任务拆解

一、 捕获鼠标输入

这部分调用 api 就可以，存储到一个静态变量的数组中就可以

```
void MouseButtonCallback(GLFWwindow* window, int button, int action, int mods)
{
 if (button == GLFW_MOUSE_BUTTON_LEFT && action == GLFW_PRESS)//此时需要增加一个点
 {
  if (points_count >= 0 && points_count < /*kTotalPointsNum*/100)
  {
   mouse_points.get()[points_count * kEachPointsSize + 0] = float(mouse_x - (width / 2)) / (width / 2);
   mouse_points.get()[points_count * kEachPointsSize + 1] = float((height / 2) - mouse_y) / (height / 2);
   mouse_points.get()[points_count * kEachPointsSize + 2] = 0.0;
   points_count++;
  }
 }
 if (button == GLFW_MOUSE_BUTTON_RIGHT && action == GLFW_PRESS) {//此时需要减少一个点
  points_count--;
 }
}
```

二、 存储输入点并传给贝塞尔曲线的生成函数

```
auto bezier_points = [](int i,int pickedCount) {//pickedCount=5
 float t = float(i) / total_points;
 vector<float> b(pickedCount,0);//有i个元素，且初始化为0

 for (int j = 0; j < b.size(); ++j) {

  b[j] = C(pickedCount-1,j)*pow(1 - t, pickedCount-1-j)* pow(t,j);
 }
 float x = 0;
 float y = 0;

 for (int j = 0; j < pickedCount; ++j) {
  x += mouse_points.get()[kEachPointsSize * j] * b[j];
  y += mouse_points.get()[kEachPointsSize * j + 1] * b[j];
 }

 float z = 0.0;
 return std::vector<GLfloat> {x, y, z};
};
```

这里使用了贝塞尔的通项公式直接求解出贝塞尔点，C 是排列组合的计算函数

三、 允许增减控制点，实时更新贝塞尔曲线

主要是修改 pickedCount 的值并且每一帧都重新计算一次贝塞尔曲线，比较简单。

四、 绘制贝塞尔曲线的动画

这一步其实本来可以在二中直接推导出来，但是二中选择了套公式而不是手动计算所以此处要重新计算贝塞尔曲线各个中间线段的端点的位置。我这次使用的是递归的求解方法，向 lambda 函数中传入当前鼠标捕获点的位置，函数捕获一个 vector，经过正递归调用后这个 vector 里的值装的是所有当前参数 t 对应的中间线段的两端。在每次递归中会向 vector 中添加线段的两个端点，然后生成 t：1-t 部分的点作为下一次递归的输入。

```
auto getTPoint=[](glm::vec2 pointA, glm::vec2 pointB, float t) {
    return pointA + t * (pointB - pointA);//得到t等分点
};
```

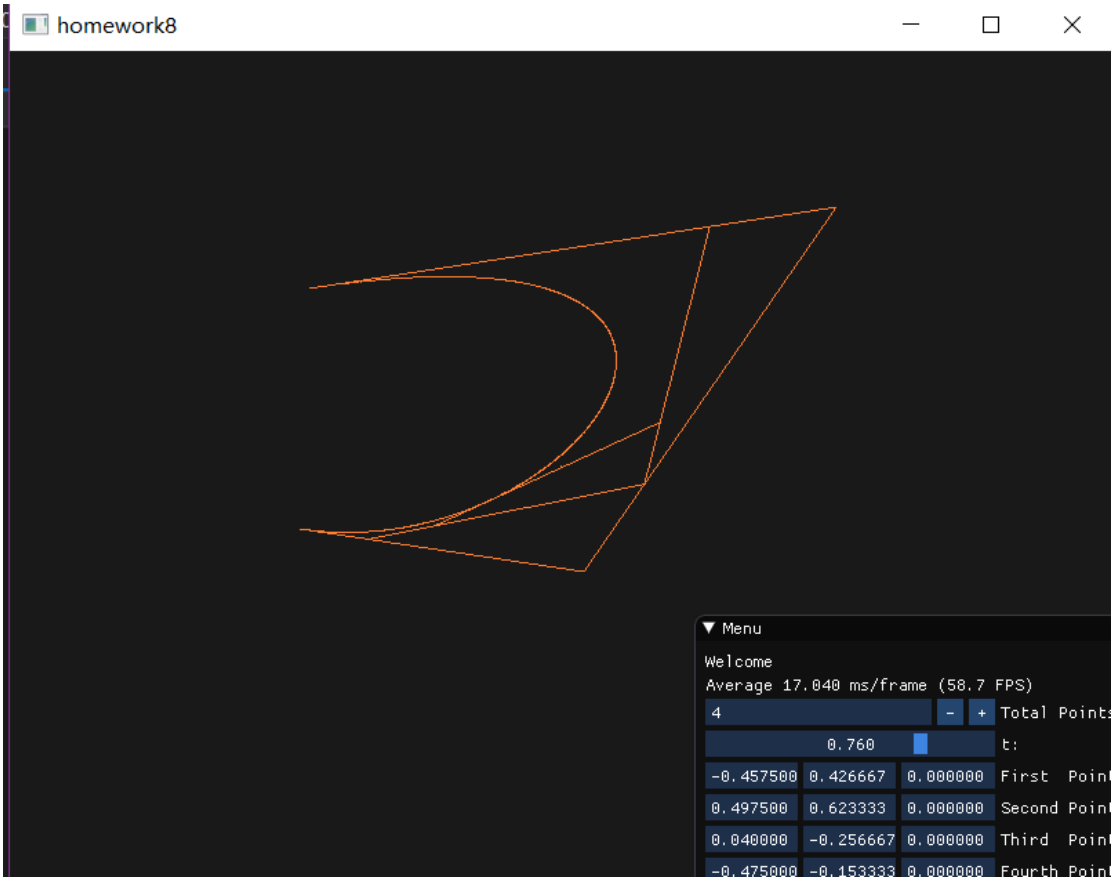
```
std::function<void(vector<glm::vec2> InputPoints, float t)> CalculatePath
    =[&lineResult,getTPoint,&CalculatePath](vector<glm::vec2> InputPoints, float t) {
    //进行画线


    vector<glm::vec2> newPoints;
    for (int i = 0; i < InputPoints.size() - 1; ++i) {//
        glm::vec2 newPoint = getTPoint(InputPoints[i], InputPoints[i + 1], t);
        lineResult.push_back(InputPoints[i].x);
        lineResult.push_back(InputPoints[i].y);
        lineResult.push_back(0);

        lineResult.push_back(InputPoints[i + 1].x);
        lineResult.push_back(InputPoints[i + 1].y);
        lineResult.push_back(0);

        newPoints.push_back(newPoint);//得到了新的新点组合
    }
    if (newPoints.size() >= 2) {
        CalculatePath(newPoints, t);//进行下一轮的计算
    }
    else {
    }
};
```

接下来按照存储结构绘制图像即可。Vertices 中分别是以两个点为单位的线段（随着 t 的动画因子变化而变化），和其后的若干点组成的整个的贝塞尔曲线。



这样一来，漂亮的贝塞尔曲线就绘制完成了