第34卷 第20期 计 算 机 工 程 2008年10月
Vol.34 No.20 Computer Engineering October 2008
·图形图像处理· 文章编号：1000—3428(2008)20—0233—02 文献标识码：A 中图分类号：TP393

基于GPU的水面实时渲染算法

王道臣[1]，万旺根[1]，唐经洲[2]，陈华杰[1]
(1. 上海大学通信与信息工程学院，上海 200072；2. 南台科技大学电子工程系，台湾 73502643)

摘 要：提出基于可编程图像硬件实时生成真实水面的渲染方法，通过实现水面建模、水面折射和反射完成整个渲染过程。在正弦波叠加的同时，利用2个凹凸纹理实现水面的动画效果，通过实时纹理映射技术实现水面的反射、折射和菲涅尔等水面光照效果。实验证明该算法能够很好地满足人们对真实感和实时性的要求，适用于虚拟现实中真实水面的生成。

关键词：可编程图形硬件；渲染；虚拟现实

Real-time Rendering Algorithm for Water Surface Based on GPU

WANG Dao-chen[1], WAN Wang-gen[1], TANG Jing-jou[2], CHEN Hua-jie[1]
(1. School of Communication and Information Engineering, Shanghai University, Shanghai 200072;
2. Department of Electrical Engineering, Southern Taiwan University of Technology, Taiwan 73502643)

【Abstract】This paper presents an algorithm to generate and render water surface in real-time based on programmable graphics hardware. The algorithm completes the rendering process by realization of water surface modeling and reflection and refraction of water. It models the water surface based on sine wave overlapped, using tow bumping-maps to achieve action effect of the water, realizes the reflection, refraction and fresnel illumination effects with texture mapping. Experimental result shows the algorithm meets the photorealism and real-time requirement very well and can apply to the generation of photorealistic water in visual reality.

【Key words】programmable graphics hardware; rendering; visual reality

1 概述

对自然景物的建模和渲染一直是虚拟现实的重要组成部分，也是计算机图形学研究的重点和热点。水作为自然景物的一部分，在增加虚拟现实的真实感和沉浸度方面有着非常重要的作用。国内外一些专家提出了很多关于水面生成与绘制的算法和技术，这些技术有些虽然能生成具有真实感的水面效果，但往往无法满足实时绘制的速度要求。有些能实时生成，但通常采用一些非常简单的建模和光照模型， 绘制出的水面效果真实感不强。随着计算机图形硬件性能的不断提高，复杂的图形计算已经逐步从 CPU 转向图形硬件的图形处理单元(Graphics Processing Unit, GPU)。本文介绍的绘制算法充分利用 GPU 提供的可编程特性及强大的计算能力，既能绘制具有真实感的水面，又能满足实时绘制的速度要求。

目前对水面的建模可以分为3类：(1)基于几何模型的方法，即直接由波型函数构造参数曲面来表示海浪表面，如Peachey[1]采用正弦函数和二次函数的线性组合来模拟波浪的外形。由于波型函数本身反映了海水表面的高度变化，因此可以模拟海浪的运动，但不能模拟波浪的破碎等现象。该算法简单直观，实时性相对较好，能满足对真实性要求不高的情况。(2)基于物理模型的方法，常用Navier-Stokes方程[2]，即经典流体力学来建立水波模型，用求得的方程数值解得到海浪的具体形状，该方法是在给定初始条件和边界条件下自动产生的，因此，它所生成的海浪形状非常接近真实的物理现象。缺点是方程的求解很困难，目前还不能满足实时性的要求。(3)基于谱的分析方法，利用海洋统计和经验模型，通过大量正弦波的叠加来模拟海面，采用FFT(快速傅里叶变换)合成一个类似海浪谱分布的高度场，文献[3-4]分别描述了采用统计模型和FFT方法模拟海浪的方法。由于在合成过程中常采用规则矩形粗网格来完成实时绘制和避免FFT方法产生的视觉上明显的重复性，因此降低了图像的质量和真实感。另外还有混合了以上2种或多种的方法，如文献[5]提出的基于cellular automat的实时海浪模拟方法、尹勇的基于浪级划分和海浪谱的实时模拟方法。

在水面光照效果的绘制方面，文献[3]对深度海水动画及海面光照效果绘制算法进行了介绍，也讨论了硬件加速方面的绘制技术。文献[6]提出了基于图像空间的菲涅耳效果的绘制方法。本文介绍了一种实时绘制水面的方法，该方法在借助可编程图形硬件通过正弦波叠加的同时，利用2个凹凸纹理实现水面的动画效果，通过实时的纹理映射技术实现水面的反射、折射和菲涅尔等水面光照效果。

2 水波建模和动画

水波建模是模拟水面效果的关键因素，考虑到对渲染真实性和实时性的要求，选择使用基于物理的正弦波模型。把水波看成是一系列不同频率和振幅的正弦波的迭加。运用正弦波叠加技术的波纹函数定义为

$$H(x,y,t)=\sum_{i=1}^{n}A_i\times\sin((D_{ix}\times x+D_{iy}\times y)\times 2\pi/L_i+t\times S_i\times 2\pi/L_i) \quad (1)$$

其中，A_i为第i个正弦波的振幅；L_i为第i个正弦波的波长；

基金项目：信息产业部电子信息产业发展基金资助项目(2005688)
作者简介：王道臣(1981－)，男，硕士研究生，主研方向：互动数字娱乐；万旺根，教授、博士生导师；唐经洲，教授；陈华杰，硕士研究生
收稿日期：2007-12-05 **E-mail：**Daochen001@163.com

D_{ix}, D_{iy}为第i个正弦波的前进方向；S_i为第i个正弦波的前进速度；t为时间。

如果时间 t 作为一个连续时间变量，$H(x,y,t)$ 表示 t 时刻水波的高度场。通过网格顶点中的坐标与该函数叠加，产生顶点位移，实现水波的动态效果。虽然通过正弦波叠加能够得到任意波形，但是实际模拟产生的只是水的大概轮廓。为了提高真实性，弥补细节上的缺失，可将一张凹凸纹理映射图(Bump mapping)映射到水面轮廓上，来展示水面细小的波纹，从而提高真实感。该方法暂时解决了水面的细节问题，但是当近距离观察水面时，可以看到水面的细小波纹是静止的。为了进一步增加真实性，可以将 2 张 Bump mapping 交替映射到水面轮廓上，随时间的不同动态替换 Bump mapping，图 1 和图 2 为 2 张生成好的 Bump mapping。图 3 是用 2 张 Bump mapping 交替产生的水面效果，通过一张凹凸纹理来扰动水面的法向量，从而实现水面的细微效果。

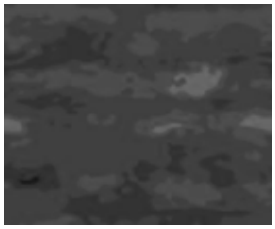

图 1　Bump mapping 1

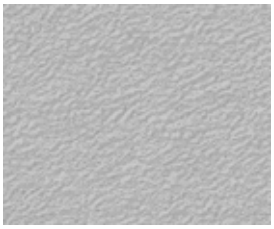

图 2　Bump mapping 2

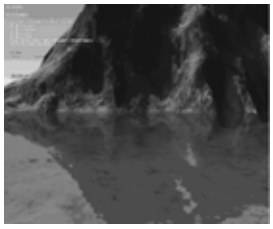

图 3　带有波纹的水面

由于 Bump mapping 中保存的是相对于(0,0,1)的扰动量，因此需要将它与光亮公式中经常使用的插值法向量对应。这可以通过在每个顶点位置构造一个坐标系来实现，在该坐标系中，顶点法向量总是指向 z 轴正方向。另外，在每个顶点位置还需要有 2 个与表面相切的向量，这 2 个向量形成正交基。这样形成的坐标系统称为切线空间。在具体计算时，需要将光线向量转到这个切线空间内。然后利用 3×3 的转换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中。

$$\begin{aligned}\boldsymbol{B}&=\frac{\partial(x,y,H(x,y,t))}{\partial x}\\ \boldsymbol{T}&=\frac{\partial(x,y,H(x,y,t))}{\partial y}\\ \boldsymbol{N}&=\boldsymbol{B}\times\boldsymbol{T}=(-B_z,-T_z,1)\end{aligned}\tag{2}$$

$$\boldsymbol{M}_{33}=\begin{pmatrix}B'_x & T'_x & N'_x\\ B'_y & T'_y & N'_y\\ B'_y & T'_z & N'_z\end{pmatrix}\tag{3}$$

其中，$\boldsymbol{B}$, $\boldsymbol{T}$ 分别表示顶点处的切向量；$\boldsymbol{N}$ 表示顶点的法向量；$\boldsymbol{M}_{33}$ 表示转换矩阵；$B'_x, B'_y, B'_y, T'_x, T'_y, T'_z, N'_x, N'_y, N'_z$ 表示切线空间的 3 个坐标轴。下面部分程序采用 HLSL 语言编写，并在 GPU 上运行。

```
TPsInput_1_4 VS_1_1_PS_1_4_Fluid(TVsInput input)
{    TPsInput_1_4 output;
 half4 pos = input.Position;
    //按照投影空间转换坐标
    output.Position = mul(pos, gWorldViewProj);
    //影射纹理坐标
    output.ReflectionTexCoord = mul(pos, gReflection);
    output.RefractionTexCoord = mul(pos, gRefraction);
    //增加一个凹凸法向坐标的偏移量
    output.BumpTexCoord0 = frac(gBumpDir0.xy×gTime) + (input.
TexCoord×gBumpScale0×gScale);
    output.BumpTexCoord1 = frac(gBumpDir1.xy×gTime) + (input.
TexCoord×gBumpScale1×gScale);
    output.FluidToCam = normalize(gCameraPosition - pos);
    output.SurfaceDetailTexCoord = frac(gDetailDir.xy×gTime) +
input.TexCoord;
    return output; }
```

3　水面的折射和反射

水面的折射和反射是以水面为裁减面，把场景(除水面本身)实时渲染成折射和反射纹理，投影到水面上。折射和反射虽然原理不同，但在代码实现过程中类似，所以本文重点说明反射的实现过程。渲染反射纹理时以水面为参考面，把摄像机翻转到水面以下，实时地把场景渲染成一张纹理，投影纹理时以顶点位置作为参数，来动态调整纹理坐标，反映出水波的流动效果。本文用一个渲染通道 FluidPass 将水面以上的场景渲染成一张纹理贴图，通过以下的投影纹理坐标变换实现该纹理与水面细节贴图的混合。

$$\begin{aligned}\boldsymbol{M}_{\text{final}}&=\boldsymbol{M}_{\text{world}}\cdot\boldsymbol{M}_{\text{view}}\cdot\boldsymbol{M}_{\text{proj}}\cdot\boldsymbol{M}_{\text{remp}}\\ Tex_{\text{proj}}&=(P+\lambda\boldsymbol{N})\cdot\boldsymbol{M}_{\text{final}}\end{aligned}\tag{4}$$

其中，$\boldsymbol{M}_{\text{world}}$，$\boldsymbol{M}_{\text{view}}$，$\boldsymbol{M}_{\text{proj}}$ 分别是水面的世界、相机和透视变换矩阵；P 是顶点坐标；$\boldsymbol{N}$ 是法向量；λ 为常量系数；Tex_{proj} 用来实现投影纹理坐标对反射纹理图进行查找；而 $\boldsymbol{M}_{\text{remp}}$ 是把裁减空间坐标变换到纹理空间的变换矩阵，其取值如下：

$$\boldsymbol{M}_{\text{remp}}=\begin{vmatrix}\frac{1}{2} & 0 & 0 & 0\\ 0 & -\frac{1}{2} & 0 & 0\\ 0 & 0 & 1 & 0\\ \frac{1}{2} & \frac{1}{2} & 0 & 1\end{vmatrix}\tag{5}$$

4　菲捏尔效果

对水面来说，观察者和水面的角度越小，反射效果越明显；角度越大，折射效果越明显，这称为菲捏尔效果。所以，需要根据观察者的角度来计算反射、折射贴图以及水面颜色的混合方式。物理上正确的菲捏尔计算比较复杂，通常使用以下近似的计算方法：

$$TFresnel = r + (1-r)\times pow(1.0-dot(viewDir, \boldsymbol{normal}),5.0)\tag{6}$$

这里，空气到水面的反射系数为 0.020 37。最终水面颜色为

$$finalColor = watercolor + reflectionColor \times TFresnel + reftraction Color\times (1-TFresnel)$$

其中，$TFresnel$ 表示菲捏尔系数；$viewDir$ 为 camera 观察的方向；$\boldsymbol{normal}$ 为法向量；$finalColor$ 为最终水的颜色；$waterColor$ 为初始水的顶点颜色；$reflectionColor$ 是反射颜色；$refracction Color$ 为水面折射颜色。

5　性能测试

根据上述方法，选择 1 024×768 的全屏模式模拟水面生成的效果图，如图 4 和图 5 所示。测试平台为：编译环境 VC.NET，2.4 GHz 奔腾 4 CPU，ATI Radeon 9550 显卡，128 MB 显存，512 MB DDR333 内存，7 200 rpm 80 GB WD 硬盘。

(下转第 237 页)