Homework 8 - Bezier Curve

16340082 黄俊凯

添加删除控制点

要完成这个目标，我们首先要查询一下 OpenGL 相关的API，如鼠标点击事件的捕获，但仅仅点击事件是不够的，我们还需要鼠标所指位置，所以我们要给以下两个事件添加 Callback：

```
// 鼠标点击事件
glfwSetMouseButtonCallback(window, mouse_button_callback);
// 光标位置变换事件
glfwSetCursorPosCallback(window, cursor_position_callback);
```

相应的两个 Callback：

```
// Points Vertices
std::vector<Point> points;
// Current Cursor Position
float xpos, ypos;

void mouse_button_callback(GLFWwindow* window, int button, int action, int mods) {
    if (action == GLFW_PRESS) {
        switch (button) {
        case GLFW_MOUSE_BUTTON_LEFT:
            points.push_back(Point(xpos, ypos));
            break;
        case GLFW_MOUSE_BUTTON_RIGHT:
            points.pop_back();
            break;
        default:
            break;
        }
    }
}

void cursor_position_callback(GLFWwindow* window, double x, double y) {
    xpos = float( (x - WIDTH  / 2) / WIDTH)  * 2;
    ypos = float(-(y - HEIGHT / 2) / HEIGHT) * 2;
}
```

其中，`mouse_button_callback` 会判断是否是点击，然后根据点击按钮的类型，进行不同的操作，左键则给全局变量 `points` 添加一个新的控制点，右键则删除最新添加的点； `cursor_position_callback` 修改全局变量 `xpos` `ypos` ，需要注意的一点是：这里的坐标系跟OpenGL里的不太一样，它以左上角为起点，y向下递增，x向右递增，为了方便，我们可以进行上述操作，将坐标变换到 [-1,1]。

绘制 Bezier 曲线

如果是立即绘制的话是很简单的，首先是先写一个 Bernstein 基函数：

```
float Bernstein(float t, int i, int n) {
    float nFactorial = 1.0,
          iFactorial = 1.0,
          niFactorial = 1.0;
    for (int k = n; k > 1; k--) {
        if (k <= n - i) niFactorial *= k;
        if (k <= i) iFactorial *= k;
        nFactorial *= k;
    }
    return nFactorial * pow(t, i) * pow(1 - t, n - i) / (iFactorial * niFactorial);
}
```

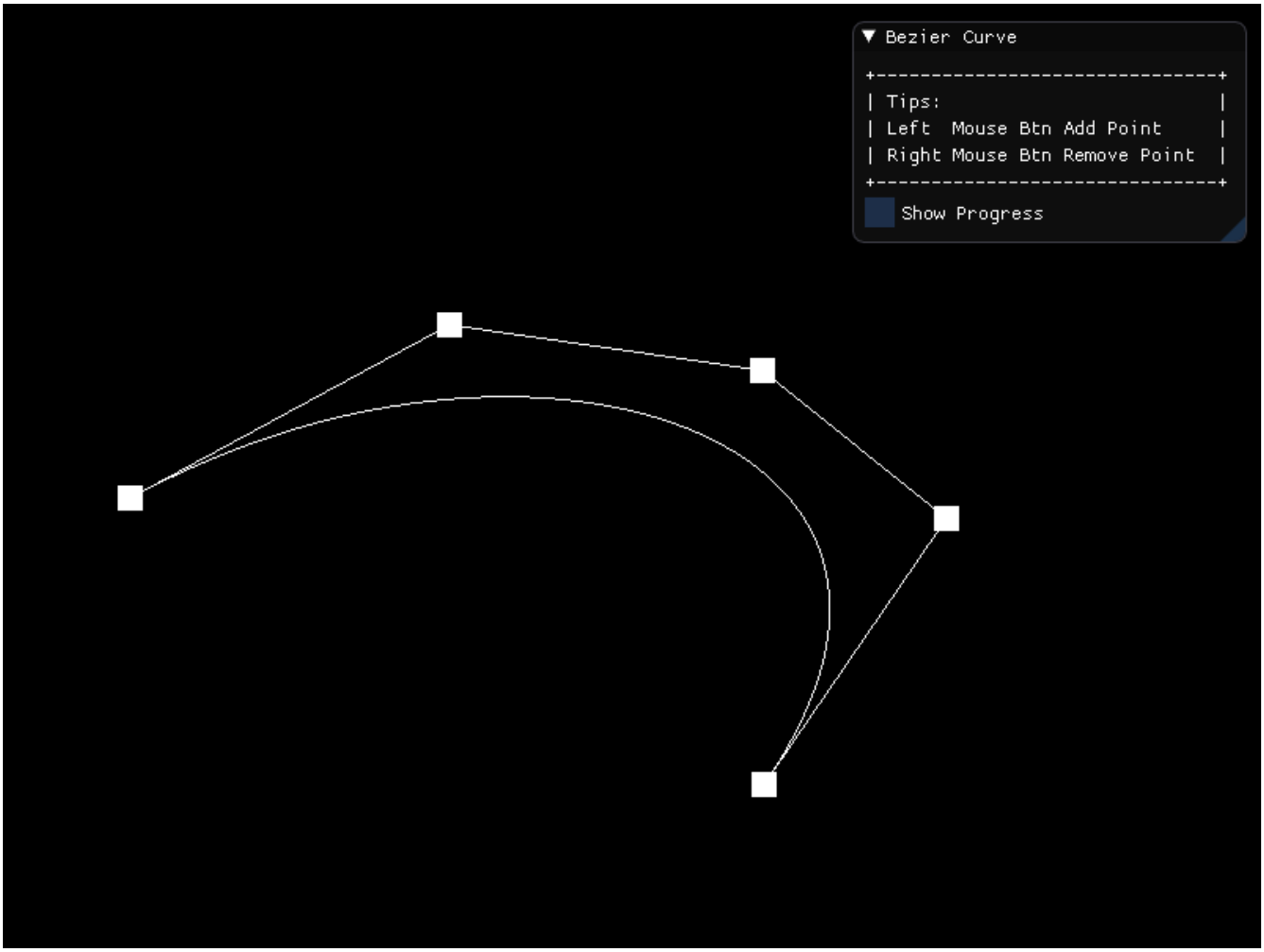
我们将要绘制的曲线离散化为 `NUM_POINT_TO_PAINT` 条直线，这里我取100，然后根据公式 $Q(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$ 计算出每一时刻 t 对应点的坐标即可，这里 P_i 是第 i 个点的坐标，B 是 Bernstein 基函数：

```
for (int j = 0; j <= NUM_POINT_TO_PAINT; ++j) {
    float t = (float)j / NUM_POINT_TO_PAINT;
    float currentX = 0.0f, currentY = 0.0f;
    for (int i = 0; i < points.size(); ++i) {
        float b = Bernstein(t, i, points.size() - 1);
        currentX += points[i].x * b;
        currentY += points[i].y * b;
    }

    curveVertices[6 * j + 0] = currentX;
    curveVertices[6 * j + 1] = currentY;
    curveVertices[6 * j + 2] = 0.0f;
    // color: (255, 255, 255) -> white
    curveVertices[6 * j + 3] = 1.0f;
    curveVertices[6 * j + 4] = 1.0f;
    curveVertices[6 * j + 5] = 1.0f;
}

// for curve
glBindVertexArray(VAOs[1]);
glBindBuffer(GL_ARRAY_BUFFER, VBOs[1]);
glBufferData(GL_ARRAY_BUFFER, (NUM_POINT_TO_PAINT + 1) * 6 * sizeof(float), curveVertices,
GL_STATIC_DRAW);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)0);
glEnableVertexAttribArray(0);
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)(3 *
sizeof(float)));
glEnableVertexAttribArray(1);
glDrawArrays(GL_LINE_STRIP, 0, NUM_POINT_TO_PAINT + 1);
```

效果



动态绘制

这个实现起来也不难，首先我们可以使用两个 vector 来存放过程中产生的点，如一开始我们有 `n-1` 个控制点，那么我们根据 t 时刻和上述控制点，可以计算出 `n-2` 个临时控制点，依次类推，代码如下：

```
std::vector<Point> tmpPoints1(points);

while (tmpPoints1.size() > 1) {
    int pointsCounter = 0;
    std::vector<Point> tmpPoints2;
    for (int i = 0; i < tmpPoints1.size() - 1; ++i) {
        float
            b0 = Bernstein(T, 0, 1),
            b1 = Bernstein(T, 1, 1);
        float
            tmpX = b0 * tmpPoints1[i].x + b1 * tmpPoints1[i + 1].x,
            tmpY = b0 * tmpPoints1[i].y + b1 * tmpPoints1[i + 1].y;
        tmpPoints2.push_back(Point(tmpX, tmpY));
        pointVertices[6 * pointsCounter + 0] = tmpX;
        pointVertices[6 * pointsCounter + 1] = tmpY;
        pointVertices[6 * pointsCounter + 2] = 0.0f;
        // color: (0, 255, 0) -> green
        pointVertices[6 * pointsCounter + 3] = 0.0f;
        pointVertices[6 * pointsCounter + 4] = 1.0f;
```

```
        pointVertices[6 * pointsCounter + 5] = 0.0f;
        pointsCounter += 1;
    }

    // for point
    glBindVertexArray(VAOs[0]);
    glBindBuffer(GL_ARRAY_BUFFER, VBOs[0]);
    glBufferData(GL_ARRAY_BUFFER, pointsCounter * 6 * sizeof(float), pointVertices,
GL_STATIC_DRAW);
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)0);
    glEnableVertexAttribArray(0);
    glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)(3 *
sizeof(float)));
    glEnableVertexAttribArray(1);
    glPointSize(4);
    glDrawArrays(GL_POINTS, 0, pointsCounter);
    glDrawArrays(GL_LINE_STRIP, 0, pointsCounter);

    tmpPoints1.assign(tmpPoints2.begin(), tmpPoints2.end());
}
```

效果