作业 1-闪烁的五角星

1. OpenGL 环境

本次作业采用 Freeglut+Glew 可编程管线，使用 CLion+CMake(MinGW)进行编译。

2. 窗口初始颜色

白色五角星窗口的初始颜色为天蓝色，立体五角星窗口的初始颜色为黑色。

3. 白色五角星

3.1. 绘制思路

五角星边框为闭合线框，直接使用 GL_LINE_LOOP 模式绘制即可，而内部并非凸多边形，无法直接使用 GL_POLYGON 绘制，因此将内部划分为 10 个小三角形进行绘制。渲染白色五角星的着色器为 simple.vert 与 simple.frag.

3.2. 顶点坐标与索引

利用数学方法计算每个顶点的坐标，为方便之后立体五角星的渲染，中心点的 z 坐标稍大于边框顶点的 z 坐标。内部三角形为白色，外边框为黑色实线，绘制外边框时使用相同的顶点缓存，事先保存一份索引数据。考虑到之后计算顶点法向量时各个三角形需要分开处理，因此将各个三角形的顶点数据分别保存，各个顶点的缓存索引如图 1 所示。

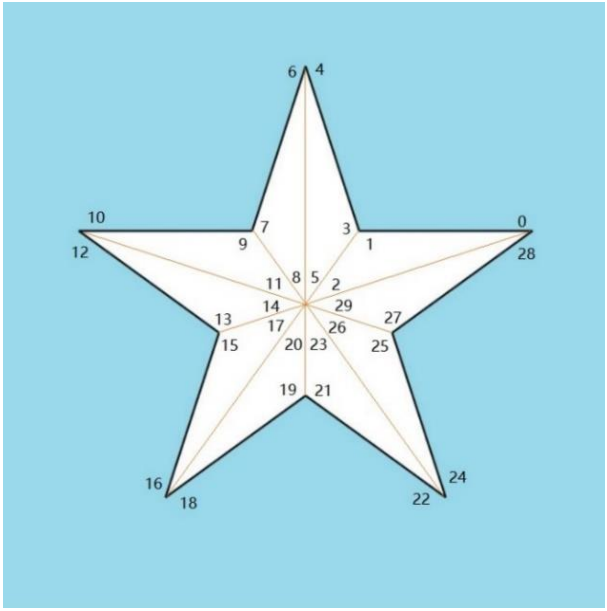

图 1 内部三角形顶点索引

3.3. 抗锯齿

绘制时启用颜色混合与 GL_LINE_SMOOTH 模式对黑色边框进行抗锯齿处理，而内部三角形的外边框被实线遮挡，无需进行处理，若处理反而会使三角形交界处出现割裂。

4. 立体五角星

4.1. 绘制思路

立体五角星没有外边框，利用 3.2.小节中的顶点缓存绘制 10 个小三角形即可。与白色三角形不同，立体三角形采用交大红作为基础颜色，使用光照明模型进行渲染。最终呈现效果如图 2 所示。包含光照明模型的着色器为 illumination.vert 与 illumination.frag.

图 2 立体三角形实现效果

4.2. 光照明模型

通过查询资料与学习，本次作业采用点光源模型，包括：环境光、漫射光以及镜面反射光。所有光照模型均根据基础算法实现，在后续课程学习中将进一步实现更为复杂化的模型。

4.2.1. 环境光

环境光与光源无关、没有方向，对场景中所有位置产生相同点亮效果，即基础亮度。环境光的实现较为简单，计算公式如下：

$$Ambient = \begin{bmatrix} R_l \\ G_l \\ B_l \end{bmatrix} * I$$

其中$R_l G_l B_l$为环境光的颜色值，I 为强度。在本项目中环境光强度 I 默认为 1.0

4.2.2. 漫射光

光线入射到物体表面时，反射的漫射光无方向区别，强度与光线入射的方向有关。当光线垂直入射时强度最大，随着入射光线与发现夹角的增大，漫射光强逐渐减小；夹角大于 90°时，光强为 0（照射到材质的另一面）。计算公式如下：

$$Diffuse = \begin{cases} \begin{bmatrix} R_l \\ G_l \\ B_l \end{bmatrix} * \cos(< -vec_{in}, vec_{nor} >), \text{夹角小于 } 90° \\ \mathbf{0}, \text{夹角大于 } 90° \end{cases}$$

其中vec_{in}，vec_{out}分别为为入射方向向量和法线向量，计算$\cos(< -vec_{in}, vec_{nor} >)$时将向量标准化后求点积即可。

关于顶点法向量vec_{nor}的计算：一般光滑模型表面顶点法向量的计算方式为所有包含该顶点的三角形面片的法向量求和，而五角星各面交界处非连续，需要对各个三角形的顶点分别保存一份法向量数据。计算法向量时，将相交两边的向量求叉积并标准化。

4.2.3. 镜面反射光

镜面反射光与漫射光类似，不同之处在于反射光线的强度与视角有关。当反射光线与视线重合时强度最大，随着两者夹角的增大而下降，夹角大于 90°时光强为 0.计算公式如下：

$$Specular = \begin{cases} \begin{bmatrix} R_l \\ G_l \\ B_l \end{bmatrix} * \begin{bmatrix} M_R \\ M_G \\ M_B \end{bmatrix} * cos^p(< vec_{out}, vec_{eye} >), \text{夹角小于 } 90° \\ \mathbf{0}, \text{夹角大于 } 90° \end{cases}$$

其中vec_{out}，vec_{eye}分别为为出射方向向量与入射点到视点的向量，$M_R M_G M_B$为镜面反射强度，p 为镜面发光参数（表现为反射光斑的边缘衰减率），M 和 p 均由照射物体的材料特性决定。本次作业中 M 的三色反射强度均设为 0.82，p 设为 8.0，接近于金属材料。

4.2.4. 点光源

照射五角星的点光源位于$(-0.3, 0.7, 1.0)$，颜色为白色，在每个顶点处以上三种光线叠加，与材质颜色混合计算出最终渲染到窗口中的 RGB 值，计算公式如下：

$$Color = (Ambient + Diffuse + Specular) * Texture$$

4.3. 增加贴图

仅仅绘制出以上五角星稍显单调，故在其一角处增加体现交大属性的贴图。贴图以材质的形式导入，对需要贴图的三角形的每个顶点绑定材质坐标，写入顶点数据缓存中一并输入着色器程序，最终将贴图与五角星本身的颜色混合后绘制在窗口中。下图为贴图（左）与增加贴图后的五角星（右）：

图 3 贴图与立体五角星

4.4. 抗锯齿

与 3.3.小节相同，开启颜色混合与 GL_POLYGON_SMOOTH 模式对五角星的外边框进行抗锯齿处理。此时为了防止三角形边界处割裂，事先额外绘制一层作为底色。

5. 动态背景

5.1. 绘制思路

动态背景设计的关键在于动画的实现，即每一帧图像各元素的属性如何更新。本次作业实现的动态背景由许多小五角星构成，每个小五角星按照一定的方式动态向外发散，随时间改变位置与颜色。动态背景的着色器为 particle.vert 与 particle.frag.

5.2. 每个小五角星的实现

为减少代码工作量，每个小三角形均以材质贴图的形式呈现。在指定位置绘制指定出一个正方形，正方形的 4 个顶点分别绑定五角星材质的 4 个顶点。由于需要每一帧更新正方形的位置，因此顶点数据采用 GL_STREAM_DRAW 模式保存。

5.3. 颜色混合模式

为呈现良好的视觉效果，为背景绘制开启颜色混合，抗锯齿采用默认的混合方式，而此处的混合方式为：

$$Color_{src} * Alpha_{src} + Color_{dst}$$

开启颜色混合前后的效果如下图所示：

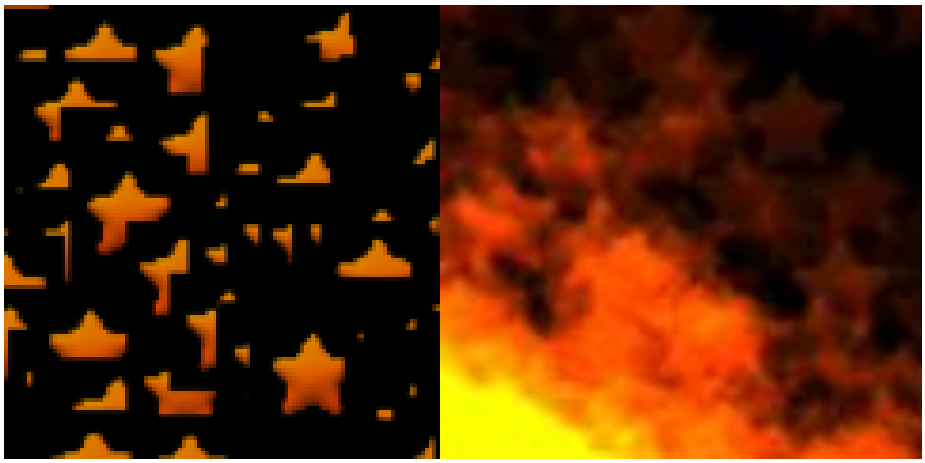

图 4 开启颜色混合前（左）后（右）

5.4. 位置计算

每个小五角星（以下简称为粒子）的属性包括：位置、速度（矢量）、加速度（矢量）、生命值、生命值衰减率，运动方式为：在指定位置生成，每一帧之后更新位置与速度，当某个粒子的生命值衰减为 0 时，重置生命值并回到起始位置，由此形成循环往复的动画。

程序启动时，首先初始化指定数量的粒子，设置各项属性的初始值。粒子的初始位置均匀分布于一个中心与大五角星重合的圆周上，如下图所示：

图 5 背景粒子初始位置示意图

共 N 个粒子，圆周半径为 r，第 i 个粒子的位置计算公式为：

$$angle = \frac{2\pi}{N} * i$$

$$\begin{cases} x = r * \cos(angle) \\ y = r * \sin(angle) \end{cases}$$

初始速度方向的设置方式有两种，一种为固定由屏幕中心指向初始位置，即由中心向外发散，以上背景粒子都采用这种方式，第二种为随机指向四周，形成点状扩散斑点，这种方式稍后会用到。速度大小在适当的范围内随机生成，加速度同理。初始速度是粒子生命值重置时恢复速度的 5 倍，以此造成启动背景时的爆炸效果

5.5. 颜色计算

粒子的颜色与时间有关。随着粒子向外扩散，在生命值降低的同步减小粒子的 alpha 值，

当生命值为 0 时 alpha 也为 0，表现为逐渐淡去直至消失。另外，随着时间推移所有粒子的整体颜色基调也发生变化，变化公式为：

$$angle = \frac{clock()}{3000.0} * \pi$$

$$\begin{cases} Color.R = Texture.R * Mask.R * (\sin(angle) * 0.4 + 0.6) \\ Color.G = Texture.G * Mask.G * (\sin(angle) * 0.4 + 0.6) \end{cases}$$

由于原材质颜色的 B 值恒为 0，因此不考虑 B 值的计算，最终效果为以 6 秒的周期按红绿黄三色交替变换。由于颜色混合计算的特性，所有粒子颜色的加和值一定为黄色，因此中心环形亮带的颜色始终为黄色。

每一帧更新粒子属性，最终形成连续的动画效果，如下图所示：

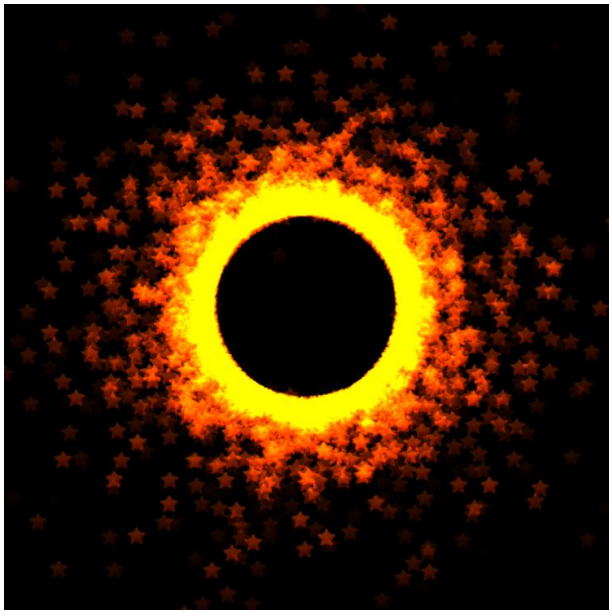

图 6 背景粒子动画截图

加上主体五角星后的显示效果如下图所示：

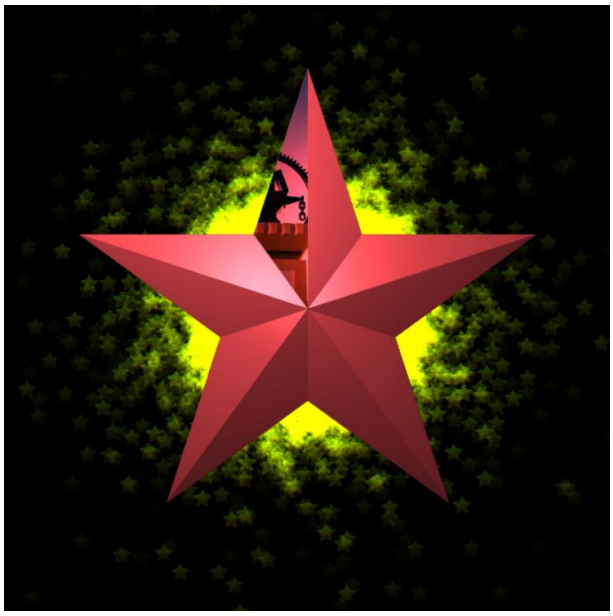

图 7 最终呈现效果

以上视觉效果类似于主体五角星向屏幕前方飞行，并带有彗星般的粒子轨迹，由于时间

精力有限，没有实现背景粒子的透视效果，如有则会更加真实立体。

6. 增加的鼠标功能

使用 freeglut 注册鼠标事件回调函数，实现鼠标粒子特效。在屏幕上点击鼠标左键，将部分粒子的初始位置设置到鼠标点击的位置，并使用 5.4.小节中提到的第二种速度初始化方式，会在该位置出现粒子爆炸特效；移动鼠标时动态更改粒子生命值为 0 时的重置位置，则会生成一条划过的轨迹，效果如下：

图 8 鼠标粒子特效

7. 其他功能

7.1. Reshape 回调函数

当窗口尺寸发生改变时，将视口扩展为当前尺寸，同时也更新所有顶点的坐标以保持原有比例，实现方式类似于固定管线的投影。为此需要在内存中重复保存一份顶点数据，每次窗口大小发生更改时，根据原有数据以及窗口长宽比例生成新的顶点数据并重新传入 VBO. 实现效果如下：

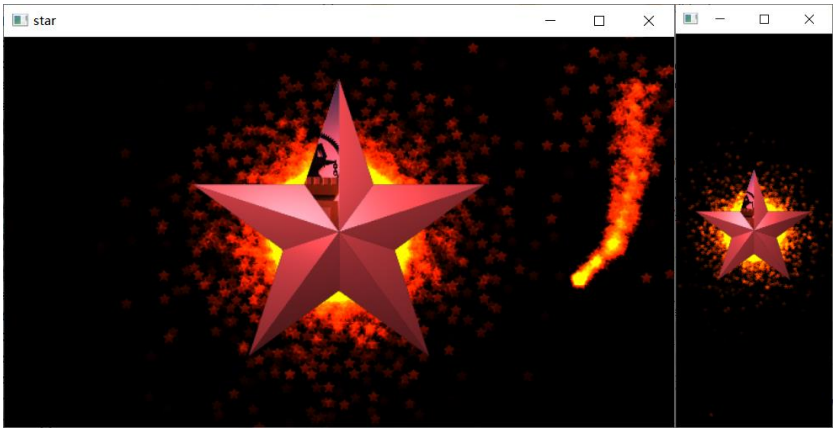

图 9 窗口尺寸发生改变时的效果

由于视口仍然为窗口的全部，显示范围并没有限制，如上图左侧窗口的右边部分仍然可以显示鼠标点击的粒子特效。

7.2. 资源加载

在开发过程中，着色器代码和材质的导入直接通过读取文件完成，需要配置相对路径，代码和材质将对用户可见，破坏程序的封装性，因此通过编写 python 程序，将以上内容以字符串形式生成至 c 头文件中，程序运行时直接从内存读取。在更改着色器代码和材质图片后，分别运行两个文件夹下的 loader.py 即可完成导入。实现此项功能后，Release 版本的可执行程序将没有任何依赖文件。