计算机图形学作业 8 实验报告

16340203 谭江华

基本要求：

1. 用户能通过左键点击添加 Bezier 曲线的控制点，右键点击则对当前添加的最后一个控制点进行消除
2. 工具根据鼠标绘制的控制点实时更新 Bezier 曲线

Bonus：

1. 可以动态地呈现 Bezier 曲线的生成过程

实现思路&算法：

1. 首先使用回调函数捕捉 mouse 的移动和点击。

```
glfwSetCursorPosCallback(window, cursor_pos_callback);
glfwSetMouseButtonCallback(window, mouse_button_callback);
```

其中 cursor_pos_callback 函数是 mouse 移动的回调函数，而 mouse_button_callback 是 mouse 点击的回调函数

在 cursor_pos_callback 函数中，记录下每个时刻鼠标所在的位置。这里需要注意的是捕捉到的鼠标位置要做一些转换变成对应的窗口坐标。

```
//光标位置回调函数
void cursor_pos_callback(GLFWwindow* window, double x, double y)
{
    //xpos = float((x - window_size_x / 2) / window_size_x) * 2;
    //ypos = float(0 - (y - window_size_y / 2) / window_size_y) * 2;
    xpos = float((x - (window_size_x/2)) / (window_size_x/2)) * 1;
    ypos = float(((window_size_y/2) - y) / (window_size_y/2)) * 1;
    return;
}
```

在 mouse_button_callback 函数中，如果点击左键，则将此时鼠标的位置保存到一个结构体数组中，如果点击右键，则撤销上一次的坐标点。

```
//鼠标点击回调函数
void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
    if (action == GLFW_PRESS)
    {
        //点击鼠标左键
        if (button == GLFW_MOUSE_BUTTON_LEFT)
        {
            point_num++;
            points.index = point_num;
            //cout << "xPos: " << xpos << "  yPos: " << ypos << endl;
            points.x[points.index - 1] = xpos;
            points.y[points.index - 1] = ypos;
        }
        //点击鼠标右键
        else if (button == GLFW_MOUSE_BUTTON_RIGHT)
        {
            if (point_num > 0)
            {
                point_num--;
                points.index = point_num;
            }
        }
    }
    return;
}
```

这里的撤销操作是通过在结构体中储存一个 index 来完成的，当点击了右键，index 就相应地减少，表示坐标点的个数少了一个；同样当点击了左键，增加坐标点之后，index 相应地增加。

```
struct points_array
{
    float *x;
    float *y;
    int index = 0; // 当前控制点的个数
};
```

2. 首先根据控制点画出直线，使得生成 bezier 曲线的过程更加直接。这里每两个控制点形成一条直线。

```
float *line_vertices = new float[6 * 2 * (point_num - 1)];
//cout << point_num << endl;
for (int i = 0, k = 0; i < point_num - 1; )
{
    line_vertices[k * 6 + 0] = points.x[i];
    line_vertices[k * 6 + 1] = points.y[i];
    line_vertices[k * 6 + 2] = 0.0f;
    line_vertices[k * 6 + 3] = 0.0f;
    line_vertices[k * 6 + 4] = 0.0f;
    line_vertices[k * 6 + 5] = 1.0f;
    i++;
    k++;
    line_vertices[k * 6 + 0] = points.x[i];
    line_vertices[k * 6 + 1] = points.y[i];
    line_vertices[k * 6 + 2] = 0.0f;
    line_vertices[k * 6 + 3] = 0.0f;
    line_vertices[k * 6 + 4] = 0.0f;
    line_vertices[k * 6 + 5] = 1.0f;
    k++;
    //cout << i << " " << k << endl;
}
```

得到数组之后，即可画出直线。

3. 然后根据控制点通过 bezier 算法得到曲线，具体实现见 bezier()函数。根据 bezier 曲线的一般参数公式：

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} Pi(1-t)^{n-i} t^{i}$$

这里另外使用了两个函数 C()和 N()来求排列组合以及浮点数的 n 次方

```
//求n的阶乘
int JieCheng(int n)
{
    if (n == 1 || n == 0)
    {
        return 1;
    }
    else
    {
        return n * JieCheng(n - 1);
    }
}

//求组合排列
float C(int n, int i)
{
    return ((float)JieCheng(n)) / ((float)(JieCheng(i)*JieCheng(n - i)));
}

//求浮点数的n次方
float N(float u, int n)
{
    float sum = 1.0;
    if (n == 0)
    {
        return 1;
    }
    for (int i = 0; i < n; i++)
    {
        sum *= u;
    }
    return sum;
}
```

然后根据公式求得各处的点：

```
points_array bezier()
{
    points_array p;
    p.x = new float[bezier_point_num];
    p.y = new float[bezier_point_num];
    int num = 0;
    for (float i = 0; i < 1; i = i + 0.001)
    {
        float x = 0, y = 0;
        for (int j = 0; j < point_num; j++)
        {
            x += C(point_num - 1, j) * N(i, j) * N(1 - i, point_num - 1 - j) * points.x[j];
            y += C(point_num - 1, j) * N(i, j) * N(1 - i, point_num - 1 - j) * points.y[j];
        }
        p.x[num] = x;
        p.y[num] = y;
        num++;
    }
    return p;
}
```

4. 最后根据得到的点进行渲染，得到曲线。

```
// 最少要3个控制点
if (point_num >= 3)
{
    float *vertices = new float[6 * bezier_point_num];
    points_array bezier_points = bezier();
    for (int i = 0; i < bezier_point_num; i++)
    {
        vertices[i * 6 + 0] = bezier_points.x[i];
        vertices[i * 6 + 1] = bezier_points.y[i];
        vertices[i * 6 + 2] = 0.0f;
        vertices[i * 6 + 3] = 0.0f;
        vertices[i * 6 + 4] = 0.0f;
        vertices[i * 6 + 5] = 1.0f;
    }
```

```
// 将所有点都画出来
glBindBuffer(GL_ARRAY_BUFFER, 0);
glBindVertexArray(VAO);
glDrawArrays(GL_POINTS, 0, bezier_point_num);
```

最终效果如图：