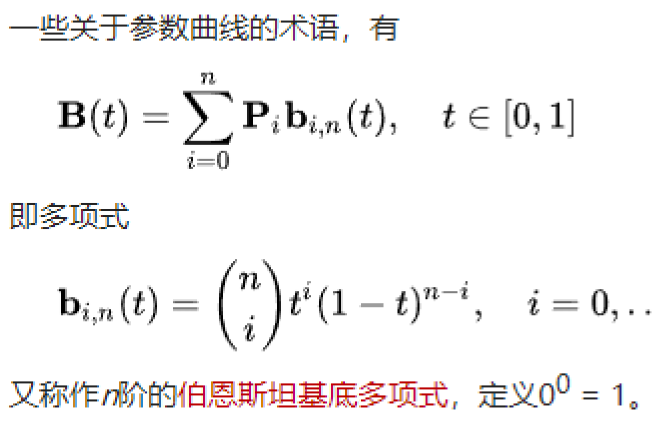
计算机图形学 Homework 9 - Bezier Curve

16340299 赵博然

使用这个公式描绘贝塞尔曲线. 节选自中文维基百科.



一些关于参数曲线的术语，有

$$\mathbf{B}(t) = \sum_{i=0}^{n} \mathbf{P}_i \mathbf{b}_{i,n}(t), \quad t \in [0,1]$$

即多项式

$$\mathbf{b}_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad i = 0, \ldots n$$

又称作n阶的伯恩斯坦基底多项式，定义$0^0 = 1$。

部分算法.

全局变量如下.

```c
// 数组最大长度
#define MAX_SIZE 1024
// 最大点数, 点数过多会溢出, 即使用了unsigned long long数据类型, 懒得搞高精度
#define MAX_POINT 20

const unsigned int window_width = 800;
const unsigned int window_height = 600;

const float point_size = 8.0f;

// 步长
const float step = 0.01f;

float currentX;
float currentY;
float vertices[MAX_SIZE];
float bonus_vertices[MAX_SIZE];
float q_vertices[4];

// 当前点数
int number = 0;

// 时间, 放在while (!glfwWindowShouldClose(window))循环中累加, 用来动态生成曲线, 其实就是曲线函数的自变量
float TIME = 0;
#define MAX_TIME 1.000
```

迭代求组合数.

```
unsigned long long binomial_coefficient(const int &n, const int &k) {
    if (n > k && k >= 0) {
        unsigned long long numerator = 1;
        unsigned long long denominator = 1;
        for (int i = 0; i < n - k; i++) {
            numerator *= n - i;
            denominator *= i + 1;
        }
        return unsigned long long(numerator / denominator);
    }
    else {
        return 1;
    }
}
```

鼠标监听器.

```
void mouse_callback(GLFWwindow* window, int button, int action, int) {
    if (action == GLFW_PRESS) {
        switch (button) {
        case GLFW_MOUSE_BUTTON_LEFT:
            TIME = 0;
            if (number < MAX_POINT) {
                vertices[number * 2] = currentX / (float)(window_width / 2);
                vertices[number * 2 + 1] = currentY / (float)(window_height / 2);
                number++;
            }
            break;
        case GLFW_MOUSE_BUTTON_RIGHT:
            TIME = 0;
            if (number > 0) {
                number--;
            }
            break;
        default:
            break;
        }
    }
}
```

生成贝塞尔曲线.

```
q_vertices[0] = vertices[0];
        q_vertices[1] = vertices[1];
        // 生成曲线
        for (float t = 0.0f; t < 1.0f; t += step) {
            for (int i = 0; i < number; i++) {
                q_vertices[2] += vertices[i * 2] * binomial_coefficient(number - 1,
i) * pow(t, i) * pow((1 - t), number - 1 - i);
                q_vertices[3] += vertices[i * 2 + 1] * binomial_coefficient(number -
1, i) * pow(t, i) * pow((1 - t), number - 1 - i);
            }
            unsigned int qVAO, qVBO;
            glGenBuffers(1, &qVBO);
            glBindBuffer(GL_ARRAY_BUFFER, qVBO);
            glBufferData(GL_ARRAY_BUFFER, sizeof(q_vertices), q_vertices,
GL_STATIC_DRAW);
            glGenVertexArrays(1, &qVAO);
            glBindVertexArray(qVAO);

            glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 2 * sizeof(float),
(void*)0);
            glEnableVertexAttribArray(0);
            glBindBuffer(GL_ARRAY_BUFFER, 0);
            glBindVertexArray(0);
            if (number > 0) {
                glBindVertexArray(qVAO);
                glDrawArrays(GL_LINE_STRIP, 0, 2);
            }

            glDeleteVertexArrays(1, &qVAO);
            glDeleteBuffers(1, &qVBO);

            q_vertices[0] = q_vertices[2];
            q_vertices[1] = q_vertices[3];
            q_vertices[2] = 0.0f;
            q_vertices[3] = 0.0f;
        }
```

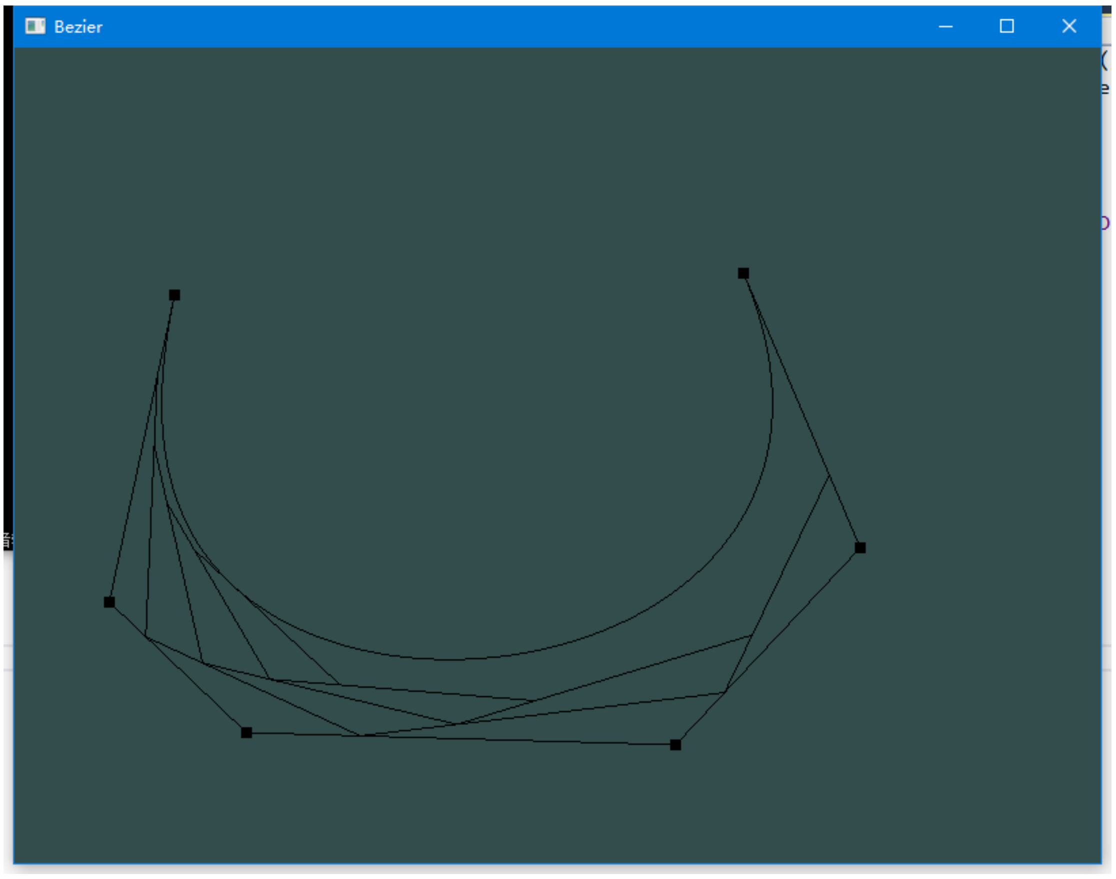
bonus 部分.

```
// bonus部分, 显示曲线的生成过程
        for (int i = 0; i < MAX_SIZE; i++) {
            bonus_vertices[i] = vertices[i];
        }
        for (int i = number; i > 2; i--) {
            for (int j = 0; j < i - 1; j++) {
                bonus_vertices[j * 2] = bonus_vertices[j * 2] * (1 - TIME) +
bonus_vertices[(j + 1) * 2] * TIME;
```

```
                bonus_vertices[j * 2 + 1] = bonus_vertices[j * 2 + 1] * (1 - TIME) +
bonus_vertices[(j + 1) * 2 + 1] * TIME;
            }
            unsigned int bonus_VAO, bonus_VBO;
            glGenBuffers(1, &bonus_VBO);
            glBindBuffer(GL_ARRAY_BUFFER, bonus_VBO);
            glBufferData(GL_ARRAY_BUFFER, sizeof(bonus_vertices), bonus_vertices,
GL_STATIC_DRAW);
            glGenVertexArrays(1, &bonus_VAO);
            glBindVertexArray(bonus_VAO);

            glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 2 * sizeof(float),
(void*)0);
            glEnableVertexAttribArray(0);
            glBindBuffer(GL_ARRAY_BUFFER, 0);
            glBindVertexArray(0);

            glBindVertexArray(bonus_VAO);
            glDrawArrays(GL_LINE_STRIP, 0, i);

            glDeleteVertexArrays(1, &bonus_VAO);
            glDeleteBuffers(1, &bonus_VBO);
        }

        TIME = TIME <= MAX_TIME ? TIME + step / 4.0 : 0;
```

代码详见 main.cpp.

程序截图.



演示视频见 video.mp4.
源程序见 program.exe.