魁地奇桌球 - 第二次迭代

杨铭
5130379022

2015 年 12 月 16 日

1 开发环境

操作系统 windows7 旗舰版

开发软件 Visual Studio 2015 community

图形库 OpenGL:gl、glut、glaux

2 简述

在第一次迭代的基础上，进行了旗帜的建模，使用 opengl 完成了下列功能：

- 加入了光照效果
- 用各种曲线函数设计并构建旗帜模型
- 在旗帜模型上加入自己独特的贴图
- 在旗帜模型中加入旗帜飘扬的相关动画

3 实现过程

3.1 概括

在工程中新建了 **flag.h,flag.cpp,glhf.h,glhf.cpp**四个文件，在 **flag.h** 和 **flag.cpp**中定义并实现了旗帜类，在 **glhf.h**中包含了 opengl 的一些核心库和辅助库并且为读取纹理包装了一个函数

3.2 读取纹理

使用 glaux 库的 AUX_RGBImageRec 类读取文件并且使用 glTexImage2D 函数将纹理映射到内存中

3.3 旗帜绘制

```
class Flag
{
public:
  Flag();

  void init();
  void render();
  void update();

  void setWind(int level);
  void windUp();
  void windDown();

private:
  GLfloat ctrlPoints[X_CTR_NUM][Y_CTR_NUM][3];
  GLfloat texpts[2][2][2];
  GLuint tex_ID;
  GLfloat dt;
  int w_level;
  GLfloat post_length, post_radius;
  GLfloat y_d, y_offset;
};
```

说明 类似其他的实体类，主要方法为 init、render、update 分别实现初始化、绘制和更新数据。成员变量中包含两个 3 维数组 **ctrlPoints**、**texpts**，分别为旗帜的曲面控制点和纹理的映射点。**tex_ID**保存着之前为旗帜读取的纹理的编号 **dt**是旗帜更新的时间间隔。**w_level**记录模拟风力的大小，它影响着 **y_d**、**y_offset**，从而影响旗帜飘动的频率、幅度和 y 轴偏移量，可以通过调用 **setWind**、或者 **windUp**、**windDown**来调节风力大小

绘制 在 render 函数中，使用 opengl 的二维差值器，实现了贝塞尔曲线的旗帜绘制和纹理贴制。

```
void Flag::render()
{
  glMatrixMode(GL_MODELVIEW);

  // draw flagpole
  glPushMatrix();
  glTranslatef(0.0f, 24.0f, 0.0f);
  glColor3f(0.5f, 0.25f, 0.0f);
  GLUquadric *pObj;
  pObj = gluNewQuadric();
  gluCylinder(pObj, post_radius,
              post_radius, post_length, 12, 1);
  glPopMatrix();

  // draw flag
  glPushMatrix();

  glTranslatef(3.8f, 24.0f, 8.5f);
  // bind texture id
  glBindTexture(GL_TEXTURE_2D, tex_ID);
  // enable 2D texture and 2D calculator to draw surface
  glEnable(GL_TEXTURE_2D);
  glEnable(GL_MAP2_VERTEX_3);
  glEnable(GL_MAP2_TEXTURE_COORD_2);

  glMap2f(GL_MAP2_VERTEX_3,0.0f,10.0f,3,5,
          0.0f,10.0f,15,2,&ctrlPoints[0][0][0]);
  glMap2f(GL_MAP2_TEXTURE_COORD_2,0.0f,10.0f,2,2,
          0.0f,10.0f,4,2,&texpts[0][0][0]);
  glMapGrid2f(10, 0.0f, 10.0f, 10, 0.0f, 10.0f);

  glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
            GL_DECAL);
```

```
34    glEvalMesh2(GL_FILL, 0, 10, 0, 10);
35    glPopMatrix();
36
37    glutSwapBuffers();
38    glDisable(GL_TEXTURE_2D);
39  }
```

更新　在 update 里更新控制点的坐标，用三角函数模拟旗帜迎风飘动的效果
本来想使用质点 -弹簧模型的布料模拟，但是时间所限导致实现效果不理想，暂时使用这种方式模拟旗帜的物理模型

3.4　光照

在 **main.cpp**中的 **initlights**中开启光照效果
首先开启光照效果和材质反光和颜色反光，设置环境光和光源位置、漫反射光和镜面发射等属性

```
1  void initlights(void)
2  {
3    GLfloat ambient[] = { 0.2, 0.2, 0.2, 1.0 };
4    GLfloat position[] = { 0.0, 0.0, 2.0, 1.0 };
5    GLfloat mat_diffuse[] = { 0.6, 0.6, 0.6, 1.0 };
6    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
7    GLfloat mat_shininess[] = { 50.0 };
8    glEnable(GL_LIGHTING);
9    glEnable(GL_LIGHT0);
10   glEnable(GL_COLOR_MATERIAL);
11   glEnable(GL_AUTO_NORMAL);
12   glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
13   glLightfv(GL_LIGHT0, GL_POSITION, position);
14   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
15   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
16   glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
17 }
```

4 结果展示

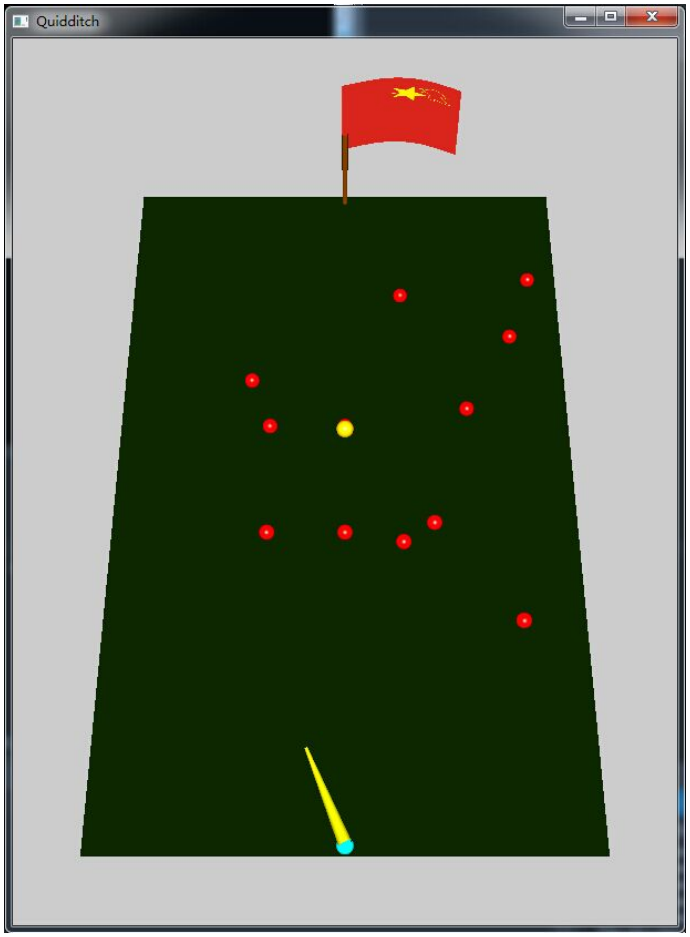

Figure 1: 效果图