Basic

1. 通过左键点击添加点，右键消除点

- 在GLFW中, 通过glfwSetMouseButtonCallback注册鼠标点击事件, 并通过glfwGetCursorPos获取鼠标位置

```
void mouse_button_callback(GLFWwindow* window, int button, int action, int mods) {
    if (action == GLFW_PRESS) {
        switch (button) {
            case GLFW_MOUSE_BUTTON_LEFT:
                //  获取鼠标位置
                glfwGetCursorPos(window, &mouseX, &mouseY);
                //  放入顶点数组
                points.push_back(Point(mouseX, mouseY));
                break;
            case GLFW_MOUSE_BUTTON_RIGHT:
                //  弹出顶点
                points.pop_back();
                break;
            default:
                return;
        }
    }
    return;
}


//  注册鼠标点击事件
glfwSetMouseButtonCallback(window, mouse_button_callback);
```

- 当鼠标左键点击时, 使用vector保存生成的顶点; 当鼠标右键点击时, 通过pop将顶点从vector弹出

```
//  顶点
struct Point{
    float x;
    float y;
    Point(float x_ = 0, float y_ = 0) {
        x = x_;
        y = y_;
    }
};

vector<Point> points;

void mouse_button_callback(GLFWwindow* window, int button, int action, int mods) {
```

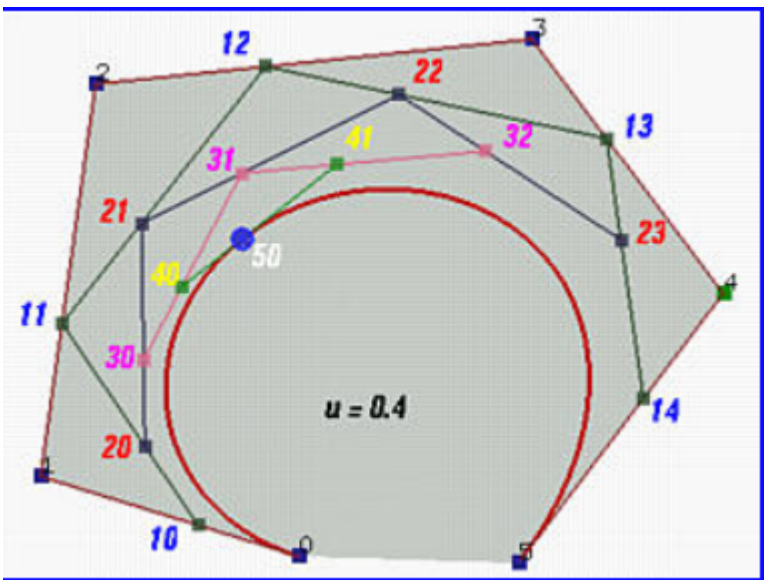
```
    if (action == GLFW_PRESS) {
        switch (button) {
            case GLFW_MOUSE_BUTTON_LEFT:
                //  获取鼠标位置
                glfwGetCursorPos(window, &mouseX, &mouseY);
                //  放入顶点数组
                points.push_back(Point(mouseX, mouseY));
                break;
            case GLFW_MOUSE_BUTTON_RIGHT:
                //  弹出顶点
                points.pop_back();
                break;
            default:
                return;
        }
    }
    return;
}
```

2. 使用de Casteljau算法计算Bezier曲线

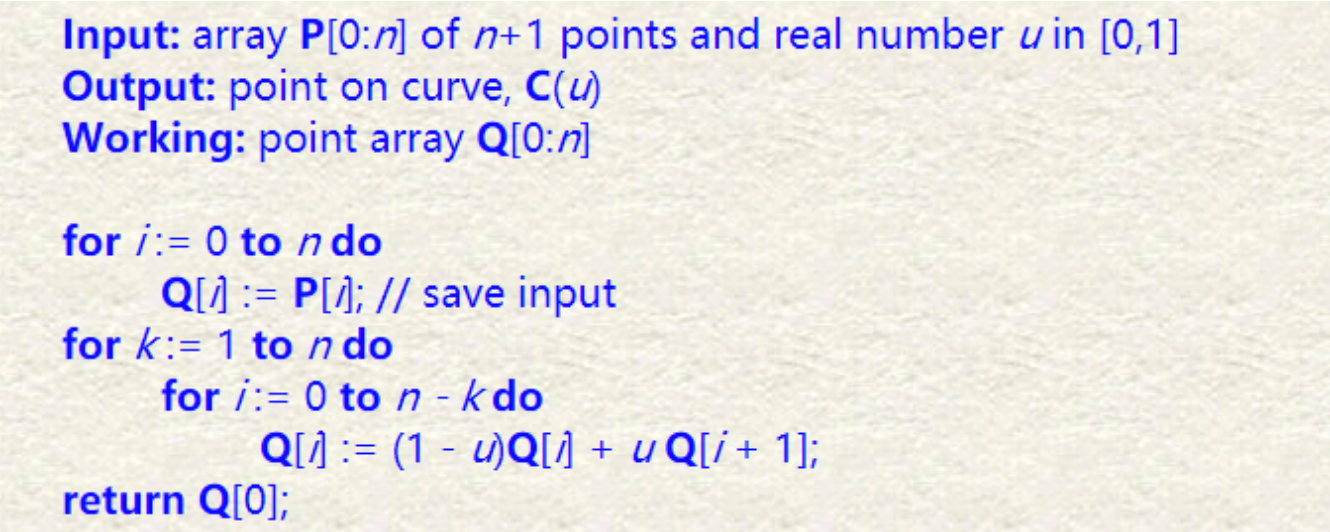
参考链接:

http://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/Bezier/de-casteljau.html

de Casteljau算法思想:



de Casteljau算法伪代码:



程序实现

```
void Bezier() {
    Point* Q = new Point[points.size()];
    for (double t = 0.0; t <= 1; t += 0.005) {
        vector<vector<Point>> temp1;
        for (int i = 1; i < points.size(); ++i) {
            vector<Point> temp2;
            for (int j = 0; j < points.size() - i; ++j) {
                if (i == 1) {   // i==1时,第一次迭代,由已知控制点计算
                    Q[j].x = points[j].x * (1 - t) + points[j + 1].x * t;
                    Q[j].y = points[j].y * (1 - t) + points[j + 1].y * t;
                }
                else {
                    // i != 1时,通过上一次迭代的结果计算
                    Q[j].x = Q[j].x * (1 - t) + Q[j + 1].x * t;
                    Q[j].y = Q[j].y * (1 - t) + Q[j + 1].y * t;
                }
            }
        }
        bezierPoints.push_back(Point(Q[0].x, Q[0].y));
    }
    delete Q;
}
```

Bonus

- 在Bezier函数中, 计算一次要画的动态点个数, 并保存要画的动态点

```
void Bezier() {
    //  1. 统计一次要画的动态点的个数
    dynamicCount = 0;
    for (int i = 1; i < points.size(); ++i) {
        for (int j = 0; j < points.size() - i; ++j) {
            dynamicCount++;
        }
    }

    Point* Q = new Point[points.size()];
    for (double t = 0.0; t <= 1; t += 0.005) {
        vector<vector<Point>> temp1;
        for (int i = 1; i < points.size(); ++i) {
            vector<Point> temp2;
            for (int j = 0; j < points.size() - i; ++j) {
                if (i == 1) {
                    Q[j].x = points[j].x * (1 - t) + points[j + 1].x * t;
                    Q[j].y = points[j].y * (1 - t) + points[j + 1].y * t;
                }
                else {
                    Q[j].x = Q[j].x * (1 - t) + Q[j + 1].x * t;
                    Q[j].y = Q[j].y * (1 - t) + Q[j + 1].y * t;
                }
                //  2. 保存动态点
                dynamicPoints.push_back(Point(Q[j].x, Q[j].y));
            }
        }
        bezierPoints.push_back(Point(Q[0].x, Q[0].y));
    }
    delete Q;
}
```

- 为防止动态点更新过快, 每50次循环更新一次动态点

```
while (!glfwWindowShouldClose(window)) {
    ...

    t++;
    if (t == 50) {
        i++;
        t = 0;
    }

    if (points.size() + bezierPoints.size() + i * dynamicCount < result.size()) {
        int c = 0;
        for (int j = points.size() - 1; j >= 2; j--) {
            glDrawArrays(GL_LINE_STRIP, points.size() + bezierPoints.size() + i *
dynamicCount + c, j);
            c += j;
        }
    }
```

```
}
```

- 动态绘制节点

假设有5个节点，则第一轮迭代产生4个动态点，第二轮迭代产生3个动态点，第三轮迭代产生2个动态点. 又因为每一轮迭代中，动态点两两生成一条线，故第一轮迭代产生3条线，迭代产生2条线，第三轮迭代产生1条线

推广到n个节点，则第一轮迭代产生n - 1个动态点，...，第m轮迭代产生n - m个动态点

```
if (points.size() + bezierPoints.size() + i * dynamicCount < result.size()) {
    int c = 0;
    for (int j = points.size() - 1; j >= 2; j--) {
        glDrawArrays(GL_LINE_STRIP, points.size() + bezierPoints.size() + i * dynamicCount
+ c, j);
        c += j;
    }
}
```