****

**《计算机图形学》**

**（作业三）**

|  |  |  |
| --- | --- | --- |
| **学院名称** | **：** | 数据科学与计算机学院 |
| **专业（班级）** | **：** | 17计算机科学与技术 |
| **姓名学号** | **：** | 17341067江金昱 |

# 方案

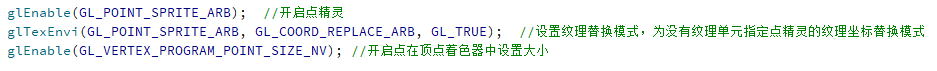
利用Opengl中的点精灵配合点的绘制方式来模拟球体粒子。

通过shader达到对每个粒子的控制，其中顶点着色器用来模拟粒子大小以及粒子运动，片元着色器用来控制粒子的形状以及粒子的色彩表现。

每帧绘制采用绑定顶点数组对象（VAO）的方式，来实现对顶点数据的便捷访问与绘制。

# 功能解析

## 点精灵

 想要自定义粒子，需要使用点精灵，并开启相关设置，之后才能在shader中变更点精灵的样貌，具体配置如下：

## Shader使用

使用glCreateShader创建shader

glShaderSource载入shader代码

glCompileShader编译Shader

glGetShaderiv可知道编译是否成功

glGetShaderInfoLog可得到编译输出的日志

glAttachShader得到shader的控制权

glLinkProgram链接shader

glGetProgramiv可知道链接是否成功

glGetProgramInfoLog可得到链接输出的日志

当编译成功之后调用glUseProgram即可使用shader

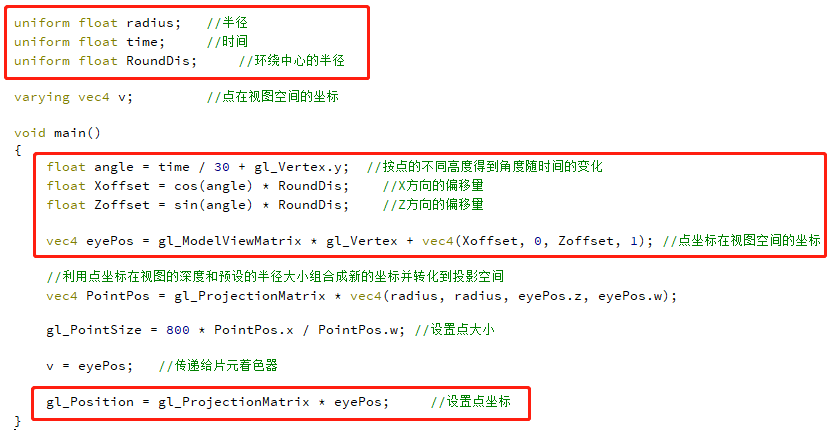
通过glGetUniformLocation可得到参数位置，并使用glUniformXX系列函数来配置参数

## 顶点着色器

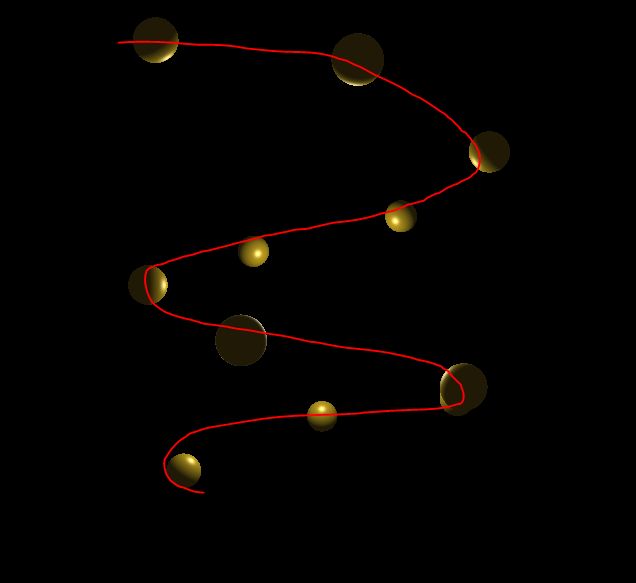
### 球体粒子运动

如上图在已知角度θ的情况下，假设轨迹为半径为1的单位圆，可以得出圆上点（x,y）的坐标为（cosθ,sinθ）。

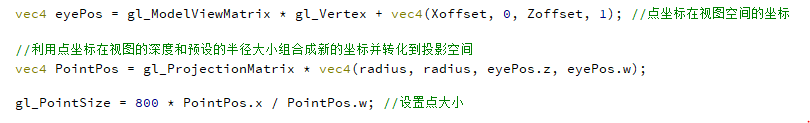
参照这个坐标公式，以粒子原始坐标为中心O，环绕半径设定为RoundDis，则粒子的位置偏移量为RoundDis\*（cosθ,sinθ）。此时只要不断改变θ的值，粒子就能绕圆进行运动，因此以时间作为θ的变化因子，θ就能随时间变化而变化。因为同时绘制的点有多个，为了让粒子间的运动产生区别，于是将粒子的高度y也引入到θ作为变化因子。

具体代码如下：

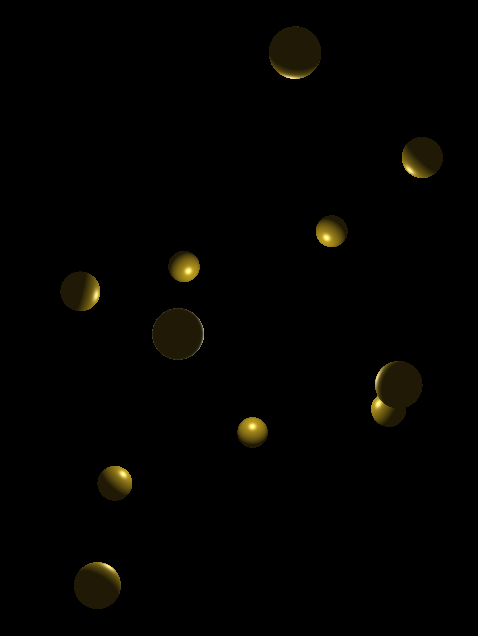
最终效果：

粒子以螺旋形的方式在空间中排布旋转。

### 球体粒子大小控制

利用粒子在视图空间内位置的深度（Z方向）作为大小的参照因子，并假设粒子位于屏幕正中心，将粒子半径radius映射到X轴Y轴，并与实际坐标结合形成一个新的坐标（radius, radius，z，eyePos.w），通过将该坐标变换到投影空间，访问最新的X或Y的映射数据即可得到最终的粒子半径大小，具体代码如下：

具体效果，近大远小：

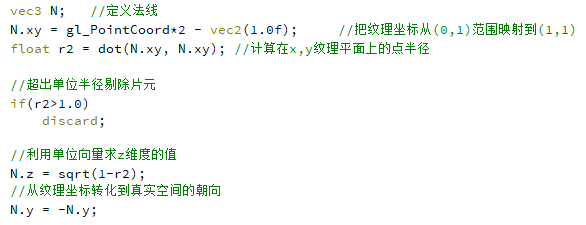


## 片元着色器

### 法线与形状

如图因为一开始配置过粒子的纹理，所以此时所有片元的纹理坐标值大小在0-1之间。这时候相当于球心的中心纹理坐标为（0.5， 0.5）。为了更方便运算，将球心的纹理坐标移动到（0,0）位置，并将纹理坐标映射到（-1，1）这个范围，之后只需要通过纹理坐标就可以判断该片元是否落在圆形范围内，不落在该范围的片元剔除即可。

因为圆表面点的法线必定通过圆心，而上述操作已经将圆心移动到了（0,0）点，因此可以利用单位向量长度为1，在已知X,Y坐标的情况下求得Z的坐标，法向量就能求出来了。

具体代码如下：

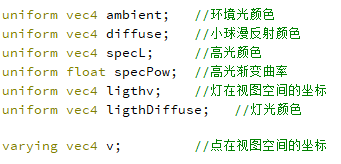
### Phong Shading

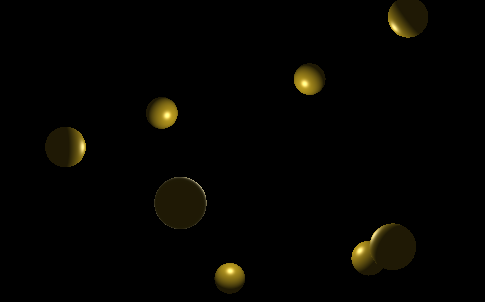
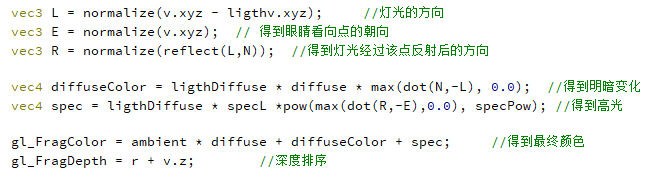
Phong Shading由环境光，漫反射以及高光组成。

通过已知灯和点的坐标可得灯光方向L，点积灯光和法线方向可以得到物体表面的明暗关系即漫反射，为了防止这个结果为负数，这里使用了max函数。

之后通过反射L得到出射光的方向R，点积R和点到眼睛的方向-E，可以得到在该观察视角下的表面高光亮度。

因为物体背面的法线N与灯光L点积之后的结果通过max函数后肯定会为0也就是黑色，为了让颜色不那么黑，引入了环境光，将其与漫反射相乘得到背后的颜色。

 需要向片元着色器传递的信息：

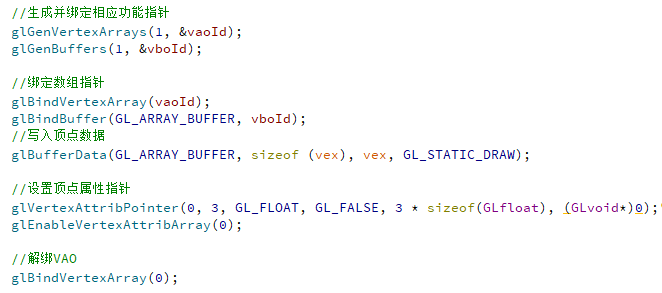
 具体代码如下：

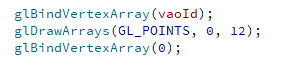
最终效果：

## VBO/VAO

顶点缓冲对象(Vertex Buffer Objects, VBO)，它会在GPU内存（通常被称为显存）中储存大量顶点，这使得我们可以一次性的发送一大批数据到显卡上，而不是每个顶点发送一次。

顶点数组对象(Vertex Array Object, VAO)，用于存储顶点属性调用。当配置好顶点属性指针时，你只需要将那些调用执行一次，之后再绘制物体的时候只需要绑定相应的VAO就行了，这使在不同顶点数据和属性配置之间切换变得非常容易。

具体代码如下：

之后在渲染循环中就能很方便的使用：

# 总结

1. 困难：在实现Phong Shading的时候缺少了粒子表面的法线这个关键变量，导致后续的光照计算无法进行下去。

解决方法：通过开启点精灵的纹理替换模式，使用纹理坐标来模拟出表面法线的方向。

1. 困难：通过纹理坐标直接转化出来的法线方向角度整体有偏转

解决方法：后来发现是因为纹理坐标系的方向和3维坐标系的Y轴方向是相反的，所以只需要乘以-1翻转即可。

1. 困难：多个粒子间深度无法确定，导致渲染顺序出错

解决方法：片元着色器中的片元深度采用粒子在空间坐标中的深度并累加上像素距离球体中心的距离来解决。

通过这次的作业，了解到了粒子的实现思路，并且学习到了shader的用法，给了自己很大的灵感在之后的图像处理上，未来可以利用shader慢慢实现很多炫酷的效果。