****

**本科毕业论文**

****

**题 目: 基于贴图的水渲染算法探讨**

**Texture Based Water Rendering Alogirthm**

**学 院: 电子信息学院**

**专 业: 电子信息科学与技术**

**学生姓名: 李卓坤**

**学 号: 2012141451029 　 年级: 2012**

**指导教师: 宋翠家**

目录

[Abstract(英文摘要） 3](#_Toc481954514)

[摘要 4](#_Toc481954515)

# Abstract

**Keywords**: Computer Graphics,Water Rendering, OpenGL

Rendering water is different from rendering many other objects in 3D scene. Firstly, it doesn’t have a static shape, therefore its mesh is changing every frame. Secondly, its shape changes from one scenario to another. It can be a glistening lake, wavy sea, splashing stream, a pool with ripples, a cup of tea or a rain drop from the sky. It also changes with the position of our eyes. What we see above the surface is different from what we see under the surface. For simulating the scenorios mentioned above, researchers have developed many algorithms for each of them. With all these complex situations and algorithms, the final issue comes to the conflict between effeciency and realilty. Computer graphics techniques are mostly applied in video games which requires both realtime rendering speed and convincing effect. The choice algorithm is a trade off between effeciency and reality.

This thesis introduces two common used texture based alogirhtms for water rendering in two different scenarios. The first one focus on rendering small to medium wave on a flat surface without complex physics simulation. In this mothod, we use textures to render relection and refraction image on the water surface and we also use more than one normal map to simulate the perturbation caused by the wave. This method is efficient for realtime rendering and its result is convincing for small to medium wave. Different from the first scenario which is water surface rendering, the second meothod focus on rendering rain drops. This meothod is not going to render textures on a surface, but render textures as particles in the scene. It uses billboarding technique to render particles and a simple particle system is used in this project to manage particles’ spawn and death.

This thesis is basically divided into three sections and each section has its own demo which implements the theory described:

1. OpenGL rendering basis (City Model Demo)
   1. First Person Camera implementation
   2. Rendering basic objects with texture
   3. Light effect
   4. Skybox
   5. Physical Collision
2. Texture based water surface rendering(Water Quad Demo)
   1. Reflection & refraction rendering
   2. Wave generation and rendering
3. Rendering rain as particles(Particle System Demo)
   1. Particle system implementation
   2. Rendering rain drops with texture

# 摘要

**关键词：**计算机图形学，水渲染，OpenGL

**在3D场景中，渲染水跟渲染其他物体不大一样。首先，水没有固定的形态因此水的网格在每一帧当中都在变化。其次，水的形态也随着情景的不同而不同。水可以表现为一个波光粼粼的湖面，波涛汹涌的大海，水花四溅的急流，丝丝涟漪的池塘，甚至一杯热茶或者从天空降下的一滴雨。随着我们身处的位置，看到的水也不一样， 比如在水面上跟在水底看到的是不一样的场景。为了模拟上面提到的各种情况，许多学者已提出各种算法来应对各种渲染需求。各种纷繁复杂的情况和算法最终都会导致我们面对效率和效果这一对矛盾。计算题图形学技术最多被应用在游戏当中， 而游戏对实时渲染速度以及真实的效果都有着很高的要求。对算法的选择就是对效率和效果的取舍。**

**本论文介绍了两种常用的基于贴图的水渲染算法，这两种算法是用于渲染不同的场景。第一种关注于在一个水平面上渲染小到中型水波。该算法不需要用到复杂物理模拟计算因此有着较高的效率。这方法在水的表面使用贴图来渲染出反射与折射的画面，然后用多余一张的法向量贴图来模拟水波荡漾的效果。这种方法有着很好的实时渲染效率和对渲染中小型波纹有着比较真实地效果。与第一个算法是用来渲染水表面的不一样，第二个算法是渲染雨滴。这种方法不采用在一个平面贴图来渲染水效果，而是把贴图渲染到分布在场景中的粒子上。这方法用到广告版技术（**billboarding**）去渲染粒子，在实现程序中用到一个简单的粒子系统去管理粒子的诞生与湮灭。**

**本论文大致分为三个部分，每一个部分有着各自的展示程序去实现当中提到的理论：**

1. **OpenGL 渲染基础（City Model 程序）**
   1. **第一人称摄像头实现**
   2. **基础物体贴图渲染**
   3. **光照效果**
   4. **天空盒子**
   5. **物理碰撞**
2. **基于贴图的水表渲染（Water Quad 程序）**
   1. **反射与折射的渲染**
   2. **水波的生成与渲染**
3. **渲染雨滴（Particle System 程序）**
   1. **粒子系统的实现**
   2. **用贴图渲染雨滴**

## 1.1研究背景

从上世纪九十年代开始，GPU的功能日益强大。从一开始的利用固定流水线（Fixed Pipeline）架构来减轻了CPU的计算量，到可编程流水线（Programmable Pipeline）, GPU不仅从硬件上提升了其运算速度以及并发运算能力，而且从软件上也给程序员提供了很多可灵活控制GPU的功能。程序员可以深入到着色器（shader）程序当中通过编码操控网格顶点的变换，然后甚至操控每一个像素的着色。强大的硬件使得水渲染渐渐从线下渲染转换到实时渲染而灵活的流水线驱动了很多学者和程序员开发出不同的算法去模拟和渲染水。

Jerry Tessendorf在他的论文中提出一种的基于快速傅里叶转换（FFT）的算法。该算法使用大量的网格节点，通过模拟水的物理运动形态来渲染出及其真实地海浪效果。在该算法中，对水体运动复杂的物理计算是在CPU上进行的，这也是该算法的一个瓶颈，大量的占用了CPU的资源使其难以在游戏中实现。Mark Finch 在《GPU Gems》里提出的一种算法对此进行了优化，能更有效的利用GPU的计算能力，平行CPU和GPU的负荷。Mark的算法简化了水的网格顶点计算，通过不同频率的几个正弦波叠加的方法来模拟水体的运动，同时把这些计算移到GPU上进行使得游戏可以有充足的CPU资源进行AI和物理计算。Jerry Tessendorf 和 Mark Finch 的算法都用到网格来表现水表的形态，但是网格的点的密度是平均分布的。Yuri Kryachko在《GPU Gems2》里阐述了一种利用位移贴图来模拟水的算法。该算法通过对高度图采样来决定网格点的位置，但是网格的密度不是一致的，而是以摄像机为圆心，越靠近摄像机密度越多，否则密度越少的径向网格。这也是一种细节等级机制（LOD），可以减少渲染不必要的顶点。

## 1.2 水渲染技术变迁



图1.1：游戏中水渲染示例截图，左一为CS1.6，其余两张为Call of Duty 4

对比下2000年和2007年的游戏我们可以深刻的感受到这短短七年间GPU渲染技术的进步以及它所带给玩家的真实感和震撼感。在2000年的Counter Strike里我们看到的大多数水渲染都是通过简单的贴图来模拟，玩家看到的只是一张来回移动的画有水纹的贴图。这种简单的方法没有考虑到折射反射等水的光学特征，只是作为一款竞技游戏，相比速度，画质的细节往往不是玩家的关注点。当到了2007年Infinity Ward 发布的Call of Duty4，当中渲染的大海的画面已经足以让玩家震撼，加上真实地光照计算以及大雨滂沱的渲染更让玩家沉醉其中。

## 1.3 研究内容

本论文将着重探讨简单的水表波纹以及下雨场景的模拟和渲染并提供实现方法。水表渲染的算法将采用基于贴图的方法来实现，而不需要通过渲染真正的网格来营造海浪的形状。该算法借用了Mark Finch 提出的不同频率波叠加的理论来让水波看上去更真实。下雨场景的渲染将采用粒子系统的方法来模拟雨滴。论文的最后分析了这两个算法的优劣之处以及其在程序运行时的实际效率。

## 1.4 论文章节

本文章节安排如下：

第一章：绪论，主要阐明了项目的研究背景，引出了其后的具体研究内容。

第二章：OpenGL渲染基础技术。主要介绍本项目大致架构以及所用到的基本渲染技术。

第三章：基于贴图的水表渲染。

第四章：下雨场景渲染。介绍简单粒子系统的实现及如何利用该系统实现下雨的渲染。

第五章：分析对比这两个算法。

第六章：总结。

引用

1. Tessendorf, Jerry. 2001. "Simulating Ocean Water."
2. Mark Finch. “GPU Gems Chapter 1. Effective Water Simulation from Physical Models”
3. Yuri Kryachko “GPU Gems 2 Chapter 18. Using Vertex Texture Displacement for Realistic Water Rendering”