

基于 GPU 实时的大范围海水模拟

杨首峰^{1,2}, 刘建波^{1,2}

(1. 四川大学 计算机学院, 四川 成都 610065;

2. 四川大学 视觉合成图形图像技术国防重点学科实验室, 四川 成都 610065)

摘要:为了模拟出具有实时性和真实感的大范围海水场景,提出了一种绘制的方法,该方法使用了存在于投影空间的海面网格模型。同时,为了使海浪模拟的真实感更强,也未采用以往基于 Perlin 噪音的产生海浪高度图的方法,而是采用了更复杂的基于统计模型与快速傅立叶变换(FFT)的波浪生成方法。其次通过立方体纹理实现了海水对天空盒的反射效果,并运用 Phone 光照模型实现了海浪的反射。此外充分地利用了 GPU 硬件的特性和 ping pong 绘制技巧。实验证明该方法能快速模拟出实时、逼真的海水的场景。

关键词:基于 GPU;海浪模拟;投影网格;快速傅立叶变换;实时渲染

中图分类号:TP391.9

文献标识码:A

文章编号:1673-629X(2013)04-0011-04

doi:10.3969/j.issn.1673-629X.2013.04.003

Real-time Simulation of Large Scale Ocean Scenes Based on GPU

YANG Shou-feng^{1,2}, LIU Jian-bo^{1,2}

(1. Computer College of Sichuan University, Chengdu 610065, China;

2. State Key Laboratory of Fundamental Synthetic Vision, Sichuan University, Chengdu 610065, China)

Abstract: In order to simulate large scale ocean scenes with real-time and realistic sense, a rendering technique is proposed. Firstly, use the concept of projected grid. And the intent of the projected grid is to create a grid mesh whose vertices are projective-spaced, not in world-space. At the same time, to make the ocean wave more realistic, do not select the simple method based on the Perlin noise but using the sophisticated algorithms which is based on statistical model and using fast Fourier transformation (FFT) to produce tillable height map. Secondly, the sun and sky reflections are achieved via cube mapping texture. Phong light and Fresnel reflection are discussed and applied. What's more, the scenes are rendered in an efficient manner taking full advantage of the modern graphic hardware and using the ping pong technique. Experiments show that the method is efficient for realistic wave modeling and has better rendering speed.

Key wordst: GPU-based; ocean wave simulation; projected grid; FFT; real-time rendering

0 引言

海水的模拟有其固有的难点和挑战性,相比于对其他自然现象的模拟。比如地形的模拟是静态的,云和火的模拟可以通过粒子系统实现,相对比较容易实现。而对海水的模拟主要是因为海水是动态变化的,而且其动态变化的过程比较复杂,很难对其进行准确地进行描述,详细原因请参考文献[1]。

在海水的模拟过程中,一般主要是需要对海水表面进行建模,自然而然的会想到使用一张网格来对海

水的表面进行描述,而且代表海水表面的网格模型需要时刻依据海水的情况动态地变化起来,因此,困难其实也就是对海浪的动态模拟,目前,以海水模拟在运行时间的性能上为依据,可分为实时性的模拟,和非实时性的模拟,而从模拟效果的真实性的角度又可分为,基于动力模型的建模方法、基于几何的建模方法^[2]、以及基于物理的建模方法和基于海浪谱的建模方法。也有基于动力模型建模的方法,其具有代表性的是粒子系统^[3]和细胞自动机。文中是基于海浪谱的模型,具体的讲,就是将海浪视为由许多不同大小的振幅,和不同的角频域以及不同的随机相位的波组合而成,然后通过线性叠加的方法叠加成的海浪。

文中首先重点介绍了海面网格的一种实时生成方法——投影网格^[4]。同以往在世界空间(world-space)中为海面进行网格建模的方法不同的是,投影网格是在(post-perspective-space)投影空间进行建模

收稿日期:2012-07-04;修回日期:2012-10-10

基金项目:国家自然科学基金资助项目(60736046);国家“973”重点基础研究发展计划项目(2009CB320803)

作者简介:杨首峰(1987-),男,陕西汉中,人,硕士,CCF会员,研究方向为虚拟现实、计算机仿真;刘建波,博士,教授,硕士生导师,研究领域是计算机图像处理和图形学、多源数据融合。

的,这种方法没有采用多分辨率的 LOD (level-of-detail) 算法^[4],但是也能得到相对较为逼真的海水模拟效果,而且比前者有更好的绘制效率。接下来文中又重点地介绍了基于统计模型与 FFT 的波浪生成方法^[5,6],以及简单地介绍了通过立方体纹理映射实现了海水对天空盒的反射效果^[7]。

文中的实验主要是基于 GPU 硬件特性,使用 OpenGL 图形 API 及 GLSL 语言^[8],利用现代图形硬件的加速技术优势,海面网格高度场的生成、网格顶点的坐标变换、法向量的计算等是在顶点 Shader 中完成的,而片元 Shader 则主要完成光照模拟和对纹理的像素渲染。

1 网格模型

1.1 投影网格

在对海浪的模拟过程中,常见的一种方式,是在世界坐标系中,创建等间距的顶点网格。但这种建模方法的缺点是,需要渲染的顶点数量过于庞大,因此计算量太大,尤其是针对于大规模的实时海面模拟影响很大。一般改进的思路是减少需要渲染的顶点数量。常见的算法有 LOD 技术^[4],以及在 LOD 思路改进的一些方法^[9],该方法的思路是,依据到视点的距离不同,进而划分出不同粗细程度的多层次网格,距离近的网格划分的细一些,远的划分的粗一些。这样可以在保证实时性渲染的情况,最大程度地绘制出质量比较好的图像效果。但这种方式也有它的缺点,比如在各层次网格的交接处容易产生图像的明显突变。影响最终渲染出的图像效果。

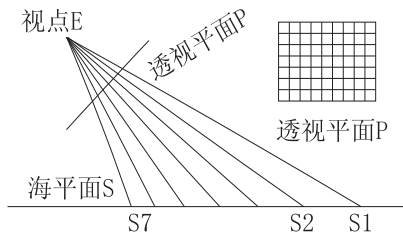


图 1 投影示意图

另一种效率比较高的方法就是使用投影网格。如图 1 所示:思路是这样的,假如在视点 E 的前方放置一块透明的平面 P ,平面 P 上画了一些等间距的网格,然后从视点 E 处以某种角度看过去,那么平面 P 上网格的顶点将投影在海平面 S 上,而且那些网格就会变的不规则。如果视点 E 移动的话,那么投影在海平面 S 上的网格也会发生相应的变化。把透明平面 P 上的网格看作在视点坐标系内建模的投影网格,而把对应投影在海平面 S 上的网格经过变换到世界空间,就是为海水在世界空间的建模网格。然后再对这些网格叠加

上相应的高度,这样波动的海面模型就建成了。

1.2 投影网格基本原理

对于计算机中图形渲染管线的步骤流程,可以参考文献[10],有详细的讲述,这里假设在顶点的变换过程中, M_{world} 、 M_{view} 、 $M_{\text{perspective}}$ 依次表示模型变换矩阵、视点变换矩阵、投影变换矩阵。

在世界坐标系中的点 P_{world} 可以通过式(1)的坐标变换后,便会相应地得到在透视投影空间的坐标系中的点 $P_{\text{projector}}$ 。

$$P_{\text{projector}} = M_{\text{perspective}} \bullet M_{\text{view}} \bullet P_{\text{world}} \quad (1)$$

因此反过来,也可以将一个点从透视投影空间变换到世界空间中,这个变化是可逆的,如式(2):

$$P_{\text{world}} = [M_{\text{perspective}} \bullet M_{\text{view}}]^{-1} \bullet P_{\text{projector}} \quad (2)$$

$$\text{令 } M_{\text{projector}} = [M_{\text{perspective}} \bullet M_{\text{view}}]^{-1}$$

则式(2)可以写成式(3)

$$P_{\text{world}} = M_{\text{projector}} \bullet P_{\text{projector}} \quad (3)$$

如图 2 所示:在投影空间中首先创建一个平面网格 P ,并且使得它与视景体的中心线保持垂直关系,然后再把它投影到世界空间中的某一平面 S_{base} 上,再通过式(3)的变化,把 S_{base} 上的网格变换到世界空间中去。如式(4),在投影空间,坐标的取值范围在 $[-1, 1]$ 。

$$P_{\text{projector}} = \{(x, y, z) \mid x, y, z \in [-1, 1]\} \quad (4)$$

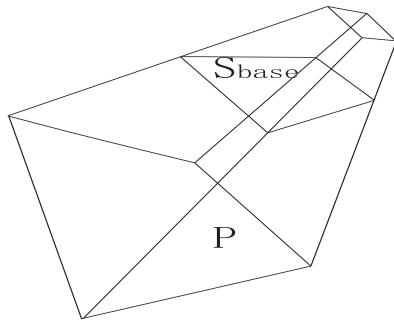


图 2 视景物与平面的位置关系示意图

1.3 投影网格算法

算法的具体过程描述如下:首先在投影空间中创建一个网格大小为 $m \times m$ 的平面网格,依次将网格中的每个顶点的 y 坐标值设置为 1 和 -1,然后再用 $M_{\text{projector}}$ 矩阵做两次投影变换,并将变换后的点连线,同时求得与 S_{base} 平面的交点,这些交点便确定了由投影点构成的可见集合 Q 。而对于其它的点,这里指网格以内的点,可以通过线性插值的方式采样获得。通过这种方式后,只要对 Q 集合中的点进行计算,减少了计算量。假设已经得到了海浪高度场函数 $H(x, y, t)$ 。

在应用时,将会进一步根据海水的实时情况,在海面上各点处依据其法线的方向叠加一个高度值。可

式(12)中, F 是 Fresnel 系数, 可由式(13)计算:

$$F = \frac{1}{2} \frac{(g - k)^2}{(g + k)^2} \left(1 + \frac{(k(g + k) - 1)^2}{(k(g - k) - 1)^2} \right) \\ (k = \cos\theta = L \bullet H, g = (n_a/n_b)^2 + k^2 - 1) \quad (13)$$

上式(13)中, θ 表示的是经过单位化后的入射光线 L 和法线 H 的夹角, 而 n_a 和 n_b 则分别表示的是空气和水两种介质的折射率, 两者比值为常量 1.333, 因此 k 的取值也就决定着 F 的取值情况。例如, 当海水处于波动状态的时候, 也就意味着, 局部小镜面的法向量在不停地变化。进而导致入射角也跟着不停地变化, 因此 F 也就不停地变化, 最终的结果就是海水表面的颜色的闪烁变化。

而对于反射的颜色 C_{reflect} 除了考虑海水本身的颜色外, 还考虑了天空光 C_{skylight} 和太阳光 C_{sunlight} 综合起来就是下面的式(14):

$$C_{\text{reflect}} = C_{\text{skylight}} + C_{\text{sunlight}} \quad (14)$$

而对光的折射颜色 C_{refract} , 这里又分两部分构成, 其中一部分是海水自身的颜色 C_{vol} , 而另一部分便是海水下方物体的颜色 $C_{\text{underwater}}$, 如式(15)所示:

$$C_{\text{refract}} = C_{\text{vol}} + C_{\text{underwater}} \quad (15)$$

4 系统实现及其结果分析

4.1 系统实现流程

在实现的过程中使用了 ping pong 绘制技巧。进而充分地利用 GPU 的硬件优势。因为对 GPU 而言, 它的数据输入和输出都是 2 维纹理。所谓的 ping pong 技术, 简要地说就是在 GPU 上计算的时候, 作为输入输出的数据结构是纹理, 在计算时第一个 pass 将 ping 纹理做为输入, pong 纹理作为输出, 然后接着计算接下来的 pass 时, 则反过来将 pong 纹理作为输入, 而 ping 纹理做为输出, 依次类推的方式直至算法结束。



图 4 系统实现流程图

如图 4 所示场景可分 4 个主要部分依次绘制实现: 绘制静物体, 包括山、船以及天空盒等静止的对象; 绘制海浪, 是四个部分中最复杂的, 也是实现过程中的重点, 可用 4 个 pass 实现, 依次为, 第一个 pass 依据模拟的时间, 求解频谱函数 $\tilde{h}(k, t)$, 此时输入是 ping 纹理, 输出是 pong 纹理。第二个 pass 可根据频谱值作 2 维 FFT, 进而产生海浪的高度图, 此时输入是第二个 pass 的输出, 即 pong 纹理, 输出则是 ping 纹理。后面依次。第三个 pass 主要改变海浪高度等调节操作。第四个 pass 此时便可产生大海的网格; 计算法线, 是根据前面所得的网格顶点数据作法向量的计算; 绘制

大海, 这一部分, 主要是渲染光照效果。

对于第四部分的光照绘制, 下面是这部分的 shader 核心代码如下:

```
skyLight = textureCube(uSkyMap, vReflectVec.xyz); float
sunSpot = pow(clamp(dot(vReflectVec, uSunDir)), gShineness);
sunlight = sunColor * sunspot * 0.5; watercolor = watercolor + sky-
Light + sunlight
```

4.2 实验结果及分析

文中实验硬件环境为 PC 机 (AMD Athlon64 X2 Dual Core Processor4400 +; 2048M RAM; NVIDIA GTX460), 软件环境包括 OSG 图形开发软件、着色器语言 GLSL、VS2008。

下面几张图分别是实验运行后的场景截图。图 5 为太阳光比较强时, 天空盒采用的是晚霞时的场景纹理。图 6 的左图是晴天时, 太阳光并不强时, 并且天空盒采用的是天空多云时的纹理。并且有静止物体如小山的绘制截图。图 6 的右图是在天空盒和海水连接处加入了一定的雾的场景截图, 因为真实的海景是人眼并不能很清楚看清海天相接处, 因为水汽的影响, 因此加入少许的雾做模糊处理, 使整个场景更真实。当网格为 256×512 时帧率在 78 到 94 之间。效率还是可以接受。



图 5 高光照晚霞时的渲染图

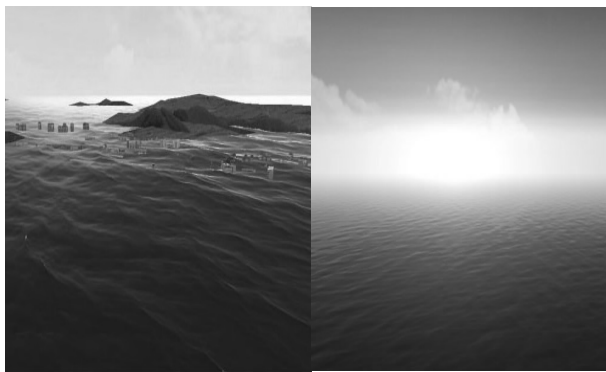


图 6 渲染场景图

5 结束语

文中介绍了一种高效的基于 GPU 的海浪实时模

(下转第 19 页)

eo 类存储着一个视频信息的名称、存储位置、视频标号等信息,这样就可以在其它地方调用平台资源了。同时为了方便程序员的开发,系统提供了客户端 SDK (Software Development Kit),SDK 包装了客户端访问服务器端的细节,提供了详细的 Open API 开发文档、示例代码、支持性的技术注解和支持客户端开发的 JAR 包,程序员在项目开发过程中只要导入 JAR 文件,就可以像使用普通的接口一样进行应用开发,客户端到服务端的操作细节由 JAR 文件来完成。

3 结束语

文中阐述了采用 Tuscany+Spring+DAO 模式,实现文档数字化与资源共享平台 Open API 的开发步骤。系统采用了松耦合的开发模式,文档数字化与资源共享平台在自身内部升级中对 Open API 的影响很小,在不改变数据库原结构和资源存储路径的情况下,甚至没有影响;Open API 的发布和构件的实现之间也是松耦合的,构件功能的变动和增减对 Open API 发布影响甚小;同时客户端 SDK 的引入,屏蔽了 Open API 对开发人员构建的新系统的影响。同时,实现了 API_KEY、SHARED_SECRET 和客户端 IP 地址的集中验证,访问权限的集中注入和解除,方便权限的变更。共享平台 Open API 不但方便在其它地方访问平台的资源,而且方便平台系统在以后升级、改造过程中的系统集成,同时对其它网站实现 Open API 具有一定的参考价值。

(上接第 14 页)

拟的全过程,着重介绍了投影网格,基于统计模型和 FFT 合成的实时海浪的模拟,尤其是基于统计模型和 FFT 合成,不但从理论上详细地给出推导,而且最后给出实验的过程及结果,以及最后简单地介绍了光照模型。实验结果还是非常不错,但为了更进一步增强整个海水的真实感,可以加入阴影,以及考虑实现交互性的海浪。

参考文献:

[1] 李向伟. 一种海水实时绘制方法的研究与实现[EB/OL]. 2005-10-18. <http://www.paper.edu.cn/index.php/default/releasepaper/content/200510-188>.

[2] Fournier A,Reeves W T. A simple model of ocean waves[J]. Computer Graphics,1986,20(4):75-84.

[3] Reeves W T. Particle System-A Technique for Modeling a Class of Fuzzy Objects[J]. Computer Graphics,1983,17(3):359-376.

[4] Johanson C. Real-time water rendering-Introducing the pro-

参考文献:

[1] 李余琨,杨平,朱桑权. 支持开放的 API 接口的增强型业务[J]. 计算机工程与应用,2004,10(4):63-68.

[2] 刘鹏,顾军,周勇. 面向下一代网络的开放式 API 技术研究[J]. 计算机技术与发展,2006,16(2):56-58.

[3] 裴珊珊,叶小梁. 国外 Open API 发展现状及趋势研究[J]. 情报科学,2009(12):1896-1900.

[4] 邓子云. SOA 实践者说:分布式环境下的系统集成[M]. 北京:电子工业出版社,2010.

[5] 余名高,贾秀峰,林坤江,等. 基于 Web 服务的企业应用集成[J]. 计算机技术与发展,2007,17(5):55-57.

[6] 凌晓东. SOA 综述[J]. 计算机应用与软件,2007,24(10):122-124.

[7] 梁爱虎. SOA 思想、技术与系统集成应用详解[M]. 北京:电子工业出版社,2007.

[8] Laws S,Combellack M,Feng R,et al. Tuscany SCA in Action[M]. Stamford:Manning Publications Co.,2011.

[9] Segal A,Beisiegel M,Delfino J S. Deploy an SCA application using the Tuscany domain manager[EB/OL]. [2008-10-02]. <http://www.ibm.com/developerworks/webservices/library/ws-sca-tuscany/index.html>.

[10] Ramalingam R. Design and develop SCA components using the Spring Framework,Part 1:The trifecta:Spring,SCA, and Apache Tuscany[EB/OL]. [2009-10-06]. <http://www.ibm.com/developerworks/webservices/library/os-springsca/index.html>.

[11] 刘泉,周颖,汪雪梅. 安全扩展的 UDDI API 实现架构[J]. 武汉理工大学学报,2003(10):77-80.

jected grid concept[D]. Lund:Lund University,2004.

[5] Mitchell J L. Real-time Synthesis and Rendering of Ocean Water[R]. [s.l.]:[s.n.],2005:2-4.

[6] 夏新华,潘志庚. 基于统计模型的海水运动仿真[J]. 计算机仿真,2005,22(1):62-63.

[7] Chiu Y F,Chang C F. GPU-based Ocean Rendering[C]//2006 IEEE International Conference on Multimeadia and Expo.. [s.l.]:[s.n.],2006:3-4.

[8] Rost R J. OpenGL 着色语言[M]. 天宏工作室译. 北京:人民邮电出版社,2006.

[9] 王纲,季振洲,张泽旭. 大范围动态海浪实时渲染[J]. 哈尔滨工业大学学报,2012,44(3):60-61.

[10] Shreiner D,The Khronos OpenGL ARB Working Group. OpenGL 编程指南[M]. 李军,徐波译. 第7版. 北京:机械工业出版社,2010:78-80.

[11] Perlin K. An image synthesizer[J]. SIGGRAPH Comput. Graph.,1985(19):287-296.

[12] 吕文静. 基于真实感图形技术的虚拟海面场景研究[D]. 天津:河北工业大学,2007.