中文摘要

海水的绘制一直是计算机图形学中的热点研究问题之一。在 3D 游戏、电影、虚拟现实、飞行模拟、地理信息系统等许多领域，都需要绘制有不同程度的真实感的水面场景，并且大多数的这些应用都要求以可交互的速度来绘制水面。要绘制大面积的水面场景，除了要绘制出真实的水面反射、折射等光的效果外，更重要的是保持实时绘制，提高绘制速度，通常的办法是加入细节层次控制，减少要绘制的多边形数目。

本文首先对水面建模进行了研究。快速傅立叶变换法是在傅立叶域对水面进行建模的一种有效方法，使用快速傅立叶变换构建水面模型能在任意的时间点上生成需要的水面高度场，能得到比较逼真的水波效果。

其次，绘制大面积的水面需要图形处理单元处理数目巨大的顶点和片元数据，细节层次控制是一种常用的模型简化方法，加入细节层次控制后，能大大减少需要绘制的三角形数量。在对水面高度场的绘制中，本文使用几何裁剪图细节层次控制算法来提高绘制速度。几何裁剪图算法以视点为中心进行绘制，能提供稳定的帧速率。

另外，为了绘制更加真实的水面光影效果，模拟光线在水面的反射和折射现象，光线反射和折射产生的 Fresnel 现象增加了水面场景的逼真度。

本文实现了一个细节层次控制的大面积水面绘制框架，该系统可以保证水面模拟的逼真度，并同时达到很好的实时速率，完全适合于三维应用中水体场景的模拟。

关键词： 海水绘制 快速傅立叶变换 细节层次控制 几何裁剪图

ABSTRACT

Ocean water rendering has become a very popular field in computer graphics in recent years. Water scenery is essential for many applications such as 3D games, digital movies, virtual reality, flight simulation and GIS, usually interactive speed is needed. To render a large scale water surface, besides the water light effects like reflection and refraction, the most important thing is real-time rendering. Level of detail control is used to diminish the rendering triangle numbers, so that we can get a faster rendering speed.

This paper firstly studied on water surface modeling. The Fast Fourier Transformation method is an effect approach in the Fourier domain, by this approach, we can get the water height field at any given time, and the wave effect is close to reality.

Rendering larger scale water surface needs the GPU to process tremendous vertex and fragment data, level of detail is a model simplifying method. By level of detail control, the rendering triangle number is regularly cut down. When rendering the water height field, we use a geometry clipmaps method to control the level of detail. Geometry clipmaps is a view centered approach, and can provide a stable rendering speed.

Finally, we render the visual effect by apply an environment map, the reflection and refraction are both modulated, the Fresnel phenomenon of reflection and refraction then greatly increases the scene's reality.

A large scale water rendering framework using level of detail control is implemented in this paper, and its reality and interactive rendering is suitable for rendering water scenery in 3D applications.

KEY WORDS: Ocean Water Rendering, Fast Fourier Transformation, Level of Detail Control, Geometry Clipmaps

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 天津大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：李志勇　　签字日期：2007 年 6 月 19 日

学位论文版权使用授权书

本学位论文作者完全了解 天津大学 有关保留、使用学位论文的规定。特授权 天津大学 可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：李志勇　　导师签名：

签字日期：2007 年 6 月 19 日　　签字日期：2007 年 6 月 19 日

第一章 绪论

1.1 引言

三维真实感图形绘制技术，是计算机图形学的一个重要研究领域。计算机图形处理单元（GPU，Graphics Process Unit）将构建的三维模型，根据视点和视线的方向，进行变换和处理，最终在显示设备上绘制逼真的三维场景。长期以来，对绘制技术的研究一直围绕绘制的真实性和实时性这两个方面展开。然而，要快速地绘制非常逼真的自然场景，如森林、云、非静止的水面等，通常比绘制其他的场景要困难得多。首先，这些场景通常面积较其他场景大；其次，这些场景对细节的要求很高。

海水的绘制，一直是三维真实感图形绘制技术中的热点研究问题，在科学可视化和虚拟现实技术中有着重要的地位，被广泛地应用在3D游戏、电影、虚拟现实、飞行模拟、地理信息系统等领域。过去，在PC机上实时生成海水场景，几乎是一件不可思议的事，像在早期的一些电影《未来水世界》、《泰坦尼克号》中，我们看到的海水场景都是由专业图形工作站绘制的。自从NVIDIA公司在1999年发布显卡GeForce256后，PC机上的实时渲染功能越来越强。随后该公司推出的GeForce3显卡，是第一块提供可编程功能的显卡，拥有独立的顶点处理单元和片断处理单元。图形处理单元的快速并行处理能力和可编程功能，使得像绘制海水这些原本只能在图形工作站中进行的任务也能在普通的PC机上完成。

1.2 海水绘制研究现状

虽然图形处理单元的处理速度有了提高，但是海水绘制依然存在绘制的真实性和实时性的问题，这是由海面水体场景的一些性质所决定的。

海面环境是一个复杂的自然场景，通常绘制海面时的场景面积较大，加上一些自然因素如风的影响，水体通常是动态和随机的，海面会出现非常明显的变化，水面上波纹等交互现象的模拟也十分复杂。要真实地绘制水面，就必须建立正确的水波模型。如果要保证水面的真实感渲染，只有动态的水波还不够，必须建立合适的光照模型，否则生成的海水看上去不会太真实。水面的颜色要涉及到光的

反射、折射、Fresnel 现象、太阳的高光以及刻蚀（Caustics）现象等等。复杂的海面场景要在计算机中达到实时绘制，通常是对建立的模型进行一定程度的简化，使需要在图形处理单元中处理的顶点数减少。

近些年来，水体效果的实时模拟一直是真实感图形学的研究热点。早期的水体模拟，主要采用的是基于几何模型（Geometry Model）的方法。Perlin 采用对高度场施加随机噪声的方法得到简单的水面效果[1]；Fournier 和 Reeves 使用改进的 Gerstner 波来模拟水面[2]；Peachy 通过叠加多个正弦波的方法得到合适的水面波形[3]。总的来讲，基于几何模型的方法简单直观，接近实时计算的目标，但是生成的水面场景不是很逼真，尤其是在仅仅用一两个波浪序列来模拟海浪时。基于海洋统计学的快速傅立叶变换（FFT，Fast Fourier Transforms）模型是另一类水面建模方法，通常将某种类型的噪声转换到傅立叶域，然后使用海洋统计波模型进行滤波。Mastin 将白噪声转换到傅立叶域，通过 Pierson-Moskowitz 谱进行滤波[4]。Tessendorf 在他的 1999 年的 Siggraph 课程[5]中详细地介绍了基于快速傅立叶变换的海浪模拟方法，他采用的是 Phillips 谱来进行滤波，通过对 Phillips 谱的参数进行调制，可以很好地体现风速和风的方向对波浪的影响。快速傅立叶变换模型方法的主要优点是可以同时模拟许多不同的波浪，能获得逼真的视觉效果，只是由于在合成的过程中采用规则的矩形粗网格来绘制，略微降低了最终水面的渲染质量和真实感。最后，还有基于 Navier-Stokes 方程的方法，即基于经典流体动力学方程来模拟流体的运动，采用数值计算方法来得到水面的具体形状，Kass 通过对浅水方程加以简化来模拟水波动画[6]，Chen 采用数值迭代方法求解二维的 Navier-Stockes 方程[7]，Foster 等则直接用数值方法求解三维流场从而更加真实全面地模拟了流体运动[8]。基于 Navier-Stokes 方程的方法所生成的水面形状非常接近真实的物理现象，但缺点是水面面积较大时方程的求解非常困难，目前还不能满足实时性的要求。

海洋模拟在实时系统中最难以解决的问题就是效率问题。由于算法复杂性和数据庞大性，动态水面模型使得绘制的实时性仍然很难满足，降低模型复杂度可以明显的提高系统效率，但是同时也降低了视觉效果。如何在保证模型真实度的前提下降低实时系统的硬件要求成为当前图形研究工作者关注的主题。在众多的加速图形绘制的方法中，细节层次模型（LOD，Level of Detail）是其中最主要的一种方法。LOD 技术由 Clark 在 1976 年首次提出[9]，之后在地形绘制领域得到了广泛的应用，该技术在不影响画面视觉效果的条件下，通过逐次简化景物的表面细节来减少场景的几何复杂性，从而提高绘制算法的效率。

1996 年以来 LOD 实时连续绘制算法取得了非常有意义的成果，下面介绍几个主要的相关工作：

1996年，Lindstrom等提出了基于视点的连续LOD细节层次方法（CLOD，Continuous LOD）[10]，该方法基于规则网格，通过一个用户可以控制的屏幕空间误差阈值控制细节简化程度，使用层次四叉树，从最高精度的底层自底向上对整个场景地形逐步简化三角形直到达到指定控制量。该方法通过维持地形块的活动分割和只访问在连续的帧间发生改变的顶点实现帧约束，通过维持顶点拓扑关系的二叉树消除裂缝。

1997年，Duchaineau等提出了实时自适应网格方法（ROAM，Real-Time Optimally Adapting Meshes）[11]，该方法使用二元三角形树（Binary Triangle Tree）基于优先级来产生连续的地形网格。该方法通过两个优先级队列进行分割和合并三角形操作。一个队列负责维持具有优先级顺序的待分割三角形的列表，该队列中优先级最高的三角形首先细分，另一个队列包含具有优先级顺序的待合并的三角形列表。这使得ROAM方法具有帧连续的特点，分割和合并操作可以通过顶点渐变平滑进行。

1998年，Rottger等人在Lindstrom的基础上，提出了一种基于四叉树结构的自顶而下的策略，与Lindstrom的自底而上的方法不同，该策略每帧只访问整个地形的一部分，但该方法需要分析整个地形数据[12]。

1998年，Hoppe等人在他们以前提出的可以从任意网格增减三角形的方法[13]的基础上提出了基于视向的渐进网格方法（VDPM，View-Dependent Progressive Meshes）[14]。该方法提供了一个基于非规则格网（TIN，Triangulated Irregular Networks）的框架，能够提供更多渐进格网优化，但是该方法需要跟踪网格的拓扑结构，难于生成和裁剪。

2000年Willem提出了几何多重映射（Geometrical MipMapping）的方法[15]，将类似于多重纹理映射的概念应用到了地形模型网格上，是一种基于3D加速硬件的高速渲染地形的方法。由于3D加速硬件能够处理大量的三角形，算法利用了这一点来保证CPU处于低负荷状态。由于CPU的低负荷，基于块的LOD处理被证明是一种很有用的方法。该方法使用屏幕空间的像素错误以及三向滤波来解决突变跳动的问题。

2001年，Lindstrom和Pascucci提出了一种新的简单的容易实现的，易于内存管理和可靠的误差分析的地形LOD方法[16]，该工作在过往许多地形可视化工作的基础上提供一个基于规则网格的自顶而下的框架，能够由里至外地基于视向对大规模地形进行细化，并且支持快速的视锥剔除，三角带渲染，可选择的细化。该方法在磁盘上组织数据，通过操作系统内存映射，能够动态加载分块地形数据到物理内存。

2002年，Ulrich提出了分块LOD（Chunked LOD）[17]，该方法是对Lindstrom

算法的改进，采用的数据结构是分块四叉树，主要致力于解决大规模静态地形数据的渲染。与 ROAM 一样，该方法也是从粗糙到精细的分解。

2004 年，Losasso 和 Hoppe 又提出了一种新的几何裁剪图（Geometry Clipmaps）方法[18]，并且该方法在 GPU 上得到实现[19]，最近又被扩展到球面坐标下[20]。与以往的方法不同的是，几何裁剪图将高度图缓存在一系列嵌套的规则网格内，所有嵌套网格随着视点的移动而不断移动，该方法具有简单的网格结构，能带来平滑的视觉效果和稳定的帧速，使用 GPU 加速还能充分利用图形硬件，减轻 CPU 负担。

由于水面网格通常都是规则的三角形网格，基于规则网格的 LOD 算法在目前的海水绘制中比较流行，主要是由 Lindstrom 提出的连续 LOD 方法和 Rottger 等作出的改进算法，在 Xudong Yang 等人的论文[21]中，采用了 Ulrich 改进的分块四叉树方法进行海面网格的 LOD 控制，达到了不错的效果。

1.3 本文主要研究工作

本文的主要研究方向是实现大面积水面的实时绘制，实现一个 LOD 控制的大面积水面绘制框架，如图 1-1 所示。图 1-1 同时也反映了海水绘制的一般流程：首先建立合时的水波模型，根据需要进行适度的网格简化，最后施加光照进行绘制。

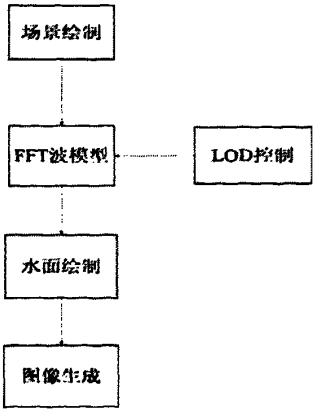


图 1-1 绘制框架

本文将使用快速傅立叶变换（FFT）模型对水面建模，快速傅立叶变换法是在傅立叶域对水面建模的一种有效方法，使用快速傅立叶变换建模能在任意的时间点上生成需要的水面高度场，能得到比较逼真的水波效果。其次，绘制大面积

的水面需要图形处理单元处理数目巨大的顶点和片元数据，在对水面高度场的绘制中，本文使用几何裁剪图方法来提供 LOD 控制。之后，通过模拟光线在水面的反射、折射、Fresnel 现象，可以增加水面场景的逼真度。

1.4 本文结构

本文各章节的组织结构如下：

第二章将从海水绘制的物理基础出发，简单介绍海水绘制的常用方法，然后介绍 LOD 模型简化技术。

第三章描述 FFT 水波模型的生成，在建立 FFT 水波模型的同时，对水面高度场进行优化，包括波形锐化、生成水面凹凸映射图等。

第四章从理论和具体实现上详细描述几何裁剪图 LOD 控制方法，对原始水面网格生成的多分辨率模型，利用几何裁剪图算法进行细节层次简化。

第五章会描述水面的光照渲染算法和实现，包括水面的反射、折射、Fresnel 现象等效果的模拟。

第六章对全文工作作出总结，并讨论未来工作方向。

第二章 海水绘制技术

2.1 海水模拟的流体力学基础

在物理学中，在为复杂现象建模时，通常要进行简化假设，流体模拟也不例外。我们假定流体是不可压缩的和均匀的。如果某流体的任何局部区域的体积不随时间而改变，那么这种流体是不可压缩的。如果它的密度在空间中是个常数，则流体是均匀的。既有不可压缩性又有均匀性，则意味着密度对时间和空间都是常数。这是流体力学的普遍假设，适用于水和空气等真是流体的模拟。

从数学上来看，流体在某一时刻的状态可以由一个速度向量场来表示，即由一个向量函数确定空间中的每一点的速度向量。由于受到各种力如重力、压力等的影响，流体的速度场分布是十分复杂的，我们用空间坐标 x 和时间变量 t，在一般的笛卡尔坐标系上模拟流体动力学。如前所述，流体由它的矢量速度场 $u(x,t)$ 和标量压力场 $p(x,t)$ 表示。这些场既随时间又随空间而改变。不可压缩的流体，其速度和压力在起始时间 $t=0$ 已知，那么流体状态随时间的变化，可以用 Navier-Stokes 方程式描述[22]：

$$\frac{\partial u(x,t)}{\partial t} = -(u(x,t)\cdot\nabla)u(x,t) - \frac{1}{\rho}\nabla p + \nu\nabla^2 u(x,t) + F \qquad \text{公式（2-1）}$$

$$\nabla\cdot u(x,t) = 0 \qquad \text{公式（2-2）}$$

这里 ρ 是流体密度（常数），ν 是动力黏度，F 代表作用于流体的外力。将公式（2-1）分解，可得 Navier-Stokes 方程的各项：

1. 平流项

流体的速度使得流体沿着流动传送物体、密度和其他的量。想象把染料喷射进入流动的液体之内，染料沿着液体的速度场被传送，这种流体的现象称为平流。事实上，正如液体的速度携带染料一样，它也携带液体本身。在右边的第一项表示这个速度场的自身平流，被称为平流项。

2. 压力项

因为流体的分子能够彼此运动，它们倾向于“挤压”和“碰撞”。当外力施加于流体时，外力不是立即地传导到整个流体所在空间，而是离外力近的分子把那些远一些的分子推开，从而产生压力。因为压力是单位面积上的力，所以在流

体上的任何压力自然地产生加速度，第二项代表这个加速度，称为压力项。

3. 扩散项

根据对真实液体的经验可以知道，有些液体比其他的更黏稠。具体来说，糖蜜和糖浆流动得很慢，而医用酒精却流动得很快，即稠的液体黏度高。黏度是液体流动受到阻碍的衡量，这个阻碍造成动量的扩散。因此，第三项称为扩散项。

4. 外力项

第四项是由于外力施加到流体上而增加的加速度。这些力或许是局部力，或许是整体力。局部力施加到流体的特定区域，例如船桨拨动水的力，而像重力这样的整体力则均匀地施加于整个流体。

2.2 海水绘制方法分类

海水绘制首先建立水波模型，然后施加光照进行绘制。根据水面模型建立的方法不同，可以将海水的模拟方法分为大致的四类：流体力学模型，噪声函数模型，正弦波叠加模型，快速傅立叶变换模型，下面将分别介绍这四种不同的方法。

2.2.1 流体力学模型

流体力学模型根据经典流体力学，即 Navier-Stokes 方程来建立水波模型，在给定初始条件和边界条件下用求得的方程数值解来得到海浪的具体形状。

根据求解方法的不同，流体力学模型也可分为两种[23]：第一种方法是从研究流体所占据的空间中各个固定点处的运动着手，分析运动着的流体所充满的空间中每一个固定点上的流体的速度、压强、密度等参数随时间的变化，以及研究由某一空间点转到另一空间点时这些参数的变化，该方法被称为欧拉法，是一种基于网格的方法；第二种方法是从分析流体各个微团的运动着手，即研究流体中某一指定微团的速度、压强、密度等描述流体运动的参数随时间的变化，以及研究由一个流体微团转到其他流体微团时参数的变化，以此来研究整个流体的运动，被称为拉格朗日法，是一种基于粒子的方法。

基于网格的欧拉法直接将 Navier-Stokes 方程离散到网格上，然后计算各个固定网格节点上状态量的变化，从而来得到整个场。这里有两种思路进行网格化，一种是交错网格，即一般情况下标量，如压强，分布在网格单元的中心，而速度之类的量分布在单元表面，这种离散的好处是容易保证守恒性条件，目前多采用此思路；另一种思路则是所有的量都处于同一个位置，这种方法简单，不需太多的插值运算，对各个变量也无不需区别对待。真正采用三维 Navier-Stokes 方程来模拟流体运动始自文献[24]，其中利用 MAC（Marker and Cells）求解流体，但

由于采用显式格式，时间步长必须满足 CFL（Courant-Friedrichs-Lewy Condition）条件以使整个计算收敛。文献[25]采用半拉格朗日法求解对流项，并结合隐式求解器，从而保证计算绝对稳定。

拉格朗日法这种基于粒子系统的方法，则将 Navier-Stokes 方程改写为：

$$\frac{du}{dt}=\frac{1}{\rho}\nabla p+\nu\nabla^2 u(x,t)+F \qquad \text{公式（2-3）}$$

如果将右边整理成一个力，则退化为牛顿第二定律：

$$a_i=\frac{du_i}{dt}=\frac{F_i}{\rho} \qquad \text{公式（2-4）}$$

这里 a_i 为粒子 i 的加速度，u_i 为粒子 i 的速度，F_i 为该粒子受到的合力，ρ 为该粒子所在位置的密度。显然，该类方法就是对于各个相对独立的粒子进行力的分析，通过积分计算出这些粒子下一个时刻的位置和其他状态量。

Reeves 首先引入粒子系统[26]，之后由于粒子系统对于非规则物体具有灵活的表现能力，因此被用来模拟流动[27]。除此之外，对于流动中出现的飞溅、泡沫很容易想到采用粒子系统来表现。文献[28]引入 SPH（Smoothed Particle Hydrodynamics）方法来模拟火和其他气态现象。SPH 方法是粒子系统的一种插值方法，通过引入平滑核来表示周围粒子的影响，该方法同时被用来求解柔性物体的大变形。拉格朗日法的方法的优点为容易表达，不需要对整个空间进行处理，而且容易保证质量守恒，而且比较容易实施控制。但拉格朗日法对于平滑运动界面的重建比较难处理，而且自由界面拓扑的改变必须采用复杂的算法才能构造出该表面的几何，计算量随着粒子数的增多而加大。

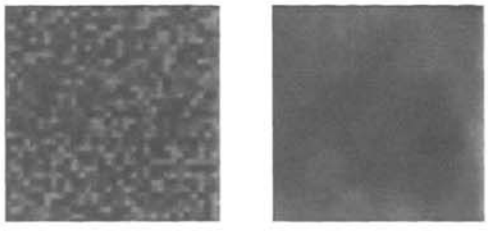
欧拉法和拉格朗日法各有优缺点，为了更真实地模拟流动，基于网格的欧拉算法往往结合拉格朗日的粒子算法一起使用，比如得到广泛应用的半拉格朗日算法。

毫无疑问采用基于经典流体动力学的 Navier-Strokes 方程来模拟流体的运动，是精确度最高的方法，所生成的水面形状非常接近真实的物理现象，但缺点是 Navier-Strokes 方程的求解非常困难，很难满足实时性的要求。

2.2.2 噪声函数模型

噪声函数模型使用连续噪声函数作为水面高度场动态变化的激励源，依靠噪

声的平滑特性来模拟水面连续动荡变化的效果。所谓连续噪声，是指对于空间中的任意两个点，当从一个点移动到另外一个点时，噪声的值是平滑变化的。Perlin 提出了噪声图像的合成方法，他用连续的噪声函数生成二维的噪声曲面[1]。Perlin 噪声是能够在空间中产生连续噪声的函数，也是在各种近似模拟中最常用的噪声函数。如图 2-1（a）是使用不连续噪声生成的纹理，图 2-1（b）是使用 Perlin 噪声生成的纹理。



（a）　　（b）

图 2-1 噪声纹理

噪声纹理图像在某个坐标位置处的颜色、灰度值就代表了高度。通常为了避免对每个水面网格点进行噪声计算，可以使用插值函数如均匀 B 样条函数进行插值构造水面。将噪声施加于 B 样条曲面的特征控制点，间接地控制水面高度场，这样在不增加噪声计算量的情况下可以对水面进行高分辨率的细分。图 2-2 是经过插值后形成的分辨率较高的噪声水面高度图。

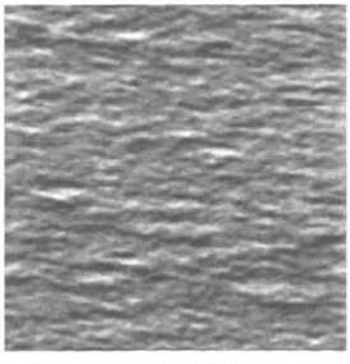


图 2-2 插值形成的噪声水面

基于噪声函数来模拟水面，是一种相当粗糙的方法，该方法所生成的水面形状波动效果不强，不过噪声函数计算比较简单，适于在对实时性要求比较高而对水面真实感要求很低的应用中使用。

2.2.3 正弦波叠加模型

正弦波叠加模型将海洋的表面运动看作是由一系列的正弦波组成，当海风作用于海面时，海面会产生主方向与海风同向的波族。这些波的主要运动方向和海风同向，但是由于海面本身的流体性质，这些波之间会通过黏性阻力等内在作用力相互作用，产生如漩涡等自然现象。并且内在的作用力会使得每个波都会沿与海风不完全平行的方向进行扩散。

为了渲染海面，通常使用一系列波长、振幅和频率都不相同的正弦波作用于一个 $M\times N$ 的顶点序列，每个顶点的高度根据通过该点的所有正弦波在时刻 t 的偏移进行叠加。根据正弦波动的等相位面，可以将正弦波动分为方向波和圆形波，等相位面为平行平面的正弦波动，是方向波；而等相位面为一系列圆柱面的是圆形波。一般来说，在风驱动的广阔水体适合采用方向波，而对于一些较小的水体，输入激励近似为点激励的情况下则适合采用圆形波。

Gestner 波是最早的用于计算机图形学海水仿真的方法[2]，在计算机图形学出现之前海洋学家们就提出了 Gestner 波函数。Gestner 波通过描述水面单个点的运动来体现整个水面的运动趋势，当一个波形通过时，该点作圆周运动。假设最初水面上未经扰动的一点被标记为 $X_0=(x_0,z_0)$，其初始高度值 $y_0=0$，当振幅为 A 的单个波形通过时，该点的运动状态可用公式（2-5）和公式（2-6）表示。

$$X=X_0-K/k\cdot A\sin(K\cdot X_0-\omega t) \qquad \text{公式（2-5）}$$

$$y=A\cos(K\cdot X_0-\omega t) \qquad \text{公式（2-6）}$$

其中 K 是一个表示波的前进方向的二维向量，其幅值满足 $k=2\pi/\lambda$，ω 表示波形的振动频率。

通常单一的 Gestner 波呈现出来的水面振动效果非常单一，为了模拟出更为真实的海水效果，可以采用多个正弦波叠加的方法来模拟复杂的波动效果，以此来改善单一的 Gestner 波的局限性。公式（2-7）和公式（2-8）表示 N 个正弦波进行叠加的结果。

$$X=X_0-\sum_{i=1}^{N}K_i/k_i\cdot A_i\sin(K_i\cdot X_0-\omega_i t+\phi_i) \qquad \text{公式(2-7)}$$

$$y=\sum_{i=1}^{N}A_i\cos(K_i\cdot X_0-\omega_i t+\phi_i) \qquad \text{公式(2-8)}$$

图 2-3 是多个 Gestner 波叠加后的波形效果。

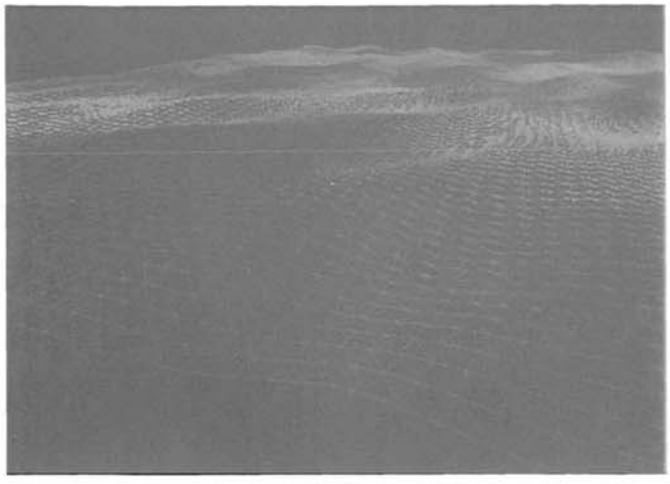


图 2-3 Gestner 叠加波形

正弦波叠加模型算法简单直观，实时性相对较好，能满足对真实性要求不是很高的情况，能够生成外观上比较真实的海面，而且性能的损耗可以控制在合理的范围内。但是，这种技术并不是在真正模拟海面，它无法根据海面所受到的力的变换来反映出真实的交互，例如不能模拟波浪的破碎等现象，只能模拟小振幅的水波。

2.2.4 快速傅立叶变换模型

快速傅立叶变换模型算法是在正弦波算法的基础上，利用快速傅立叶变换和空间频谱函数方程推导出来的海浪生成算法，它可以生成较真实的非线性波浪。该方法是根据要模拟的海浪模型，设计出一个过滤器，在这个过滤器的一端输入某一已知的随机过程，通常采用白噪声，在过滤器的输出端，即可得到所需模拟的波面过程，在计算上可通过 Pierson-Moskowitz 谱或者其他海浪谱模型在频域内过滤一个白噪声，然后采用逆向傅立叶变换（IFFT）转换到空间域，得到一系列的波高[4][5]。由于快速傅立叶变换具有周期性，我们可以生成一小块海面，然后拼接成较大的海面。在本文的第三章，将更加详细地描述快速傅立叶变换模型的水波生成方法。

快速傅立叶变换模型算法在大尺寸取样栅格中产生了难以置信的逼真效果，在消费级 PC 上对中等大小的取样栅格也可以进行实时处理。虽然还没达到 Enright 等人 2002 年创建的离线流体模拟那样的尖端水平，但差距正在缩小。

2.3 LOD 模型简化

实时真实感图形学技术是在当前图形学算法和硬件条件的限制下提出的在一定的时间内完成真实感图形图像绘制的技术。图形的渲染算法通常是固定的，要加快绘制速度实现实时性，只有从需要绘制的场景本身入手，通过损失一定的图形质量来达到实时绘制真实感图像的目的。就目前的技术而言，主要是通过降低现实三维场景模型的复杂度来实现的。

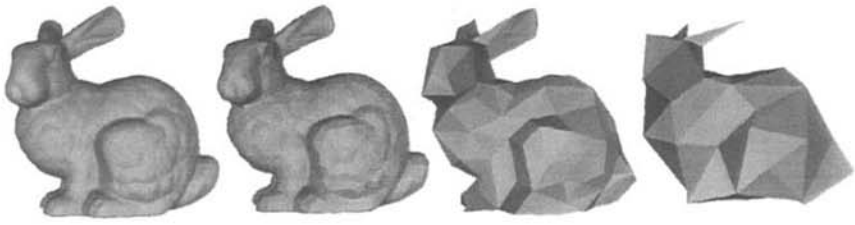
细节层次简化（LOD，Level of Detail）技术在不影响画面视觉效果的条件下，通过逐次简化景物的表面细节来减少场景的几何复杂性，从而提高绘制算法的效率。该技术通常对每一原始多面体模型建立几个不同逼近精度的几何模型。与原模型相比，每个模型均保留了一定层次的细节。在绘制时，根据不同的标准选择适当的层次模型来表示物体。

2.3.1 LOD 简介

在当前的真实感图形学中，需要绘制的三维场景的复杂度都非常高，一个复杂的场景可能会包含几十万甚至几百万个多边形，一种很自然的想法就是通过减少场景中的要绘制的多边形数目，来提高图像绘制的速度，LOD 技术就是在这种背景下提出来的。

从视觉效果的角度来说，距离我们越远的物体人眼看起来会越觉得模糊，越近的物体就会觉得相对地清晰。因此近处的物体模型需要精细绘制，远处物体模型则可以进行粗糙处理。这也符合人们观察世界的方式——人们通常对眼前的物体更为关心，而远处的关注程度则没有那么高。所以即使是非常大范围的场景，我们也只需要对眼前的小范围场景进行精确描绘，而对远处大范围的场景可以进行一定程度的简化。

基于这样的原因，LOD 技术通常对一个原始多面体模型建立几个不同逼近程度的几何模型。与原模型相比，每个模型均保留一定层次的细节，当从近处观察物体时，采用精细的模型；而当从远处观察物体时，则采用较粗糙的模型。这样对于一个较复杂的场景而言，可以减少场景模型绘制时的多边形数目，同时使得生成的真实感图像的质量的损失保持在用户给定的阈值以内，而生成图像的速度可以大幅度的提高，这是 LOD 技术的基本原理[9]。图 2-4 中是四个拥有不同精细程度的物体模型，精细程度越高则需要绘制的三角形数目也较多。图 2-5 则是当物体远近不同时，采用 LOD 技术进行的绘制，当物体离得较近时，采用精细模型进行绘制，当物体离得很远时，则可以采用粗糙模型进行简单绘制。



69451个三角形　　2502 个三角形　　251个三角形　　76个三角形

图 2-4 不同精细程度的模型

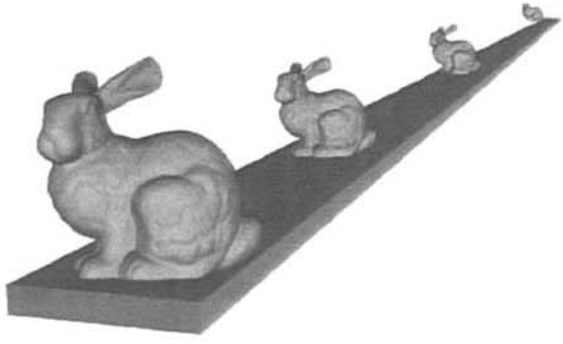


图 2-5 模型的 LOD 绘制

但是需要注意的是，当视点连续变化时，在两个不同层次的模型之间就会出现一个明显的跳跃，有必要在相邻层次的模型之间形成光滑的视觉过渡，使生成的图像序列是视觉光滑的。在用三角形网格表示地形时，还要注意避免 T 型连接，所谓 T 型连接，就是指一个三角形或者多个三角形共享另外一个三角形的一条边，形成 T 型连接之后边的重复绘制在视点改变时就很容易出现裂缝。

LOD 技术的研究主要集中于如何建立原始网格模型的不同细节层次的模型以及如何建立相邻层次的多边形网格模型之间的几何形状过渡。

2.3.2 LOD 模型生成算法分类

由于人们通常用多边形网格（特例为三角形网格）来描述场景中的图形物体，因而 LOD 模型的生成就转化为三维多边形网格简化问题。网格简化的目的是把一个用多边形网格表示的模型用一个近似的模型表示，近似模型基本保持了原模型的可视特征，但顶点数目少于原始网格的顶点数目。多边形网格简化算法进行分类的方法有多种：

1. 按网格形状分类

不规则网格：网格由不规则三角形构成，三角形之间共享顶点和边的关系比

较复杂。

规则网格：网格三角形比较规则，通常是等腰直角三角形，两个三角形之间可以是共享一条直角边，也可以共享斜边，但必须一致。

2. 按简化机制不同分类

自适应细分型：要求首先建立原始模型的最简化形式，然后根据一定的规则，通过细分把细节信息增加到简化模型中，从而得到较精细的模型表示。

采样型：类似于图象处理中的滤波方法，有时不能保持拓扑结构不变。这类方法对原始模型的几何表示进行采样，其中一种方法是从模型表面选择一组点；另一种方法是把一个三维网格覆盖到模型上，并对每个 3D 网格单元进行采样。

几何元素删除型：通过重复地把几何元素（点、边或面）从三角形中移去，从而得到简化模型。有三种形式的删除：直接删除；通过合并两个或多个面来删除边或面；对边或三角形进行折叠。移去或删除操作反复进行，直到模型不能被简化或达到了用户指定的近似误差为止。在进行几何元素删除时，绝大多数算法要求不能破坏模型的拓扑结构。

3. 局部算法/全局算法

全局算法是指对整个物体模型或场景模型的简化过程进行优化，而不仅仅根据局部的特征来确定删除不重要的元素。局部算法是指应用一组局部规则，仅考虑物体的某个局部区域的特征对物体进行简化。

4. 其他分类方法，如视点相关、误差可控性等

视点相关性：把算法分为两大类，即与视点无关的模型简化算法和与视点相关的模型简化方法。早期的算法都与视点无关，最近出现了一些与视点相关的方法，这是一个重要的发展趋势。

误差可控性：有两层含义，一是用户对整个模型的近似误差是否可以控制（全局）；二是指用户对局部误差是否可以控制。进一步讲，用户可以有选择地对模型的不同部分使用不同的误差度量。

2.3.3 LOD 算法回顾

在地形网格绘制研究中，已经提出了许多不同的 LOD 算法，下面将对一些比较经典的算法进行回顾。

1. CLOD 算法

Lindstrom 等在 1996 年提出了基于视点的连续 LOD 细节层次方法（CLOD，Continuous LOD）[10]，该算法对地形分块，每一地形块根据采样间距分成几个细节层次。算法为每一个块估计了一个误差范围，如果视点的移动使得当前块的误差超出预估范围，则调用块相应的低或高细节层次。由于采用了四叉树结构，细

节层次的简化与细化比较方便。当前细节层次下，对块中的每一顶点计算误差值，与误差阈值比较后决定该点是否删除。CLOD 算法的这种顶点删除操作等价于相邻三角形的合并操作，每删除一顶点相当于合并以该点为直角点的两相邻三角形，如图 2-6 所示。算法从底至上地合并三角形，最终得到块的连续细节层次。

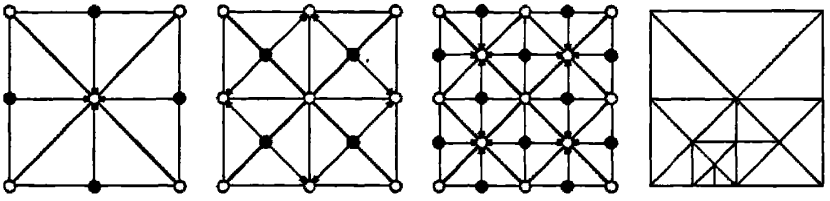


图 2-6 CLOD 算法块分解和合并

算法通过保持顶点间依赖关系来消除不同细节层次间的裂缝。顶点之间存在着依赖关系，若某顶点的存在依赖于其它顶点，则后者是前者的父顶点，前者是后者的子顶点。算法建立顶点树存储顶点间的依赖关系，当需要保留块中某顶点时，递归检查该顶点的父顶点，若所有父顶点都存在（没有删除）则终止递归；否则插入那些被删除的父顶点。这样算法得到的所有顶点都保持了正确的依赖关系，从而消除了裂缝。至于块与块之间的裂缝，算法建议共享边界顶点，允许依赖关系越过边界传播来实现相邻块在边界处有相同的细节层次。

Lindstrom 首次提出的屏幕空间误差控制方法被广泛使用，是一种评价原始网格顶点重要性的有效方法。该方法先计算当前顶点在物空间的实际误差值，再将该误差值投影到屏幕空间，得到顶点的屏幕误差（像素单位）。地形简化时以屏幕误差与给定误差阈值的比较结果作为简化条件。算法利用了帧间相关性来提高实时性能。算法记录当前帧中每个顶点的状态（是否保留，误差是否足够大），在下一帧中只对状态发生变化的顶点进行操作。当视点变化很小时，帧中大多数顶点的状态不变，因此算法只对少量的顶点进行操作，从而提高了算法的效率。

2. ROAM 算法

ROAM 算法即实时优化自适应网格（Real-Time Optimally Adapting Meshes），是由 Mark Duchaineau 等人提出的[11]。ROAM 算法基于三角形二叉树结构表示地形网格，并在三角形之间建立显示相邻关系。如图 2-7 所示，二叉树结构中的根 T 是一个等腰直角三角形，以底边中线将三角形对分，得到下一级的两个子三角形 T0 和 T1，T0 、T1 互为邻居，如果需要则依次细分。在需要合并三角形时，进行相反的过程。

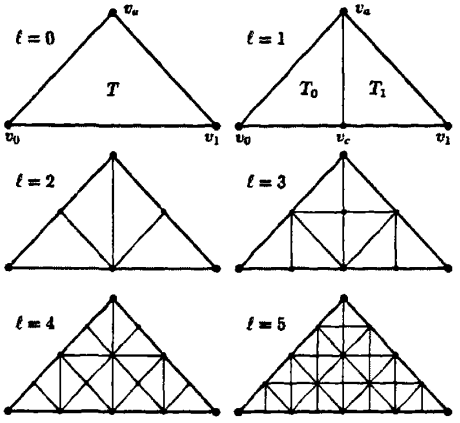


图2-7 ROAM算法三角形二叉树结构

算法为了提高实时分裂和合并的效率，采用了双队列优化的方法。一个分裂队列用来专门保存需要分裂的三角形，另一个合并队列则专门对三角形进行合并。队列均为优先级队列，队列中的三角形按照设置的优先级的高低进行排列。以分裂队列为例，首先把由前一帧中获得的网格中的所有三角形插入到队列中，插入时按照优先级排序，然后执行循环操作，在需要分裂时从队列中取出优先级最高的三角形进行分裂，得到的两个三角形重新插入到队列中，这样每次分裂得到的三角形网格都是最优的。双队列优化的关键是获得严格单调的三角形优先级，算法中采用了和 Lindstrom 算法类似的三角形平面误差，该误差严格单调，因此被用作为三角形的优先级。

ROAM 算法同样利用了帧间相关性来提高实时性能，不过在高分辨率的地形绘制中 ROAM 算法效率不高，在三角形数在 10 万以上时，其交互绘制性能会大大降低。

3. VDMP 算法

Hoppe 在 1996 年提出了渐进网格（PM，Progressive Meshes）算法[13]。该算法首先搜索平面区域的特征边，然后通过使用边折叠操作来进行模型简化。当边折叠产生一个新顶点时，移去两个面和一个顶点，结果将产生一个简化的基网格；而顶点分裂操作（边折叠的逆），可用于把细节增加到基网格上，如图 2-8 所示。使用这种渐进网格，可抽取多个连续的细节层次。算法分为以下几个步骤：

（a）使用基于能量方程表示的最小简化代价对边进行排序，并放入一个表中；

（b）对表中前面的边进行边折叠操作，记录下对应的顶点分裂操作；

（c）新顶点可选在边的两个端点或边的中点；

（d）重新计算那些受简化基本操作影响的边的代价函数；

（e）重复 b-d，直到表为空，或者简化代价超过给定值。

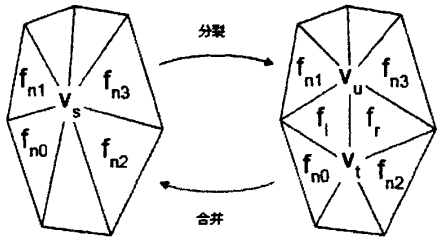


图 2-8 PM 算法分裂合并过程

1997 年，Hopper 对 PM 算法进行改进，得到视相关的渐进网格算法（VDPM，View-Dependent Progressive Meshes）[14]，该算法能够根据视相关参数对模型进行简化。1998 年，Hoppe 把 VDPM 算法应用到地形领域，实现了大规模地形的实时漫游。算法首先将地形分块，对每块分别用渐进网格预处理，根据视点和误差阈值生成连续细节的 TIN 模型。VDPM 记录简化过程的信息，并用这些信息根据不同的视点位置来实时生成不规则三角形网格。地形块通过共享边界顶点实现无缝连接，避免块与块之间的裂缝。该算法也支持在两个 LOD 之间基于时间的几何变形。

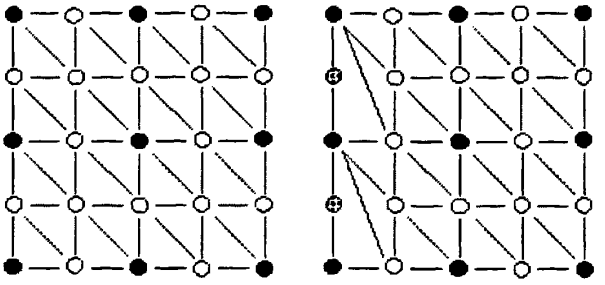
VDPM 算法中的误差考虑了模型表面的颜色、纹理、表面法向量等特征信息，并将这些信息引入到模型的能量函数中来控制模型的简化，简化速度快。

4. Geometrical MipMapping 算法

Willem 在 2000 年提出了几何多重映射（Geometrical MipMapping）的方法[15]，将类似于多重纹理映射的概念应用到了地形模型网格上。

算法将地形分成多个小区域，每个小区域为一个块，每个块都拥有多层次的分辨率。分辨率最高的层次由地形网格直接生成，往后分辨率依次减为原来的 1/4，如图 2-9（a）所示，该块的最高分辨率由所有黑色和白色顶点构成，去掉白色顶点后分辨率降为原来的 1/4。算法以块为简化的基本单位，每个块按照离视点的远近选择合适自己的分辨率，块内部分辨率是相同的，相邻的块分辨率不一定相同。

当不同分辨率的块共享边界时，就会出现 T 型连接，几何多重映射算法通过改变分辨率较高的块的内边界三角形连接方法来消除 T 型连接，图 2-9（b）中的灰色点是引起 T 型连接的点，通过改变分辨率较高的块的内边界三角形连接方式，即忽略引起 T 型连接的灰色点，就可避免 T 型连接。该方法使用屏幕空间的像素错误以及三向滤波来解决突变跳动的问题。



(a) (b)

图 2-9 块结构及 T 型连接消除

几何多重映射法是一种基于 3D 加速硬件的高速渲染地形的方法，3D 加速硬件能够处理大量的三角形，算法利用了这一点来保证 CPU 处于低负荷状态。由于 CPU 的低负荷，基于块处理的多重映射被证明是一种很有用的方法。

2.3.4 LOD 模型的选择标准

恰当地选择 LOD 模型能在不损失图形细节的条件下加速场景显示，提高系统的响应能力。选择的方法可以分为如下几类：一类是去掉那些无法用图形硬件绘制的细节，如基于距离和物体尺寸标准的方法；另一类是去掉那些人类视觉觉察不到的细节，如基于偏心率、视野深度、运动速度等标准的方法；此外，还有一种方法考虑的是保持恒定帧速率[32]。

1. 距离标准

这种方法考虑的是物体到观察者的距离，即从视点到物体内指定点的欧氏距离。这种方法的理论依据是当一个物体距离视点越远，此物体能够被观察到的精细的细节部分就越少。这就意味着选择较粗糙的细节层次来表示物体不会对显示的逼真度有很大影响。

2. 尺寸标准

这种方法利用了人眼辨识物体的能力随着物体尺寸的减小而减弱的特性。它考虑到待表示物体的尺寸，较小的物体用较粗糙的细节层次，较大的用较精细的细节层次。

3. 偏心率

此方法利用了人眼辨识物体的能力随着物体逐渐远离视域中心而减弱的特性。视网膜的中心对物体细节的分辨能力较强，视网膜边缘的分辨能力较弱。根据这个原理，将显示的场景分为具有较精细细节层次的中心部分（对应于眼睛视域的中心）和外围部分（对应于视域的外围部分）。

4. 视野深度

这种方法根据观察者眼睛的焦距来为物体选择合适的细节层次。在聚焦区域的前面或者后面的物体不被聚焦。

5. 运动速度

该方法是根据物体相对于观察者视线的角速度选择合适的细节层次。运动着的物体相对观察者的速度也影响着人眼观察到的物体的清晰程度，在屏幕上快速运动的物体看起来是模糊的，这些物体只在很短的时间内被看到，因而观察者可能看不清它们。这样，就可以用较粗糙的细节层次来表示它们。

6. 固定帧速率

保持一个较高并且趋于稳定的帧速率对于良好的交互性能是非常有意义的。这就意味着一旦选定一个帧率，就要保持一定范围内的恒定，不能随场景复杂度的改变而出现大幅度的变化。

第三章 基于 FFT 的水波模型建立

海水是一个大范围的水体，如果把占据某个三维空间的水体看作一个整体，则这个水体可以看作是由很多很多的小水立方体构成的。如果不对水体做空间位置上的限制，这些小水立方体可以看作三维空间中的一个个小粒子，粒子运动和之间的互相作用形成了水体的流动。基于粒子的方法物理上比较直观，但是粒子的不固定性要求计算每个粒子的状态，这样的模拟效率通常比较低。如果对粒子的空间位置在水平面上进行限定，将水平面分成二维网格，则水体可看作由不同网格上的水柱构成，水柱的高度就是该网格上水面的平均高度。

基于水体的网格表示，我们只需要知道水面的高度起伏形状来渲染最终的水体，因此水体通常被简化为高度场（Height Field）。一个高度场等同于一个二维空间中的方程，根据给定点的 x 和 z 方向上的坐标，计算出点在 y 方向上的高度。高度场被存储为高度图（Height Map），用高度图来渲染表面的方法也被称为置换映射。

由于图形硬件采用扫描线法来渲染，因此平面必须要用多边形来表示。我们把表面近似表示为网格平面（网格不一定规则），然后生成高度场来采样和置换网格平面上的点，最后进行渲染和着色。

生成高度场的方法有多种，二维 Navier-Stokes 方程、波形函数、快速傅立叶变换等等。基于 Navier-Stokes 方程的方法所生成的水面形状非常接近真实的物理现象，但缺点是水面面积较大时方程的求解非常困难，目前还不能满足实时性的要求；基于波形函数方法生成的水面则过于单调且交互性比较差。因此，本文中我们选择快速傅立叶变换的方法。

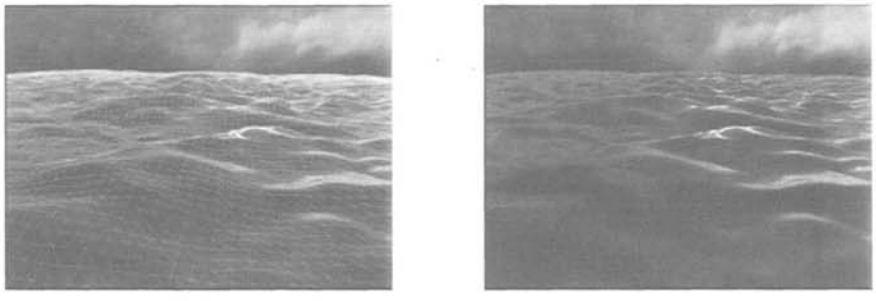
3.1 FFT 模型简介

在海洋统计学中，简单的 Gerstner 正弦波模型已经不能满足根据实际海洋观察得来的统计规律。水波的统计学描述同样将水面高度场分解成一系列正弦和余弦波，但是为了建模方便，这些正弦波的波幅值有着很好的数学和统计学规律。在将高度场分解成正弦波时，一般用快速傅立叶变换算法来求解，因此统计波模型也通常称为 FFT 模型。

水波的 FFT 模型由于基于海洋统计学，该方法具有如下优点：

1. 可以直接利用现有的海洋学观测和研究的成果。如：海浪频谱和方向谱的相关公式。
2. 可以通过参数调整波浪模拟的效果。如：改变风速可以得到不同的波浪形状；改变频率分割数和方向分割数，可以调整波浪模拟的细节。
3. 模拟效果比较真实。
4. 可以实时的模拟波浪运动。

FFT 模型采用 Mastin 提出的基于傅立叶域的方法[4]，该方法首先将某一类型的波形噪声进行傅立叶变换转换到傅立叶域中，然后通过要模拟的海浪模型设计的过滤器进行滤波，在过滤器的输出端，即可得到所需模拟的波面过程。在计算上，Mastin 通过 Pierson-Moskowitz 谱在频域内过滤一个白噪声，然后采用逆向傅立叶变换（IFFT）转换到空间域，得到一系列的波高，Tessendorf 在他的的 Siggraph 课程中的介绍则采用的是 Phillips 谱[5]，如图 3-1 所示，图 3-1（a）是采用 FFT 方法生成的海面框图，图 3-1（b）是其绘制结果[30]。



（a）　　（b）

图 3-1 FFT 生成的海面

由于 FFT 模型生成的海面可以很好地通过海浪谱的参数进行调节，在绘制的实时性和真实性方面平衡性较好，我们将采用该模型来生成水面高度场。

3.2 FFT 水波模型建立

在 FFT 模型里，将水面高度作为一个独立变量，记为 $h(X,t)$，表示在时间 t 时，点 X 的高度，点 $X=(x,z)$ 表示水平面上某一点。

计算方法如下：

$$h(X,t)=\sum_{K}\tilde{h}(K,t)e^{iK\cdot X} \qquad \text{公式（3-1）}$$

计算时使用的是离散的傅立叶变换，其中 K 是一个表示波的前进方向的二维向量，表示为 $K=(k_x,k_z)$，如果要绘制的高度场的实际尺寸用 L 表示，其网格分辨率为 $N\times N$，则 $K=(k_x,k_z)=(2\pi n/L,2\pi m/L)$，$n$、$m$ 是满足关系 $-N/2\le n,m\le N/2$ 的整数，其幅值满足 $|K|=2\pi/\lambda$。

$\tilde{h}(K,t)$ 是一个复数的形式，是水面高度场的傅立叶系数，同时表示时间 t、以方向 K 前进的波的波幅和相位。

公式（3-1）计算所得的结果是在离散点 $X=(nL/N,mL/N)$ 的高度值。

在绘制高度场时，为了计算高度场中每一点的法向量，通常需要得到该点的梯度值。传统的差分形式的梯度计算方法比较简单，但是在波长较小时逼近程度不够精确，更好的计算梯度的方法是对公式求导，得到梯度的傅立叶求和形式：

$$\nabla h(X,t)=\sum_K iK\tilde{h}(K,t)e^{iK\cdot X} \qquad \text{公式（3-2）}$$

由公式（3-1）和公式（3-2）可知，$\tilde{h}(K,t)$ 决定了波浪的表面状况，由它来生成水面高度场。Tessendorf 介绍了一种由 Phillips 谱来生成表示波幅和相位的复值场 $\tilde{h}(K,t)$ 的方法。

$$\tilde{h}(K,t)=\tilde{h}_0(k)e^{i\omega(K)t}+\tilde{h}_0^*(-K)e^{-i\omega(k)t} \qquad \text{公式（3-3）}$$

$$\tilde{h}_0(K)=\frac{1}{\sqrt{2}}(\xi_r+i\xi_l)\sqrt{P_h(K)} \qquad \text{公式（3-4）}$$

其中，$\tilde{h}_0(K)$ 是 $\tilde{h}(K,t)$ 当 $t=0$ 时的初始值，ξ_r 和 ξ_l 是两个互相独立的值在 0 和 1 之间的高斯随机数，$\omega(K)$ 为角频率函数，$P_h(K)$ 即为 Phillips 谱。

$\omega(K)$ 满足如下关系：

$$\omega^2(K)=gk\tanh(kd) \qquad \text{公式（3-5）}$$

其中，$k=|K|$，d 表示水面深度。

Phillips 谱 $P_h(K)$ 由以下的公式定义：

$$P_h(K)=a\frac{e^{-1/(kl)^2}}{k^4}|K\cdot W|^2 \qquad \text{公式（3-6）}$$

其中，$l=v^2/g$ 是在风速为 v 时可能产生的最大波长（g 为重力加速度常数

$9.8m^2/s$），W 表示风的传播方向，K 表示波的传播方向，a 为比例常数。

3.3 高度场优化

3.3.1 波形锐化

由公式（3-1）计算出来的水面高度波形总体上波形比较平滑，当风速较大时，水波的叠加往往会产生很多小的波形比较尖锐的水波。如果水波达到的高度足够高，还可能在最高点分裂，产生泡沫、水花、水泡等现象。在这里，我们只考虑在风速不是特别大的时候产生的锐化波形。

我们通过对波形中的每一点施加微量扰动的方法来使波形显得锐化，其扰动公式为：

$$X = X + \lambda \sum_K -i\frac{K}{k}\tilde{h}(K,t)e^{iK\cdot X} \qquad \text{公式（3-7）}$$

λ 为控制参数，它的大小将直接影响生成的波形的锐化程度。

3.3.2 水面凹凸映射图

真实的水面并不是完全的光滑的，即使是在风速很小的情况下，水面也会有明显的凹凸起伏，呈现出褶皱的效果。在传统的纹理贴图技术下，只能将平面贴图机械地附着在由多边形构成的 3D 骨架上，不能真实地表现出物体细节表面的凹凸起伏，要想实现褶皱的效果，只能在纹理贴图上添加一些阴影。然而这些阴影在遇到动态光源时不会即时改变，给人的感觉很不真实。

凹凸纹理映射（Bump Mapping）是一种新的纹理贴图方式，它不需改变物体的几何造型，使用凹凸贴图为三角形上的每个顶点赋予虚拟的法线。因此，光线反射不是按照原来真正的平面三角形法线计算，而是按照虚拟的法线计算，模拟出表面凹凸不平的质感。

在三维网格平面中，向量（0，0，1）在凹凸纹理映射中表示未经扰动的法向量，可以看作是一个从表面指向表面上部的方向向量。当表面凹凸不平时，朝上的法向量将根据凹凸情况的不同而适当倾斜。通常将倾斜扰动后的表面法向量存储在一个二维数组里，称为凹凸映射图。

将一个高度场转化为一个凹凸映射图是一个非常直接的过程，对在高度场中的每一点，只需要根据它正上方和右方的点对高度进行差分，法向量是两个差分

向量叉积的规范化结果。在已有的高度图中，用$h(x,z)$表示储存在(x,z)位置的高度值。于是，可先计算出该位置的在X和Z方向上的切向量$U(x,z)$和$V(x,z)$，进而通过$U(x,z)$与$V(x,z)$的叉积得到该处的法向量，计算公式如下：

$$U(x,z)=(1,0,h(x+1,z)-h(x,z)) \quad \text{公式（3-8）}$$

$$V(x,z)=(h(x,z+1)-h(x,z),0,1) \quad \text{公式（3-9）}$$

$$N(x,z)=nomalize(U(x,z)\times V(x,z)) \quad \text{公式（3-10）}$$

正确的凹凸映射还需要光线入射方向及其与视线的半角方向和凹凸映射图中的法向量三者在同一个坐标系中，否则近处观察就会发现，在场景中的光照与实际的光与眼睛的位置并不一致。在物体空间中，几何形状的法向量是（0，0，1），显然前述运用 FFT 模型技术生成的波形曲面的法向量不会是（0，0，1），所以无法直接使用凹凸映射图。解决的方法是旋转物体空间的光线向量和半角向量到凹凸映射图法向量的坐标系，即构造一个旋转矩阵：首先在波纹曲面的每一个顶点位置，利用该处的切向量构造一个切线坐标系，在该局部坐标系下，顶点的法向量总是指向 z 轴正方向，即（0，0，1），这样就可以根据二维的纹理地址直接获取凹凸映射图里扰动后的法向量。剩下的问题构造每个顶点的旋转矩阵，利用该矩阵将对应的凹凸映射图法向量从局部切线空间转换到世界坐标系，旋转矩阵的三个列向量通常分别用 T、B、N 表示，它们是切线空间的三个坐标轴，即切线方向、副法线方向和法线方向，旋转矩阵及其计算公式如下：

$$R(x,y,z)=\begin{pmatrix} T_x & B_x & N_x \\ T_y & B_y & N_y \\ T_z & B_z & N_z \end{pmatrix} \quad \text{公式（3-11）}$$

$$T=\left\langle \frac{\partial x}{\partial x},\frac{\partial h(x,z)}{\partial x},\frac{\partial z}{\partial x}\right\rangle=\left\langle 1,\frac{\partial h(x,z)}{\partial x},0\right\rangle \quad \text{公式（3-12）}$$

$$B=\left\langle \frac{\partial x}{\partial z},\frac{\partial h(x,z)}{\partial z},\frac{\partial z}{\partial z}\right\rangle=\left\langle 0,\frac{\partial h(x,z)}{\partial z},1\right\rangle \quad \text{公式（3-13）}$$

$$N=T\times B=\left\langle -\frac{\partial h(x,z)}{\partial x},1,-\frac{\partial h(x,z)}{\partial z}\right\rangle \quad \text{公式（3-14）}$$

在根据指定的参数计算出水面高度场之后，我们在切线空间中计算每个顶点

的扰动后的法线向量，并最后计算出由切线 T，副法线 B 和法线 N 组成的转换矩阵，具体方法参见公式（3-12）、公式（3-13）和公式（3-14），该矩阵将会在渲染水面时把凹凸纹理的切线空间上的法线向量转化到世界空间，来计算立方体凹凸环境映射。

3.4 实验结果

在最后的实现中，我们将约束随机产生波的参数，包括风速、风向、波幅等，来模拟出比较真实的水面波形。此过程中这些参数是相关的，必须对这些参数进行组合，以达到水面的合理效果。

首先，我们必须根据场景的需要选择合适的风速和风向。风速小时水面将比较平静，显得风和日丽；风速很大则波浪起伏，显得更加激烈。风向则决定了波浪运动的大致方向。之后，需要根据风速来选择合适的波幅，波幅可以由 Phillips 谱中的比例参数来控制。在风速较小时，我们选择较小的波幅；在风速较大时，我们选择偏大的波幅，比如暴风雨中的波浪显然比风和日丽场景中的波浪要大。

根据公式（3-1）至公式（3-6），设定合时的风速大小和风向后，生成的 FFT 模型的波形图如图 3-2 和图 3-3 所示。从图中可以看出，在风力的作用下，水面呈现出波浪起伏的效果：图 3-2 中，风速较小，波形显得比较平滑；图 3-3 中，风速加大，水面波形起伏得比较厉害。

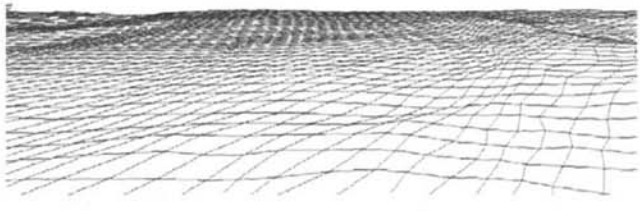


图 3-2 风速较小时 FFT 波形水面

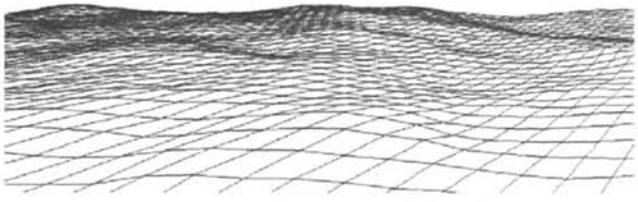


图 3-3 风速较大时 FFT 波形水面

图 3-4 中是利用公式（3-7）进行波形锐化后的水面波形图，与图 3-3 相比，波浪看起来起伏得不像没锐化之前那么平滑了，在有些浪尖的地方波形过渡比较明显，更贴近风力较大时的现实水面情况。

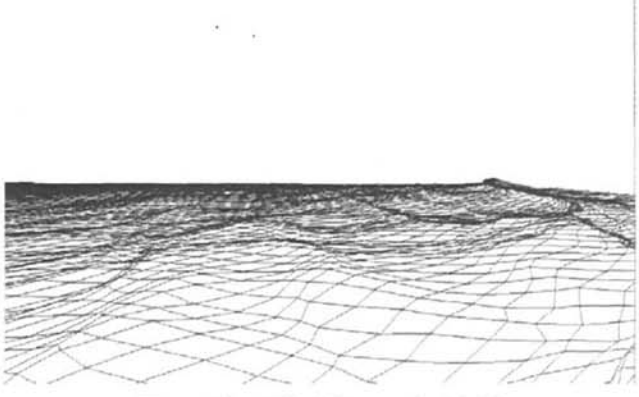


图 3-4 波形锐化后的 FFT 波形水面

由于 FFT 算法的 CPU 耗费较大，水面网格的分辨率对其绘制速度有很大的影响，在 1024×1024 的大分辨率下，水面的波形生成已经比较耗时，实验证明，采用 512×512 或者 256×256 的网格分辨率在 FFT 模型下是比较合适的。

3.5 本章小结

本章基于快速傅立叶变换模型对水面高度场生成方法进行了研究，使用快速傅立叶变换模型能在任意的时间点上生成需要的水面高度场，能得到比较逼真的水波效果。在根据快速傅立叶变换模型的相关理论生成了水面高度场后，针对生

成的水面比较平滑的问题，提出了水面高度场优化的方法，可以生成不同场景中的水面。

第四章 水面网格的LOD控制

LOD 技术在不影响画面视觉效果的条件下，通过逐次简化景物的表面细节来减少场景的几何复杂性，从而提高绘制算法的效率。从大面积海水绘制的实时性角度出发，使用适当的LOD简化技术是十分必要的。

几何裁剪图算法是在2004年首次提出的一种新的基于规则网格的LOD控制方法[18]。在以往的LOD控制方法中，大多数需要在运行时创建或者修改网格结构，即使使用顶点缓存和索引缓存，这种耗费对现在的图形卡来说也是比较昂贵的。另外，那些使用不规则网格的方法则需要CPU来进行更多的计算，使得CPU成为提高整体性能的一个瓶颈。与以往的方法不同的是，几何裁剪图将高度图缓存在一系列嵌套的规则网格内，所有嵌套网格随着视点的移动而不断移动。这种规则网格结构带来了以往的LOD方法所不具有的一些好处：

1. 简单的数据结构：没有在基于指针、索引结构上的不规则移动，嵌套网格的环形寻址非常高效。
2. 优化的渲染吞吐量：顶点数据存储在显示内存之中，规则的网格结构允许我们以三角形带的方式减少渲染次数，提高渲染吞吐量。
3. 稳定的渲染：由于网格采样完全独立，无需参数用于动态调整，所以使得渲染速度接近常数。
4. 带宽控制：因为有固定的裁剪图大小，能够收缩渲染区域以减少渲染负荷，这样可以控制裁剪图的更新带宽。

几何裁剪图当然也存在着一些局限。一方面，最差情况下的地形带有同一层次的细节，这样不利于局部调整。另外一个限制是高度图被假设为拥有边界频谱密度，即高度图的频谱通常是比较均匀的，非常高的高度频谱将影响到绘制效果。举个例子，一个高的像针的特征地形进入观察范围会带来延迟。实际上我们要绘制的水面网格是比较平滑的，不会出现这种高度突变的问题。值得注意的是，船、植物和其它对象被组合到环境上时，我们就需要使用其它LOD技术分别渲染。

我们将使用几何裁剪图（Geometry Clipmaps）的方法进行水面网格的LOD控制来保持恒定帧速率。几何裁剪图来源于纹理裁剪图（Texture Clipmaps），但有一些关键的不同点：纹理裁剪图在计算每个像素的LOD时基于屏幕空间几何投影，但是对于高度网格，屏幕空间几何直到LOD选择前都是不能获得的，取而代之，几何裁剪图使用一系列以视点为中心的嵌套的矩形区域在世界空间基于

观察距离选择 LOD。

4.1 几何裁剪图的 Clipmap 机制

几何裁剪图思想来源于 Clipmap 机制，Clipmap 是在 Mipmap 基础上提出的一种纹理映射机制。Mipmap 是过去应用最为广泛的纹理映射技术之一，Willams 将低一级图像的每边的分辨率取为高一级图像的每边的分辨率的二分之一，而同一级分辨率的纹理组则由红、绿、蓝三个分量的纹理数组组成。由于这一个查找表包含了同一纹理区域在不同分辨率下的纹理颜色值，因此被称为 Mipmap[15]。

Mipmap 可以用一个四棱锥来描述，如图 4-1 所示。最底层图像为给定的原始图像，第二层图像可以由最底层图像与边长为 2 个像素的正方形滤波器做卷积运算得到。一般地，第 n 层图像可以由原始图像与边长为 2^{n-1} 个像素的正方形滤波器做卷积运算得到，每一层图像分辨率是上一层图像的 1/4。

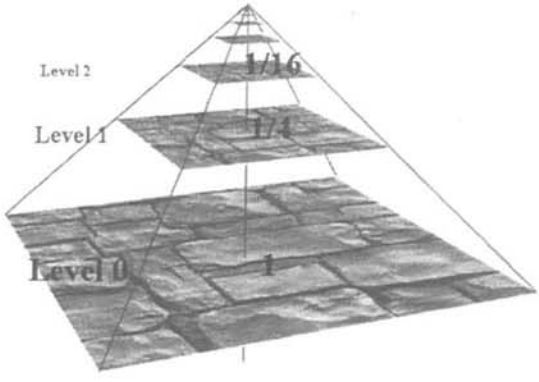


图 4-1 Mipmap 多分辨率纹理

在这种框架下存在以下缺点：

（1） 多边形可能细分达到没有纹理与之对应的程度。

（2） 需处理边界问题，因为一幅纹理的滤波在边界无法顾及其它相邻区域纹理的像素，因此在过滤后两相邻纹理间会出现不连续现象。

（3） 这种几何模型与纹理映射的过紧结合，使得对纹理的任意改动都会导致大量重复工作。

针对 Mipmap 的一些缺点，Tanner 提出了 Clipmap 方法[28]。Clipmap 方法首先对原始大纹理创建 Mipmap，然后选择一个分割尺寸，再把每一个 Mipmap 层分割成矩形网格状的片，避免了传统方法先把大纹理分割成小纹理再进行

Mipmap 过滤，而必须处理纹理边界的做法。

Clipmap 方法首先在 0 层（最高分辨率层）指定一个裁剪中心，裁剪中心是一个任意的纹理坐标，其它各层可由此导出，各层中心都沿一条线对准，从 0 层的中心直到 Mipmap 的顶点。在 Mipmap 的每一层，通过裁剪中心和裁剪尺寸的定义就精确地选择了要裁剪的区域，裁剪下来的这些固定大小的矩形区域叫裁剪区。裁剪后的 Clipmap 的整个形状呈方尖塔形，它由两部分构成，即裁剪图栈和裁剪图字塔，如图 4-2 所示。

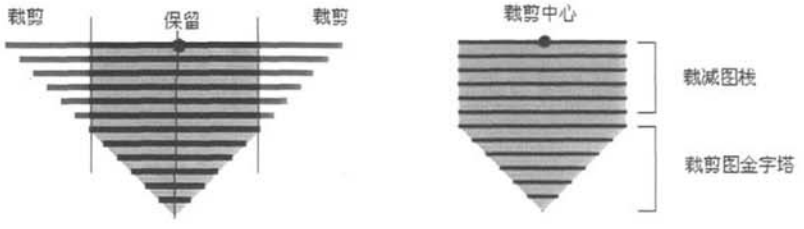


图 4-2 Clipmap 裁剪机制

Clipmap 的优点在于没有把整个影像金字塔的纹理装入内存，而是将裁剪中心周围一定范围内的纹理通过裁剪装入内存。通过这种方法，分辨率最高的纹理只覆盖了几何模型的一小部分，而分辨率较低的纹理占了大部分，需要贴图的部分只是 Clipmap 的一小部分。

裁剪纹理的实现思想对于解决图形学中相关问题有非常大的指导意义，这种方法不仅对显示任意大范围的纹理是重要的，同时还能减少几何建模工作。裁剪纹理采用动态从大量数据上裁剪一小部分用于显示，使用了一些缓存预测算法思想提高漫游的实时性和运用环形装载模式进行数据重用等，这些方法对处理大量几何数据的显示也很重要，利用裁剪纹理的纹理数据显示算法，整体的数据流可以被轻易地以应用需要按一定大小裁剪。几何裁剪图这种 LOD 算法正是借鉴了这一思想。

4.2 几何裁剪图算法

4.2.1 多分辨率模型

几何裁剪图算法借鉴了 Clipmap 的思想，将高度图多分辨率模型存储在一个如图 4-3 所示的高度图金字塔中，该金字塔由多分辨率的 m 个层次以嵌套形式构成，表示 m 个不同级别的分辨率。位于金字塔最上层的是分辨率最低的一层，

从上往下分辨率逐级提高，位于最底层的金字塔层存储的是分辨率最高的高度图。这些金字塔的各层都以视点为中心，投影到屏幕空间中时，紧靠视点的是分辨率最高的一层，分辨率越低则离视点距离越大，如图 4-3 右边所示。

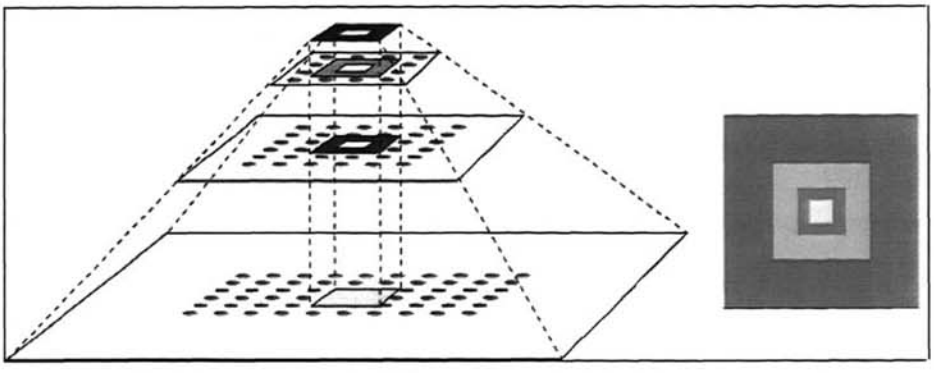


图 4-3 几何裁剪图金字塔

几何裁剪图金字塔的每一个层次都是包含 $N \times N$ 个顶点的数组，称为一个裁剪图层。由于网格数相同，则分辨率高的层网格间距较小，分辨率低的层网格间距较大，分辨率高的层嵌套于次高的层中，如图 4-4 所示。将这些裁剪图层分别编号，最粗糙的层标号为 $l=0$，分辨率最高的层标号为 $l=m-1$。

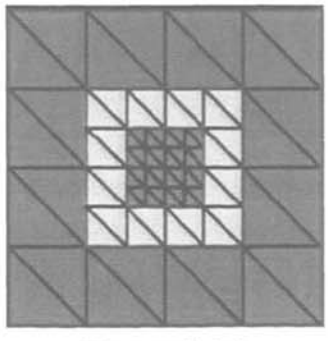


图 4-4 网格嵌套

裁剪图层中的顶点数据存储在显存中的顶点缓存中，由于网格非常规则，在绘制时可以很方便地将多个三角形合成一个三角形带，这样绘制将更加高效。当观察者移动位置时，相应的裁剪图也必须作出更新，这样才能使观察区总是以视点为中心。为了能够不断地进行有效增量更新，只有刚进入视野的顶点数据被加入进来，替换掉那些不再被观察者看见的区域。顶点数据以环形数组的方式进行存取，通过在 x 和 z 方向上的求模操作形成环形编码来进行读取操作。

4.2.2 裁剪区定义

对于每一个裁剪图层次，我们定义一系列矩形区域，如图 4-5 所示。裁剪区域（Clip Region）是世界空间内每一层存储的 $N \times N$ 规则网格数据。活动区域（Active Region）是位于裁剪区内期望渲染的区域，为一个以观察者为中心的 $N \times N$ 区域。当观察者移动时，我们通过更新每一层裁剪区域来匹配期望的活动区域。但是，这样的更新在快速移动时是很耗费的，可以让裁剪区域落后于观察者，在需要时裁剪活动区域到现有数据范围。最终，渲染区域（Render Region）被设计成环形的，如图中绿色斜纹部分，第 l 层渲染区域的外部边界是第 l 层活动区域 *active_region(l)*，内部边界则是第 l+1 层活动区域 *active_region(l+1)*。对于分辨率最高的 m-1 层，第 m 层活动区域 *active_region(m)*被定义为空，即分辨率最高层为实心正方形区域。

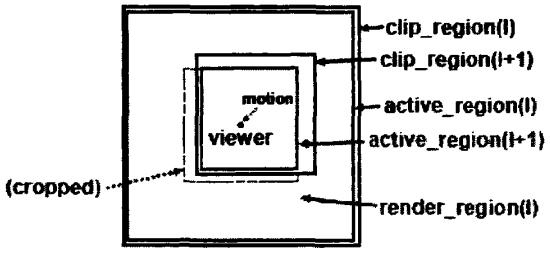


图 4-5 裁剪区域

为了施加表面光照，每一个裁剪图层都拥有自己单独的法向图，当裁剪图进行更新后，法向图也相应地作出更新。

4.3 算法实现

4.3.1 算法流程

根据 FFT 水波模型生成的高度场，经过 B 样条插值可以获得高度场多分辨率模型。在建立水面高度场多分辨率模型后，几何裁剪图算法以视点为中心指定一个裁剪中心，通过裁剪中心和裁剪尺寸就确定了各个裁剪图层的裁剪区。根据视点的方向，可以确定裁剪区内的活动区，如果活动区位置发生改变，则需要更新相应的网格，然后裁剪新的活动区域到裁剪区，之后渲染出来，下面步骤是每一帧算法的主要步骤：

1. 根据视点和裁剪区检测期望的活动区域；

2. 更新网格信息；
3. 裁剪活动区域到裁剪区域，并且渲染；

4.3.2 活动区域计算

几何裁剪图以视点为中心的更新主要是通过对每一层的裁剪图选择合适的活动区来实现的。我们通过一个简单的策略，对于在世界空间中网格间距为 $g_l = 2^{-l}$ 的每一层 l，我们让期望的活动区域为以视点为中心的 $Ng_l \times Ng_l$ 区域。也就是说，期望的剪切图被定位到以观察者为中心，我们希望渲染每一层的全部区域。

我们通过考虑屏幕空间三角形大小和剪切图尺寸之间的关系来确定裁减图的大小。我们假设网格平面有很小的坡度，所以每一个三角形近似一个大小为 g_l 的直角三角形。对于世界空间中任意的可见的点，其屏幕投影大小与屏幕空间的深度成反比例。如果视线方向是水平的，屏幕深度可以在 XZ 平面量度。观察者坐落于 l 层渲染区域（*render_region(l)*）的中心，它的外边界长度为 Ng_l，内边长度为 $Ng_l/2$。对于 90 度宽的视野，屏幕空间平均深度（遍及各个方向）大约 $0.4Ng_l$，所以近似的屏幕空间以像素表示的三角形大小如下：

$$s = \frac{g_l}{0.4Ng_l}\frac{W}{2\tan\frac{\varphi}{2}} \qquad \text{公式（4-1）}$$

W 是窗口大小，我们定义 W=640 像素，φ=90 度，当剪切图尺寸 N=256 就能获得比较好的结果，这符合一个屏幕空间三角形 3 像素的大小。

当我们的视线方向不是水平时，渲染区域 l（render_region（l））的屏幕深度大于上面期望的 $0.4Ng_l$，并且因此屏幕空间三角形变得小于 s，如果直接从地形上空向下俯视，三角形大小是很小的，并且明显变得混淆难以分辨。解决的方法是不渲染不必要的好的精度的层次。特别地，我们计算观察者处于地形上方的的高度通过访问有效的最好的裁剪图层次。对于每一个层次 l，如果观察者高度大于 $0.4Ng_l$，我们就把该层的活动区域设置为空。

把活动区域简单定位于观察者中心的缺点是，当视域变窄时，裁剪图的大小 N 必须增大，解决的方法是相对于视锥调整裁剪图的位置和大小。这里我们采取了不考虑视觉变窄的简单替代方案，直接选择以观察者为中心的区域，因为它们能够让观察立即围绕当前视点旋转。这已经能够满足多数应用的需求，比如飞行模拟器通过可以转换遥控杆让用户观察各个方向。

综上所述，活动区域定义为以视点坐标为中心的 $Ng_l \times Ng_l$ 区域，如果观察者高度大于 $0.4Ng_l$，我们把活动区域设置为空。

4.3.3 裁剪图更新

当期望的活动区域随着观察者的运动而改变时，裁剪区域应该同样跟着改变。注意到我们采用的是上面提及的环形方式编址，这样在改变一个层次时不必要复制旧的数据，而是简单填充新的 L 形暴露的区域。

我们对裁剪区域定义下面约束：

1. $clip\_region(l+1) \subseteq clip\_region(l) \otimes 1$，即 l 层裁剪区域属于 l+1 层裁剪区域，$\otimes$ 表示一个单位的腐蚀。我们需要裁剪区域嵌套以进行从粗糙到好的地形几何预测，预测时至少需要在各个方向维持一个网格的间距。
2. $active\_region(l) \subseteq clip\_region(l)$，即 l 层活动区域属于 l 层裁剪区域，因为渲染数据必须是呈现的在裁剪图中的数据的一个子集。
3. l 层活动区域的边界必须落在偶数的顶点上，以便于不会在和比较粗糙的 l-1 层的边界之间存在缝隙。
4. $active\_region(l+1) \subseteq active\_region(l) \otimes 2$，$l$+1 层活动区域属于 l 层活动区域，并且渲染区域必须至少占两个格网单元宽度，以允许在两个层次之间进行渐变。

当观察者快速运动时，不仅活动区域的位置会发生改变，该活动区域需要的分辨率也可能发生变化，这就需要将活动区域进行裁剪，通过裁剪来更新相邻的裁剪区域。我们以从粗糙到好的顺序更新各个层次，在达到需要的处理精度时停止，一般选择以更新的采样数量超过 N^2 时停止。

更新裁剪图的算法用伪码可以表示为：

```
//更新裁剪图
从粗糙到好顺序遍历每一个层次
{
    裁剪 l 层活动区域到 l 层裁剪区域
    裁剪 l 层活动区域到 l-1 层活动区域
}
```

4.3.4 裁剪图渲染

在对裁剪图进行更新时，活动区域被裁剪到裁剪区，并[illegible]至 4。注意到，如果一个活动区域 k 为空，则构造的活动区域 l 当满足 $l>k$ 时同样为空。对于分辨率更高的层次，活动区域为空是相当普遍的，或者由于它们的

裁剪区域已经在时间内被更新了，例如观察者运动得很快；或者由于分辨率更高的层次是无必要的，例如观察者远高于地形表面。

既然分辨率高的层次是离观察者更近的，在根据确定的活动区域对裁剪图进行更新后，我们按照相反的顺序即从分辨率较高到更粗糙的顺序对裁剪图进行渲染，这样还可以利用硬件进行遮挡消隐。l 层渲染区域被分割为四块矩形区域，它们分别通过绘制一个三角形带来进行渲染，如图 4-6 所示。

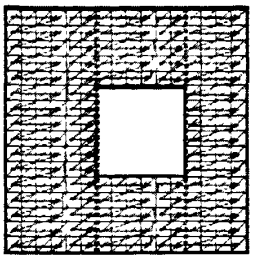


图 4-6 渲染区域的三角带生成

渲染裁剪图的算法用伪码表示为：

```
//渲染裁剪图
从好到粗糙顺序遍历每一个层次
{
   l 层渲染区域=l 层活动区域-（l+1）层活动区域
   渲染该渲染区域
}
```

4.3.5 渐变区域可视连贯性

在绘制裁剪图时，由于相邻的层分辨率不同，在边界上的网格三角形并不是完全匹配的，会出现不同渲染区域之间的裂缝。为了除去裂缝并且提供时间上的连续性，我们让每层渲染区域外边界附近的三角形网格发生渐变，以便于过渡到较为粗糙的 l-1 层。渐变是相对于视点空间网格坐标的函数，所以这种渐变不是基于时间的，而是连续跟踪观察者的位置。

通过实验发现，当渐变区的宽度 w 约为 N/10 个网格单元时效果很好，如果 w 更加小，层次边界会变得明显，如果 w 比较大，好的细节会存在不必要的丢失。如果分辨率高的 l+1 层活动区域太接近观察者，我们计算 $w=\min(10n,\min\_width(l))$，从条件 4 可以得出，$min_width(l)$最少为 2。

为了便于进行渐变，实际上每个顶点被存储为（x，y，z，yc）向量，yc 是

该顶点在下一个相邻的更粗糙的层中的高度，我们通过下面的插值公式进行渐变：

$$y' = (1-\alpha)y + \alpha yc, \alpha = \max(\alpha_x, \alpha_z) \quad \text{公式(4-2)}$$

$$\alpha_x = \min\left(\max\left(\left(\left|x - v_x^l\right| - \left(\frac{x_{\max} - x_{\min}}{2} - w - 1\right)\right)/w, 0\right), 1\right) \quad \text{公式(4-3)}$$

$$\alpha_z = \min\left(\max\left(\left(\left|z - v_z^l\right| - \left(\frac{z_{\max} - z_{\min}}{2} - w - 1\right)\right)/w, 0\right), 1\right) \quad \text{公式(4-4)}$$

这里的(v_x^l, v_z^l)指示在 l 层裁剪区域中视点的连续的坐标，$(x_{\min}, z_{\min})$和$(x_{\max}, z_{\max})$是以整数表示的 l 层活动区域的范围，期望的插值系数α的取值区间为 0 到 1 之间。

尽管几何渐变能够避免裂缝，但是边界上的 T 型连接仍然存在，为了缝合相邻的层以形成紧密网格，我们使用在边界上渲染零面积三角形的简单方案。

4.4 实验结果

采用几何裁剪图的 LOD 控制算法，在保持以视点为中心的区域的网格拥有高分辨率的条件下，适当降低视点远处网格的分辨率。可以从两个方面来理解几何裁剪图算法的网格简化功能：第一，在绘制的网格实际面积一定的情况下，几何裁剪图算法下网格分辨率由外而内逐级增加，在视点中心几何裁剪图算法可以获得很高的分辨率，因此，在几何裁剪图算法控制下，实际的分辨率提高；第二，在最高级别网格分辨率相同的情况下，裁剪图网格覆盖的面积将随裁剪图层次的增加而增加，每增加一层裁剪图，实际的覆盖面积将增加为原来的四倍，这样在最高分辨率相同的情况下几何裁剪图算法绘制面积增大。

图 4-7 和图 4-8 是当观察者由比较平滑的区域进入高度起伏较大区域时网格多分辨率模型的变化情况。图 4-7 中区域比较平滑时，裁剪图活动区域层次较少，当观察者移动到图 4-8 中所示区域时，根据绘制点到视点的距离，所处的裁剪图层次会发生变化，当视点移近时到一定值时，裁剪图活动区域层数便会增加。

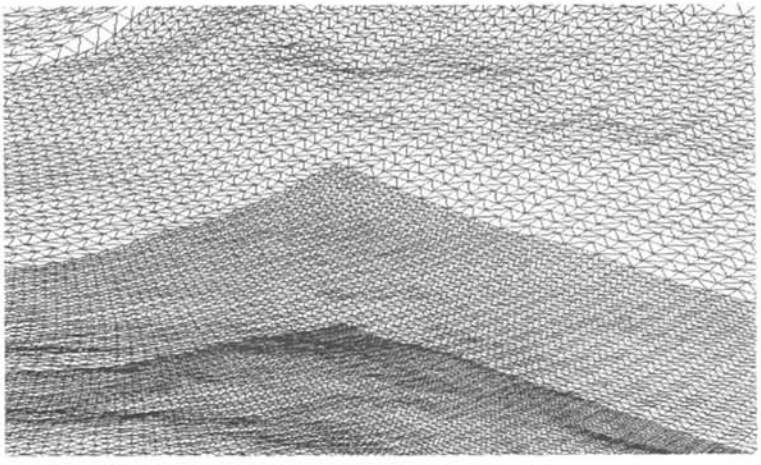


图 4-7 视点处于平滑区域

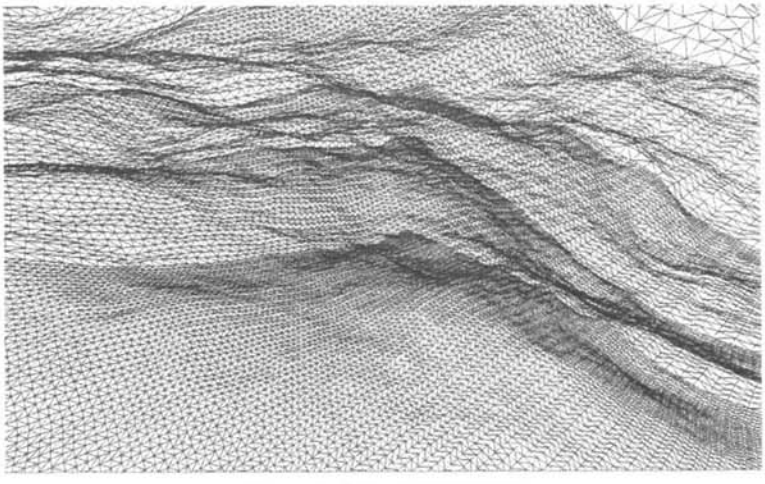


图 4-8 视点处于非平滑区域

使用几何裁剪图算法，绘制网格时能保证良好的更新。当观察者迅速移动的时候，更新带宽会成为绘制速率的瓶颈。几何裁剪图算法中固定的裁剪图大小能够收缩渲染区域，以减少渲染负荷，这样可以控制纹理裁剪图的更新带宽。而且在一次预算当中更新尽可能多的层次，使得观察者移动时需要更新的裁剪图层次减少。

由于网格采样完全独立，不依赖于粗糙度，无需参数用于动态调整，所以使得渲染速度非常稳定，这样几何裁剪图算法在以视点为中心时，能提供非常稳定的帧速率，这是几何裁剪图算法很大的优点之一。

图 4-9 和图 4-10 当观察者围绕观察点快速旋转时网格多分辨率模型的变化情况，二者在绘制时的帧速率基本相同。几何裁剪图的采用的顶点存取的环形寻

址方式，使得观察者旋转时数据的更新仅发生在已有的裁剪图层次中，这样旋转时的更新速度不会下降。而像以前的分块四叉树的方法，观察者旋转时，若在不同的分块间转换，则数据更新量比较大，自然会影响到绘制速率。

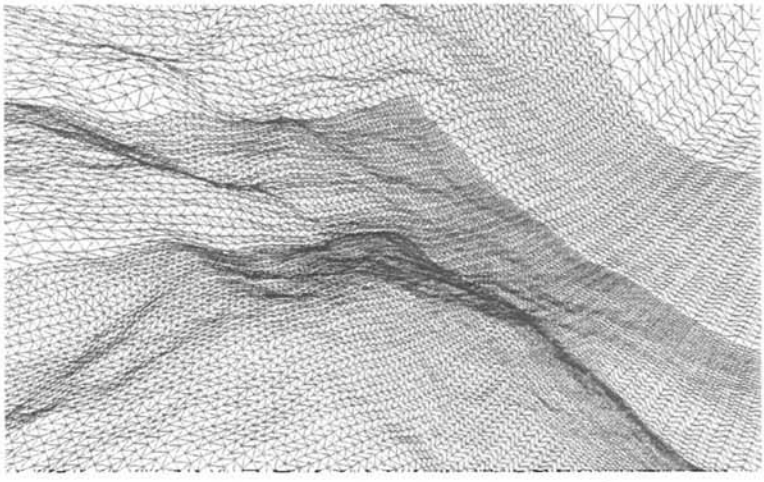


图 4-9 观察者旋转时水面网格一

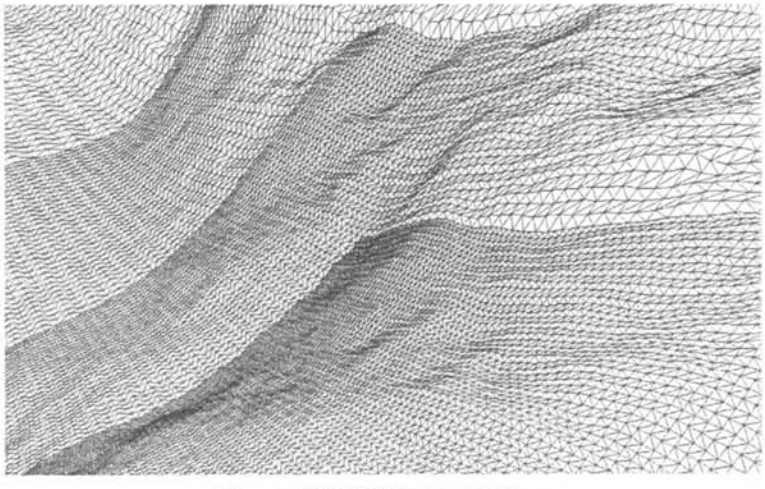


图 4-10 观察者旋转时水面网格二

在绘制速率上总的来说，裁剪图层次数 m 越大，花费在建立多分辨率模型上的时间越多，绘制速率越低。当层次数 m 固定时，裁剪图网格分辨率的大小是影响绘制速度的主要因素，另外由第三章我们知道，分辨率太高，由 FFT 算法生成原始网格也将会比较费时。裁剪图层次数目不同时，不同的网格分辨率下，得到的几何裁剪图算法绘制速度如表 4-1 所示。

表 4-1 几何裁剪图算法绘制速度

裁剪图层数 / 裁剪图分辨率	m=6		m=8	
	三角形数目	绘制速率（fps）	三角形数目	绘制速率（fps）
128×128	177419	158.3	225806	112.7
256×256	715275	91.3	910350	71.9
512×512	2872331	48.7	3655694	30.4

总的来说，通过几何裁剪图进行水面网格的 LOD 控制，能够以视点为中心进行水面的更新，当水面面积很大时仍然能够达到稳定的实时帧速率。

4.5 本章小结

本章基于几何裁剪图的方法对水面高度场的 LOD 控制进行了研究。在生成水面高度场基础之上，利用几何裁剪图算法对原始水面网格生成多分辨率模型。实验证明，通过几何裁剪图进行 LOD 控制能够以视点为中心进行水面的更新，当水面面积很大时仍然能够达到稳定的实时帧速率。

第五章 光照效果的实现

5.1 光照过程

要生成真实的水面效果，水面光照必不可少。只有在光线的作用下，水面才会显现出动态的环境变化。水面光照环境十分复杂，为了绘制出的水面效果更加真实，我们考虑的水面光照环境由四个部分组成：大气、太阳、水面和水面以下的水体。光线从光源如太阳出发投射到水面上，一部分的光线在水面发生镜面反射，一部分的光线则在水面发生折射进入水体。折射进入水体的光线在水中还会被水中的杂质散射，一部分被散射的光线又会透出水面，射向大气中。由于水面的不规则，一些被反射或者散射的光线甚至会二次到达水面，重新被水面反射或折射出去，如图 5-1 所示。为了简单起见，对光线被水面二次反射或折射的情况将不予考虑，这不仅减轻了计算负担，在多数情况下也能保证绘制效果的精确度。

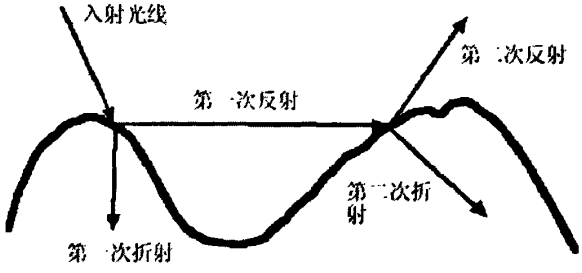


图 5-1 水面二次反射和折射过程

在不考虑二次反射和折射的情况下，光线和水面的相互作用可以以水面为分界线分为两个部分。首先是水面上的光照，我们把水面看成是由许多小三角平面构成的，来自天空和太阳的光线射入水面时，被分成两部分，一部分被直接反射进入空气中，另一部分则折射进入水中，折射光线的方向可由 Snell 定律来确定，反射光线和折射光线各自所占的比率由 Fresnel 公式确定。另一部分是水面下的光照，这一部分由于属于水下场景，在绘制水面时我们将忽略这部分光线[31]。

5.2 水面光照效果

5.2.1 反射

水面的镜面反射非常简单，入射视线 I 从视点到达景物表面 O 点，O 点的法向量为 N，经物体的表面反射后生成一条反射光线 R。

对于给定的视线向量 I，和水面的法线 N，反射的光线 R 可以表示为：

$$R = I - 2N(N \cdot I) \qquad \text{公式（5-1）}$$

根据反射光线，就可以从相应的反射纹理中读出反射颜色值。

仅仅有环境反射的颜色是不够的，我们还要考虑到全局光源的光线，比如太阳光，在水面产生的反射。当太阳照耀在海面上，会形成强烈的高光。与环境反射类似，可以通过反射光线来从相应的高光纹理来读取反射的太阳光线颜色值。如果太阳的位置也是不断变化的，则需要使用原始的 Phong 光照模型来直接计算。全局的反射反映了环境中的物体在水面的反射情况，另外还需考虑场景中的物体在水面上的局部反射。与全局反射不同的是，局部反射不只是与反射光线的角度有关，还涉及到反射点在水面的具体位置，反射点不同，产生局部反射效果是不一样的。采用光线跟踪的方法计算局部反射是非常费时的，我们采取简化的方法。首先，在绘制水面前，我们将水面近似看成一个平面，根据镜面反射可以很简单地获得物体的反射图像，作为物体的局部反射纹理；之后，以水面的实际屏幕坐标为纹理坐标，可以在绘制水面时获得局部反射值，通过读取凹凸映射图，对纹理坐标施加一个与表面法向量相关的偏移值，可以控制生成的局部反射图像的扭曲程度，让水面的凹凸效果显得更加真实。

最后得到反射光线的公式表示为：

$$C_{ref} = C_{ref}^{local} \cdot \alpha + \left(C_{ref}^{global} + C_{ref}^{sun}\right) \cdot (1 - \alpha) \qquad \text{公式（5-2）}$$

其中 C_{ref} 是水面反射的颜色，C_{ref}^{local} 表示局部反射的颜色，C_{ref}^{global} 和 C_{ref}^{sun} 分别表示环境反射和太阳反射，α 表示某一点上局部反射在反射光线中所占的比例。如果希望局部反射更加强烈，则 α 值接近 1。

5.2.2 折射

由于空气和水的密度不同，光线到达水面时方向便会发生改变产生折射，如图 5-2 所示。我们将会使用 Snell 定律来计算折射光线，Snell 定律描述了两种媒质的分界面上光线的折射规律，其用方程式表示为：

$$n_1 \sin\theta_1 = n_2 \sin\theta_2 \qquad \text{公式（5-3）}$$

其中 θ_1 为入射角（视线和表面法线之间的夹角），θ_2 为折射角（反射向量和表面法线之间的夹角），n_1 和 n_2 是两种不同物质的折射指数，空气和水的折射指数分别为 1.0 和 1.33。

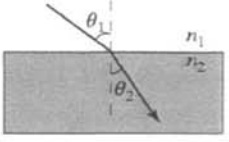


图 5-2 光线折射

我们由 Snell 定律得到折射光线的方向，折射颜色也由两部分组成，即水体本身的颜色和水下物体如礁石等的颜色。由于人眼不可能到深水如海洋底的物体的颜色，所能看到的只能是水面以下一定范围内的物体颜色。水体本身由于入射光线方向，视线方向和水中所含矿物质成分的不同，也呈现不同的颜色。我们只考虑了固定的几种水体颜色，并事先计算当水面是一个纯粹的平面时，水面以下物体的颜色和水体本身颜色的结合，存储在一个折射纹理中。

5.2.3 Fresnel 现象

Fresnel 现象可以说是渲染真实感水面最重要的视觉效果之一，是指当光线到达水面时，有部分的光线反射，部分的光线折射，Fresnel 现象展示了光线反射和折射的混合。Fresnel 公式定义了反射光线颜色和折射光线颜色在最后水面颜色中的相对比重，称为 Fresnel 系数。Fresnel 系数的取值范围是[0.0，1.0]，可由公式（5-4）得到。

$$F = \frac{(g-k)^2}{2(g+k)^2}\left(1+\frac{(k(g+k)-1)^2}{(k(g-k)+1)^2}\right), \left(k=\cos\alpha, g=n_1/n_2+k^2-1\right) \qquad \text{公式（5-4）}$$

其中 k 和 g 是 α、n_1 和 n_2 的函数，α 是入射光线与表面法线的夹角，n_1 和 n_2 分别是空气和水的折射率，近似我们取它们的比值为 1：1.33。

当由表面法向量变化时，入射角不停地变化，Fresnel 系数就产生变化，从而导致了海水表面颜色组成的变化。举例来说，如果 Fresnel 系数是 0.8，那么反射光线比例为 80%，而通过水下折射到达海水表面的光线的比例为 20%。当视线与法向量的夹角接近 90 度时，Fresnel 系数达到最大；当视线与法向量的夹角接近 0 度时，Fresnel 系数最小。

根据 Fresnel 系数，我们可得到包含了反射和折射的水面颜色方程为：

$$C_{water} = FC_{ref} + (1-F)C_{refract} \qquad \text{公式（5-5）}$$

其中 C_{water} 表示水面颜色，C_{ref} 是水面反射的颜色，$C_{refract}$ 是水面折射的颜色，F 是 Fresnel 系数。

5.3 光照效果实现

本文使用立方贴图纹理来模拟水面的周围的环境，利用了现代可编程 GPU 的强大处理功能，使用 Cg 语言（C for Graphics）进行编程。

最新一代的图形处理器的图形处理流水线中同时提供了顶点级和像素级的可编程能力，图 5-3 显示了当今图形处理器所使用的图形硬件流水线。在流水线中，原来固定的硬件顶点处理单元和片断处理单元分别被可编程的顶点处理器和可编程的片断处理器代替。Cg 所提供的则是一种类似于 C 的图形编程语言和一个编译器，将光照算法翻译成图形处理器硬件能够执行的形式。

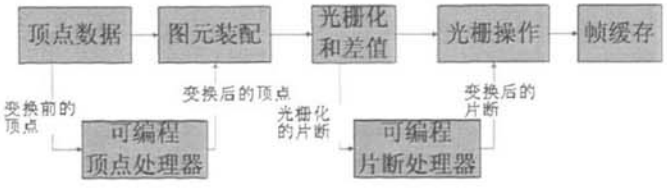


图 5-3 图形硬件流水线

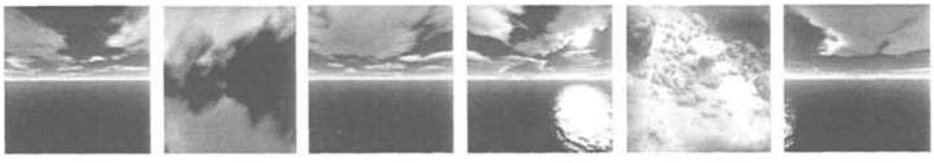
5.3.1 立方贴图纹理

近来的图形处理器都支持一种被称为立方贴图（Cube Mapping）的环境映射。

环境映射是指将周围的环境事先以纹理图像的方式存储在显示内存中，以纹理映射的方式取出。当你看一个高度反射的物体例如铬合金球体的时候，你看到的不是物体的本身，而是物体反射的它周围的环境。假如光线从眼睛出发，到达高度反射的表面上某一点，然后被反射到环境中去，这样眼睛看到的不是表面，而是环境在反射光线的方向上看上去的样子。

一个立方贴图不是由一幅而是六幅正方的刚好能组合在一起形成一个立方体的表面的纹理图像组成，六幅图像一起形成了用来编码环境贴图纹理的全方位图像。一个 2D 纹理只能映射一个 2D 纹理坐标集到一幅单独的纹理图像，要映射一个立方贴图纹理需要一个 3D 方向向量，可以把这个向量看成是从立方体中心射出的光线，当光线与立方贴图的六个面之一相交时，读取的就是该交点的纹理值。

除了实时的渲染环境贴图之外，另外一个比较常见的做法是从文件中读取预先生成的环境贴图。使用静态贴图实现起来比实时渲染简单，它不需要把场景绘制到环境贴图上，只需要在创建环境纹理的时候从硬盘文件中载入即可。使用静态贴图会降低一定的真实性，但是可以大大降低渲染开销，是一种适合在低端环境中使用的技术。图 5-4 是在程序中使用的静态立方贴图：



（a）x 轴正　（b）y 轴正　（c）z 轴正　（d）x 轴负　（e）y 轴负　（f）z 轴负

图 5-4 海面立方贴图

5.3.2 实现步骤

光照效果的计算大部分都由相应的 Cg 程序来完成，Cg 中的许多内置的光线计算函数，我们都可以直接利用。蚀刻效果由于要进行光线跟踪，是在 CPU 中单独计算的，不过由于跟踪过程比较简化，对 CPU 的计算负担增加不多。以下是具体的实现：

1. 光线计算

入射光线：入射光线 I 是从眼睛（坐标 eyePositionW）到顶点（坐标 postionW）的向量：I=postionW–eyePositionW。

反射光线：根据入射光线 I 和表面法向 N，有反射光线 R=reflect(I, N)。

折射光线：根据入射光线 I 和表面法向 N，以及空气和水的相对折射率

etaRatio(1/1.33)，折射光线 T=refract(I, N, etaRatio)。

2. 反射

全局反射：C_{ref}^{global}=texCUBE(sky, R)，根据反射光线对立方环境贴图采样。

太阳反射：C_{ref}^{sun}=tex2D(sun, R)，直接根据反射光线对高光纹理采样。

局部反射：C_{ref}^{local}=tex2D(local, screencoords+ strength*N) ，根据法向 N 对纹理坐标施加扰动，然后局部反射纹理进行采样。

最后根据公式（5-2）便可计算出反射颜色值。

3. 折射和 Fresnel 现象

根据水体本身颜色以及由折射光线对立方环境贴图采样可以得到折射颜色值，然后根据公式（5-4）可以计算出 Fresnel 系数，再由公式（5-5）便可计算出带有 Fresnel 效果的颜色值。

5.4 绘制结果

光照实验同样在不同的条件下进行，首先我们不考虑任何反射和折射现象，仅是根据海水自身的性质对水面上色，作为海水的基色，如图 5-5 和图 5-6 所示，两幅图的区别仅在于风速的大小不同，图 5-5 中，风速与图 5-6 中相比要小一些，这从波浪的起伏幅度大小上我们可以看出。

图 5-5 风速较小时水面

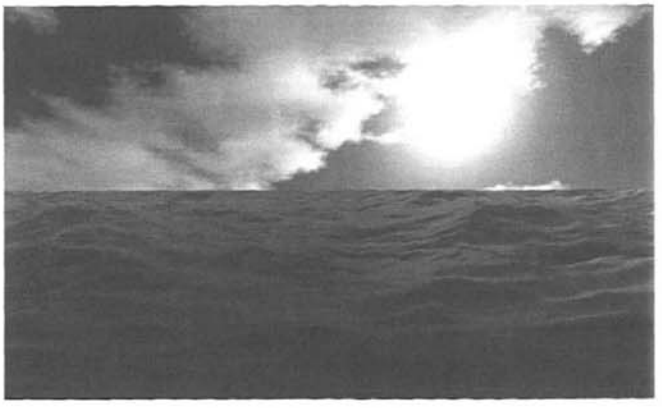


图 5-6 风速较大时水面

如果只是简单地给海面上色，海面各处显然都是同一颜色，不仅看上去比较单调，通常也与海面的真实光照情况有很大的差别，因此给海面添加反射、折射等效果是非常必要的。图 5-7 和图 5-8 分别是施加了反射效果和折射效果之后的绘制结果。

图 5-7 中，虽然水面比较平静，但是水面对周围环境的反射很明显，由于对环境的反射，水面的颜色已经与反射颜色进行融和，在太阳反射比较集中的地方，可以看到反射的太阳高光效果。

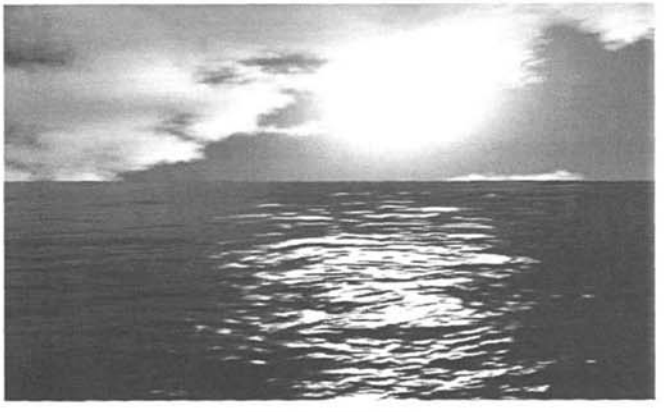


图 5-7 水面反射

图 5-8 中，加入了光线折射并考虑 Fresnel 现象，在图片的右下角我们可以看到物体在水下的一部分。

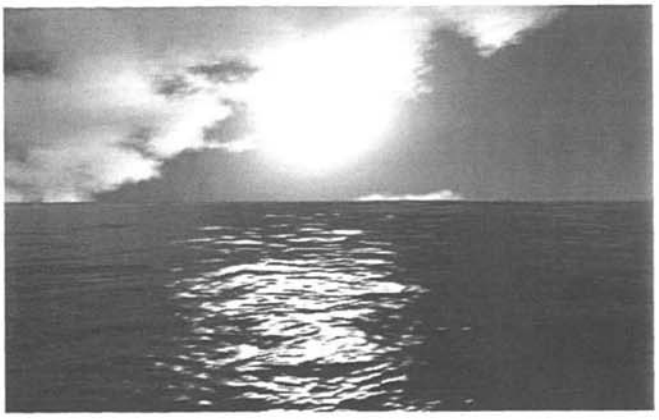


图 5-8 水面折射

从绘制结果图中可以看出，水面场景的反射、折射效果都比较明显，基本上可以满足普通的水面实时模拟的需要。

施加光照效果的海面绘制速率与纯 LOD 控制的绘制相比，帧速率必然有所降低，不同的网格分辨率下，得到的平均绘制速度如表 5-1 所示。

表 5-1 不同裁剪图分辨率下绘制速度

裁剪图分辨率	128×128	256×256	512×512
绘制速率（fps）	93.0	61.7	41.4

在 Xudong Yang 等人的论文[21]中，使用随机噪声来生成水面波形，采用了 Ulrich 改进的分块四叉树方法进行海面网格的 LOD 控制，在 256×256 分辨率下最后达到的绘制帧速率为每秒 55 帧左右。我们实现的绘制系统除了生成的水面波形效果肯定比他们用随机噪声生成的要好的多外，在该分辨率下绘制速率与之相比要稍高一些。

5.5 本章小结

本章在前面两章生成水面高度场、进行高度场网格的 LOD 控制后，对水面光照效果的实现进行了研究。在不考虑二次反射和折射的情况下，实现了水面的反射、折射、Fresnel 现象等效果，可以保证水面模拟的逼真度。通过水体光照效果的施加，才真正实现了大面积的水体场景模拟。

第六章 总结和展望

本文针对海水绘制中存在的实时性和真实性问题进行了研究，在此基础之上实现了一个由 LOD 控制的大面积水面绘制框架。具体的研究内容主要包括：

1. 基于快速傅立叶变换模型对水面高度场生成方法进行了研究，使用快速傅立叶变换建模能在任意的时间点上生成需要的水面高度场，能得到比较逼真的水波效果。在根据快速傅立叶变换模型的相关理论生成了水面高度场后，针对生成的水面比较平滑的问题，提出了水面高度场优化的方法，可以生成不同场景中的水面。

2. 基于几何裁剪图的方法对水面高度场的 LOD 控制进行了研究。在对原始水面网格生成多分辨率模型之后，利用几何裁剪图算法进行细节层次简化。实验证明，通过几何裁剪图进行 LOD 控制能够以视点为中心进行水面的更新，当水面面积很大时仍然能够达到稳定的实时帧速率。

3. 对水面光照效果的实现进行了研究。在不考虑二次反射和折射的情况下，实现了水面的反射、折射、Fresnel 现象，可以初步保证水面模拟的逼真度。通过水体光照效果的施加，真正实现了大面积的水体场景模拟。

经实验证明，该系统可以保证水面模拟的逼真度，并同时达到很好的实时速率，完全适合于三维应用中水体场景的模拟。

目前，真实感图形学技术发展非常迅速，针对本文的海水绘制实现，进一步的研究方向可以包括：

1. 基于 GPU 的几何裁剪图。本文采用的是在 OpenGL 扩展功能下实现的几何裁剪图算法，裁剪图的剪切、更新都是在 CPU 中完成，采用完全由 GPU 完成的几何裁剪图算法可以减轻 CPU 负担。

2. FFT 算法的 GPU 实现。目前在 GPU 上实现的一维 FFT 算法大约是 CPU 计算速度的四倍，不过二维的 FFT 算法在 GPU 上还没能实现，如果在 GPU 上进行二维的 FFT 计算，将使计算性能大幅提高。

3. 水体交互性应用研究。目前的很多三维应用特别是游戏特别注重交互性能，在这种要求下，水体的交互现象必不可少。本文仅简单地实现了水面尾迹，可以采用粒子系统等技术，实现水面的泡沫、浪花等效果。

参考文献

[1]Perlin K.，An Image Synthesizer，Computer Graphics，1985，19(3)：287~296

[2]Fournier A.，Reeves WT，A Simple Model of Ocean Waves，Computer Graphics，1986，20(4)：75~84

[3]Peachey D.R.，Modeling Waves and Surf，Computer Graphics，1986，20(4)：65~74

[4]Mastin G.A.，Watterberg P.A.，Mareda JF，Fourier Synthesis of Ocean Scenes，IEEE Computer Graphics and Applications，1987，7(3)：16~23

[5]Tessendorf J.，Simulating Ocean Water，SIGGRAPH Course Notes，1999，1~18

[6]Kass M.，Miller G.，Rapid，Stable Fluid Dynamics for Computer Graphics，Computer Graphics，1990，24(4)：49~57

[7]Chen J.X.，Lobo N.V.，Real-Time Fluid Simulation in a Dynamic Virtual Environment，IEEE Computer Graphics and Applications，1997，17(3)：52~61

[8]Foster N.，Metaxas D.，Realistic Animation of Liquids，Graphical Models and Image Processing，1996，58(5)：471~483

[9]Clark J.H.，Hierarchical Geometric Models for Visible Surface Algorithms，Communications of the ACM，1976，19(10)：547~554

[10]Lindstrom P.，Koller D.，Ribarsky W.，Real-Time，Continuous Level of Detail Rendering of Height Fields，Computer Graphics，1996，30(3)：109~118

[11]Duchaineau M.，Wolinsky M.，ROAMing Terrain：Real-time Optimally Adapting Meshes，IEEE Visualization，1997，81~88

[12]Rottger S.，Heidrich W.，Real-time Generation of Continuous Levels of Detail for Height Fields，In Proceedings of the 6th International Conference in Central Europe on Computer Graphics and Visualization，1997，315~322

[13]Hoppe H.，View-Dependent Refinement of Progressive Meshes，Computer Graphics，1997，33 (3)：189~198

[14]Hoppe H.，Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering，IEEE Visualization，1998，35~42

[15]Willem H.，Fast Terrain Rendering Using Geometrical MipMapping，http://www.flipcode.com/tutorials/geomipmaps.pdf，2000

[16]Lindstrom P.，Pascucci V.，Visualization of Large Terrains Made Easy，IEEE Visualization，2001，363~370

[17]Ulrich T.，Chunked LOD：Rendering Massive Terrains using Chunked Level of Detail Control，SIGGRAPH Course Notes，2002

[18]Losasso F.，Hoppe H.，Geometry Clipmaps：Terrain Rendering Using Nested Regular Grids，ACM Transactions on Graphics，2004，769~776

[19]Asirvatham A.，Hoppe H.，Terrain Rendering Using GPU-Based Geometry Clipmaps，GPU Gems 2，Addison-Wesley，2005，27~46

[20]Clasen M.，Hege H-C，Terrain Rendering using Spherical Clipmaps，Symposium on Visualization，2006，91~98

[21]Goss，Xudong Yang，Xuexian Pi，GPU-Based Real-time Simulation and Rendering of Unbounded Ocean Surface，Ninth International Conference on Computer Aided Design and Computer Graphics，2005

[22]Harris M.J.，Fast Fluid Dynamics Simulation on the GPU，GPU Gems，Chapter 38，Addison-Wesley，2004

[23]Youquan Liu，Xuehui Liu，Physically Based Fluid Simulation in Computer Animation，Pacific Graphics，2004，171~173

[24]Foster N.，Metaxas D.，Realistic Animation of Liquids，Graphical Models and Image Processing，1996，58(5)：471~483

[25]Stam J.，Stable Fluids，Proceedings of SIGGRAPH，New York：ACM Press，1999，121~128

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

[27]Miller G.，Pearce A.，Globular Dynamics：A Connected Particle System for Animating Viscous Fluids，Computers and Graphics，1989，13(3)：305~309

[28]Stam J.，Fiume E.，Depicting Fire and Other Gaseous Phenomena using Diffusion Processes，Proceedings of SIGGRAPH，Los Angeles，1995，129~136

[29]Tanner C.，Migdal C.，The Clipmap：A Virtual Mipmap，Computer Graphics，1998：151~158

[30]Mitchell J.L.，Real-time Synthesis and Rendering of Ocean Water，ATI Research Technical Report，2005

[31]Wand M.，Straßer W.，Real-Time Caustics，Computer Graphics Forum，2003，22(3)：611~620

[32]郭阳明，翟正军，陆艳红，虚拟场景生成中的LOD技术综述，计算机仿真，2005，22(12)：180~184

致　谢

本论文的工作是在我的导师孙济洲教授的悉心指导下完成的，孙济洲教授严谨的治学态度和科学的工作方法给了我极大的帮助和影响。在此衷心感谢多年来孙济洲教授对我的关心和指导。

张加万副教授悉心指导我们完成了实验室的科研工作，在学习上和生活上都给予了我很大的关心和帮助，在此向张加万老师表示衷心的谢意。

在实验及撰写论文期间，张红颖、白玉超等同学对我论文中的研究工作给予了热情帮助，刘磊、刘志辉对我的论文写作问题给予了细致的解答，在此向他们表达我的感激之情。

另外也感谢家人和其他朋友，他们的理解和支持使我能够在学校专心完成我的学业。