_points: 总点表
- classified_edges: 分类边表

5. 算法流程

- 主要函数在 scan.cpp 中实现
- 添加虚拟背景面，无穷远处，设置颜色为黑色。
- 根据 y_max 值构造分类边表。
- 从上到下扫描。
  - 在活化边表中剔除已经扫描完成的边 (dy <= 0) 。
  - 根据分类边表，把扫描线恰好触碰到的所有边加入活化边表。
  - 将活化边表根据节点的 x 值排序。
    - 遍历当前活化边表中每两个元素之间的线段，同时将左端点所在的面标记置反，若置反后为真，则将该面加入活化面表，反之则将其从活化面表剔除。
    - 获取当前活化面表中深度值最大的面，深度值由当前线段中点的深度值表示，由平面方程计算而来。
    - 绘制该线段，颜色为该深度值最大的面。

6. 加速分析

1) 数据索引

为了减少空间消耗，程序中所有涉及到不同类之间的从属关系时都用一个整型表示，即存入 id 索引，待到使用时从 total_*表中通过索引下标获取具体值，类似于指针。经实测发现，在时间效率上也有所提升。

2) 存储结构

在存储方式上采用 vector 的连续空间存储方式，也尝试过 list 的非连续空间存储方式，经比较，即使在删除随机元素时需要 O(n)的时间复杂度，连续空间存储时间效率较高（release 模式）。

3) OpenMP

此外，还考虑过使用 OpenMP 加速，但在性能瓶颈处（从活化面表中剔除面&获取深度值最大的面）效果不佳。

- 从活化面表中剔除面：由于采用连续空间存储，单个线程删除一个元素需要 O(n)复杂度，并行无法提高效率。
- 获取深度值最大的面：在迭代更新时需要将目标最值上锁，以防止误更新，而上锁会降低并行的时间效率，最后接近串行算法。树结构求最值的并行算法一般用于大规模核心数的 GPU 并行，并不适用核心数较少的 CPU。

4) 单精度 or 双精度

使用单精度浮点数 (float) 和双精度浮点数 (double) 绘制效果差别不大，时间效率略有差别，故选择使用单精度浮点数。

5) 单缓冲 or 双缓冲

在左键旋转时调用了 glutIdleFunc() 显示动画，故使用双缓冲可使绘制和计算并行操作，起到加速效果。

7. 实验数据

模型	顶点数	面片数	加载模型耗时（ms）	区间扫描线耗时（ms）
al.obj	3618	3440	11.4	33.7
bunny.obj	34834	69451	129.2	225.7
dolphins.obj	856	1692	5.7	13.7

f-16.obj	2344	2366	9.4	9.3
male.obj	705208	705204	2160.4	374.0
rose+vase.obj	2185	3360	28.7	57.3
soccerball.obj	1760	1992	7.6	29.7
teapot.obj	530	992	2.6	9.0

表格 1 各个模型时间效率

实验中用到的 obj 模型均从网上下载而来。在 scale = 1 / 3.0 时除 bunny.obj 和 male.obj 两个模型以外，基本能做到实时交互。

8. 绘制结果

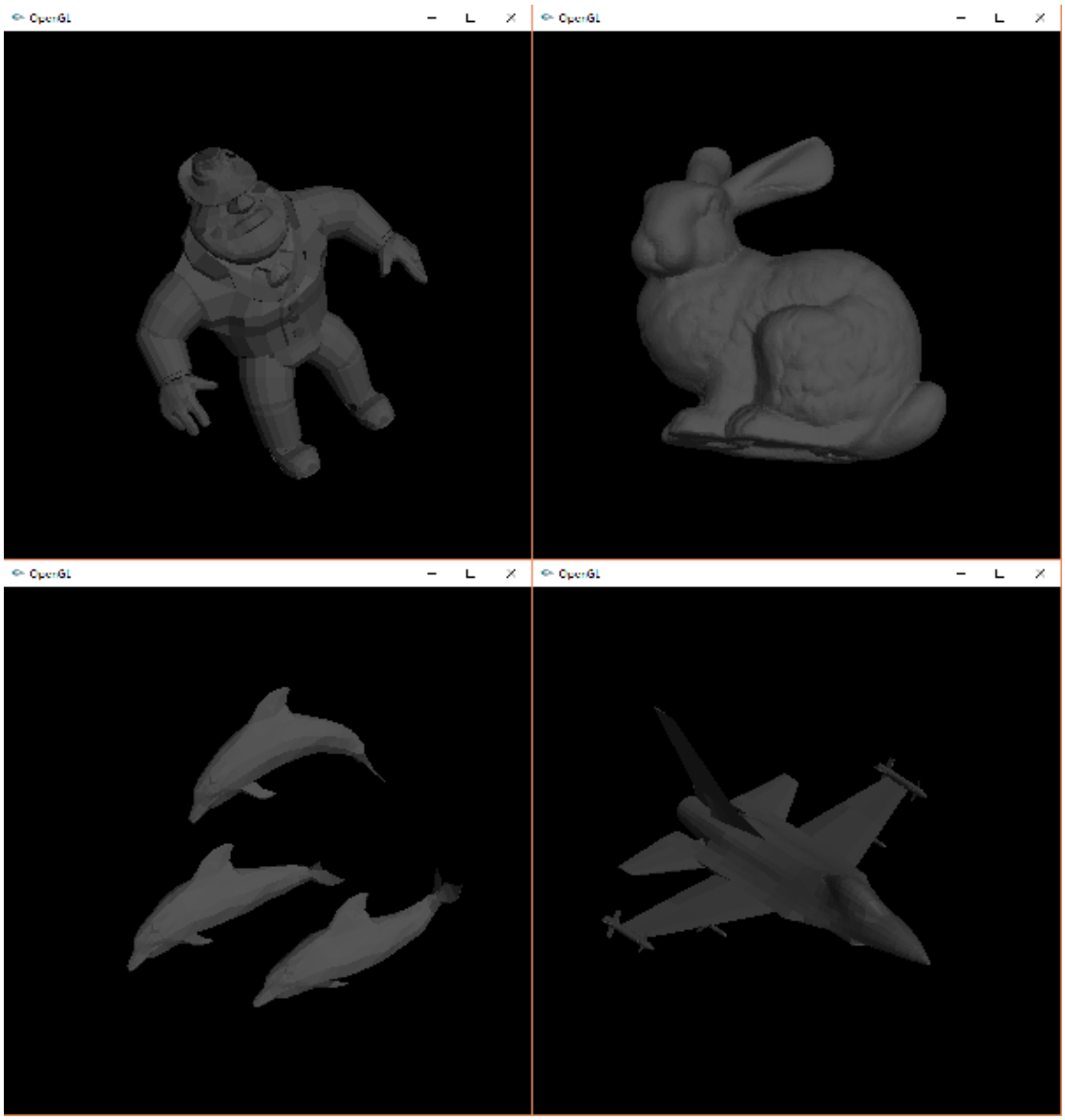

图 1 前四个模型绘制结果

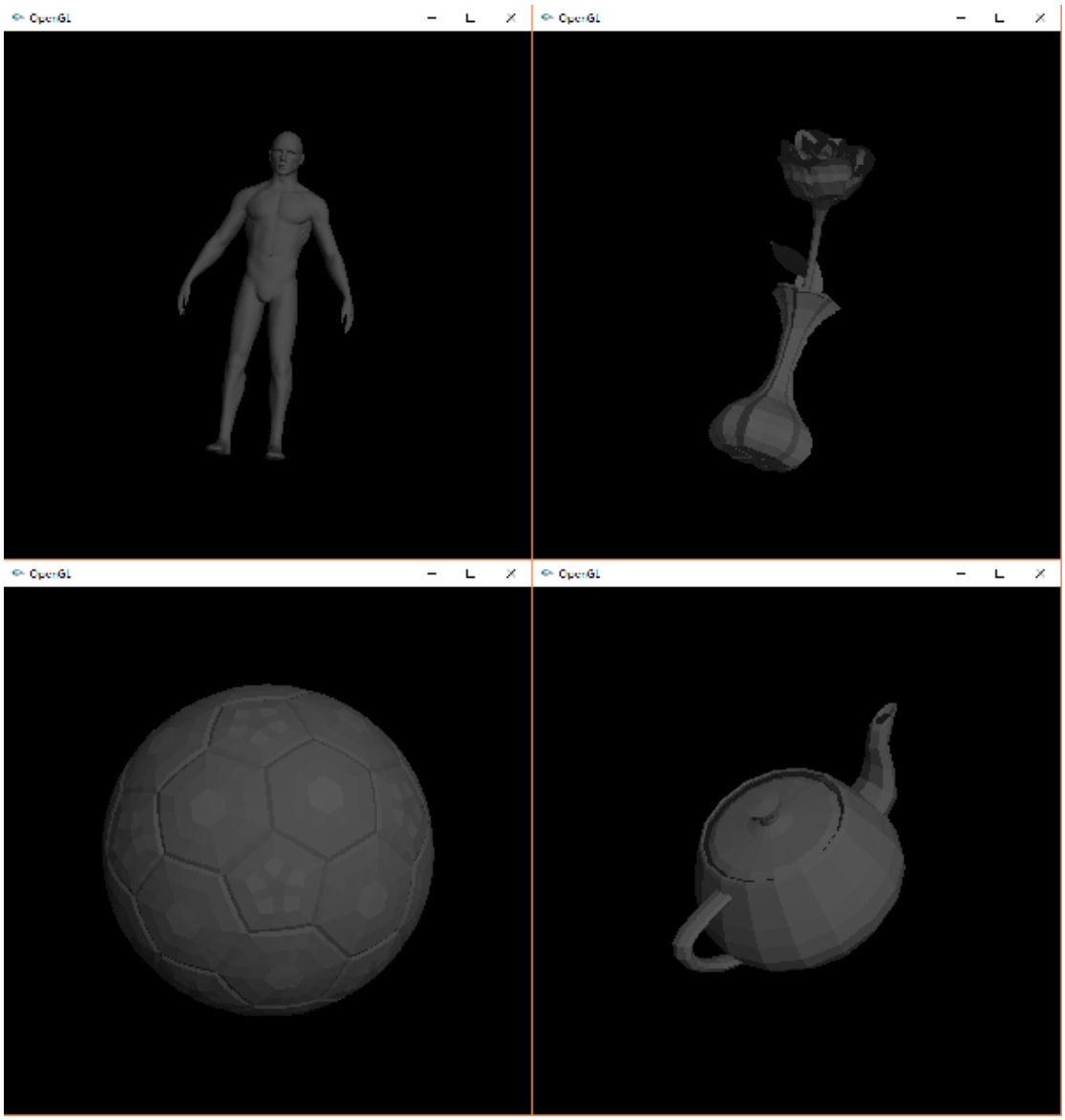

图 2 后 4 个模型绘制结果

9. Reference

[1]. (美)罗杰斯(Rogers D. F.) 著，石教英等 译. 计算机图形学的算法基础(原书第 2 版)[M]. 北京:机械工业出版社, 2002. 308-319

[2]. https://github.com/sccbhxc/ScanLineZBuffer

[3]. http://blog.csdn.net/th_gsb/article/details/50999307

[4]. https://free3d.com/3d-models/all/1/obj

[5]. http://blog.csdn.net/timzc/article/details/6290004

[6]. http://blog.csdn.net/u014030117/article/details/46427599

[7]. https://stackoverflow.com/questions/14378/using-the-mouse-scrollwheel-in-glut

[8]. http://blog.csdn.net/szchtx/article/details/8628265

[9]. http://blog.csdn.net/a_big_pig/article/details/51039581