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）
21301003	程轩	6	24	17	37	84

存在行距问题，留白过多

北京交通大学

计 算 机 图 形 学 课 程 论 文

实时粒子系统的动态效果应用研究

Application of dynamic effects in real-time particle systems

学院：软件学院

专业：软件工程

学生姓名：程轩

学号：21301003

指导教师：吴雨婷

北京交通大学

2024 年 6 月

中文摘要

摘要：

粒子系统作为计算机图形学中的一种重要技术，广泛应用于模拟复杂的自然现象，如烟雾、火焰、雨雪等。本文以实时粒子系统的设计与实现为核心，探讨其在动态效果中的具体应用。首先，介绍了粒子系统的基本原理和实现方法，包括粒子的生成、属性设置、运动更新和渲染技术。其次，基于一个具体的项目实例，详细描述了粒子系统在三维场景中的集成过程，并分析了不同参数对粒子效果的影响。最后，通过对粒子系统的性能优化和实际应用案例研究，验证了其在提升视觉真实感和动态表现力方面的有效性。本文通过对粒子系统的仔细学习并运用到计算机图形学实验，对一项以前鲜有了解的技术有了新的认识和掌握。

关键词：粒子系统；实时渲染；粒子特效

ABSTRACT

ABSTRACT:

As an important technology in computer graphics, particle system is widely used to simulate complex natural phenomena, such as smoke, flame, rain and snow. This paper takes the design and implementation of real-time particle system as the core, and discusses its specific application in dynamic effect. Firstly, the basic principle and implementation method of particle system are introduced, including particle generation, property setting, motion updating and rendering technology. Secondly, based on a concrete project example, the integration process of particle system in 3D scene is described in detail, and the influence of different parameters on particle effect is analyzed. Finally, through the performance optimization of the particle system and practical application case study, the effectiveness of the particle system in improving visual realism and dynamic performance is verified. Through the careful study of particle systems and the application of computer graphics experiments, this paper has a new understanding and mastery of a previously little-understood technology.

KEYWORDS： Particle system; Real-time rendering; Particle effect

目录

1.引言

本论文所研究的项目在计算机图形学课程第二次实验的基础上增加对粒子系统的实现。本章首先阐述了实时粒子系统的动态效果应用研究的项目背景及意义，接着分析了国内外相关应用的研究现状，最后介绍了本文的主要内容以及组织结构。

1.1 研究背景及意义

自 1983 年 William Reeves 提出粒子系统以来，它在电影、游戏和虚拟现实等领域得到了广泛应用。粒子系统能够高效地模拟诸如火焰、烟雾、爆炸、水流等自然现象，这些现象通常难以通过传统几何建模来实现。传统的粒子系统由于计算复杂性和资源消耗大，往往难以满足实时性的要求。随着计算机硬件性能的提升和图形处理算法的优化，实时粒子系统成为可能，被广泛应用于需要高度互动和即时反馈的场景中，比如实时渲染和虚拟环境。

本研究聚焦于实时粒子系统的动态效果应用，旨在提高粒子系统的视觉表现力和实时渲染性能。通过优化粒子生成、更新和渲染策略，提升系统的互动性和用户体验。研究成果不仅有助于丰富三维场景的动态效果，还为科学可视化、虚拟现实训练和医学模拟等领域提供技术支持。

1.2 国内外研究现状

粒子系统经过多年发展相当成熟，已经运用在动画制作、游戏、场景模拟等多个不同领域。国产动画电影《深海》电影创作团队别出心裁地将粒子系统与中国传统水墨相结合，全片中数十亿粒子构成的粒子系统所营造的氛围，造就了《深海》电影中富有细节且极具梦幻感的深海世界[1]；李松维等人实现了一个粒子系统和浓度场相结合的烟雾模型，研究了浓度场模型的建立以及粒子在浓度场作用下如何分裂在风场作用下如何运动，在虚拟战场环境中应用本方法模拟了浓烟[2]；余少波等人基于粒子系统对尾焰红外图像实时仿真技术进行了研究，利用实时生成的尾焰红外图像驱动电阻阵列产生红外辐射，进行了半实物仿真试验[3]。

2.粒子原理

（1）粒子属性

粒子系统由多个粒子组成，每个粒子的基本属性包括初始位置、初始速度、加速度、颜色、透明度、形状和生命周期，初始位置决定粒子的生成位置。初始速度和加速度描述粒子生成后的运动轨迹，可以用随机函数来描述其运动路径。颜色和透明度是粒子的视觉特征，这些属性随时间变化。形状则决定粒子的几何形态，如球形、矩形等。生命周期表示粒子的存在时间，从生成到消失。

粒子的初始位置由生成函数决定，生成函数可以是点、线、面或体积内的任意一点。例如，在模拟喷泉时，粒子可能从一个小区域内的随机位置生成。初始速度和加速度决定了粒子的运动轨迹，可以模拟重力、风力等自然力的影响。颜色和透明度通常随着时间变化，例如火焰粒子在燃烧时从橙色逐渐变为灰色。粒子的形状可以是点、线、三角形或复杂的纹理图案等简单的几何形状。生命周期则用来控制粒子的生存时间，模拟粒子的生成、存在和消亡过程。

（2）基本模型

粒子系统的基本模型包括粒子的生成、运动轨迹、粒子之间的影响、粒子的消失和渲染。粒子的生成通过随机函数实现，决定单位时间内生成的粒子数量。生成粒子的数量公式如下：

newParticle = meanParticle + Rand() * varParticle

其中，newParticle 是新粒子数，meanParticle 是粒子平均数目，varParticle 是其方差

新粒子的生成数量可以取决于系统的屏幕大小，以减少渲染时间。

newParticle = (meanParticle + Rand() * varParticle) * ScreeArea

其中，ScreenArea 是屏幕区域大小。

粒子的运动需要确定初始位置、初始速度、加速度、颜色、透明度、形状和生命周期等属性。粒子的运动由初始速度和加速度决定，可以用随机函数描述其运动路径。粒子之间可以相互影响，如通过碰撞和引力作用。粒子根据其生命周期消失，或者当粒子离开视口时被主动销毁。渲染时根据粒子的位置和属性，在每一帧中进行渲染。

（3）生命周期

粒子的生命周期由帧数衡量，每一帧生成时，粒子的生命周期递减。当粒子的寿命为零时，它将消失。此外，粒子在离开视口时也会被销毁，以减少渲染负担。一旦计算出一帧所有粒子的位置和属性参数，渲染算法就会生成图像。粒子可以是透明的，也可以在其他粒子上投影阴影。为了简化渲染算法，可以假设粒子不与其他物体相交，仅处理粒子的渲染。

（4）层次结构

粒子系统可以有层次结构，父粒子系统生成的粒子可以构成子粒子系统。父粒子系统的变换影响所有子粒子系统及其粒子，形成一个树形结构，用于控制复杂的对象模型。当父粒子系统发生变换时，它的所有子粒子系统及其粒子也会随之变换，用这种方式可以对由许多粒子系统组成的复杂对象模型进行全局控制。例如，在模拟一朵云时，云的整体移动可以由父粒子系统控制，而云中的每个小涡流可以由子粒子系统生成和控制。这种方式允许更精细的控制和更复杂的效果，通过对父粒子系统的操作，可以实现对整个粒子系统的全局控制，同时对子粒子系统的操作，可以实现局部的细节调整，实现更加逼真和复杂的模拟效果。

3. 实验环境

操作系统：Windows 11

编程语言：C++

开发工具：Microsoft Visual Studio 2022
GNU Compiler Collection(GCC)

图形库：OpenGL，GLUT(OpenGL Utility Tookit)

硬件配置：

（1）处理器：Intel Core i5 或同等处理器
（2）内存：16 GB RAM
（3）显卡：NVIDIA GeForce RTX 3060 Laptop GPU，支持 OpenGL 4.5

其他工具：CMake（项目配置和管理）

4. 方法描述

4.1 方法概述

为了研究实时粒子系统的动态效果，在第二次实验的基础上实现了动态生成粒子的功能该系统主要由三个部分组成：

（1） 粒子生成模块：负责粒子的初始化和生成。

（2） 粒子更新模块：根据物理规则和随机函数更新粒子的状态。

（3） 粒子渲染模块：负责粒子的绘制和显示。

4.2 粒子生成模块

粒子生成模块负责初始化粒子的属性，包括位置、速度、颜色和生命周期。每个粒子的属性通过随机函数生成。粒子的初始位置既可以是固定的，也可以是一个区域内的随机点。初始速度决定了粒子生成后的运动方向和速度，而颜色和生命周期则决定了粒子的视觉效果和存在时间。

```
struct Particle {
    float life;      // 粒子的生命值
    float fade;      // 衰减速度
    float x, y, z; // 位置
    float xi, yi, zi; // 速度
    float r, g, b; // 颜色
};

const int numParticles = 1000;
Particle particles[numParticles];

void initParticles() {
    for (int i = 0; i < numParticles; i++) {
        particles[i].life = 1.0f;
        particles[i].fade = float(rand() % 100) / 1000.0f + 0.003f;
        particles[i].x = 0.0f;
        particles[i].y = 0.0f;
        particles[i].z = 0.0f;
        particles[i].xi = float((rand() % 50) - 26.0f) * 10.0f;
        particles[i].yi = float((rand() % 50) - 25.0f) * 10.0f;
        particles[i].zi = float((rand() % 50) - 25.0f) * 10.0f;
        particles[i].r = 1.0f;
        particles[i].g = 0.5f;
        particles[i].b = 0.2f;
    }
```

}

在这个函数中，每个粒子的初始属性通过随机函数生成。粒子的生命周期（life）被初始化为 1.0，表示粒子刚生成时完全可见。衰减速度（fade）决定了粒子的生命值减少的速度。粒子的初始位置（x, y, z）被设为(0.0, 0.0, 0.0)，表示从原点生成，速度（x_i, y_i, z_i）决定了粒子的运动方向和速度，颜色（r, g, b）则决定了粒子的颜色。

4. 3 粒子更新模块

粒子更新根据物理规则和随机函数更新每个粒子的状态，通过对每一帧的计算，模拟粒子的运动。

```
void updateParticles() {
    for (int i = 0; i < numParticles; i++) {
        if (particles[i].life > 0.0f) {
            particles[i].x += particles[i].xi / 1000.0f;
            particles[i].y += particles[i].yi / 1000.0f;
            particles[i].z += particles[i].zi / 1000.0f;
            particles[i].life -= particles[i].fade;
        } else {
            particles[i].life = 1.0f;
            particles[i].fade = float(rand() % 100) / 1000.0f + 0.003f;
            particles[i].x = 0.0f;
            particles[i].y = 0.0f;
            particles[i].z = 0.0f;
            particles[i].xi = float((rand() % 50) - 26.0f) * 10.0f;
            particles[i].yi = float((rand() % 50) - 25.0f) * 10.0f;
            particles[i].zi = float((rand() % 50) - 25.0f) * 10.0f;
            particles[i].r = 1.0f;
            particles[i].g = 0.5f;
            particles[i].b = 0.2f;
        }
    }
}
```

通过这个更新粒子函数，如果粒子仍然存活 life > 0，则更新其位置和生命值。位置更新根据速度（x_i, y_i, z_i）进行的，生命值的更新通过减少衰减速度 fade 来实现的。如果粒子的生命值≤0，重新初始化该粒子的属性，模拟粒子的循环生成和消失过程。

4. 4 粒子渲染模块

粒子渲染负责在每一帧中绘制所有活跃的粒子，使用混合模式和点光滑功能，让粒子看起来更柔和具有透明度。在主循环中调用粒子更新和渲染函数，实现粒

子的动态效果。

```
void drawParticles() {
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE);
    glEnable(GL_POINT_SMOOTH);
    glPointSize(5.0f);

    glBegin(GL_POINTS);
    for (int i = 0; i < numParticles; i++) {
        if (particles[i].life > 0.0f) {
            glColor4f(particles[i].r, particles[i].g, particles[i].b, particles[i].life);
            glVertex3f(particles[i].x, particles[i].y, particles[i].z);
        }
    }
    glEnd();

    glDisable(GL_POINT_SMOOTH);
    glDisable(GL_BLEND);
}
```

通过渲染函数的实现，启用混合（GL_BLEND）和点光滑（GL_POINT_SMOOTH），粒子渲染的效果真实。根据当前的生命周期进行设置每个粒子的颜色，使用透明度（alpha）值来表示粒子的生命值。

5. 实验结果与分析

5.1 动态效果

在前实验的基础上添加的实时的粒子系统进行研究，粒子系统能够实时更新和渲染，表现出动态的粒子效果。通过对每帧粒子的更新计算，粒子系统模拟了粒子的运动、变化和消失过程。

（1） 运动轨迹：

通过随机生成初始速度和加速度，粒子在三维空间中表现出多样的运动轨迹比如如上升、扩散和旋转。粒子的运动表现出一定的物理规律，同时通过随机函数引入了不确定性，使得运动轨迹更加自然。

（2） 生命周期：

每个粒子的生命周期由帧数决定，粒子生成后生命值逐渐减少直至消失。通过设置不同的衰减速度，粒子拥有不同的存在时间，呈现出不同粒子多样的运动。生命周期的管理让粒子系统可以动态调整粒子的生成和消亡，保持系统的稳定性。

（3） 视觉效果：

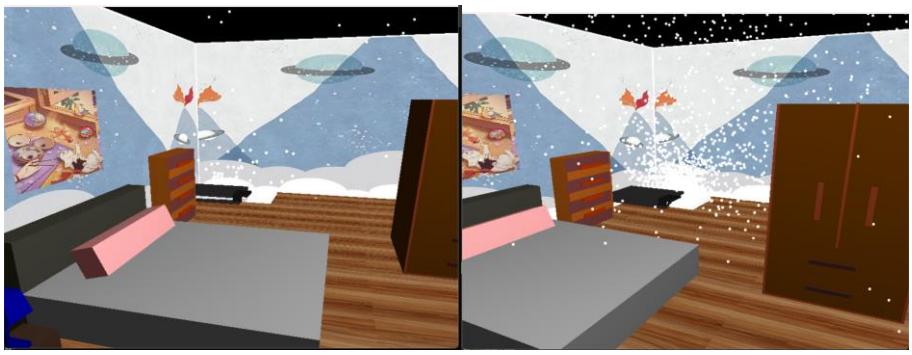
本实验粒子没有设置复杂的颜色和形状变化，但通过设置粒子的透明度和衰减速度，实现视觉上的渐变效果。粒子的简单视觉效果可以用于模拟星空、雪花等场景。同时，通过调整混合模式和点光滑功能，让粒子更加真实。

图 5.1 粒子动态效果

从图片可以看到，屏幕中央的粒子数量最多，向四周运动呈现出不同的运动轨迹。随着向四周扩散运动，粒子透明度和数量逐渐减小，直至到屏幕边缘完全消失

5.2 性能分析

通过调整不同参数观看实验结果，我们可以发现粒子系统的性能体现在每秒帧数（FPS）和处理的粒子数量上。初始条件下生成了 500 个粒子，在每帧中更新和渲染这些粒子。随后逐渐增加粒子数量，得到了不同粒子数量下的平均 FPS。



粒子数量	平均 FPS
500	65
1000	60
2000	55
5000	45
10000	30

表 5.1 不同粒子数量下的平均 FPS

从上表可以看出，随着粒子数量的增加，每秒帧数逐渐降低。当粒子数量达到 10000 时，系统的 FPS 降至 30，仍能保持实时渲染的效果。该粒子系统具有较好的扩展性，能够处理一定数量的粒子而不明显降低性能。。

5.3 优化改进

本次实验中通过以下提升了粒子系统的性能和视觉效果：

（1）**减少粒子数量**：限制粒子生成的区域和数量，减少渲染负担，提高系统的 FPS。合理控制粒子数量可以提高系统性能，保证视觉效果的连贯性。
（2）**使用混合模式**：在渲染粒子时，使用混合模式和点光滑功能，让粒子看起来更柔和具有透明度。混合模式的使用模拟出更加逼真的光影效果，使粒子的视觉表现更加自然。
（3）**优化更新算法**： 简化粒子的更新计算，减少每帧的计算量，提高系统的性能。优化更新算法不仅可以提高系统的效率，还能增强粒子系统的响应速度，使得实时效果更加流畅。

6. 结论

本次研究成功实现了一个实时粒子系统，对实时粒子系统的动态效果进行研究应用。该系统在生成、更新和渲染粒子方面表现出良好的性能和稳定性，能够有效处理不同数量的粒子，并保持较高的 FPS。通过本次实验，我们可以发现粒子系统通过随机生成粒子的属性，展示了多样的运动轨迹和视觉效果，能够模拟不同场景下的动态效果。

虽然实验中的粒子效果较为基础，通过合理的算法和优化措施，如混合模式、点光滑功能和生命周期管理，让我对粒子系统有个详细的学习认识。未来研究可以引入复杂的物理模拟和视觉特效，提升粒子系统的真实性和多应用场景，如果有机会进入游戏开发、动画制作和科学模拟等领域，相信有了本次基础能够快速上手进行应用。

综上所述，本次研究验证了粒子系统在实时渲染中的可行性和有效性，为后续的深入研究和应用奠定了基础。

参考文献

[1] 蔡琦,王竞轩. 粒子系统在动画特效制作中的应用[J]. 包装世界,2023(6):37-39.
[2]李松维,周晓光,王润杰,等. 基于粒子系统烟雾的模拟[J]. 计算机仿真,2007,24(9):199-201,231.
[3]佘少波,李凡,王丙乾,等.基于粒子系统的尾焰红外图像实时仿真技术[J].系统仿真技术,2022,18(04):285-290.