层次 ZBUFFER 实验报告

李广林
CAD&CG State Key Lab
22121070 liguanglin@zju.edu.cn

2022 年 1 月 15 日

1 实验要求与原理（参照课件）

1.1 实验要求

- 实现层次 z-buffer 与场景八叉树
- 分别对比简单模式和完整模式与扫描先 z-buffer 算法的加速比

1.2 实验原理

1.2.1 八叉树

通过八叉树管理模型的三角形面片，当八叉树节点的空间被对应区域的 zbuffer 完全遮挡，则掠过该节点。在构建八叉树时，对于每个节点，如果某个面片完全在其某个儿子节点中，那么将这个面片细分到该儿子节点中，否则留在本节点中。八叉树具体实现请参照代码 OctTree.cpp。

1.2.2 层次 zbuffer

层次 zbuffer 相当于四叉树，对 zbuffer 的二维坐标进行划分，用四叉树来维护某个区间划分内的最远深度，这样可以在查询八叉树对应二维坐标区间的最大深度值时实现 log 效率快速的查找。具体实现请参考代码 zBuffer.cpp。

1.2.3 扫描线更新 zbuffer

在这里，与冯老师课件中提到的实现方法不同。我在这里具体实现是对于当前八叉树节点中的每个面片，直接在 zBuffer 中用扫描线的做法更新 zBuffer，没有使用教学中提到了活化边链表等数据结构。对于每个三角形，我们需要尽可能快的找到它对应的 zbuffer 中二维整数点坐标，在程序中采用了如图1所示的扫描算法，首先求出 y 相对于 x 的导数，然后每次边界上下一个 x 对应 y 值是在上一个的基础上加上这个的导数，实现了在循环中只有加法操作的枚举三角形内坐标的算法。对于深度值的差分也是这样。最后，在更新完每个点的深度值之后，再去更新层次 zbuffer 中每个非叶子节点的最大深度值。在这里仍有一个优化的地方，如果在扫描三角形面片的时候，从深度最大的那个点开始扫描，那么总体更新层次 zbuffer 这个四叉树的效率将会提升，大约会提升 log 倍。具体实现在 zBuffer.cpp 中的 update 函数。同样，这里如果使用区间扫描线算法实现也会快很多。

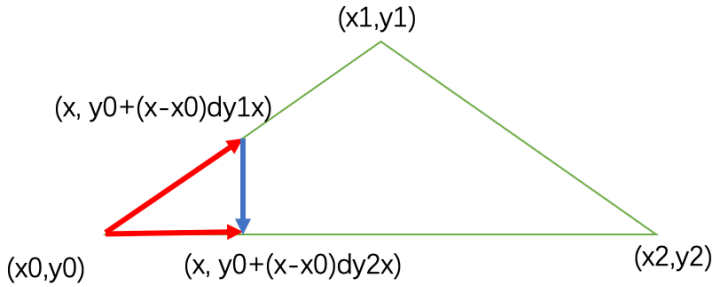

图 1: scanline 示例。

2 代码实现

请参考代码中的注释以及说明文档。

3 实验结果

3.1 效率对比

在实验中使用了 bunny、airplane、teapot 三个 obj 模型进行测试，统计了其每帧渲染所需时间以及有无八叉树或层次 zbuffer 的加速比率，如下表所示，与扫描线算法的加速比对比如表所示。

表 1: 有无八叉树、层次 zbuffer 的扫描线算法单帧绘制时间 (ms) 与面片剪枝数对比.

Model	完整模式	无八叉树	无层次 zbuffer	只有扫描线	剪枝面片数	剪枝面片百分比
airplane	24.2	28.6	28.3	33.1	18447	17.3%
bunny	42.1	47.6	70.4	78.8	11146	16.0%
teapot	29.0	29.8	58.9	73.7	108	10.9%

3.2 实验效果截图展示

如图2、3、4所示。

4 结论与展望

在本实验中，完成了扫描线 zbuffer 以及层次化 zbuffer 的数据结构，并在实验结果中证明了八叉树与四叉树数据结构对绘制的加速作用。除此之外仍然存在改进的地方，比如可以将区间扫描线加入绘制的每一步，从而提高速度，但这并不在本实验的要求范围内。

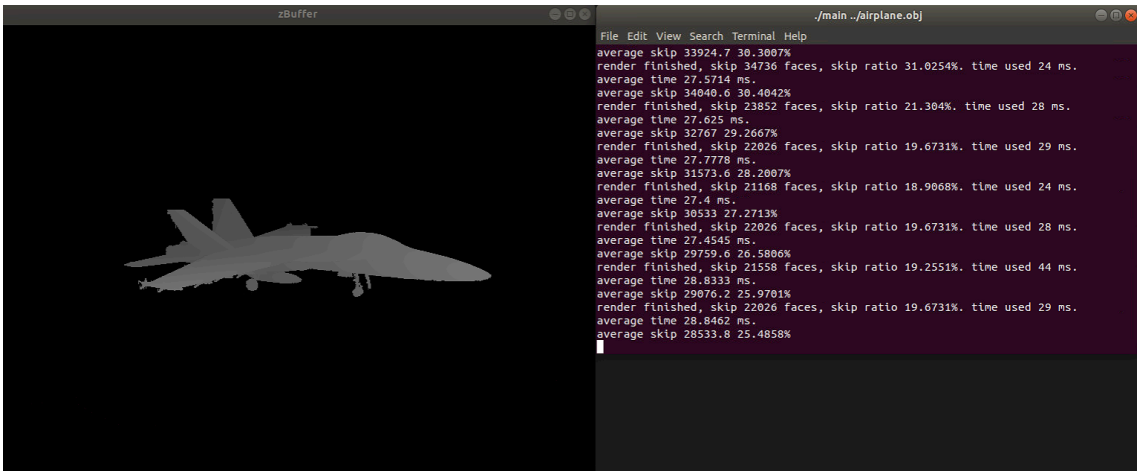

图 2: airplane 绘制结果。

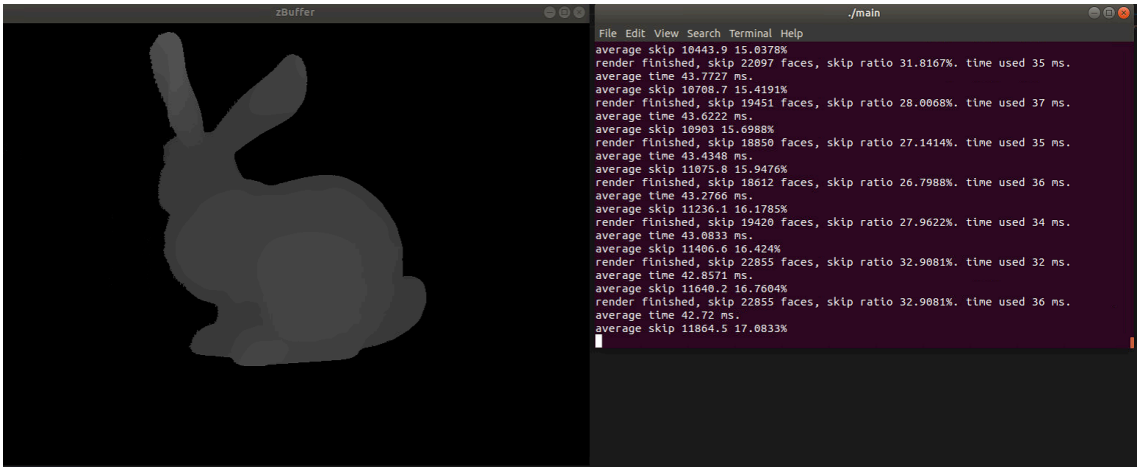

图 3: bunny 绘制结果。

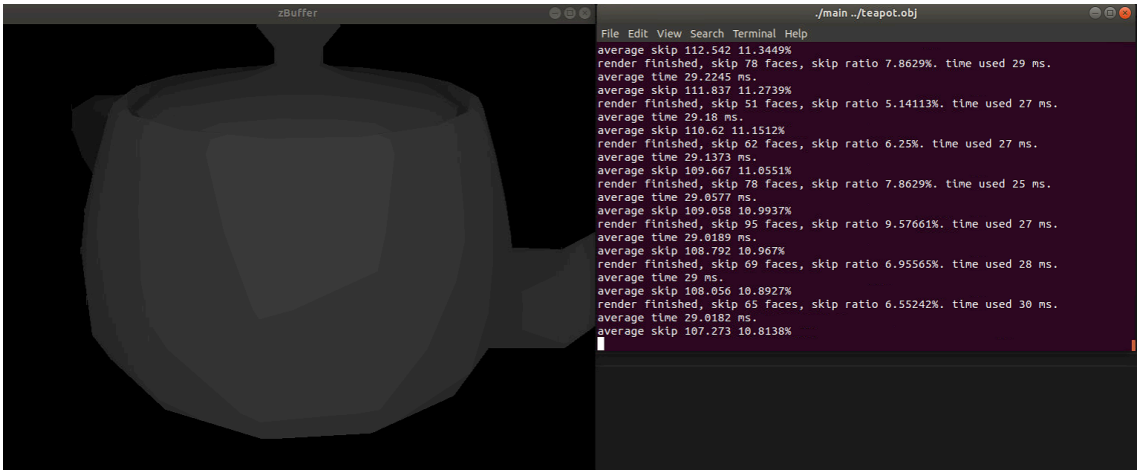

图 4: teapot 绘制结果。