doi: 10.7690/bgzdh.2016.05.018

基于 GPU 的光线追踪算法

赵建伟，班 钰，王 朝，闫双胜
（中国矿业大学矿山机器人中心实验室，北京 100083）

摘要：为在个人 PC 上实现实时光线追踪技术，设计并实现一种基于 GPU 的光线追踪算法。介绍光线追踪原理，深入分析实时光线追踪的特点，研究实时光线追踪中最困难的空间加速结构构建和搜索问题，并进行算法优化和实验。结果表明：该算法可得到非常逼真的图形效果，为人们提供深度信息，实现二维图像中的三维效果。在仿真模拟、几何造型、广告影视、指挥控制及科学计算的可视化等领域都得到广泛应用。

关键词：光线跟踪；加速结构；着色语言；GPU
中图分类号：TP301.6 **文献标志码**：A

Ray Tracing Algorithm Based on GPU

Zhao Jianwei, Ban Yu, Wang Zhao, Yan Shuangsheng
(*The Robot Center Laboratory of Mine*, *China University of Mining & Technology*, *Beijing* 100083, *China*)

Abstract: A ray tracing algorithm based on GPU is designed and implemented to realize real-time ray tracing technology on personal PC. This paper introduces the principle of ray tracing, analyzes the characteristics of real-time ray tracing, and studies the construction of the most difficult space accelerating structure and the search problem of the real time ray tracing. The results show that the algorithm can get a very realistic graphics effect, providing people with depth information, and realize the three-dimensional effect in 2D images. It has been widely used in the fields of simulation, geometric modeling, advertising, film, television, command and control, and the visualization of scientific computing.

Keywords: ray tracing; acceleration structure; shading language; GPU

0 引言

光线追踪算法是一种利用几何光学原理生成三维图形的成像算法。它模拟在空中传播过程，与空间物体发生碰撞后，渲染出不同颜色，明暗效果。由于这种方法依靠光线，物体的物理性质及其相互作用建模，利用多种观感：透明感，明暗感，反射敢和纹理等，增强图形真实感，为人们提供深度信息，实现了二维图像中的三维效果；因此，可以得到非常逼真的图形效果。在仿真模拟、几何造型、广告应是、指挥控制及科学计算的可视化等许多领域都得到广泛应用。

1 光线追踪原理

光线追踪技术模拟人眼的视觉成像原理：物体之所以被看见是因为从物体出发的光进入了眼睛。利用光路可逆原理，笔者认为从眼睛发出了一束光并照射到了物体上，并经过若干次的反射和折射又回到了光源。图 1 展示了光线追踪技术的成像原理。人眼以一定的视角向像素平面看过去，所看到的场景是这样形成的：一条光线以一定的视角穿过像素平面，照射到物体 a 上的 P_1 处，一部分光线经反射直接回到光源；一部分光线经反射照射到物体 c 上的 P_2 处。P_2 处的一部分光线经过反射穿过物体 b 回到光源，一部分光线经过 c 的折射照射到物体 d，并经 d 反射回到光源。这是从光路可逆的原理去解释光线的传播。实际上，从光源发出的光线一部分直接找到了 P_1 处；一部分直接照到了物体 b 上，由于 b 不透明，这部分光线的延长线照射到了物体 c 上，笔者称这部分延长的光线为阴影光；一部分直接照射到了物体 d 上，经物体 d 的反射照射到物体 c 上，再经过 c 的折射照射到物体 a 上。物体 a 上 P_1 的显示是根据物体 a 的材质和光的强度决定的，其中光的强度由照射到 P_1 点的所有光线叠加而成，具体来讲是由部分反射光和折射光叠加决定的；同样的道理，P_2 的显示是由照射到 P_2 处阴影光、折射光和 c 的材质决定的[1]。

2 光线追踪算法

根据光线追踪原理的描述，光线追踪算法的输入是像素平面 pixelPlane 和场景 scene，输出是经过光线追踪渲染的场景 renderedScene；首先对场景进行预处理，然后对每一个像素产生若干条光线，接

收稿日期：2016-01-13；修回日期：2016-02-28
基金项目：中国博士后科学基金(2012M510424)；中央高校基金科研业务费专项基金项目(800015FH)
作者简介：赵建伟(1979—)，男，内蒙古人，博士后，硕士生导师，从事机器人技术研究。

着光线搜索预处理后的场景找到第一个与场景相交的点，最后根据场景材质和交点信息计算像素点的颜色并显示在像素平面上[2]。处理前后效果如图 2 所示。该算法展示了光线追踪技术的 3 个特点：

1) 场景的预处理直接影响到光线搜索的性能；实际上场景的预处理主要完成的工作就是为光线搜索构建高质量空间加速结构，包括 KD-tree 和 BVH。

2) 光线搜索实际上只需找到第一个与场景相交点，即找到能被看见的物体；因此，光线追踪仅渲染能被看到的物体。

3) 每一条光线都相互独立的完成搜索和渲染任务，并行性是光线追踪的本质属性。

Algorithm 2.1: Ray Tracing
1: Procedure ray Tracing(in: pixelPlane,scene; out: rendedScene)
2: Begin
3: preprocessing(scene);
4: For each pixel in pixelPlane
5: generateRays(pixel,rays);
6: traversalRays(scene,rays,firstIntersections);
7: rendering(scene,firstIntersections,pixel);
8: End

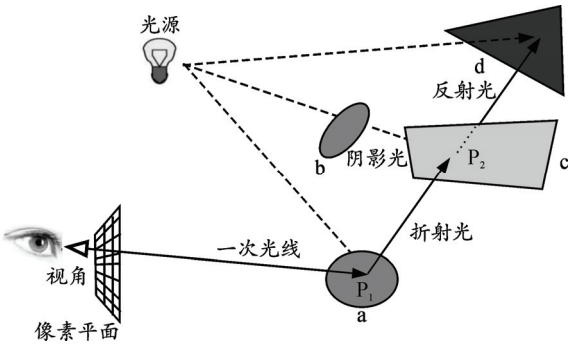


图 1　光线追踪原理

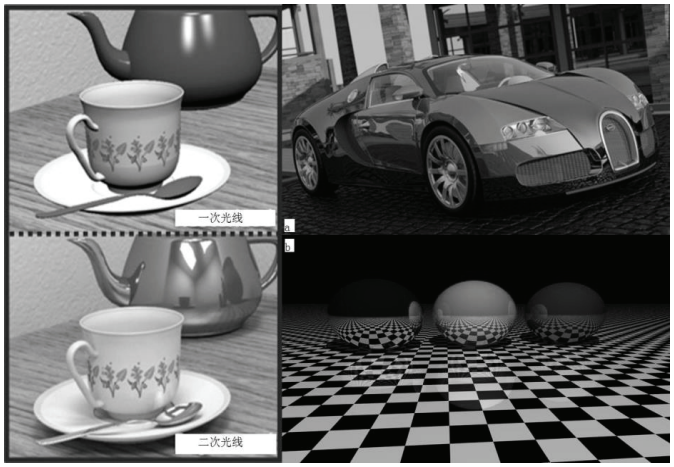
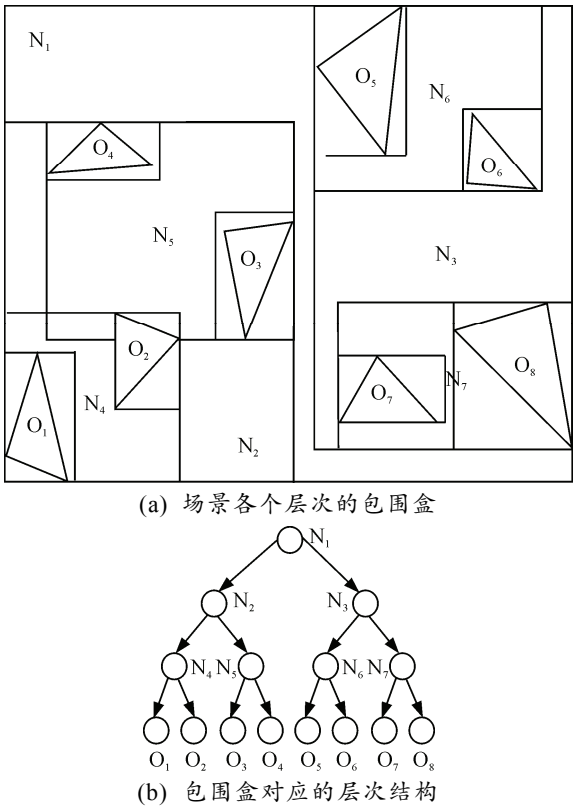


图 2　处理前与处理后光线效果图

3　空间加速结构

3.1　空间加速结构 BVH

空间加速结构(bounding volume hierarchy，BVH)就是将空间中的物体按照包围盒进行层次结构的组织，这里的层次结构通常是指树形结构。这种层次结构可定义如下：叶节点的包围盒(bounding box)是能包含该叶节点对应的所有物体的最小包围盒，其余节点是能包含孩子节点的最小包围盒。图 3 描述了二维场景的 BVH。图中 O_1～O_8 是叶子节点，它们的包围盒就是各自物体的最小包围盒；其他节点，如 N_4 是包围孩子节点 O_1 和 O_2 的最小包围盒，N_3 是包围孩子节点 N_6 和 N_7 的最小包围盒。由 BVH 的定义可以看出：BVH 的构建通常采用自底向上的方式，这种方式的第一步通常是将场景的三角形放在若干的巢里形成叶子节点，然后再一层一层的往上形成树形结构。同时高质量的 BVH 构建总是和 SAH 的方法紧密结合在一起。



(a) 场景各个层次的包围盒

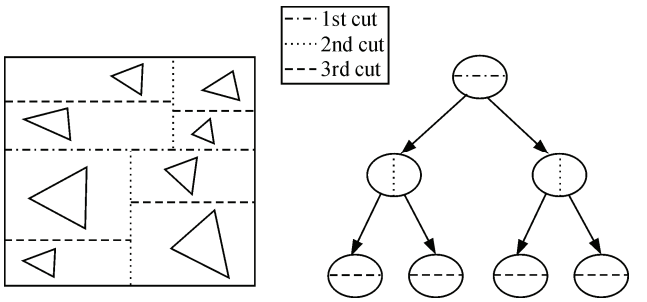
(b) 包围盒对应的层次结构

图 3　BVH 示意图

3.2　空间加速结构 KD-tree

KD-tree 是多维空间树形结构的简称。它将空间的物体按照划分平面组织成树形结构。图 4 展示了一个二维 KD-tree 的构建过程，图中“1st cut”表示第一次划分平面所在的位置；“2st cut”表示

第二次划分平面所在的位置，依次类推；由于每次划分都产生 2 个新的子空间，第二次划分实际上需要 2 个划分平面，第三次划分需要 3 个划分平面，依次类推，而 KD-tree 的层次关系正是这样建立的：新产生的 2 个子空间是本次划分平面的左、右 2 个孩子。KD-tree 的构建过程通常是自顶向下的。基于 SAH 的 KD-tree 构建通常从顶层出发逐层选择最优的候选平面进行划分[3]。



(a) 场景划分结果 (b) 形成的树形结构 KD-tree

图 4 KD-tree 的构建

4 基于 GPU 平台上的光线追踪技术

遍历开始之前，需要开辟一定空间用于存放：1) 光线的几何数据；2) 光线所需访问的加速结构(某个部分或全部)；3) 被加速结构索引的几何体数据；4) 运行时所需要的栈。遍历过程采用数据并行方式，将一条光线的遍历过程定义为基本任务单元。每个 GPU 硬件线程同一时刻负责一条光线的处理。因此，一个单指令多线程(single instruction multiple threads，SIMT)处理器同一时刻处理 32 条光线。每个线程处理过程都遵循基于栈的遍历算法。笔者在前面提到过，执行流程的分支会导致 SIMT 利用率的降低。对于遍历算法而言，具体体现在同一向量中各线程在内节点遍历步骤中以及在叶节点访问步骤中的执行时间差异[4]。为了表达简洁起见，笔者将内节点遍历步骤称为 T 操作(Traverse)，将叶节点访问步骤称为 I 操作(Intersect)。T 操作的分支出现在节点类型的判定上。当一个线程遇到叶节点时，就需要结束 T 操作而转如 I 操作。但是，由于同步执行的约束，如果向量中还存在某个线程尚未完成 I 操作，则较早完成 I 操作的线程就必须等待；因此，整个向量消耗在 I 操作上的时间，由其中最慢的线程决定。笔者将从栈中弹出的节点称为起始点，将 T 操作以起始点为输入所找到的叶节点称为终点。线程在 T 操作上的差异主要由从起始点到终点的路径上所包含的节点数量的差异决定。类似地，各线程在 I 操作中可能遇到不同数量的几何体；因此在执行时间上也会存在差异[5]，如图 5 所示。

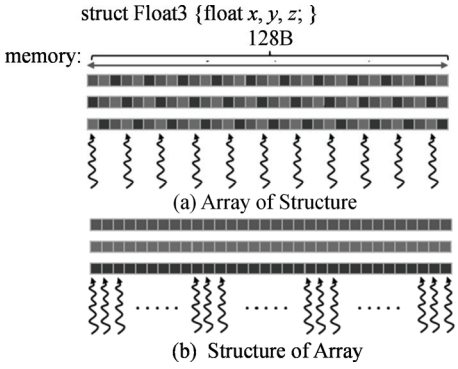


图 5 结构的数组和数组的结构对比

5 优化算法与实验结果

性能优化始终围绕 3 个基本策略：最大化并行执行以获得最大利用率、优化存储器使用以获得最大存储器吞吐量、优化指令使用获得最大指令吞吐量。对于应用的某个特定部分，什么策略会产生最好的性能收益依赖该部分的性能限制。

在原本的遍历算法中，场景节点存储在 Global memory 中。众所周知，Global memory 没有 Cache，访问速度很慢，Shared memory 访问速度虽然很快，但是容量很小。对于较大的数组，将其绑定至 texture memory 是个不错的选择。Texture memory 有 cache，而且容量很大[6]。和常量内存一样，纹理内存是另一种类型的只读内存，在特定的访问模式中，纹理内存同样能够提升性能并减少内存流量。纹理内存缓存在芯片上；因此在某些情况中，它能够减少对内存的请求并提供更高效的内存带宽。纹理缓存是专门为那些在内存访问模式中存在大量空间局部性(spatial locality)的图形应用程序而设计的。在某个计算应用程序中，这意味着一个线程读取的位置可能与邻近线程的读取位置“非常接近”，如图 6。

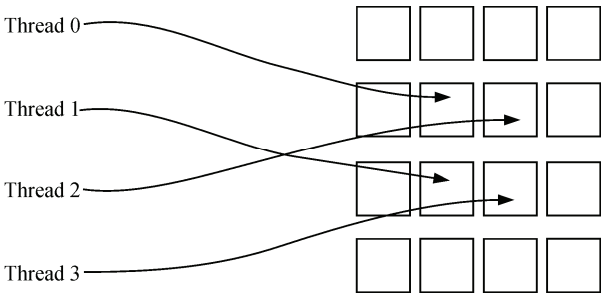


图 6 纹理内存的存储模式

这样能使程序的速度提高到之前两倍的效率。

6 结束语

光线追踪简单优雅的成像原理和高质量超逼真

的显示效果，被视为未来改善图形显示效果最具竞争力的技术之一。随着个人 PC 性能的快速提升，尤其是 GPU 通用计算的出现，人们渴望在个人 PC 上实现实时光线追踪技术。笔者深入分析了实时光线追踪的特点，围绕实时光线追踪中最困难的空间加速结构构建和搜索问题展开研究，至少在 4 方面推进了实时光线追踪技术，包含 2 个重要的创新点，而且这些方法不仅限于光线追踪应用，而且还具有一定的普适性。

参考文献：

[1] Peter Shirley, Steve Marschner, Michael Ashikhmin, et al. Fundamentals of Computer Graphics[M]. Third Edition. Boca Raton: CRC Press, 2009: 12–13.

[2] Andrew S. Glassner. An Introduction to Ray Tracing[M]. New York: Academic Press, 1989: 15–16.

[3] Pharr M, Humphreys G. Physically Based Rendering: From Theory to Implementations[M]. Second Edition. Holland: Elsevier, 2010: 34–37.

[4] 李强云，武昕伟. 基于子空间的正交匹配追踪算法[J]. 四川兵工学报, 2015, 36(6): 113.

[5] AILA, T, AND LAINE, S. 2009. Understanding the efficiency of ray traversal on gpus[C]//In Proceedings of the Conference on High Performance Graphics, New York: Lucax, 2010: 145–149.

[6] IMAGINATION.2010. Power VR. Series 5XT. GPU [EB/OL]. http:// www. imgtec. com/powervr/ series5xt.asp.

****

（上接第 55 页）

参考文献：

[1] 吕遐东，李蕾，李维林，等. 舰艇作战系统雷达统一对准模型及其误差分析[J]. 中国舰船研究，2009，12(6): 53–57.

[2] 王正湖. 舰船建造中作战系统对准精度控制的研究[D]. 大连：大连理工大学, 2014.

[3] 施宗伟，秦克，邹立. 对舰载作战系统坞内对准的探讨[J]. 舰船工程研究, 2001, 12(2): 12–14.

[4] 王伯文. 舰用作战系统零位对准方法[J]. 海军装备，2009, 8(12): 34–35.

[5] 崔琦，郑剑飞. 舰艇作战系统动态对准技术探讨[J]. 舰船工程研究, 2010, 12(4): 29–33.

[6] 马曦，冯浩，吕遐东，等. 一种舰载作战系统系泊对准模型约定真值预估方法[J]. 火力与指挥控制，2011, 36(7): 159–162.

[7] 孙世岩，张国栋. 舰艇摇摆对雷达目标跟踪精度的影响分析[J]. 火力与指挥控制, 2011, 36(5): 75–78.

[8] 吴小强，窦林涛，初阳，等. 舰艇姿态对雷达测量误差的影响分析[J]. 指挥控制与仿真, 2011, 33(6): 102–105.

[9] 李洪梅，潘江怀，何佳洲，等. 舰载雷达探测误差传递与灵敏度分析[J]. 数据采集与处理，2012，27(4): 474–479.

[10] 桑德一，赵建军，杨利斌. 航母运动对着舰引导雷达精度的影响[J]. 中国舰船研究, 2014, 9(6): 8–13.

[11] 《现代舰艇火控系统》编写组. 现代舰艇火控系统[M]. 北京：国防工业出版社, 2008: 17–21.

[12] 曹正才. 舰载雷达常用稳定方式坐标变换[J]. 雷达与对抗, 2010, 30(1): 47–52.