韩宇潇 16340069

整体代码结构：

```
class HW8 {
public:
    HW8();
    ~HW8();
    void HW8LinkShader();
    unsigned int getShaderProgram();
    void drawPoint(vector<pair<float, float> >, float);
    void drawLine(vector<pair<float, float> >);
    void Bezier(float x, float y, int, bool);
    float B(int, int, float);

private:
    Shader shader;
    vector<pair<float, float>> coordinate;
    float ratio = 0.0;
};
```

HW8LinkShader():

用来构建着色器，将在本次作业中用到的顶点着色器和片段着色器连接，在接下来的调用过程中方便使用。

getShaderProgram():

返回构建的着色器到 main 函数。

drawPoint():

画点，传入顶点和点的大小。

drawLine():

画线，传入画线所需顶点。

Bezier():

实现 Bezier Curve 的绘制，传入顶点坐标，添加顶点还是删除顶点，是否动态生成。

B():

进行 Bezier Curve 的绘制过程中的计算。

Basic:

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。

实验思路：

（1）实现鼠标的点击函数和移动函数，当用户点击鼠标时，记录此时位置和点击的状态（左键点击和右键点击）。

```
void mouse_callback(GLFWwindow* window, double xpos, double ypos) {
    if (firstMouse) {
        lastX = xpos;
        lastY = ypos;
        firstMouse = false;
    }

    float xoffset = xpos - lastX;
    float yoffset = lastY - ypos; // reversed since y-coordinates go from bottom to top

    lastX = xpos;
    lastY = ypos;

    camera.ProcessMouseMovement(xoffset, yoffset);
}

void mouse_button_callback(GLFWwindow* window, int button, int action, int mods) {
    if (action == GLFW_PRESS)
        switch (button) {
        case GLFW_MOUSE_BUTTON_LEFT:
            state = 1;
            break;
        case GLFW_MOUSE_BUTTON_MIDDLE:
            state = 2;
            break;
        case GLFW_MOUSE_BUTTON_RIGHT:
            state = 3;
            break;
        default:
            return;
        }
    return;
}
```

（2）根据鼠标点击状态判断添加顶点还是删除顶点，画出顶点。

```
if (state == 1) {
    pair<float, float> p(x, y);
    coordinate.push_back(p);
}
else if (state == 3){
    if (!coordinate.empty())
        coordinate.pop_back();
}

drawPoint(coordinate, 10.0f);
```

（3）通过 Bezier Curve 的方法，计算曲线上的点的坐标，画出曲线。

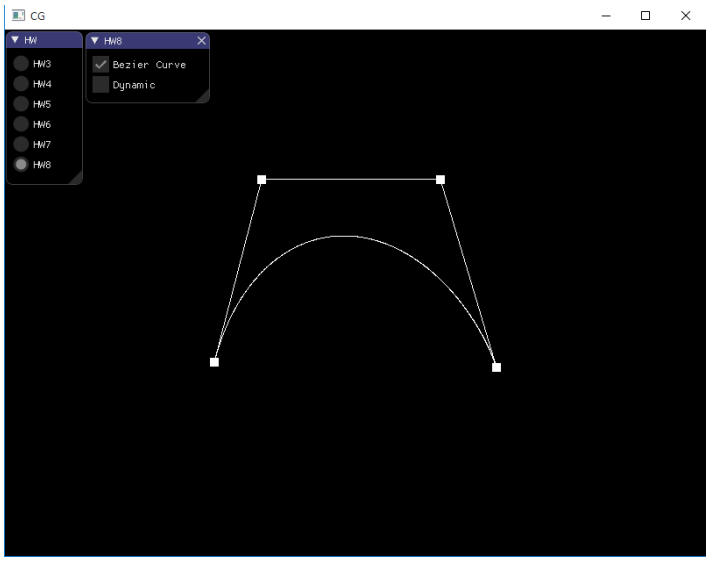
```
vector<pair<float, float>> bezier;
for (int j = 0; j < 1000; j++) {
    float Q_x = 0;
    float Q_y = 0;
    vector<pair<float, float> >::iterator it = coordinate.begin(), end = coordinate.end();
    for (int i = 0; it != end; ++it, i++) {
        Q_x += it->first * B(i, coordinate.size()-1, j / 1000.0);
        Q_y += it->second * B(i, coordinate.size()-1, j / 1000.0);
    }

    pair<float, float> p(Q_x, Q_y);
    bezier.push_back(p);
}

drawPoint(bezier, 1.0f);
```

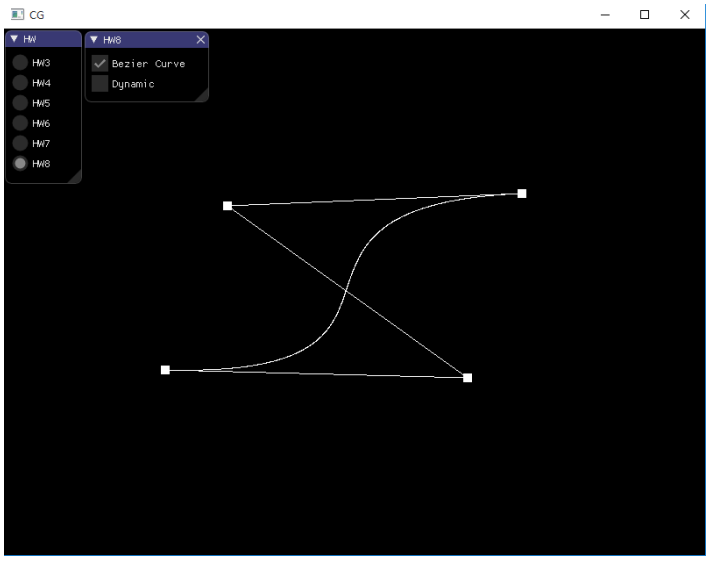
```
float HW8::B(int i, int n, float t) {
    float res = 1;
    int max_ = i > n - i ? i : n - i;
    int min_ = i < n - i ? i : n - i;

    for (int j = n; j > max_; j--)
        res *= j;
    for (int j = min_; j > 1; j--) {
        res /= j;
    }
    res *= pow(t, i);
    res *= pow(1 - t, n - i);
    return res;
}
```

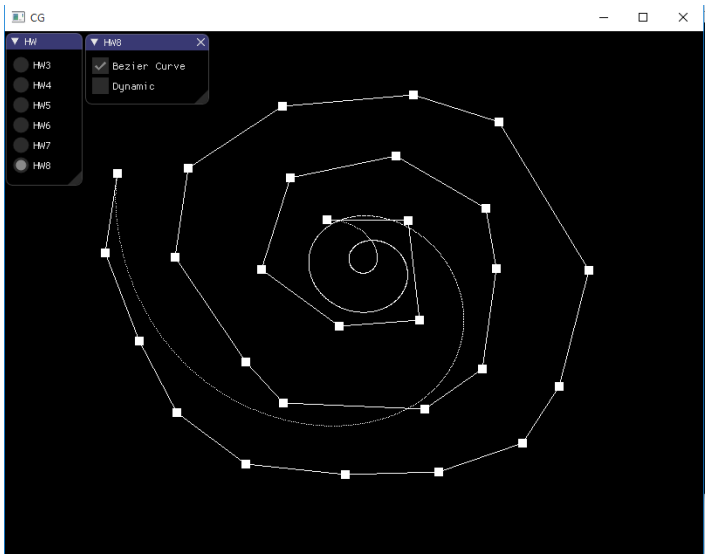
（4）画出顶点间的连线

```
drawLine(coordinate);
```

实验结果：







Bonus:

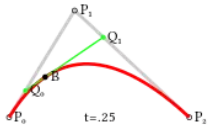
1. 可以动态地呈现 Bezier 曲线的生成过程。

实验思路：

（1）按照 Bezier 曲线的构成方法动态生成 Bezier 曲线。例如二次曲线见下图所示，高阶曲线原理相同。

- **二次曲线** 由三个点构成。建构二次贝塞尔曲线，可以使中介点Q0和Q1作为由0至1的 t：

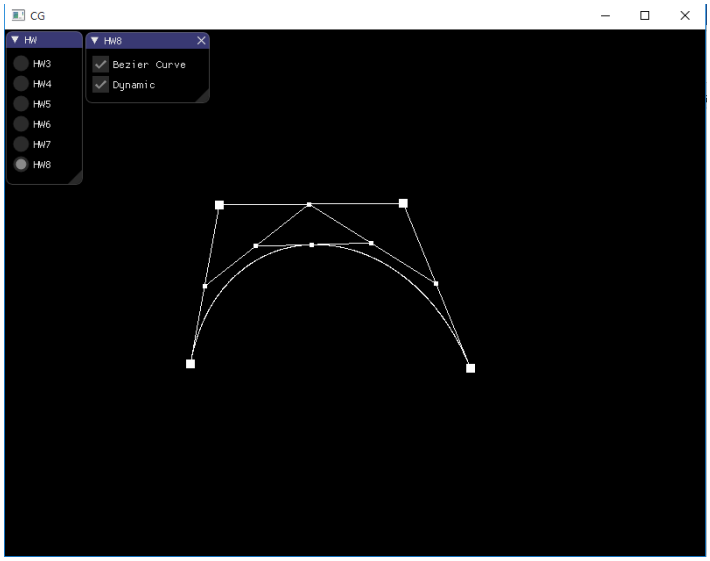
  1. 由P0至P1的连续点Q0，描述一条线性贝塞尔曲线。
  2. 由P1至P2的连续点Q1，描述一条线性贝塞尔曲线。
  3. 由Q0至Q1的连续点B（t），描述一条二次贝塞尔曲线。

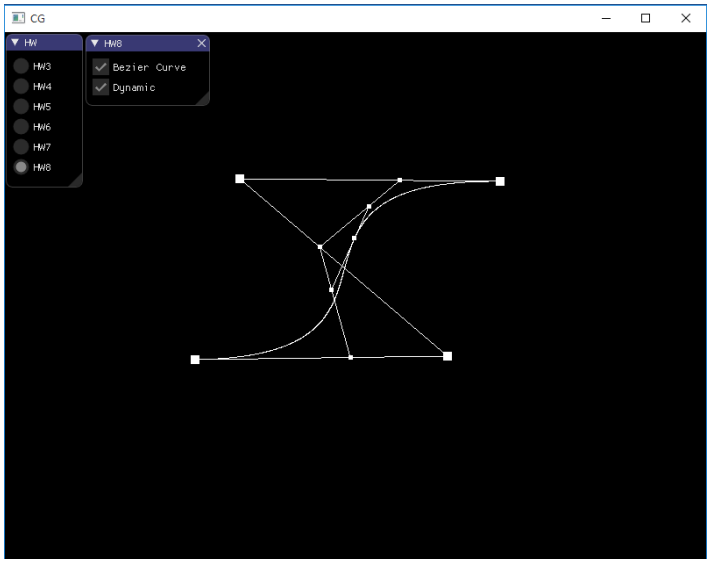


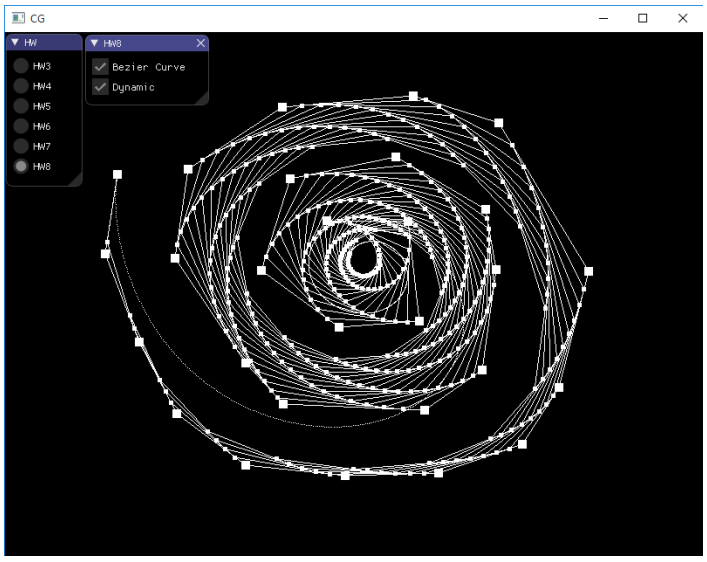
二次贝塞尔曲线的结构

```
if (dynamic) {
    if (state == 1 || state == 3)
        ratio = 0.0;

    vector<pair<float, float> > point_ = coordinate;
    int n = point_.size() - 1;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n-i; j++) {
            float Q_x = ratio*(point_[j + 1].first - point_[j].first) + point_[j].first;
            float Q_y = ratio * (point_[j + 1].second - point_[j].second) + point_[j].second;
            pair<float, float> p(Q_x, Q_y);
            point_[j] = p;
        }
        drawLine(point_);
        drawPoint(point_, 5.0f);
    }

    ratio += 1.0f / 1000;
    if (ratio >= 1.0)
        ratio = 0.0;
}
```

实验结果：


CG
HW
HW3
HW4
HW5
HW6
HW7
HW8
HW8
Bezier Curve
Dynamic


CG
HW
HW3
HW4
HW5
HW6
HW7
HW8
HW8
Bezier Curve
Dynamic


CG
HW
HW3
HW4
HW5
HW6
HW7
HW8
HW8
Bezier Curve
Dynamic