实验：粒子系统

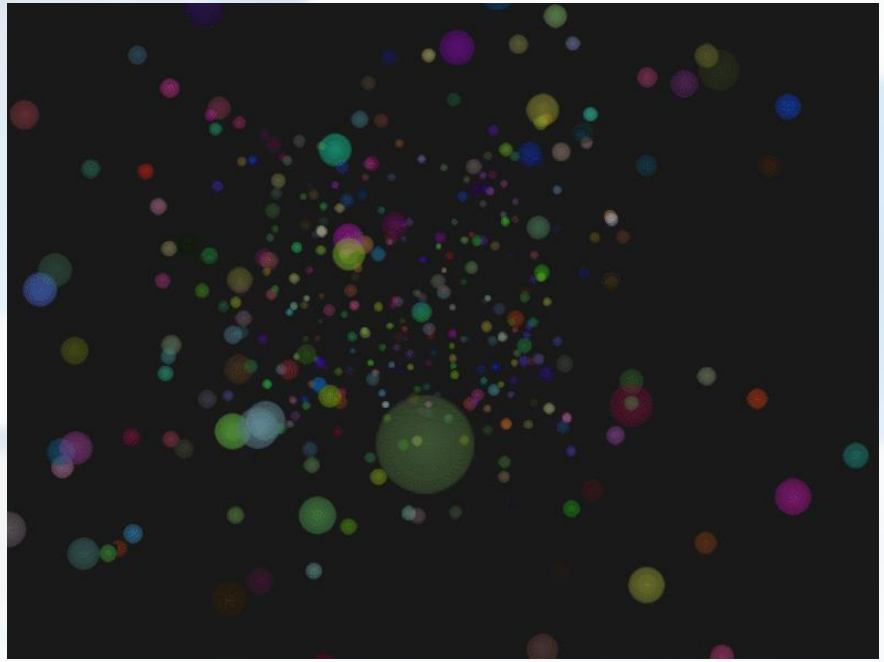
华中科技大学软件学院　万琳

提纲

1. 实验要求
2. 程序流程
3. 要点解析

1 实验要求

实现粒子系统：通过程序控制粒子的产生到消亡，以及根据粒子在它的生命周期里面的属性的变化，去动态的绘制粒子



2 程序流程

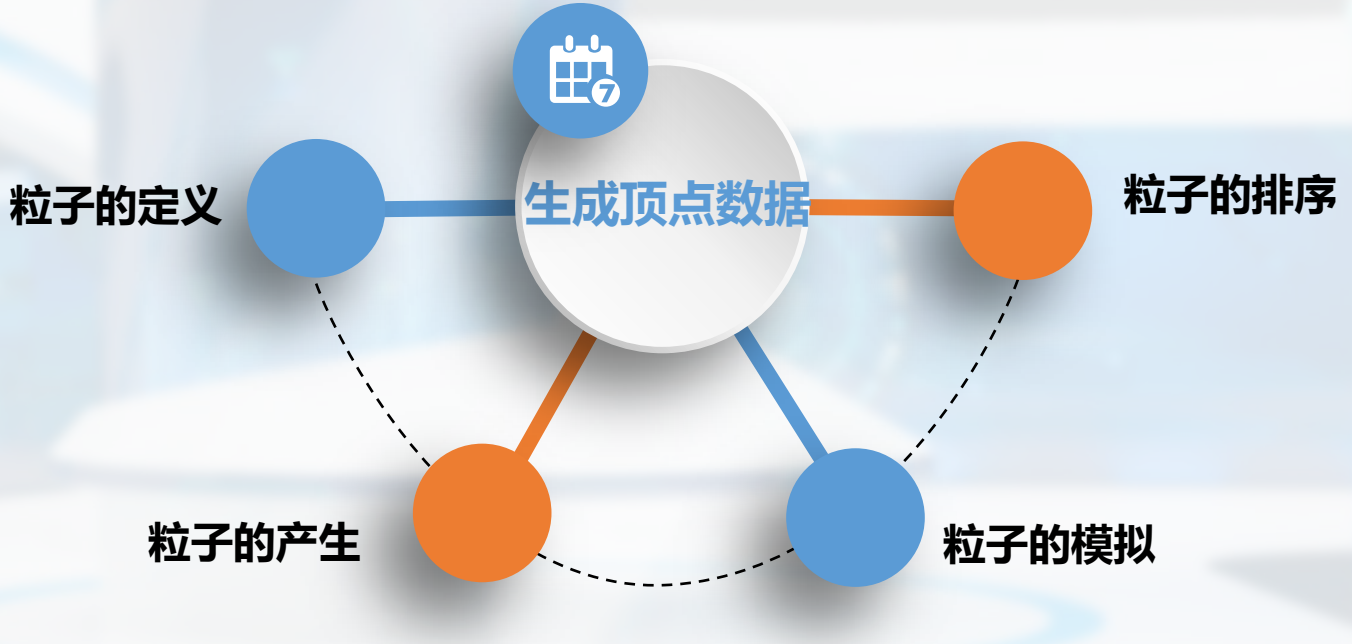
在窗口中绘制一个球。

3 要点解析

➢数据处理：生成顶点数据，生成和绑定VBO和VAO，设置属性指针

3 要点解析

➢数据处理：生成顶点数据，生成和绑定VBO和VAO，设置属性指针



3 要点解析

生成顶点数据

粒子的定义

粒子的基本属性

```
//粒子结构体定义
struct Particle {
        glm::vec3 pos, speed;
        unsigned char r, g, b, a; // 颜色
        float size; //粒子大小
        float life; // 粒子的剩余生命，小于0表示消亡.
        float cameradistance; // 粒子和摄像头的距离

        bool operator<(const Particle& that) const {
                // 逆序排序， 远的粒子排在前面
                return this->cameradistance > that.cameradistance;
        }
};
```

3 要点解析

生成顶点数据

粒子系统的属性

```
const int MaxParticles = 200; //最大粒子数
const float life = 2.0; //粒子的存活时间
glm::vec3 startPos = glm::vec3(0.0f, 0, 0.0f); //粒子起点
glm::vec3 endPos = glm::vec3(0.0f, 0, 4.8f); //粒子终点
Particle ParticlesContainer[MaxParticles];
```

粒子的结构体数组，
用于存放所有的粒子

3 要点解析

生成顶点数据

粒子的产生

找到粒子结构体数组中已经消亡的粒子的位置，并生成新的粒子去替换它

```
// 消亡多少粒子，产生多少粒子
int newparticles = deltaTime / life * MaxParticles;

for (int i = 0; i<newparticles; i++) {
        int particleIndex = FindUnusedParticle();
        ParticlesContainer[particleIndex].life = life;
        glm::vec3 maindir = glm::vec3(0.0f, 10.0f, 0.0f); //主要方向
        产生随机的位置偏差randomdOffset（使用rand()函数）
        ParticlesContainer[particleIndex].pos = startPos + randomdOffset; //粒子起点
        ParticlesContainer[particleIndex].speed = (endPos - startPos) / life;
        产生随机的颜色、透明度、大小，并赋值给ParticlesContainer[particleIndex].
}
```

3

要点解析

生成顶点数据

粒子的模拟

```
int ParticlesCount = 0;
for (int i = 0; i<MaxParticles; i++) {
    Particle& p = ParticlesContainer[i]; // 引用
    if (p.life > 0.0f) {
        p.life -= deltaTime;
        if (p.life > 0.0f) {
            p.pos += p.speed * (float)deltaTime;
            p.cameradistance = glm::length(p.pos - CameraPosition)
            将粒子p的位置、大小和颜色，填充到particle_position_size_data与particle_color_data数组中
        }
        else   //已经消亡的粒子，在调用SortParticles()之后，会被放在数组的最后
            p.cameradistance = -1.0f;
        ParticlesCount++;
    }
}
SortParticles();
```

判断粒子是否消亡

3 要点解析

生成顶点数据

粒子的排序

根据粒子的cameradistance属性，给粒子排序

```
void SortParticles() {
        std::sort(&ParticlesContainer[0], &ParticlesContainer[MaxParticles]);
}
```

其中对粒子进行比较要追溯到粒子结构体中对”<”的重载

```
struct Particle {
        ...
        float cameradistance;
        bool operator<(const Particle& that) const {
                // 逆序排序， 远的粒子排在前面
                return this->cameradistance > that.cameradistance;
        }
};
```

最终会将粒子结构体数组ParticlesContainer中， cameradistance值大的粒子排在数组的前面。

3 要点解析

➢粒子的渲染

遍历所有的粒子

绘制粒子

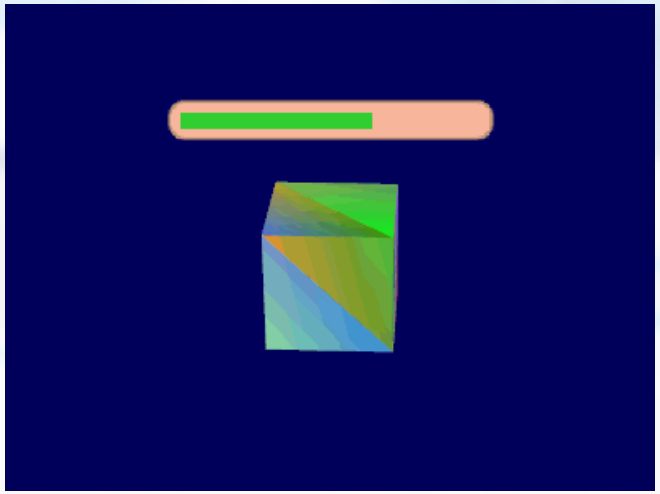
```
glBindVertexArray(VertexArrayID);
for (int i = 0; i < ParticlesCount; i++)
{
    shader.setVec4("xyzs", particle_position_size_data[4 * i + 0],
                           particle_position_size_data[4 * i + 1],
                           particle_position_size_data[4 * i + 2],
                           particle_position_size_data[4 * i + 3]);
    shader.setVec4("color", particle_color_data[4 * i + 0],
                           particle_color_data[4 * i + 1],
                           particle_color_data[4 * i + 2],
                           particle_color_data[4 * i + 3]);
    glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
}
```

3 要点解析

➢补充说明：公告板技术 Billboards （1）

为了让粒子始终面向摄像机方向，需要用到公告板（ Billboards）技术。

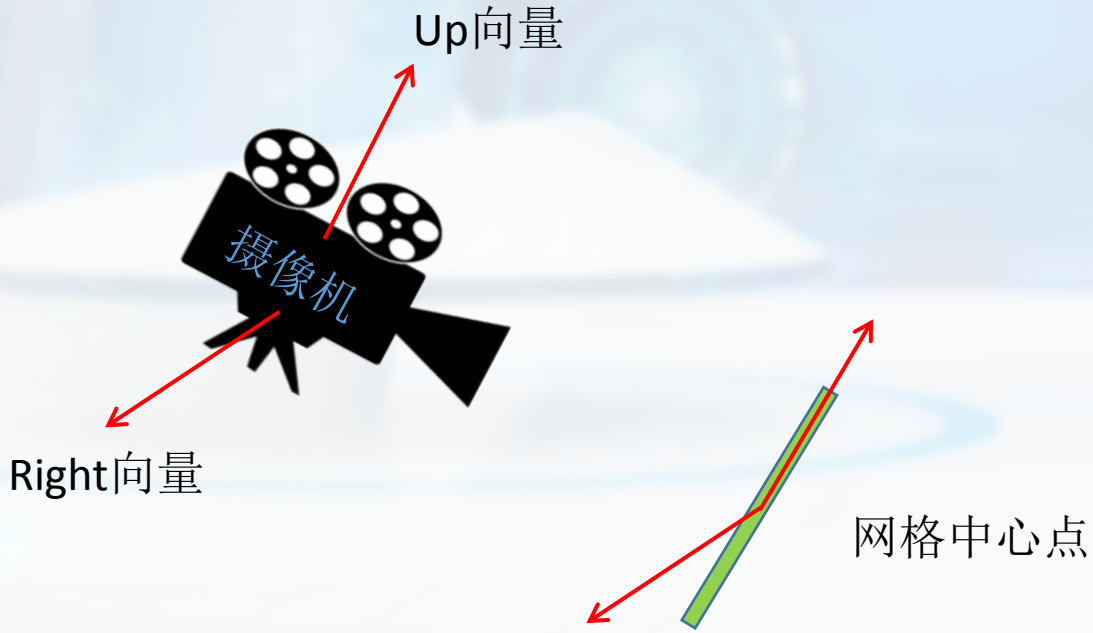
公告板技术，形象的来说，就好像一个人举着牌子，无论你从哪个方向看向牌子，那个人都会把牌子朝着你的方向旋转，你永远只能看到牌子的正面。在游戏中，npc 头上的名字或是血条的渲染用到了公告板技术。



3 要点解析

➢补充说明：公告板技术（2）

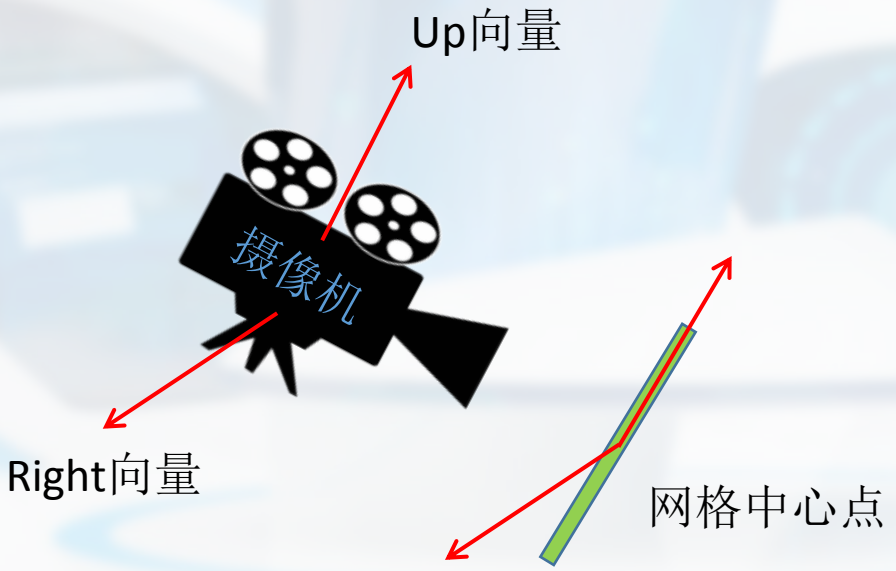
公告板技术主要利用一个正方形网格，根据当前的摄像机变换信息，调整至面向摄像机的位置。这里我们主要利用摄像机向上和向右的两个向量来调整正方形网格的点。



3 要点解析

➢补充说明：公告板技术（3）

将公告板技术应用到粒子系统当中去。根据粒子中心在世界空间的坐标，以及摄像机的UP和Right向量，算出粒子四个顶点的坐标，使得这个粒子是始终面向摄像机的。



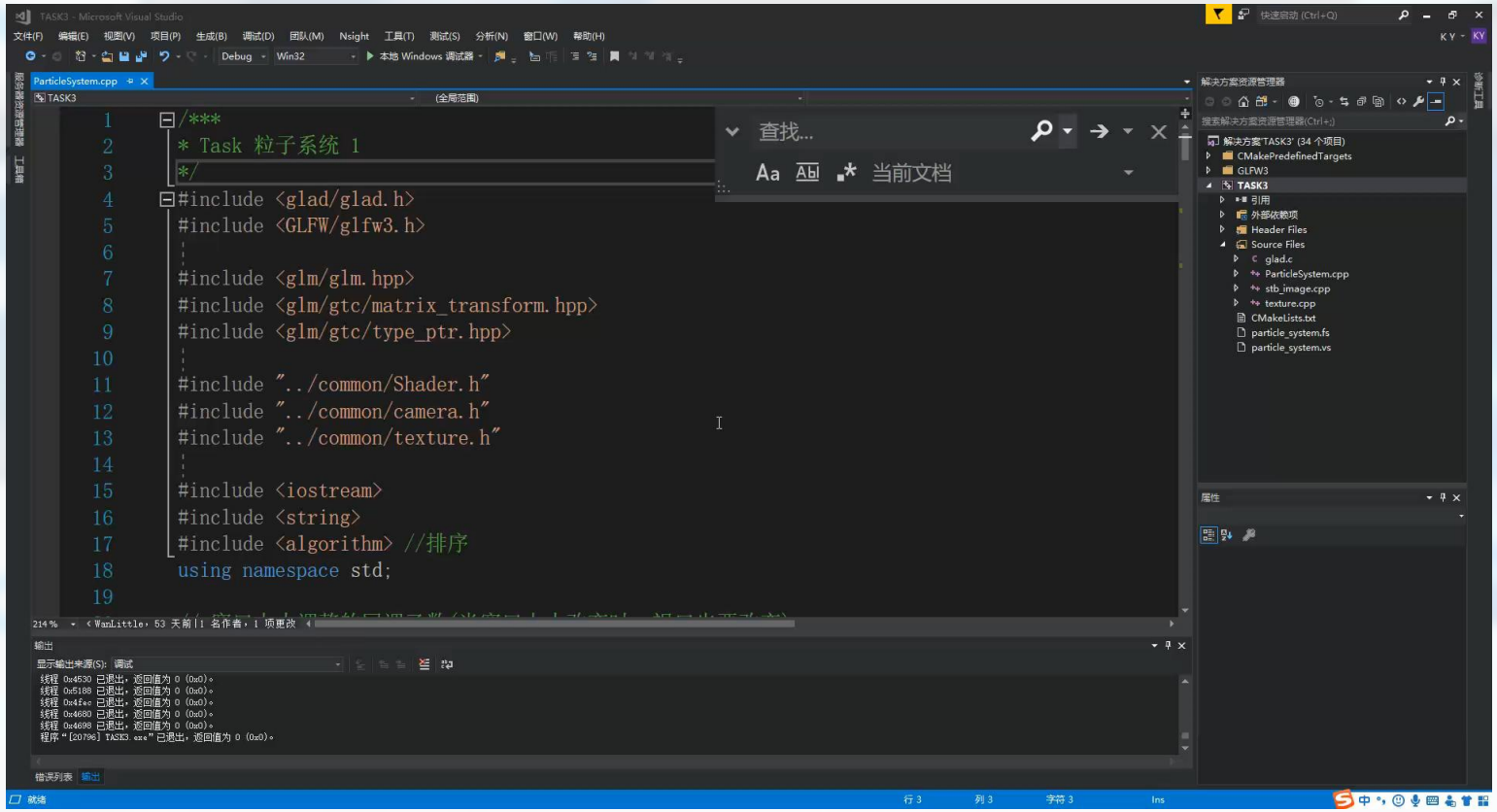

```
vec3 vertexPosition_worldspace
= particleCenter_wordspace
+ CameraRight_worldspace * particleSize
+ CameraUp_worldspace * particleSize;
```

其中
particleCenter_wordspace 对应图中的网格中心点
CameraRight_worldspace 对图中的Up向量
CameraUp_worldspace 对图中的Right向量
particleSize 粒子的大小

而Up和Right向量，可以从View矩阵中直接获取到。
CameraRight_worldspace = {ViewMatrix[0][0], ViewMatrix[1][0], ViewMatrix[2][0]}
CameraUp_worldspace = {ViewMatrix[0][1], ViewMatrix[1][1], ViewMatrix[2][1]}

4 程序演示

在下面的视频中先后对**最大粒子数**、**粒子的大小**进行修改，运行时可以看到明显的效果变化。




谢谢
软件学院　万琳