# 计算机图形学大作业报告

## 一、题目描述

**2017 大作业要求：交互控制的动态粒子系统**

场景建模及渲染，场景添加光照，光照可交互控制（20 分）

设计实现粒子系统特效，粒子运动有物理仿真，可实时切换粒子系统中的粒子三维模型（30 分）

粒子的三维模型的光照、纹理映射（20 分）

设计实现粒子系统的交互控制（20 分）

作业报告（10 分）

## 二、项目简介

本项目使用 OpenGL 开发，利用粒子系统建模实现了一个喷泉，搭配有场景建模、光照、交互控制等。

## 三、代码组织

**main.cpp：**

包含主函数、图形绘制、光照设置、交互控制、音频控制的回调函数。

其中的 Model 类保存当前时刻物体的三个坐标分量、速度分量、加速度分量。通过 glutTimerFunc 函数定时调用 move 方法，根据时间间隔，计算下一时刻的坐标和速度，如此循环。

**particle.cpp(.h)**：粒子和粒子系统

Particle 结构体：单个粒子的属性，如位置、速度、加速度、生存时间、模型、纹理等

ParticleSystem 类：整个粒子系统，包含粒子的初始化状态、粒子数量等，以及粒子的添加、删除、更新等方法

**objLoader.cpp(.h)**：从文件读入 obj 模型；从文件加载纹理

LoadModel 函数加载 obj 格式的模型，并返回一个 Obj 类的对象，可调用其 draw 方法（参数为缩放比例）绘制模型。

LoadGLTexture 函数读取 bmp 格式的纹理图片（通过 glaux 库的 auxDIBImageLoadA 函数实现），加载到 OpenGL 中，并返回纹理 ID 供后面的

glBindTexture 使用

**waveIn.cpp(.h)**：处理音频输入

程序定时采集声音数据并计算音量，根据音量改变粒子系统中粒子的初速度大小和方向。

## 四、功能实现

1、**场景**：场景建模采用天空盒，即画一个立方体，然后在六个面贴上相应的图片，在内部观察可以看到整个场景的效果。场景尺寸为 20*20*20，世界坐标系中的 1 对应真实世界中的 1 米，以方便与真实场景和物理常数（如重力加速度等）对应。

2、**粒子系统**：通常的粒子系统都是固定粒子数量，然后根据粒子生命周期进行增删。应用到喷泉这个具体场景，更适合的策略是固定粒子的产生速率，同时根据粒子的生命周期去删除，而不限制总的粒子数量。

在实例化 ParticleSystem 类时，从文件读取粒子模型、粒子纹理，读取参数以限制粒子的最大数目等。之后，由主程序中的 timerFunc 定时调用 update 方法，并传入距离上一次调用经过的时间作为参数。

粒子运动轨迹按照抛物线进行物理仿真，使用上一刻的状态计算下一刻的状态以提高仿真的真实性，同时使用中点法计算位移以减小误差。粒子的生命期从产生开始计算，到落地结束。利用随机数对初始速度和角度作一个小的扰动，增加真实感。

在 update 方法中，根据传入的时间间隔，逐一更新所有粒子的状态，删除死亡的粒子，产生一定数量的新粒子。update 方法由主程序中的 timeFunc 函数定时调用，设定为 30 次/秒。

render 方法绘制所有的粒子，同时在绘制前指定材质、纹理、模型。由主程序中的 display 函数调用。

3、**特效**：

添加了环境光和 0 号光源，并为场景模型和粒子模型指定了法向量和材质。能够明显地观察到粒子的明亮面与阴暗面。

随着喷泉水柱逐渐靠近地面，构成水柱的粒子透明度越来越大(通过修改粒子的 alpha 值并开启混合(GL_BLEND)实现)。

4、**交互**：包含鼠标、键盘、音频三种方式

*场景交互*：通过鼠标拖动，可以改变观察视角；在英文小写状态下按键盘的 j、k 键，可以改变观察点的位置；按 h、l 键，可以拉近或远离视野中的场景。

*粒子系统交互*：按 p 可以暂停/恢复粒子的运动；按+、-键，可以增加或减少喷泉水柱的个数。粒子的初速度由声音控制，声音越大，初速度和仰角则越大，喷泉喷得越高。

*光照交互*：按 r、g、b、w、d 可以切换五种颜色的光源

*模型交互*:按 1 使用从 model.obj 文件加载的模型;按 2 使用 glutSolidSphere 的小球模型；按 3 隐藏所有粒子。

## 五、其他说明

1、作业参考了网上的很多文章，无法一一列举，在此对各位原作者一并表示感谢。

2、需要 glut.h、glut.lib、glut32.lib、glut.dll、glut32.dll、glaux.h、glaux.lib、glew.h 以支持编译和运行，编译环境为 visual studio 2015。

3、如果在 visual studio 中通过 F5 或 Ctrl+F5 执行，则默认的路径是 project 文件夹（代码文件所在的文件夹）；如果单独执行生成的 exe 文件，则默认的路径是 exe 文件所处目录。默认路径下应有 model.obj 文件、water.bmp 文件和 skybox 文件夹（内含 6 张 bmp 图片），否则无法运行。

4、obj 模型导入、音频采集、音量计算的代码来自于作业 1。这里简述一下原理：

*obj 模型文件*：纯文本格式，以 v 开头的行指定若干个顶点点的三维坐标(x,y,z)；以 vt 开头的行指定若干个纹理坐标(u,v)以 vn 开头的行指定若干法向量坐标；以 f 开头的行表示面，后面是以空格分隔的几组数据，每组数据代表 顶点坐标索引/纹理坐标索引/法向量索引。

*音频采集*：调用了 windows 的 wavein 系列接口。

*音量计算*：将采样点绝对值的均值取对数，再线性调整到-96 到 0 的范围，数字越大（绝对值越小），音量越大。

5、由于项目中的粒子系统中粒子数目不确定并且经常增删，因此使用双向链表作为数据结构保存所有的粒子。同时为了避免为粒子反复释放/申请内存，在 ParticleSystem 类初始化的时候按照粒子数目上限创建粒子 Particle 结构体并还存在 ParticleBuffer 变量中。之后 addParticle 和 deleteParticle 只是 ParticleBuffer 和 ParticleList 两个链表之间的结点移动，不涉及内存申请/释放。

6、关于音量与粒子初速度的映射关系，由主程序中的 f 函数定义。f 函数接收-96 到 0 的音量值，返回 0 到 1 的一个值。实测结果发现音量值大都集中在-50 到-20 之间，为了初速度对音量变化的敏感程度，f 函数应该在中间变化幅度大，而在两边变化幅度小。最终选择了[-1,1]区间上的反正切函数作为基础，并通过线性变换和幂运算进行细致的调整。

7、在完成大作业期间遇到过的一些坑：

（1） 项目必须使用 release 模式编译，用 debug 模式会异常的卡（debug 模式画面大概 3 秒刷新一次，release 模式画面能够流畅显示）。猜测原因可能是绘制模型（Obj::draw 方法）时使用了 STL 的 vector，STL 库在 debug 模式和 release 模式下有巨大的性能差别，而绘制模型在每次页面重绘都要进行。

（2） release 模式需要添加链接选项/SAFESEH:NO 才能通过编译

（3） 项目使用了 glaux 库，不仅要把 glaux.lib 复制到 vs 的 lib 文件夹下，还要在链接器设置中手动引用 glaux.lib 和 legacy_stdio_definitions.lib，否则链接时会找不到引用。

（4） 顶点的法向量必须是单位长度，否则光照效果无法出现。

（5） 使用六个正方形进行天空盒建模，在测试中正方形之间的边界线始终无法消除。最终发现是默认显卡有问题，切换到另一块显卡后一切正常。

## 六、心得体会

通过这个项目，对计算机图形学和 OpenGL 有了初步的了解，也更进一步认识到了大型电影、游戏特效制作的复杂。这个大作业的实现效果很粗糙，没有进行更复杂的场景建模，没有实现光照阴影效果，cpu 占用较大；使用到

的 OpenGL 技术也比较落后，没有尝试 shader 等更高级的技术。未来还有很多知识等待去学习。