学号	姓名	论文规范性（10）	问题分析与调研（30）	方案创新性（20）	实验结果分析与讨论（40）	结课论文总成绩（100）
21301042	梁增权	8	21	17	36	82

缺乏相关工作梳理

计算机图形学结课论文

OpenGL 环境下的 3D 场景中粒子系统设计与开发

学　　院：软件学院

专　　业：软件工程

学生姓名：梁增权

学　　号：21301042

指导教师：吴雨婷

北京交通大学

2024 年 6 月

摘要

在虚拟现实场景中，传统的工具主要是直线、平滑的曲线、平面及边界整齐的平滑曲面，这些工具在描述模糊的物体时会遇到一些问题，例如火焰、下雪等等，这类模糊物体的表面是不规则的并且具有复杂性和不确定性，特别是表面形状的动态性和流体形的变化。这些物体的逻辑结构很难表达，传统图形学中简单的变换不能实现这些特殊的效果。随着计算机图形学的发展，粒子系统在计算机图形学领域得到了广泛应用。

2005 年，北京交通大学的马俊和朱衡君[1]结合动态纹理技术和矢量控制的方法开发了一个喷泉粒子系统模型；2007 年，魏开平等[2]结合纹理映射技术和粒子系统技术开发了喷泉模型,在他们的模型中考虑了风力对喷射出去的粒子的影响;2012 年，张磊等人提出基于 OGRE 粒子系统的喷泉模拟[3]，方便了对于粒子属性的修改,提高了开发速度。

本文阐述了粒子系统的原理及其在模拟自然现象中的应用，如火焰、雪花和蝴蝶等。通过使用 GLUT 库，实现了一个交互式 3D 场景，通过键盘和鼠标操作与场景中的物体进行互动。

关键词：计算机图形学；粒子系统；3D 场景

目录

1 引言

粒子系统是一种常见的图形技术，用于模拟复杂的自然现象，如烟雾、火焰、雨滴和雪花。粒子系统通过大量简单的粒子来构建复杂的视觉效果，这些粒子在空间中独立运动，并根据设定的规则进行更新和渲染。本文通过对雪花和火焰两种粒子效果的实现，探讨了粒子系统在不同自然现象中的应用。

2 相关工作介绍

在本实验中，我设计了两种粒子效果，分别是火焰和雪花，并将它们添加到了 3D 场景中。该实验通过 OpenGL 进行渲染，分别创建火焰和雪花粒子系统，用来模拟自然界中的下雪和火堆燃烧的现象。

3 雪花粒子系统

雪花是现实生活中不可缺少的美景，在计算机中也能看到这一美景是人们的愿望。利用计算机图形学中的粒子技术来实现下雪是切实可行的，因此我们在这一部分讨论如何设计和实现一个雪花粒子系统。

3.1 设计与实现

雪花粒子系统的设计目标是模拟雪花从空中缓缓飘落并积累的过程。实现过程中，主要步骤包括粒子初始化、粒子更新与渲染。

3.1.1 初始化粒子

为了实现下雪效果，我创建了多个雪花粒子，为粒子添加了雪花的纹理，将天空设为发射面，并将它们按照特定的高度逐渐下落，并且设定 y 轴方向上的加速度为重力加速度。但考虑到自然界中的雪花大小、方向以及速度等属性都不一样，我为所有的雪花粒子赋予了随机大小、速度、加速度和角度等属性，并且为他们添加了 0-100 之间的随机的生命周期，如图 3-1 所示。通过这些随机属性，使得生成的每个雪花粒子都独一无二，模拟了自然界中雪花的多样性。

```
particle* init_particle()
{
    float size = rand() % 20 * 0.03f;
    unsigned char color[] = { 1.0f,0.0f,0.0f };
    float speed[] = { float(rand() % 10 - 4) / 800, float(rand() % 10 - 4) / 400, float(rand() % 10 - 4) / 800 };
    float acc[] = { 1.0f*(rand()%3 - 1)/90000,-4.9 / 4000 ,1.0f*(rand() % 3 - 1) / 90000 };
    float angle[] = { rand() % 360, rand() % 360 ,rand() % 360 };
    particle* p = new particle(vec(size,size,size),vec(speed),vec(acc),vec(angle), rand()%100,texture[0]);
    return p;
}
```

图 3-1 初始化雪花粒子

3.1.2 粒子更新与渲染

在每帧渲染中，更新粒子的属性，如位置、速度和加速度，并检测粒子的生命周期。但是当程序运行时间过长时，发现雪花粒子在地面上积累的越来越多，使得渲染速度减慢，最终导致程序崩溃。因此当雪花落到地面后，将雪花交给另外一个类管理，该类将继续维护雪花的当前位置。雪花在地面上的数量超过限制后，将会使先落下的雪花自动消亡。

3.2 效果展示

如图 3-2 所示，实现了不同状态下雪花下落的场景。在图中，可以看到漂浮的白点即为雪花。每一片雪花都有独特的朝向和飘落速度，以模拟现实中的雪花。

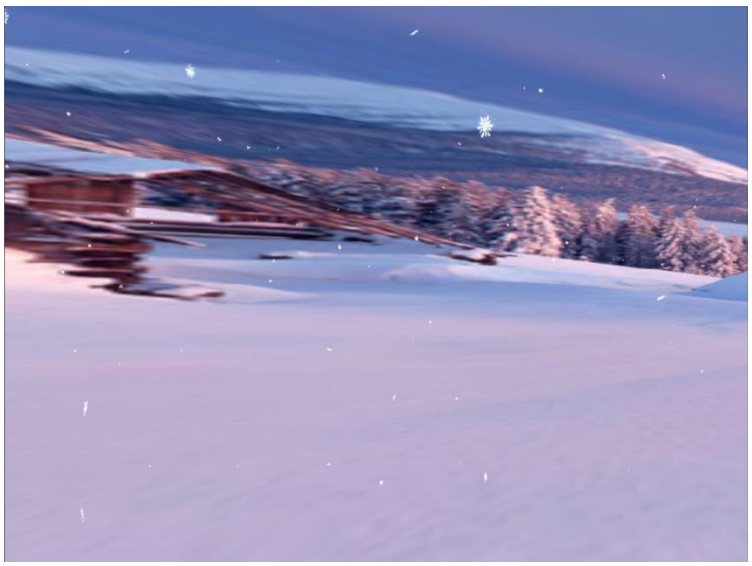

图 3-2 下雪效果展示

4 火焰粒子系统

模拟火焰粒子的系统需要考虑其独特的视觉效果，包括颜色的变化、透明度的衰减和粒子的运动轨迹。不同于雪花粒子系统，火焰粒子系统需要更加复杂的处理来实现逼真的燃烧效果。这一部分将详细探讨火焰粒子系统的设计与实现。

4.1 设计与实现

火焰粒子系统的设计目标是模拟火焰燃烧的过程，重点在于表现火焰的动态和光影效果。火焰粒子的初始化、更新和渲染过程与雪花粒子系统类似，但在粒子属性和更新逻辑上有所不同。

4.1.1 初始化粒子

火焰粒子初始化过程和雪花粒子基本一致，但不同的是火焰粒子是从比较小的面进行发射，初始速度是向上的，并且加速度和生命周期等都比雪花粒子小得多。火焰粒子初始化代码如图 4-1 所示。

```
particle* init_flame()
{
    float size = rand() % 90 * 0.02f;
    float speed[] = { float(rand() % 10 - 4) / 1600, float(rand() % 10 - 4) / 800, float(rand() % 10 - 4) / 1600 };
    float acc[] = { 1.0f*(rand() % 3 - 1) / 9000000,4.9 / 4000000 ,1.0f*(rand() % 3 - 1) / 9000000 };
    float angle[] = { rand() % 360, rand() % 360 ,rand() % 360 };
    particle* p = new particle(vec(size, size, size), vec(speed), vec(acc),vec(angle), rand() % 50 + 10, texture[2]);
    return p;
}
```

图 4-1 初始化火焰粒子

4.1.2 粒子更新与渲染

在渲染火焰粒子之前，需要启用纹理映射，以便将纹理应用到粒子上。火焰粒子的纹理如图 4-2 所示。为了让火焰更加逼真，我们设置了纹理环境模式，以使颜色和纹理能够混合。在 GL_MODULATE 模式下，顶点颜色和纹理颜色相乘，使得粒子颜色与其纹理颜色相结合，呈现出更自然的效果。如图 4-3 和图 4-4 所示，分别展示了紫色和橙色火焰。火焰粒子的位置基于其速度和时间增量进行更新，公式如下：

position += velocity * deltaTime

这一公式使得粒子的位置会随着时间推移而变化，由速度和时间增量决定了粒子移动的距离和方向。

现实中火焰随着与火源的距离增加，颜色和透明度也会变化。因此我让粒子的颜色和透明度随着时间逐渐衰减，以模拟火焰的自然消退，公式如下：

color.a -= fade * deltaTime

color.r = fmax(0.0f, color.r - 0.1f * deltaTime)

第一条公式用于减少粒子的透明度，其中 color.a 表示粒子的透明度，fade 是衰减系数，deltaTime 是时间增量。

第二条公式用于减少粒子的红色分量，其中 color.r 表示粒子的红色分量，fmax 函数确保颜色值不低于 0，防止出现负值。

图 4-2 火焰粒子纹理

4.2 效果展示

通过以上步骤，我实现了一个逼真的火焰粒子系统。火焰粒子在空间中上升、扩散，并随着时间推移逐渐消散，呈现出自然的火焰效果。如图 4-3 与图 4-4 所示，展示了不同颜色下的火焰形态。

图 4-3 紫色火焰

图 4-4 橙色火焰

5 实验结果与分析

在本次实验中，我成功实现了对下雪场景和火焰的模拟，并且在 OpenGL 中渲染绘制了出来。在这部分将会对实验结果进行分析和讨论。

5.1 实验环境

操作系统：Windows 10
开发环境：Visual Studio 2022
图形库：OpenGL

5.2 结果分析

通过对实验结果的观察，有以下结论：

1、对于雪花粒子系统，必须设置掉落在地上的雪花的最大值，否则会出现程序崩溃的情况。

2、通过对所有的雪花设置不同的速度、加速度和角度后，可以模拟不同情况的雪景，呈现较为真实的雪景效果。

3、开始时没有设置纹理环境模式，使得火焰呈现出纹理照片的颜色，如图 5-1 所示，效果较差。使纹理与颜色结合后，如图 4-3 和 4-4 所示，便能得到逼真的火焰粒子效果。

4、通过调整粒子的参数，如速度、大小和颜色等，可以产生不同形态和颜色的火焰效果，增加了系统的灵活性和可定制性。

5、通过设置发射器的的参数，保证了粒子系统的性能良好，能够在实时渲染的情况下保持流畅，没有明显的卡顿现象。

图 5-1 白色火焰

6 结论

本次实验取得了较为理想的结果，但仍有一些方面可以进一步改进和拓展。

对于本实验中的雪花粒子系统：

1、当雪花掉落在地上时不会进行积累，只会零散的排布在地上，超过一定数量后会消失，未来可以计算某一区域内的雪花数量，超过一定数量后使用雪堆进行替代，使场景更加真实。

2、对于场景中雪花数量过多时可能导致程序崩溃问题，可以使用更高效的

粒子系统算法来提高处理速度和减少资源占用。

对于本实验中的火焰粒子系统：

1、粒子纹理较为单一，使得火焰的真实感降低，可以增加纹理的样式来丰富火焰以达到更加真实的效果。

2、增加火焰粒子的运动轨迹，而不是单调的向上发射，以实现更加自然的燃烧效果。

参考文献

[1] 马骏,朱衡君.基于动态纹理和粒子系统的喷泉模拟[J].北京交通大学学报,2005(01):90-94+110.

[2] 魏开平,朱晓华,沈显君,等.基于纹理映射和粒子系统的三维喷泉实时模拟[J].计算机工程与设计,2007(11):2586-2588.DOI:10.16208/j.issn1000-7024.2007.11.026.

[3]张磊,黄亚玲.基于 OGRE 粒子系统的喷泉模拟[J].应用科技,2013,40(03):24-27.