作品信息

作品名称	基于 KD 树深度遍历的交互式光线跟踪算法
待求解的问题	
本课题利用 GPU 的高度并行计算能力，充分优化了光线跟踪算法。解决了以下几个问题： 1. 三角形与射线求交算法 2. 基于 KD 树遍历的光线跟踪算法，在 GPU 内核中的基于深度优先的 KD 树遍历算法。 3. 折射反射以及阴影效果，并且支持纹理	
使用的算法	
本课题提出了一种基于位操作的虚拟栈，从而改进了 GPU 内核中遍历 KD 树的行为，使其进行深度优先的搜索。这种搜索方式，相对于 kd-restart 以及 ropes 等前人的方法有一定改进，并且效率很高。 除了虚拟栈的算法以外，还应用了光线跟踪中很多经典的算法。	
编程和优化技巧	
1. 利用 32 位整数的位操作实现基于深度优先的 KD 树遍历 2. 代码文档规范，可读性较强。	
与传统的 CPU 开发的程序相比达到的加速比	
本课题的算法分别实现了 CPU 以及 CUDA GPU 两个版本，后者相对于前者的加速高达 78 倍，有了非常大的改进。对于官方提供的简单的场景，算法可以达到实时的效果。而对于含有 87 万三角形的 stanford dragon 模型，并带有反射属性，本文的算法可以达到每秒 2-3 帧的交互式性能。	
补充信息（可以放入任何与此作品相关的说明信息）	1. 本课题的 KD 树是离线创建的，存储在文件中，所以 xml 场景脚本中的信息是不可以随便更改的，否则 KD 树无法更新，将导致程序崩溃。 2. 由于资源文件较大，所以读取时间可能会稍微长一点，请耐心等待。 3. 本课题的算法可以达到交互式的性能，但是由于编程接口原因，导致算法实际渲染的交互并不是特别流畅，其主要原因是因为光线跟踪生成的图片是利用 MFC 的消息循环机制和 GDI 来进行动态渲染的，而这两部分效率都很低，所以我们感受到的实际的 FPS 可能要小于算法所真正能达到的性能，但这部分与算法本身是无关的。 4. 本课题的代码均为作者自己开发，没有任何抄袭。 5. 本课题在 32 位操作系统的环境下进行开发，建议 build32 位可执行程序。如果用 64 位的 sdk 以及 toolkit 进行编译，请在一些设置的地方相应调整。

提交的工作如下：

1. 模板文件 (intro.pdf)
2. 源程序 [包含 CPU 与 GPU 两个版本] (Src)
3. 编译好的可执行程序 (Bin)
4. 说明文档 (Doc->doc.pdf)
5. 支持该程序运行的其他程序和素材等。[需要 DirectX Runtime 以及 CUDA runtime 3.0 或更高版本，由于文件大小的关系，请麻烦在微软和 Nvidia 的官方网站上下载。程序运行时需要的模型及纹理资源都在相应目录下放置好了。]
6. 作品界面截图 (pic.jpg)
7. 本课题的光线跟踪算法生成的图片 (gallery)
8. 程序运行时演示 (video.wmv)