作业9

Basic:

- 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
- 工具根据鼠标绘制的控制点实时更新Bezier曲线。

鼠标点击和实时更新Bezier曲线

因为要实现鼠标点击的事件,在这里写一个函数`mouse_button_callback()`并用`glfwSetMouseButtonCallback(window, mouse_button_callback)`添加进入程序。

而`mouse_button_callback()`的函数如下所示，点击左键或者右键可以加入或消除点，之后会重新设置将存入float数组并计算bezier曲线。而这也让我们可以实时更新Bezier曲线。

具体效果见**basic.mp4**

```
void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
        double xpos, ypos;
        if (action == GLFW_PRESS) switch (button)
        {
        case GLFW_MOUSE_BUTTON_LEFT:
                glfwGetCursorPos(window, &xpos, &ypos);
                setPoint(xpos, ypos);
                setVertice(&Points, pointsVertices);
                bezierCurve();
                setVertice(&bezierLine, bezierLineVertices);
                break;
        case GLFW_MOUSE_BUTTON_RIGHT:
                glfwGetCursorPos(window, &xpos, &ypos);
                removePoint();
                setVertice(&Points, pointsVertices);
                bezierCurve();
                setVertice(&bezierLine, bezierLineVertices);
                break;
        default:
                return;
        }
        return;
}
```

Bezier曲线计算

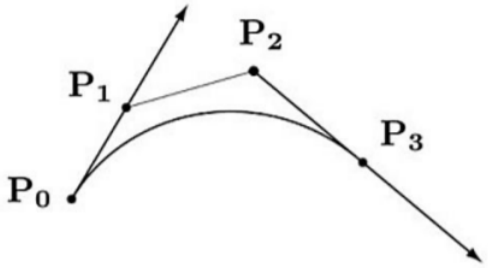
这里按照公式进行计算。理论上来说对于任意点数量的Bezier曲线均可计算，但是在计算数据可能溢出的情况在这里可以支持12个点的Bezier曲线计算。公式如下图。

- Bézier curve本质上是由**调和函数（Harmonic functions）**根据**控制点（Control points）**插值生成。其参数方程如下：

$$Q(t)=\sum_{i=0}^{n}P_iB_{i,n}(t)\,,\quad t\in[0,1]$$

- 上式为n次多项式，具有$n+1$项。其中，$P_i(i=0,1\ldots n)$表示特征多边形的$n+1$个顶点向量；$B_{i,n}(t)$为**伯恩斯坦（Bernstein）基函数**，其多项式表示为：

$$B_{i,n}(t)=\frac{n!}{i!(n-i)!}t^i(1-t)^{n-i}\,,\quad i=0,\ 1\ldots n$$



参照公式所写的算法如下图所示。其中computeNumC是计算组合数即(n!)/(i!(n-1)!)。

```
void bezierCurve() {
        if (Points.size() > 1) {
                float step = 0.001f;
                int size = 1 / step;

                float t = 0.0f;

                bezierLine.clear();
                for (int i = 0; i < size; i++, t += step) {
                        double x = 0, y = 0;
                        for (size_t j = 0; j < Points.size(); j++) {
                                x += Points[j].x * pow(t, j) * pow(1 - t,
Points.size() - 1 - j) * computeNumC(j);
                                y += Points[j].y * pow(t, j) * pow(1 - t,
Points.size() - 1 - j) * computeNumC(j);
                        }
                        Point temp(x, y);
                        bezierLine.push_back(temp);
                }
        }
        else {
                bezierLine.clear();
        }
}

int computeNumC(int num) {
        return computeFactorial(Points.size() - 1) / (computeFactorial(num) *
computeFactorial(Points.size() - 1 - num));
```

```
}

int computeFactorial(int num) {
        long int answer = 1;
        for (int i = 1; i <= num; i++)
                answer *= i;
        return answer;
}
```

Bonus:

可以动态地呈现Bezier曲线的生成过程。

动态呈现原理

动态呈现实际上就是每刷新一次就要更新点的位置和它们的连线。

在这里设计了一个递归的算法。每一次递归计算每条线段中每个点的位置，并按顺序放入vector，再进行下一次递归直到计数的time到0为止。其中每一次递归所用的构成线段的点为上一次递归所产生的点。例如，上一次递归在四条线段上生成了四个点分别为ABCD，则这一次递归所用的线段为AB，BC，CD三条线段，并在这三条线段上生成三个点并push进vector用于下一次递归。

```
void getAnimationPoints(double step, int time, bool isFirst, int alreadyPush) {
        if (time == 0) {
                return;
        }
        if (isFirst) {
                for (int i = 0; i < time; i++) {
                        animationLine.push_back(getPointFromLine(Points[i],
Points[i + 1], step));
                }
                getAnimationPoints(step, time - 1, false, 0);
        }
        else {
                for (int i = alreadyPush; i < time + alreadyPush; i++) {
                        animationLine.push_back(getPointFromLine(animationLine[i],
animationLine[i + 1], step));
                }
                getAnimationPoints(step, time - 1, false, alreadyPush + time + 1);
        }
}
```

观看动态生成过程

在这里设置的是按下空格就可以查看Bezier曲线的动态生成过程。

```
void processInput(GLFWwindow *window) {
        if (glfwGetKey(window, GLFW_KEY_ESCAPE) == GLFW_PRESS)
```

```
                glfwSetWindowShouldClose(window, true);
        if (glfwGetKey(window, GLFW_KEY_SPACE) == GLFW_PRESS)
                beginAnimation = true;

}
```

动态效果

具体效果见bonus.mp4