成员报告

郭昊谋

17341044

负责部分： 海星和鱼的构造。**starfish.cpp**,**fish.cpp**

starfish.cpp

实际上海星的结构相当对称，可以通过简单几何体和贴图的形式将其表现出来。构造相关类只需要考虑 phong 反射模型，及几何体构造。

Renderable.h有关类的头文件

```
class StarFish : public Renderable
{
private:
    static GLfloat vertex[];    /// vertex array data
    static GLfloat normal[];    /// normals for each vertex
    static GLfloat colours[];   /// colour array data
    static GLfloat material[4]; /// material RGBA
    static GLfloat shininess;   /// 高光系数（ shininess） in phong reflection
model
public:
    StarFish();             /// default constructor
    virtual ~StarFish();    /// default destructor

protected:
    void _draw(void);   /// draws the StarFish
};
```

几何体构造：将海星分为5部分，只画出其中的一条腿，然后通过旋转即可构造出完整的海星。

```
void StarFish::_draw(void)
{
    // phong 反射模型 设置RGBA，高光属性等
    glMaterialfv(GL_FRONT, GL_SPECULAR, material);
    glMaterialf(GL_FRONT, GL_SHININESS, shininess);
    glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
    glEnable(GL_COLOR_MATERIAL);

    ...

    // 旋转重复画出5条腿，构成完整的海星
    // draw all 5 legs
    GLfloat step = 360.0f / 5;
    for (int i = 0; i < 5; i++)
    {
        glPushMatrix();
        glRotatef(i * step, 0.0f, 1.0f, 0.0f);
```

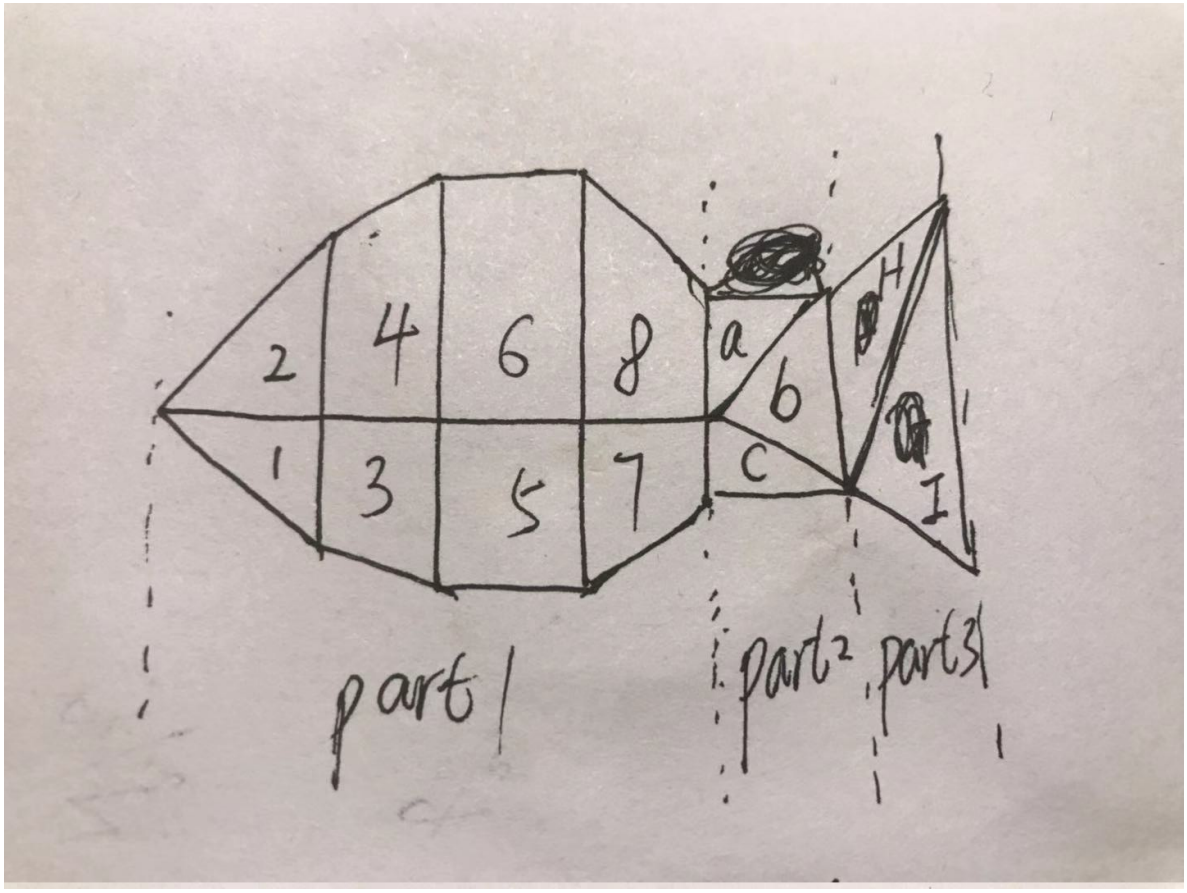
```
        glDrawArrays(GL_QUADS, 0, 4 * 5);
        glPopMatrix();
    }
    ...
}

/*
*   descrip:
*       海星可由5部分构成，只构造一条腿即可
*       vertex:
*           一条腿可以变换成四棱锥（并非真正意义上的）对应的点。
*           1,2,3,4对应4个面，X对应四棱锥的上平面
*       normal:
*           对应vertex的法线
*       colors:
*           对应vertex的颜色
*/
GLfloat StarFish::vertex[] =
    {
        //3                 |                   |                   |
        0.0f, 0.2f, 0.0f, 0.0f, 0.0f, 0.6f, 1.5f, 0.0f, 0.3f, 1.5f, 0.2f,
0.0f,
        //4                 |                   |                   |
        0.0f, 0.0f, -0.6f, 0.0f, 0.2f, 0.0f, 1.5f, 0.2f, 0.0f, 1.5f, 0.0f,
-0.3f,
        //1                 |                   |                   |
        0.0f, -0.2f, 0.0f, 1.5f, -0.2f, 0.0f, 1.5f, 0.0f, 0.3f, 0.0f, 0.0f,
0.6f,
        //2                 |                   |                   |
        0.0f, 0.0f, -0.6f, 1.5f, 0.0f, -0.3f, 1.5f, -0.2f, 0.0f, 0.0f, -0.2f,
0.0f,
        //X                 |                   |                   |
        1.5f, 0.2f, 0.0f, 1.5f, 0.0f, 0.3f, 1.5f, -0.2f, 0.0f, 1.5f, 0.0f,
-0.3f};
```

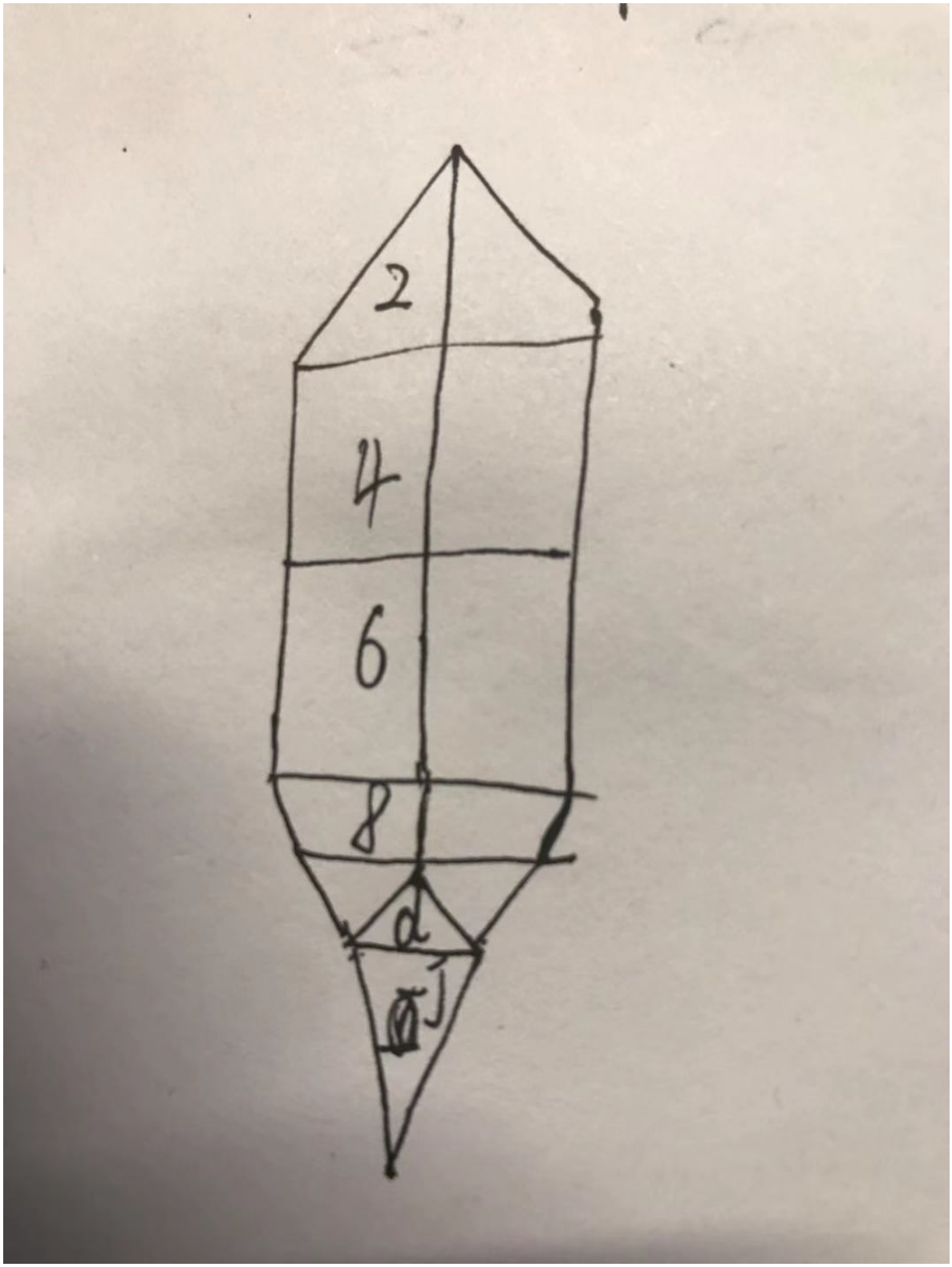
fish.cpp

鱼要实现运动，包括自身尾巴的摆动，并且需要通过贴图来实现逼真的模型。

鱼的构造可以变成对称的两部分，鱼的另一面只需要通过将原有的图形进行旋转即可获得。鱼的一面被分解为3部分，尾巴的摆动只需要part3进行简单的摇摆即可。

如图所示，以下为侧视图和俯视图（草图）,实际效果建议通过运行程序进行查看。需要注意的是，鱼尾巴的运动，实际上是尾巴相对鱼的运动，这和实验2机器人摆手是一样的。


2
4
6
8
1
3
5
7
a
b
c
H
I
part1
part2
part3



代码摘录fish.cpp

```
/// Draws the full fish
void Fish::_draw(void)
{
    //鱼的整体运动速度
    // the fish moving path
    GLfloat xInc = cos(ry * (3.14156) / 180) / 10.0f;
    GLfloat zInc = sin(ry * (3.14156) / 180) / 10.0f;

    // 允许鱼活动的范围，超出则相当于从对应的另一边的边界进入
    // the floor is 70.0 x 70.0
```

```
    // the fish keep inside a 65.0 x 65.0 area
    if (x < -35)
        x += 65.f;
    if (x > 35)
        x -= 65.f;
    if (z < -35)
        z += 65.f;
    if (z > 35)
        z -= 65.f;

    //刻画运动轨迹
    // increment the fish position
    x -= xInc;
    z += zInc;

    // set up the material properties (only front needs to be set)
    glMaterialfv(GL_FRONT, GL_AMBIENT, material);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, material);
    glMaterialfv(GL_FRONT, GL_SPECULAR, material);
    glMaterialf(GL_FRONT, GL_SHININESS, shininess);

    // enable texturing
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, FISH_TEXTURE);

    // set up texture parameters
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR_MIPMAP_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);

    // set up vertex arrays
    glVertexPointer(3, GL_FLOAT, 0, vertex);
    glNormalPointer(GL_FLOAT, 0, normal);
    glTexCoordPointer(2, GL_FLOAT, 0, texels);
    glColorPointer(3, GL_FLOAT, 0, colours);

    // enable vertex arrays
    glEnableClientState(GL_NORMAL_ARRAY);
    glEnableClientState(GL_COLOR_ARRAY);
    glEnableClientState(GL_VERTEX_ARRAY);
    glEnableClientState(GL_TEXTURE_COORD_ARRAY);

    //画出鱼的一面
    // CounterClockWise
    glFrontFace(GL_CCW);
    drawSide();

    //画出鱼另的一面
    glScalef(1.0f, 1.0f, -1.0f);
    // ClockWise
    glFrontFace(GL_CW);
    drawSide();

    // 刻画尾巴的运动，限制角度范围为tailAngleCutOff
    GLfloat pt = sin(tailAngle * 3.14159 / 180);
```

```
    tailAngle += tailAngleInc;
    if (tailAngle < -tailAngleCutOff || tailAngle > tailAngleCutOff)
        tailAngleInc *= -1;

    //由于尾巴是相对鱼其它部分运动的，需要另行操作
    vertex[143] = vertex[152] = vertex[149] = vertex[158] = vertex[167] = pt;
    glDrawArrays(GL_TRIANGLES, 6 + (4 * 6) + (3 * 5), 3 * 4);
    glScalef(1.0f, 1.0f, -1.0f);

    //鱼尾巴的另一部分
    glFrontFace(GL_CCW);
    vertex[143] = vertex[152] = vertex[149] = vertex[158] = vertex[167] = -pt;
    glDrawArrays(GL_TRIANGLES, 6 + (4 * 6) + (3 * 5), 3 * 4);

    // disable all vertex arrays and texturing
    glDisableClientState(GL_VERTEX_ARRAY);
    glDisableClientState(GL_TEXTURE_COORD_ARRAY);
    glDisableClientState(GL_COLOR_ARRAY);
    glDisableClientState(GL_NORMAL_ARRAY);
    glDisable(GL_TEXTURE_2D);
}

/*
*   descrip:
*       fish is divided into 3 part:
*           part1:
*               2 triangles: 1,2
*               6 quads: 3,4,5,6,7,8
*           part2:
*               5 triangles:a,b,c,d,e
*           part3:
*               4 triangles:H,I,J,K
*/
```

心得体会

对海星和鱼的类构造麻烦的部分仍旧是具体顶点的设置，需要对顶点位置进行设置来达到良好的近似效果。作业2中的机器人是比较简单的几何构造，不过已经有了基本的要素，包括颜色设置，整体运动，躯体相对整体的运动等等。作业2实际上对现在的实验起到了良好的承上启下的作用。通过这个实验，我实际上学到了更多的其它知识，比如光影效果相关的phong 模型，贴图方面的内容。收获良多。