计算机图形学第八次作业

作业要求

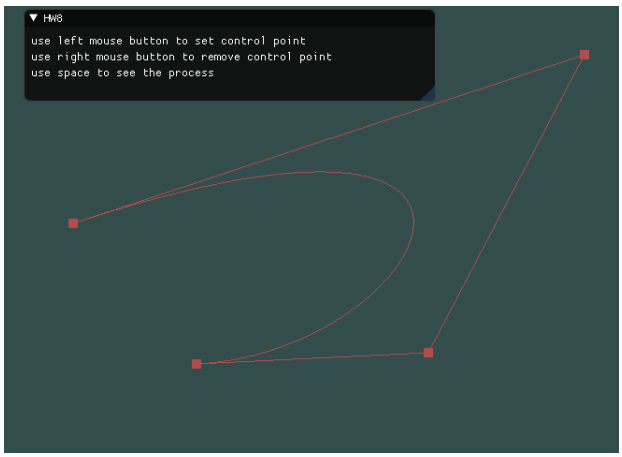
Basic:

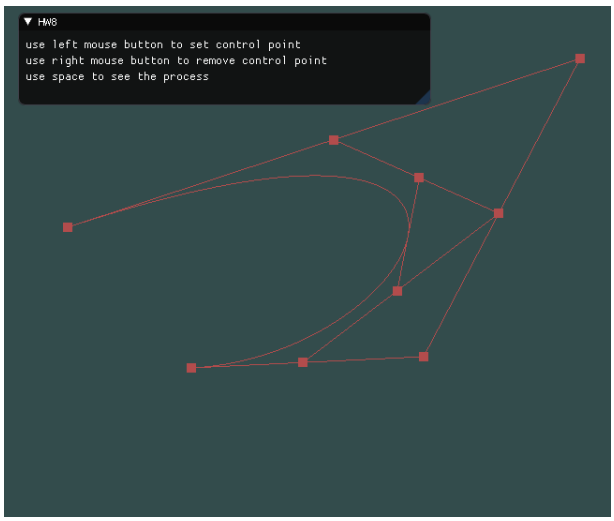
1. 用户能通过左键点击添加Bezier曲线的控制点，右键点击则对当前添加的最后一个控制点进行消除
2. 工具根据鼠标绘制的控制点实时更新Bezier曲线。 Hint: 大家可查询捕捉mouse移动和点击的函数方法

Bonus：

1. 动态呈现曲线生成过程

实验结果





Basic

首先是对bezier曲线的生成，包括两部分，一部分是对伯恩斯坦基函数的生成，一部分是求和函数：

伯恩斯坦基函数数的公式为：$\frac{n!}{k!(n-k)!}t^{n-k}(1-t)^n$

具体函数如下：

```
int JieCheng(int n) {
    if (n == 1 || n==0) return 1;
    return n * JieCheng(n - 1);
}

//求组合数
int zuhe(int n, int k) {
    return JieCheng(n) / (JieCheng(k)*(JieCheng(n - k)));
}

//求Q(t)
glm::vec2 Q(double t) {
    int n = p.size() - 1;
```

```
    double x = 0, y = 0;
    double p1 = pow((1 - t), n), p2 = 1;
    for (int i = 0; i <= n; i++) {
        x += zuhe(n, i)*p1*p2*p[i].x;
        y += zuhe(n, i)*p1*p2*p[i].y;
        p1 /= (1 - t);
        p2 *= t;
    }
    return glm::vec2(x, y);
}
```

然后是使用一个vector来控制点的添加和删除，当鼠标左键点击的时候添加，右键点击的时候删除。在左键点击时将点击的点push到vector中，右键点击时将数组中最后一个点pop。

```
void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
    if (run)
    {
        return;
    }
    if (button == GLFW_MOUSE_BUTTON_LEFT && action == GLFW_PRESS) {
        double xpos, ypos;
        glfwGetCursorPos(window, &xpos, &ypos);
        p.push_back(glm::vec2((float(xpos) / float(SCR_WIDTH) * 2.0f) - 1, -((float(ypos) /
float(SCR_HEIGHT) * 2.0f) - 1)));
    }

    if (button == GLFW_MOUSE_BUTTON_RIGHT && action == GLFW_PRESS) {

        p.pop_back();
    }
}
```

这里显示点的位置的时候要求点击点与显示框的相对位置。

然后是每次添加一个点的时候显示到该点的bezier曲线。由于我们通过控制点求出了每个t对应的bezier曲线上的点，那么我们就可以根据上一个bezier点和下一个bezier点画出两点之间的直线，然后多条直线构成曲线，因此t的每次增加的值需要很小才能显示曲线的效果。

```
vector<float> bezierPoints;
int count = 0;
for (double i = 0.0f; i < 1.0f; i+=0.0005)
{
    glm::vec2 Bpoint(Q(i));
    vector<float> point{
        Bpoint.x, Bpoint.y, 0.0f
    };
    bezierPoints.insert(bezierPoints.end(), point.begin(), point.end());
    if (count > 0)
    {
        float *ptr = bezierPoints.data() + (count - 1) * 3;
        glBufferData(GL_ARRAY_BUFFER, 6 * sizeof(float), ptr, GL_STATIC_DRAW);
```

```
        glDrawArrays(GL_LINES, 0, 2);
    }
    count++;
}
```

每次渲染只画出一条直线，循环之后就会得到一条曲线。

Bonus

在点击键盘的空格键的时候运行曲线的生成过程那部分程序，首先是将控制点数组存放到一个临时数组中，然后使用一个running_t参数得到控制点间连线上的点，当我们有n个控制点的时候，那么就在n-1条控制点连线上得到n-1个点，更新临时数组，连接这n-1个点之间的连线，然后再根据这n-1条连线得到连线上n-2个点，一直到最终更新的临时数组中只有1个点为止，这样就会画出包括控制点连线在内一共$(n-1)+(n-2)+...+1$条直线。其中最后一条直线与曲线相切。

然后在每次画完这些直线之后更新running_t这个参数得到控制点连线中的下一个点达到一个动画的效果。当running_t的值达到1的时候就是结束动画的时候。

```
vector<glm::vec2> runningPoints(p);
while (runningPoints.size() > 1)
{
    vector<glm::vec2> nextLevel;
    for (int i = 0; i < runningPoints.size()-1; i++)
    {
        float presPoint[] = {
            runningPoints[i].x, runningPoints[i].y, 0.0f,
            runningPoints[i + 1].x, runningPoints[i + 1].y, 0.0f
        };
        glBufferData(GL_ARRAY_BUFFER, 6 * sizeof(float), presPoint, GL_STATIC_DRAW);
        glDrawArrays(GL_LINES, 0, 2);
        glDrawArrays(GL_POINTS, 0, 2);
        glm::vec2 nextPoint;
        nextPoint.x = (running_t)*runningPoints[i].x + (1-running_t) * runningPoints[i +
1].x;
        nextPoint.y = (running_t)*runningPoints[i].y + (1 - running_t) * runningPoints[i +
1].y;
        nextLevel.push_back(nextPoint);
    }
    runningPoints = nextLevel;
    cout << runningPoints.size() << endl;
}
running_t += 0.001f;
if (running_t >= 1.0f)
{
    run = false;
}
```