计算机图形学基础

B 类作业实验报告

杨宇菲 计算机系 2020215224
yangyufe20@mails.tsinghua.edu.cn

实验环境：Windows 10, Microsoft Visual Studio 2012（B1, B5）
Windows 10, Microsoft Visual Studio 2019（B2）

题目一：绘制出星星旋转的效果

- **实验原理**

选取阿基米德螺旋线，其极坐标方程为

$$r = a + b\theta$$

要使星星从原点生成并开始旋转，我们令$a = 0$，即$r = b\theta$。转换到平面直角坐标系为

$$\begin{cases} x = r\cos\theta = b\theta\cos\theta \\ \mathrm{y} = r\sin\theta = b\theta\sin\theta \end{cases}$$

- **实验步骤**

1. 参数设置

设星星之间的夹角 deltaTheta 为 22.5°, 即每圈 16 颗星星。最大转角 maxTheta 为 1800.0°, 即最多 5 圈星星。当转满 maxTheta 后，屏幕上存在星星数为 starNum=80。设阿基米德螺旋线参数 b=0.0415（使 5 圈星星恰能填满屏幕），转动速度 thetaSpeed 为 45.0°/s。

2. 随机颜色生成

在进入 Game Loop 之前, 先随机生成 maxNum 个颜色, 其中三个颜色分量均用(rand() % 256) / 255.0f 随机生成，并排除三个颜色分量均为 0 的情况，将这 maxNum 个颜色储存在 glm::vec3 类型 vector 中。

3. 星星的转动和绘制

转角最大的星星的角度为 curTheta，现有星星数量为 curNum = (int)floor(curTheta / deltaTheta) + 1。从转角最大的星星（i = 0）开始，遍历 curNum 颗星星，第 i 颗星的角度为 curTheta - i * deltaTheta，按实验原理中的公式计算其 x 和 y 坐标，用 glm::translate 获取位移 model 矩阵并传入顶点着色器，获取储存星星颜色的 vector 中的第 i 个颜色并传入片段着色器中，绘制。

要使星星转动，每个 Game Loop 中，使 curTheta 累加 thetaSpeed * deltaTime。当 curTheta >= maxTheta 时，去除转角最大的星星，即让 curTheta -= deltaTheta 为下一颗星星的的转角，删除颜色 vector 中的头元素，并在其尾部加入新生成的随机颜色。

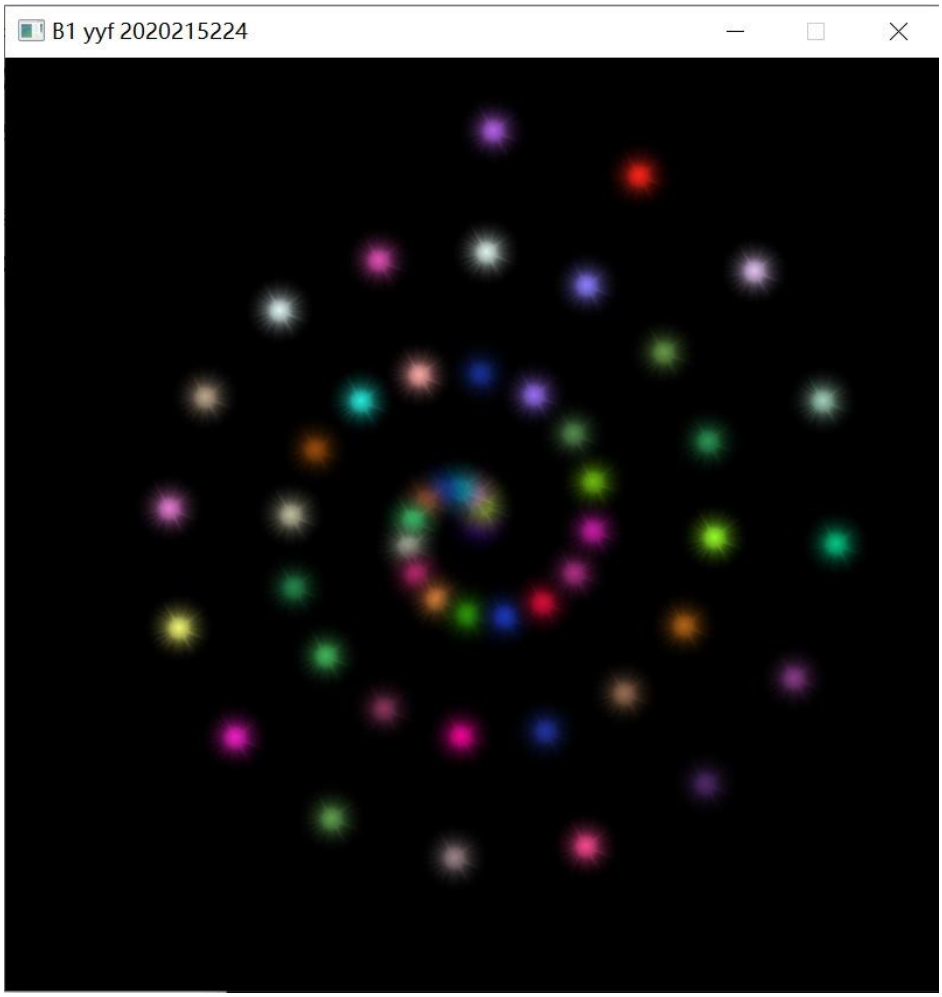
4. 对星星贴图的处理

由于贴图 Star.bmp 没有 alpha 通道，需要将其黑色背景消除，并使星星外围的光芒较为暗淡。为达到这一效果，在片段着色器中计算纹理坐标距纹理中心(0.5, 0.5)的距离 r，将 alpha 设为 r 的函数，

$$\mathrm{alpha} = \begin{cases} 0, & \mathrm{r} > 0.3 \\ 1 - \dfrac{\mathrm{r}}{0.3}, & 0 \le \mathrm{r} \le 0.3 \end{cases}$$

```
float r = sqrt(pow(TexCoord.x - 0.5f, 2.0f) + pow(TexCoord.y - 0.5f, 2.0f));
texColor = texColor * vec4(starColor, max(0.0f, 1.0f - r / 0.3f));
```

- **实验效果（见录屏）**



交互方式:
ESCAPE 键——关闭窗口

题目二：实现 5 个行星围绕太阳运动的效果同时添加字体效果

- **实验原理**

太阳系行星公转轨道面与黄道面夹角很小，可以认为其在一个平面内转动，转动轨道为以太阳为中心的椭圆。椭圆的极坐标方程为

$$r = \frac{a(1-e^2)}{1+e\cos\theta}$$

其中e为轨道偏心率，a为轨道半长轴，即(近日点距离+远日点距离)/2，$\theta = \omega t = \frac{360^\circ}{T}t$。

考虑到行星的近日点并不在一条直线上，还需要将椭圆旋转ϖ，

$$r = \frac{a(1-e^2)}{1+e\cos(\theta+\varpi)}$$

其中ϖ为近日点黄经，为升交点黄经Ω和近日点幅角ω之和。

再由椭圆极坐标转化到平面直角坐标。

- **实验步骤**

1. 行星参数

	Mercury	Venus	Earth	Mars	Jupiter
直径（km）	4879	12104	12756	6792	142984
近日点（10^6km）	46.0	107.5	147.1	206.6	740.6
远日点（10^6km）	69.8	108.9	152.1	249.2	816.6
公转周期（年）	0.24	0.61	1.00	1.88	11.86
轨道偏心率	0.205	0.007	0.017	0.094	0.049
近日点幅角（°）	29.13	54.9	102.9	286.5	274.3
升交点黄经（°）	48.3	76.7	0.0	49.6	100.4
公转方向	逆时针	顺时针	逆时针	逆时针	逆时针

设单位长度 unitLength 为 1000.0，单位公转周期 earthT 为 10.0s。

2. 绘制轨道

用绘制点的方式绘制椭圆曲线。每个椭圆用 poinNum=3600 个点来绘制，第 j 个点的角度为 360.0 / pointNum * j，用实验原理中的公式计算其 x 和 y 坐标，用 GL_POINTS 模式绘制点。

3. 绘制太阳和行星

对每个行星的角度，根据其转动方向，加上或减去$\frac{360°}{T}$，为防止角度随时间增长过大，将其限制在[0, 360°)或(-360°, 0]的范围内循环变化。用实验原理中的公式计算其x和y坐标，绘制绑定相应纹理的两个三角形。

4. 绘制文字

使用 RenderText 函数，在天体的左上角绘制文字。

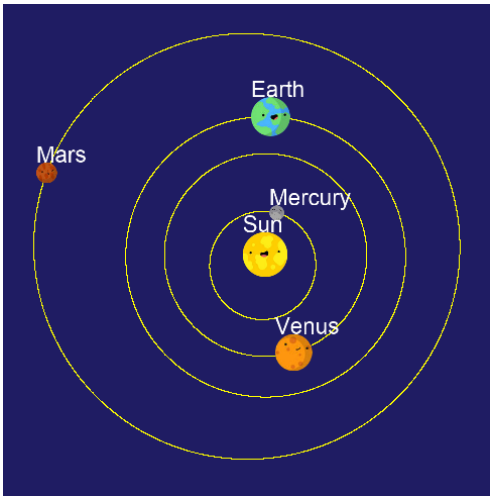
要注意的是在渲染文字时要用区别于储存天体和轨道顶点信息的另一组 VAO 和 VBO 动态存入文字顶点信息，并且绘制轨道、天体、文字使用的是三组不同的 shader，每次绘制前要进行 VAO 和 texture 的绑定以及 shader 的激活，绘制完成时解绑 VAO 和 texture。

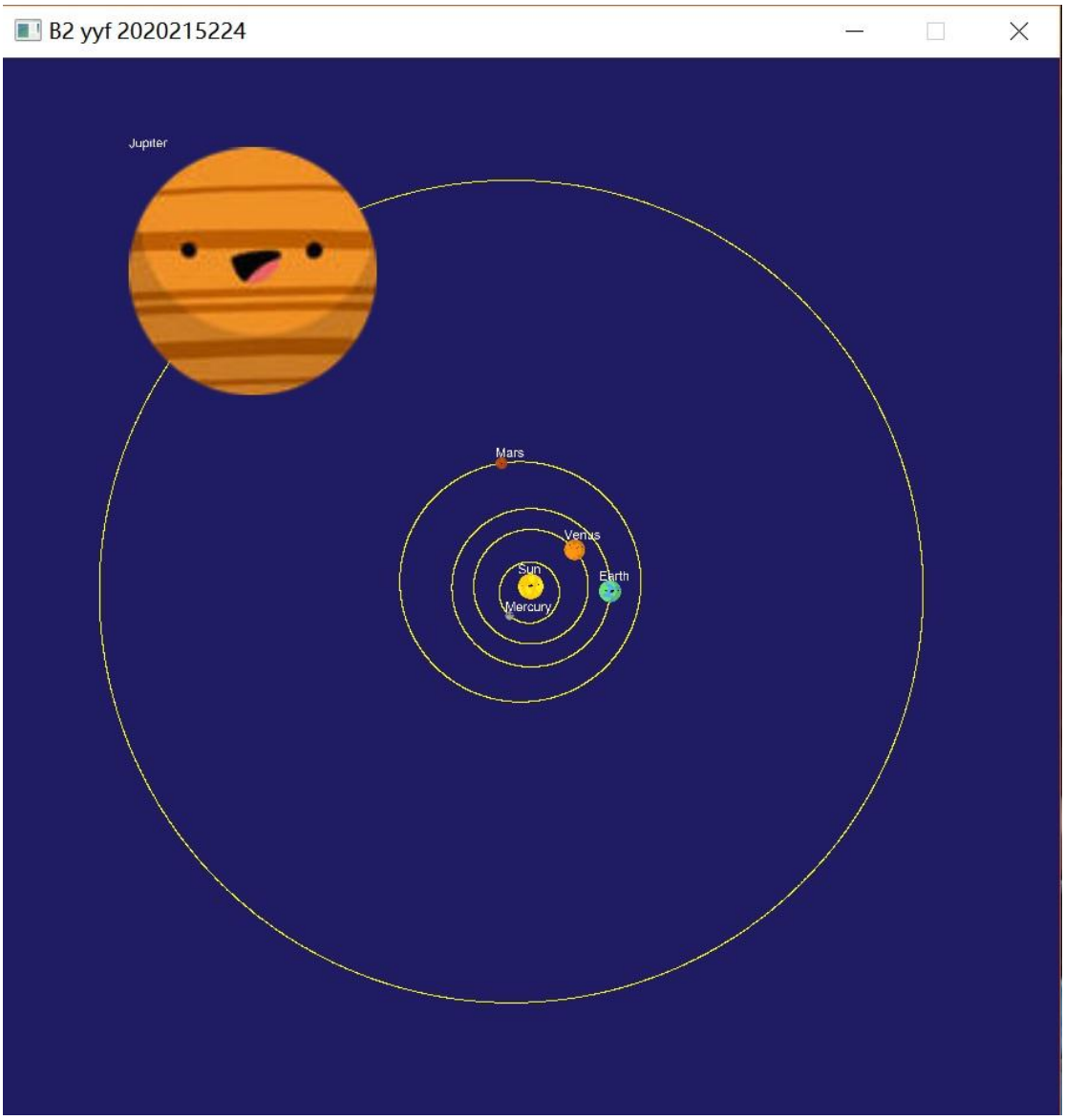
5. 交互

检测若按下方向键，单位公转周期∓deltaT * earthT（乘以 earthT 使旋转速度的变化更加平滑），使行星绕 z 轴旋转速度增大或减小。

检测鼠标滚轮位移 yoffset，单位长度 unitLength -= yoffset * deltaLength，实现放缩。

- **实验效果（见录屏）**





交互方式：

ESCAPE 键——关闭窗口

鼠标滚轮——视野缩放

方向左右键——减小或增大行星绕 z 轴旋转速度，单位（地球）公转周期最小为 2.0s，最大为 3600.0s

题目五：实现简单的粒子效果-下雪

- **实验原理**

创建 Flake 结构体储存每片雪花的三维坐标、下落速度和大小，并用 vector<Flake>进行储存，可以实现方便的遍历、生成新粒子和删除无效粒子。

- **实验步骤**

1. 创建 Flake 结构体

```
struct Flake{
    GLfloat x0;
    GLfloat y0;
    GLfloat z0;
    GLfloat speed;
    GLfloat size;
};
```

2. 参数设置

设置空间的长宽高分别为 skyLength = 40.0，skyWidth = 20.0，skyHeight = 20.0。雪花的最小速度和最大速度分别为 minSpeed = 1.0，maxSpeed = 4.0，最小尺度和最大尺度为 minSize = 0.1，maxSize = 0.5。每秒生成雪花数的初始值为 num_per_second = 1，每秒生成雪花数的最大值为 maxNum = 100，每秒生成雪花数保持恒定的时间为 constTime = 1.0。

3. 雪花的生成

GenTime 的初始值设为 0。每次 Game Loop 中，对 GenTime 累加 deltaTime，当 genTime 达到 1 / num_per_second 时，生成新的雪花，它的 x,z 坐标、下落速度和大小在给定范围内随机生成，其中产生[0, 1]之间的随机数使用(GLfloat)rand() / (GLfloat)RAND_MAX，y 坐标为 skyHeight - newFlake.size / 2.0。将这些值赋给 Flake 对象的属性变量，将此 Flake 对象加入 vector<Flake>的尾端。每生成一片新的雪花，将 genTime 重设为 0，重新开始计时。

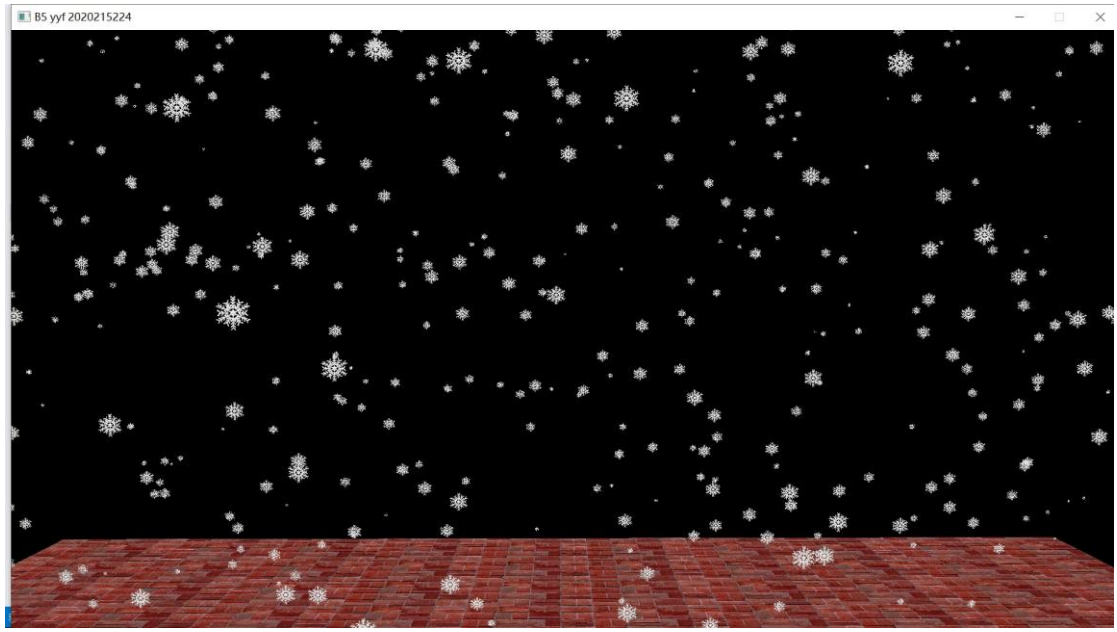
4. 雪花绘制和无效雪花的删除

使用 iterator 遍历 vector<Flake>。对每一片雪花，检测雪花的 y 坐标是否大于 size / 2.0。若大于，先后用 glm::translate 和 glm::scale 对单位 model 矩阵进行操作，实现对雪花的先放缩再移动，将 model 传入顶点着色器，然后进行绘制。每次绘制完成后，将雪花的 y 坐标减去 speed * deltaTime，实现雪花下落。若不大于，用 erase 函数删除这片雪花，并返回下一个 Flake 对象的指针。

5. 单位时间内生成雪花随时间增多

设 num 初始值为 0，每生成一片雪花 num += 1。当 num 达到 num_per_second * constTime 时，num_per_second += 2（限制最大值为 maxNum），并将 num 归零，重新计数，实现每秒生成雪花数每隔 constTime 增加 2。

- **实验效果（见录屏）**



交互方式：
ESCAPE 键——关闭窗口
W/A/S/D 键——相机位置前/后/左/右（自身坐标系）移动
鼠标滚轮——视野缩放
鼠标移动——改变相机俯仰角和偏航角