

水波现象的光学模型及其三维仿真研究与应用

陈 路, 桑 楠, 熊光泽

(电子科技大学 计算机科学与工程学院, 四川 成都 610054)

(mr_chen_lu@263.net)

摘 要:水波现象因涉及到复杂的物理过程一直是计算机图形学研究的热门领域,也是具有相当难度的。通常运用光线追踪、求解复杂的微分方程来模拟水波复杂的物理、光学现象,但这种精确的模拟方法渲染一帧往往要好几个小时。本文充分利用当今 3D 硬件的可编程能力,用 Perlin 噪声函数建立水波物理模型、立方体环境映射等技术建立水波光学模型,尽管这种方法建立的水波模型并非很准确,但视觉效果较为逼真,很适于仿真、游戏等实时性要求很高的应用领域

关键词:水波;实时性;物理建模;光学建模;着色器

中图分类号: TP391.41 **文献标识码:** A

Dynamic modeling and 3-D simulation of water surface

CHEN Lu, SANG Nan, XIONG Guang-ze

(College of Computer Science and Engineering, University of Electronic Science & Technology of China,
Chengdu Sichuan 610054, China)

Abstract: CG water has been a major research topic for a long time now. Realistically simulating liquids has proven very difficult, due to the inherent complexity of the physical processes involved. Most photorealistic solutions use a form of raytracing and complex differential equations to approximate the optical behaviour and motion dynamics of a water body. Unfortunately, such physically accurate approaches often take many hours of render time per frame. With the advance of programmable 3D hardware, the article discussed methods such as wave physical generation by Perlin function and wave optical model by cubic environment mapping. A lot of the visual properties of water could be approximated by efficient techniques that, even though not physically correct, look convincing to the human eye, and it was pretty appropriate to interactive contexts such as video games, simulation and so on.

Key words: water; real-time; physical model; optical model; shader model

0 引言

水波现象一直是计算机图形学(Computer Graphics, CG)领域的研究热门,尤其是要求高实时性更是被公认为有相当难度的研究领域。因涉及到的物理仿真过程极其复杂,CG 中通常的作法是采用光线追踪和复杂的微分方程来模拟水波光学现象(如折射、反射等)和动力学行为(如水波流动、绕流等现象)。然而,这种基于水波动力学和光学等学科建立起来的真实物理模型的方法虽然逼真度、准确性都非常高,但往往要好几个小时才能渲染一帧,实时性不高。这在实时性要求很高如:三维游戏等应用领域中是不可取的,必须要很好地权衡画面的质量与性能两方面的矛盾,本文探讨一种既不损失水波真实性又能很好满足游戏玩家视觉效果的水波渲染方法。本文探讨有限域水波的模拟,包括水波动力学特性和光学特性两个方面,当然更高级的特性需要硬件支持,光学特性是通过图形编程单元(Graphic Programming Unit, GPU)来实现。

与模拟大多数基于物理效果的自然现象类似,水波的仿真也是分为两个阶段,第一阶段是基于物理学的水波生成阶段,即水波建模阶段。该阶段要仿真出水波受内外力作用下

的表现,也就是水波如何运动、如何改变形状、如何与周围环境交互(如封闭容器中水波是如何衍射的等现象)。第二阶段是可视化阶段,这一阶段首先介绍光照模型,然后着重介绍如何采用立方体环境映射来生成周围环境的反射。

1 水波物理建模

1.1 动力学模型

1) 水波一维能量分布

根据频率分布模型(Pierson-Moskowitz Model),该能量分布模型既精确计算量又小,是目前最好的计算水波一维能量分布模型^[1],

$$E^{pm}(f) = \left(\frac{\alpha \cdot g^2}{(2\pi)^4 f^5} \right)^{\frac{5}{4} \left(\frac{f}{f_m} \right)^{-4}}$$

其中 $\alpha = 0.008$, $f_m = \frac{0.13 * g}{v}$, $g = 9.80665 \text{ m/s}^2$ (重力加速度), v 为风速, f_m 为最大频率。

2) 色散方程(Dispersion Equation)

$$\text{由 } \omega(k) = \sqrt{gk} = \sqrt{g \frac{2\pi}{\lambda}} = 2\pi f, \text{ 得: } \lambda = \frac{g}{2\pi f^2}$$

3) 相位角分布

收稿日期:2005-04-15;修订日期:2005-06-07

作者简介:陈路(1980-),男,四川宜宾人,硕士研究生,主要研究方向:计算机图形应用技术; 桑楠(1964-),男,四川营山人,副教授,博士研究生,主要研究方向:嵌入式/实时高可信技术、仿真开发平台技术; 熊光泽(1938-),男,四川丹棱人,博士生导师,主要研究方向:嵌入式实时计算系统及系统仿真。

$$D(f, \theta) = \frac{\tau(s+1)}{2\tau(s+0.5)\sqrt{\pi}} \left(\cos^2 \frac{\theta}{2} \right)^s$$

$$\text{其中 } \tau(s+1) = \int_0^\infty x^s e^{-x} dx$$

$$s = \begin{cases} 9.77 * \left(\frac{f}{f_m}\right)^{-2.5} & f \geq f_m \\ 6.97 * \left(\frac{f}{f_m}\right)^5 & f < f_m \end{cases}$$

4) 二维能量分布

$$\begin{aligned} E(f, \theta) &= E^{\text{pm}}(f) * D(f, \theta) \\ &= \left(\frac{\alpha \cdot g^2}{(2\pi)^4 f^5} \right)^{\frac{5}{4} \left(\frac{f}{f_m}\right)^{-4}} * \frac{\tau(s+1)}{2\tau(s+0.5)\sqrt{\pi}} \left(\cos^2 \frac{\theta}{2} \right)^s \end{aligned}$$

5) 振幅分布 A

$$A = \sqrt{\frac{E(f) * D(f)}{k * f} * \frac{\pi^2}{\omega}}$$

6) 二维振动方程

$$\xi = A \sin(k * x + \omega * t), k = \frac{2 * \pi}{\lambda} (\lambda \text{ 为波长}), t \text{ 为时间},$$

ω 为相位角, x 为位置。

由二维振动方程很容易推广到三维, 有了以上物理数学模型后, 再根据波的叠加原理就可以计算得到任意 (x, y) 点处波的高度值^[1]。

虽然根据动力学建立起来的水波模型能更真实地模拟水波静、动力学效果, 还可以提供充分的交互性, 但求解微分方程很费时, 这种方法不适用于实时性要求很高的游戏中。

1.2 三维 Perlin 噪声模型

虽然水波建模数学上也有很多可行的方法, 但最简单的方法就是采用 Perlin 噪声, 可以在无用户交互的情况下能大致模拟出水波的运动, 该方法简单、实时性很高, 再结合目前许多先进的图形渲染技术, 可以模拟实时性很高、又不损失视觉效果各种水波反射、折射等现象, 本文就是采用这种方法进行水波的物理建模。

Perlin 于 1985 年提出噪声函数的应用, 该方法采用三维整数网格来定义噪声函数。在每一个整数网格点 (i, j, k) (其中 i, j, k 均为整数) 上定义一随机灰度值 (0 ~ 255) 作为该函数的值。非整数网格点的函数值则通过该点相邻的 8 个整数网格点处的函数值得到。具体实现时, 可采用一个查找表来存储整数网格点处的噪声函数值, 不会出现函数值的突变现象, 但该函数 $T(u, v, w)$ 在相邻网格边界面上的剃度是不连续的, 从而导致了在某些方向上存在人工的痕迹, 这些可以通过三次插值来改善^[1]。

2 水波光学建模阶段

第二阶段是可视化阶段, 即水波渲染阶段, 该阶段采用水波建模阶段生成的水波数据然后将水波渲染到屏幕上, 同时能呈现出水表面与周围环境以及光线之间复杂的光学特性, 而如何对这些光学特性的建模对最终视觉效果是至关重要的。与前面水波物理建模阶段本质区别就是, 水波物理建模阶段数据的生成都是由 CPU 计算完成, 而光学建模阶段则是在 GPU 上通过逐像素计算完成的, 这就需要硬件支持可编程软件接口 PS2.0 及以上。而如今很多中端硬件都支持 PS2.0 及以上 (Pixel Shader2.0, 像素着色器), 如: Nvidia 公司的

Geforce 系列, ATI 公司的 R9800XT/9600XT, Nvidia 公司的 6200/6600/6800, 而 Nvidia 公司的 6200/6600/6800 支持 PS3.0 也是目前硬件支持 PS 的最高版本。硬件的发展使得开发实时性要求高的图形技术 (如实时水波) 更加方便, 简单。限于篇幅这里仅讨论反射现象, 截图中读者还可以看到采用基本光照模型实现的折射、高光等现象。

2.1 基本光照模型

光照模型或者明暗模型主要用于表示物体表面某点处的光强度的计算。实际中有两种做法: 其一是将光照模型应用于每个可见面的每个点, 另一种方法是, 将光照模型应用在面上少量点然后对亮度进行插值。前者由于计算量很大难以满足高实时性的要求, 如: 辐射度算法。故本文采用插值方法。

1) 环境光

一个物体表面即使不直接暴露于光源下, 只要其周围的物体被照明, 它也可能看得见。在基本光照模型中, 只须改变一个场景的基准光亮度, 就可以简单地模拟一种从不同物体表面所产生的反射光的统一照明, 称为环境光或背景光。环境光没有空间或方向上的特征, 在所有方向上和所有物体表面上投射的环境光数量都恒定不变。用参数 I_a 表示。

环境光计算公式: $I_{\text{ambdiff}} * I_a$

2) 漫反射

场景中每个面上的漫反射是恒定不变的, 与观察方向无关。用参数 k_d 表示入射光线中被漫反射部分的百分比, 该参数表示漫反射系数或漫反射率。如果 I_l 是光强, 则表面上某点处的漫反射方程为:

$$I_{l, \text{diff}} = k_d * I_l * \cos \theta \quad (\theta \text{ 为入射角})$$

可以看出, 若在某特定点, 入射光垂直表面, 则该点完全被照射。当光照的角度远离表面法向量时, 该点的光亮度将降低。仅当入射角在 $0^\circ \sim 90^\circ$ 时 ($\cos \theta$ 在 0 与 1 之间), 点光源才照亮面片; 若 $\cos \theta$ 为负值, 则光源位于面片之“后”。

若 \vec{N} 为物体表面的单位法向量, 且 \vec{L} 为从表面上一点指向点光源的单位矢量, 则:

$\cos \theta = \vec{N} * \vec{L}$, 且对单个点光源的光照中的漫反射方程为:

$$I_{l, \text{diff}} = k_d * I_l * (\vec{N} \cdot \vec{L}) \quad (\cdot \text{ 表示点乘})$$

3) 镜面反射和 phong 模型

当我们观察一个光照下的光滑物体表面如磨光的金属、苹果或人的前额时, 可能在某个观察方向看到高光或强光, 这个现象就是镜面反射, 因为在接近镜面反射角的一个会聚区域内入射光的全部或绝大部分成为反射光。可以用下式来计算物体表面上某点处的镜面反射:

$$I_{\text{spec}} = k_s * I_l * (\vec{V} \cdot \vec{R})^{n_s} \quad (\cdot \text{ 表示点乘})$$

其中 \vec{V} 表示从物体上某点指向视点的单位矢量, \vec{R} 单位化的反射方向, n_s 越大, 则表面的粗糙值较小 (小于 1), 对于理想反射器 n_s 是无限的, 而粗糙表面 n_s 的值接近 1^[2]。

这就是著名的 Phong 光照模型, 实际上由于计算反射向量与入射光线的点积比较费时, 往往用简化的 Phong 模型代替。

为了简化, 作如下假设, 假设水波数据是从高度图 (用灰度图来存储高度信息的图) 得到, 水波表面被分割成很多网格, 网格上每个顶点都除了位置信息外, 还有法线信息; 假设水面是完全清澈透明, 然而实际上水中因含悬浮颗粒状的杂质, 这些粒子像水分子本身一样, 会散射和吸收而使光线衰

减,也就是为什么浅水比深水更透明,光在介质中传播的距离越远,被悬浮粒吸收和散射的几率就越大。图一是采用 Phong 模型的镜面光照效果。



图 1 采用 Phong 模型的镜面光效果

2.2 水波反射——Cubemap 实现

为了表现周围环境在水面的反射效果,本文采用 cubemap(cubic environment mapping,立方体环境映射)来实现,效果如图 2 所示,采用了前述介绍的几种基本光照模型。

Cubemap 属于环境映射的一种,需硬件支持,其基本原理如下:cubemap 由 6 个正方形纹理贴图组成,6 个正方形贴图围成的立方体构成了环境贴图,视点固定在物体的中心,从而渲染出 6 个视域方向的环境贴图。设想视点固定在一间房子的中心,假设房子是空的,那么构成的 6 个视图就包括四面墙、天花板和地板。然后就建立从物体上某点的入射光方向到环境贴图之间的映射关系,假设反射光线向量和环境贴图立方体处于同一个坐标系下(假设为世界坐标系),其映射算法如下:

1) 找到入射光与哪个环境贴图相交,也就是通过比较单位化的反射光向量和位于原点的单位正方体区域,从而确定入射光与哪个面相交。

2) 将入射光映射到纹理空间 (u, v) 坐标系中,如:点 (x, y, z) 与负 z 轴的正平面相交,则映射方法为: $u = x + 0.5, v = -z + 0.5$

也就是在世界坐标系下,将物体表面上某点的入射光位置存入 cubemap 中,相当于建立一个 360 度的查找表,由 3 维方向向量入射光线索引 cubemap,这样就可以用来建立反射立方体,照相机位于 cubemap 中心,周围环境通过立方体六个面被渲染,这一步是沿主轴方向将周围环境色硬编码,也就是通过预计算完成;在第二个 pass 中,反射物体就会被渲染,反射向量是基于顶点或者基于像素计算的,反射向量用作 cubemap 的索引,以下是基于 (vertex shader) 和 PS (pixel shader) 实现的 cubemap 算法(假设顶点是在世界坐标空间):

```
void VP_cube_reflect( float4 inPos : POSITION,
    float3 inNormal : NORMAL,
    out float4 outPos : POSITION,
    out float3 outTexCube : TEXCOORD0,
    uniform float4x4 Mvp,
    uniform float3 cameraPos )
{
    //将顶点位置变换到屏幕坐标下
    outPos = mul( Mvp, inPos );
    //计算世界坐标系下从相机到顶点位置向量(入射光)
    float3 V = inPos - cameraPos;
    // 计算反射向量
    outTexCube = reflect( V, inNormal );
}

void FP_cube_reflect( float3 inTexCube : TEXCOORD0,
```

```
out float4 outCol : COLOR,
    uniform samplerCUBE EnvCubeMap )
{
    // 由反射向量取样 cubemap, 也就是查找映射表
    outCol = texCUBE( EnvCubeMap, inTexCube );
}
```



图 2 cubemap 实现的水波反射效果

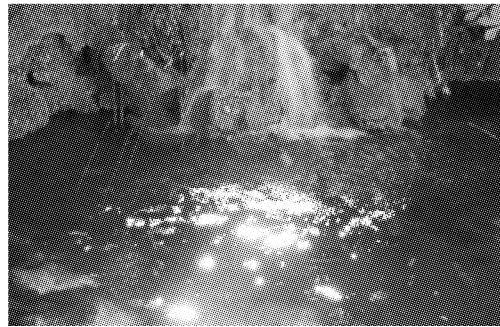


图 3 光照模型中的高光效果

3 水波数据表示

由于存在各种水波模型,也存在多种水波渲染器(也就是用来表现水波光学特性),我们可以选择不同的水波模型,也可以选择不同的渲染器,水波模型和水波渲染器的多样性就要求能通过公共的数据接口,使得渲染器和水波模型相互独立,该系统采用抽象接口来表示公共的数据,这样各模块之间具有松散的耦合性:不同水波模型和渲染器之间只需通过该接口就能完成而无需知道对方。这对将来增加新的水波物理模型或增加新的水波渲染器或同时增加二者提高了灵活性。

4 结语

通过基于物理学的研究,对水波生成用 Perlin 噪声函数建立出数学物理模型。同时,充分利用当今 3D 硬件的可编程能力,用立方体环境映射等技术建立水波光学模型,让水波表现可视化。在现今的硬件环境条件下,即非常真实的模拟了水波的光照,物理效果;又极大的提高其运行效率。

同时,对水波模型和水波渲染器的抽象接口考虑又充分体现了其无关性,对将来游戏或者 VR 应用的技术发展及开发提供了更加灵活的策略。

参考文献:

- [1] HEARN D, BAKER MP. Computer Graphics[M]. 北京:电子工业出版社,2000.
- [2] WATT A, POLICARPO F. 3D games Real-time Rendering and Software Technology[M]. Vol 1. 北京:机械工业出版社,2004.
- [3] 孙家广. 计算机图形学[M]. 北京:清华大学出版社.
- [4] Microsoft DirectX SDK Document[Z], 2005.