计算机图形学Project2程序说明

使用opengl绘制真实感图形

文件组织

> Debug
>
> Project2.exe 可执行文件
>
> main.cpp 主程序

代码说明

我绘制的图形是一个球体上运动的12辆小车

初始化

```
#define PI                  3.14159265358979323846264338327795

#define PLANET_RADIUS       100.0

#define NUM_CARS            12

#define CAR_LENGTH          32.0
#define CAR_WIDTH           16.0
#define CAR_HEIGHT          14.0

#define CAR_MIN_ANGLE_INCR  0.5     // 每帧小车绕球转的最小角度
#define CAR_MAX_ANGLE_INCR  3.0     // 每帧小车绕球转的最大角度

#define CAR_TOP_DIST        (PLANET_RADIUS + CAR_HEIGHT)  // 车顶到球心的距离

#define EYE_INIT_DIST       (3.0 * CAR_TOP_DIST)  // 人眼到球心的初始距离
#define EYE_DIST_INCR       (0.1 * CAR_TOP_DIST)  // 改变人眼距离时的增长量
#define EYE_MIN_DIST        (1.5 * CAR_TOP_DIST)  // 人眼到球心的最小距离

#define EYE_LATITUDE_INCR   2.0     // 改变眼睛的纬度时的增长量
#define EYE_MIN_LATITUDE    -85.0   // 眼睛的最低纬度
#define EYE_MAX_LATITUDE    85.0    // 眼睛的最高纬度
#define EYE_LONGITUDE_INCR  2.0     // 改变眼睛的经度时的增长量

#define CLIP_PLANE_DIST     (1.1 * CAR_TOP_DIST)  // 截面距离

#define DESIRED_FPS         60      // 每秒帧数
```

接下来对球体颜色，车轮颜色，物体材质，光源进行了参数设置，还定义了小车的数据结构，窗口大小，影响显示效果的几个变量，如眼睛的位置等。

绘制小车

画出两个方块作为车身，再画四个圆环作为轮子，使用矩阵对其大小和位置进行调整，拼出小车，代码较为繁琐，此处省略。

绘制出所有小车

先把小车平移到球体表面，再绕y轴旋转一定角度，最后绕xz平面的一个轴旋转一定角度，小车就可以沿着圆心在球心的最大圆行驶了。

```
void DrawAllCars(void)
{
    for (int i = 0; i < NUM_CARS; i++)
    {
        glPushMatrix();
        glRotated(car[i].rotAngle, car[i].xzAxis[0], 0, car[i].xzAxis[1]);
        glRotated(car[i].angularPos, 0, 1, 0);
        glTranslated(0, 0, PLANET_RADIUS);
        DrawOneCar(car[i].bodyColor);
        glPopMatrix();
    }

}
```

关于小车的参数

InitCars函数中设置了小车的车身颜色，绕各个轴的旋转角度，小车的行驶速度。

```
void InitCars(void)
{
    for (int i = 0; i < NUM_CARS; i++)
    {
        car[i].bodyColor[0] = (float)rand() / RAND_MAX;  // 0.0 to 1.0.
        car[i].bodyColor[1] = (float)rand() / RAND_MAX;  // 0.0 to 1.0.
        car[i].bodyColor[2] = (float)rand() / RAND_MAX;  // 0.0 to 1.0.

        car[i].angleIncr = (double)rand() / RAND_MAX *
            (CAR_MAX_ANGLE_INCR - CAR_MIN_ANGLE_INCR) + CAR_MIN_ANGLE_INCR;
        // CAR_MIN_ANGLE_INCR to CAR_MAX_ANGLE_INCR.

        car[i].angularPos = (double)rand() / RAND_MAX * 360.0;      // 0.0 to
360.0.

        // The following 3 items defines a random great circle.
        car[i].xzAxis[0] = (double)rand() / RAND_MAX * 2.0 - 1.0;   // -1.0 to
1.0.
        car[i].xzAxis[1] = (double)rand() / RAND_MAX * 2.0 - 1.0;   // -1.0 to
1.0.
        car[i].rotAngle = (double)rand() / RAND_MAX * 360.0;        // 0.0 to
360.0.
    }
}
```

小车行驶的实现

使用回调函数在每帧中改变小车在它的圆上的旋转角度，实现小车的移动。

```
void UpdateCars(void)
{
    for (int i = 0; i < NUM_CARS; i++)
    {
        car[i].angularPos += car[i].angleIncr;
        if (car[i].angularPos > 360.0) car[i].angularPos -= 360.0;
    }
    glutPostRedisplay();
}
```

```
void MyTimer(int v)
{
    if (!pauseAnimation)
    {
        UpdateCars();
        glutTimerFunc(1000 / DESIRED_FPS, MyTimer, v);
    }
}
```

键盘交互

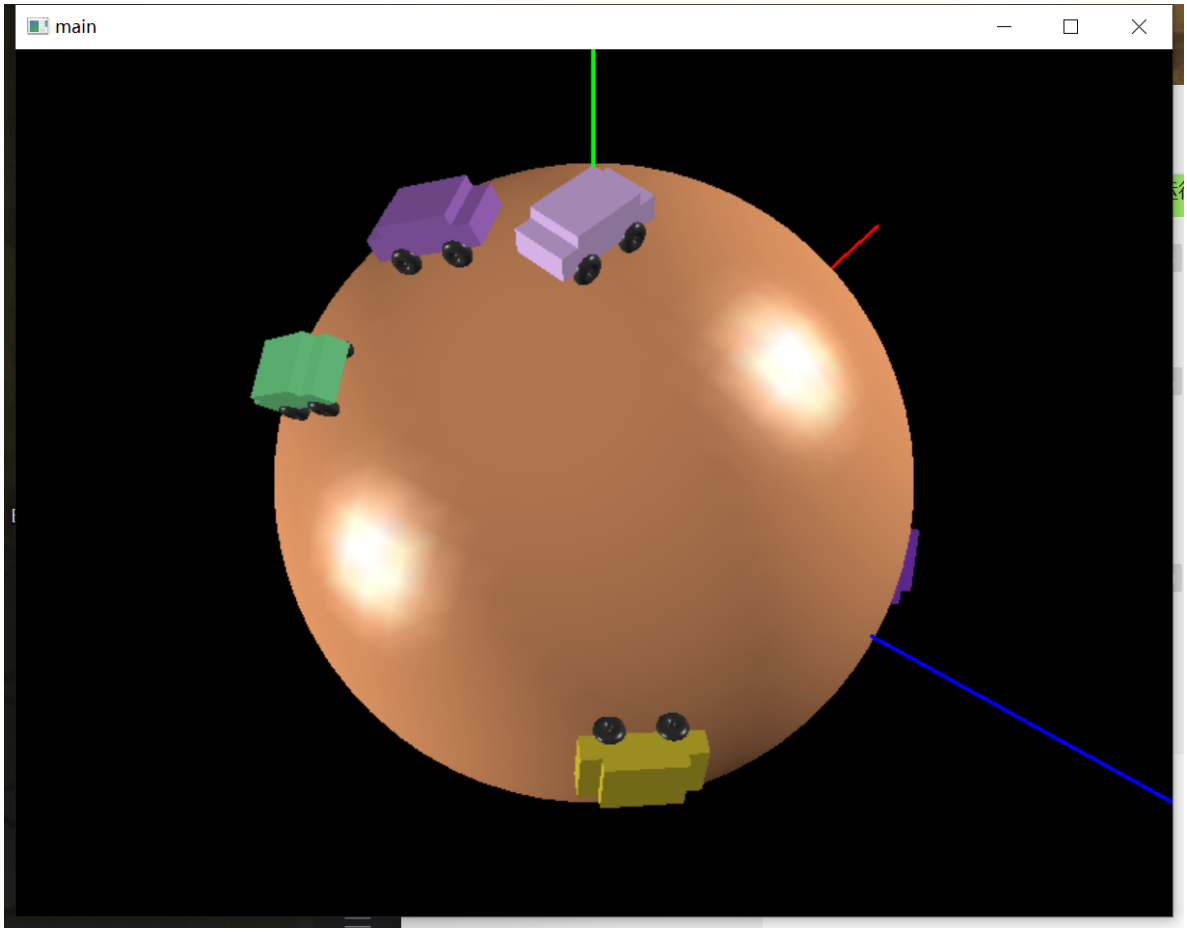
在`MySpecialKey()`和`MyKeyboard()`函数中，对各种键盘操作进行了定义

键盘操作	功能
q/Q	退出
p/P	暂停
w/W	线条还是填充
x/X	有无xyz坐标轴
r/R	视角重置
left	看左边，视点左移
right	看右边，视点右移
up	看上面，视点上移
down	看下面，视点下移
PgUp	靠近看
PgDn	离远看

下面的语句用来设置视线，上面的几个功能就是通过修改eyeLatitude，eyeLongitude，eyeDistance实现的

```
gluLookAt(eyeDistance * cos(eyeLatitude / 180.0 * PI) * sin(eyeLongitude / 180.0 * PI), eyeDistance * sin(eyeLatitude / 180.0 * PI), eyeDistance * cos(eyeLatitude / 180.0 * PI) * cos(eyeLongitude / 180.0 * PI), 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
```

效果



不足

没有解决小车碰撞之后的穿模问题，看多了还觉得很好笑。