Computer Science and Application 计算机科学与应用, 2018, 8(1), 16-20
Published Online January 2018 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2018.81003

Hans汉斯

Research on Distributed Real-Time Rendering Based on Cloud Computing

Changyin Xing, Kanjie Zhang, Xuan Liu

Jilin Animation Institute, Changchun Jilin
Email: 63080180@qq.com

Received: Jan. 2nd, 2018; accepted: Jan. 15th, 2018; published: Jan. 25th, 2018

Abstract

With the continuous development of computer technology, 3D animation industry is in a vigorous development. As the rendering process takes a lot of time, it also affects the animation production efficiency. In order to solve this bottleneck, we can adopt distributed rendering. Due to the similarities between distributed computing and cloud computing, cloud computing technology can be combined with distributed computing into a cloud computing rendering platform. This paper mainly analyzes and studies the distributed real-time rendering platform based on cloud computing.

Keywords

Cloud Computing, Render, Distributed Rendering, Real-Time Rendering

基于云计算的分布式实时渲染研究

行长印，张堪杰，刘　旋

吉林动画学院，吉林　长春
Email: 63080180@qq.com

收稿日期：2018年1月2日；录用日期：2018年1月15日；发布日期：2018年1月25日

摘　要

伴随着计算机技术的不断发展，3D动漫行业正处于蓬勃发展的态势。由于渲染过程需要消耗大量的时间，这也影响着动漫的制作效率。为了解决这一瓶颈，可以采用分布式渲染的方式。由于分布式计算与云计算具有相似性，云计算技术可以与分布式运算集合而成云计算渲染平台。本文主要对基于云计算的分布式实时渲染平台进行了分析与研究。

文章引用: 行长印, 张堪杰, 刘旋. 基于云计算的分布式实时渲染研究[J]. 计算机科学与应用, 2018, 8(1): 16-20.
DOI: 10.12677/csa.2018.81003

关键词

云计算，渲染，分布式渲染，实时渲染

1. 引言

伴随着信息技术产业的迅速发展，3D 动漫行业正处于蓬勃发展的态势。由于大众对 3D 画面的细腻程度以及画面质量要求越来越高，渲染过程中产生的计算量与日俱增。计算能力的短缺成为动画渲染的最大瓶颈[1]。为了解决这一瓶颈问题，分布式渲染是一个很好的方案。分布式渲染系统将渲染任务分成多个子任务，分配到系统中的渲染节点，进行渲染操作，可以大大提高渲染速度[2]。由于云计算与分布式计算的相似性，基于云计算的分布式渲染平台应运而生。本文的主要目的是在分布式云渲染平台的基础上，对于实现实时渲染的关键技术进行分析与研究。

2. 相关概念

2.1. 云计算

云计算是在分布式计算理论与网格计算理论的基础之上，进化得到的公共计算服务。它并不将计算分布于本地或远程服务器，而是分布到分布式网络，用户不需知晓计算的细节，只需要将自己的计算机与“云”相连，就能够得到高质量的计算服务[3]。

由于云计算具有超强的计算能力，以及大规模存储等优势，将它与分布式渲染结合而成的基于云计算的分布式渲染平台可以有效缩短渲染时间，提高渲染效率。

2.2. 实时渲染

随着科技的发展，在虚拟现实、三维游戏等领域中，越来越多地应用了实时渲染技术。实时渲染指的是根据图形学算法将三维数据绘制到二维位图之中，并将这些位图实时显示。它的本质就是对图像数据的实时计算和输出，要求在短时间内渲染出一张图片，并显示出来，同时渲染并显示下一张图片。这种技术仅仅依靠 CPU 是没办法完成的，还需要依靠显卡实现。

由于单渲染系统计算不了渲染过程中的巨大数据量，如果想实现对 3D 场景的实时渲染，最好采用分布式渲染系统。由于在一般的分布式渲染系统中，不需要考虑对场景的优化，因此同一场景在不同的系统当中会渲染出一样的效果。而分布式实时渲染系统则与之不同，要想达到好的实时渲染效果，就要采取一定技术，进行场景优化，以降低计算量，提升渲染效率与质量。

3. 关键技术

3.1. 并行算法 sort-first

根据在物象的转换过程中，拆分渲染任务的位置不同，可将并行渲染系统分为三种。sort-first 是在几何变换之前就重新分配图元；sort-middle 算法对屏幕空间的几何元素重新分配是在几何变换与光栅化间进行；sort-last 则是在光栅化的最后重新对像素进行分配[4]。

sort-first 是比较常见的一种并行算法，它首先将要输出的图像划分成为多个不重叠的区域，每块区域由它专门的渲染节点负责渲染，也就是要预分配每条流水线的渲染任务。在这种架构中，每一个渲染节点都相当于一条完整的渲染流水线，不需要在几何变换和光栅化过程中传递几何信息，也不用在最终进行深度信息的合并。

该算法的流程如图 1 所示。首先在分布图元时，需要计算图元覆盖的区域，这种计算被称为“预变换”。然后对分配好的各图元进行几何处理和光栅化操作。最后将渲染好的子图像拼接为最终图像。

3.2. 场景优化

在实时渲染过程中，对场景进行优化，降低场景的复杂程度，可以有效提高渲染效率。本文选择的是 Level of Detail (LOD)细节层次模型。

LOD 算法主要是根据三维场景中的物体的清晰度不同，生成多个具有不同层次细节的版本，从而实现对清晰度分层管理的目的。在实时渲染的进程中，依据试点不同，选取合适的 LOD 模型，可以在不损伤细节的前提下，降低场景复杂程度，提高渲染速度，并加快场景的显示速率。

为降低场景的复杂程度，首先利用八叉树算法对三维场景进行分析与管理。接着针对不同物体，以及不同需求，选择合适的 LOD 模型，例如剔除法，即找到场景中观察者无论如何都看不到的图元，并将其剔除，以加快渲染速度。另一种方法是根据观察者与场景中物体的距离进行选择，距离远的可以采用比较粗糙的细节层次。还可以依据物体的运动速度选出适应的模型，运动较快的物体，在屏幕上显示的细节比较模糊，可以使用较粗糙的细节层次表现出来。

3.3. 实时光线追踪

在计算机图形学领域里，光线追踪是十分经典的算法。光线跟踪的过程就是通过对现实世界当中的成像原理进行模拟，从而对每像素着色的过程。在光线追踪过程中，主要考虑光线传递的方法、中止位置以及光线所得的着色值等问题。

当光照射到物体表面时，会产生光的反射、折射、吸收及透射等等现象，要想处理所有状况，就要选出合适的光照模型。我们可以选用简单光照模型，即 Phong 光照模型来简化问题。该光照模型可以依据物体的材质，在场景及物体交汇的交点处计算出镜面反射(specular)值和漫反射(diffuse)值，从而确定着色值。

传统的光线跟踪方案只能通过串行的方式解决光线产生、光线和场景的碰撞检测等问题。由于光

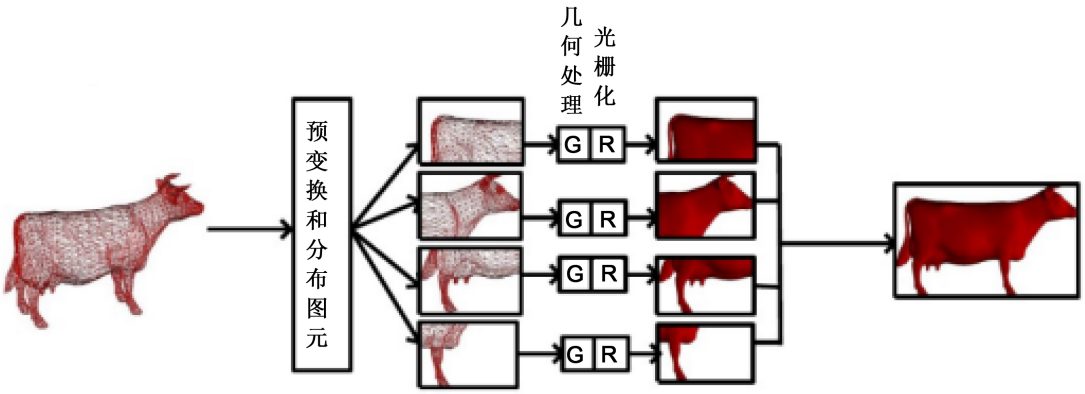

Figure 1. Sort-first algorithm
图 1. Sort-first 算法

线追踪的过程当中，不需要考虑像素间的依赖问题，因此可以利用这一特性，设计一种并行的光线追踪方案，该方案的算法结构如图 2 所示，其中初始参数包含观察方向、视角等，并利用这些参数生成初始光线：

并行光线跟踪的实现主要由以下几个部分组成：

1) 针对光线的产生问题，对每个光线都生成相对独立的基本参数。

2) 针对光线与场景碰撞检测问题，因为在光线着色的过程中，像素之间没有依赖关系，所以对每一条光线可以进行并行地着色化。

实现以上算法后，可以针对不同的需要对细节进行再度优化，从而提升算法执行的效率。例如：在创建 KD-Tree 的时候，可以先使用均匀分割的方式，直到三角形数目达到规定数目后再利用最优面的分割方式，通过这种方案可以有效地提高创建 KD-Tree 的效率。我们还可以把场景当中的一些原始几何数据、KD-Tree 等信息进行分类，并存储于不同的纹理之中。通过这种方案不仅可以利用数据的打包存放减少对同种数据资源的重复访问，还可以利用纹理提供的虚拟缓存，从而降低了数据访问的延迟，提升了访问效率。

使用这种光线追踪方案对场景进行渲染，渲染结果如图 3、图 4 所示。

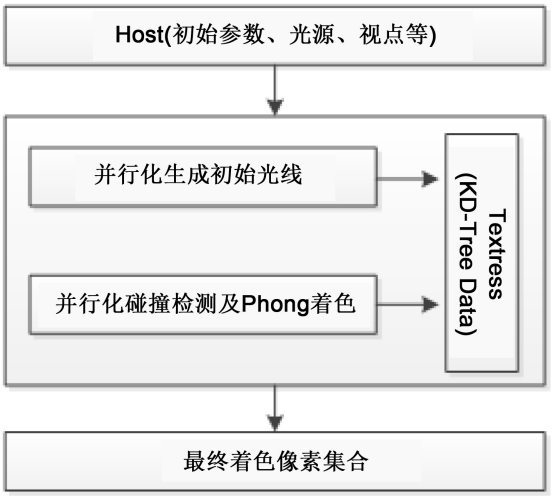

Figure 2. Structure chart of parallel ray tracing algorithm
图 2. 并行的光线追踪算法结构

Figure 3. Rendering results of character model
图 3. 人物模型渲染结果

Figure 4. Rendering results of scene
图 4. 场景渲染结果

通过对渲染结果分析可知，使用这种实时光线追踪算法，可以有效地提高渲染的质量，使场景更富真实感，从而达到更好的实时渲染效果。

4. 总结

本文主要是在基于云计算的分布式渲染平台的基础之上，对实时渲染进行了研究。通过分析实时渲染过程中需要的关键技术，提出了实现实时渲染的相关技术和算法。利用这些技术，可以有效降低渲染时长，加快图像显示速率，达到更好的实时渲染效果。

基金项目

基于云计算的下一代动画实时渲染技术研究(吉教科合字【2016】第 521 号)。

参考文献 (References)

[1] 耿蕊. 中国动漫产业发展的危机与转机[J]. 湖南社会科学, 2010(1): 216-219.

[2] 李树声. 网络集群渲染在 3D 动画制作中的应用[J]. 广播与电视技术, 2004, 31(9): 63-64.

[3] 程克非, 罗江华, 兰义富. 云计算基础教程[M]. 北京: 人民邮电出版社, 2013.

[4] 史银雪, 陈洪, 王荣静, 译. 3D 数学基础[M]. 北京: 清华大学出版社, 2005.