《计算机图形学》作业报告

专业: 计算机科学与技术 姓名: 张松韬 学号: 2021218207

1. 作业要求

本次作业使用上机实验平台和框架，实现多边形的扫描转换。具体任务包括：理解多边形扫描
转换的原理；掌握典型多边形扫描转换算法；掌握步处理、分析实验数据的能力；编程实现基本 X-
扫描线转换算法（必做）；编程实现有效边表转换算法（选做）。

2. 作业内容 (重要步骤描述、关键代码段、结果截图、结果分析、作业心得)

重要步骤描述：

1.首先，找到多边形顶点集中的最小 Y 坐标和最大 Y 坐标，确定了需要处理的扫描线的范围。

2.对每一条扫描线进行处理：

创建一个空的向量 intersections 用于存储与当前扫描线相交的边的 X 坐标。

遍历多边形的每一条边，判断该边是否与当前扫描线相交。

若相交，根据边的两个顶点的坐标以及当前扫描线的 Y 坐标，计算出交点的 X 坐标，并将其加入 intersections 向量中。

对 intersections 向量中的交点进行排序，以便后续绘制线段时按顺序绘制。

遍历排序后的交点，每两个交点之间形成一段需要填充的区域，使用 DrawPixel 函数绘制每个像素点，从而填充该区域。

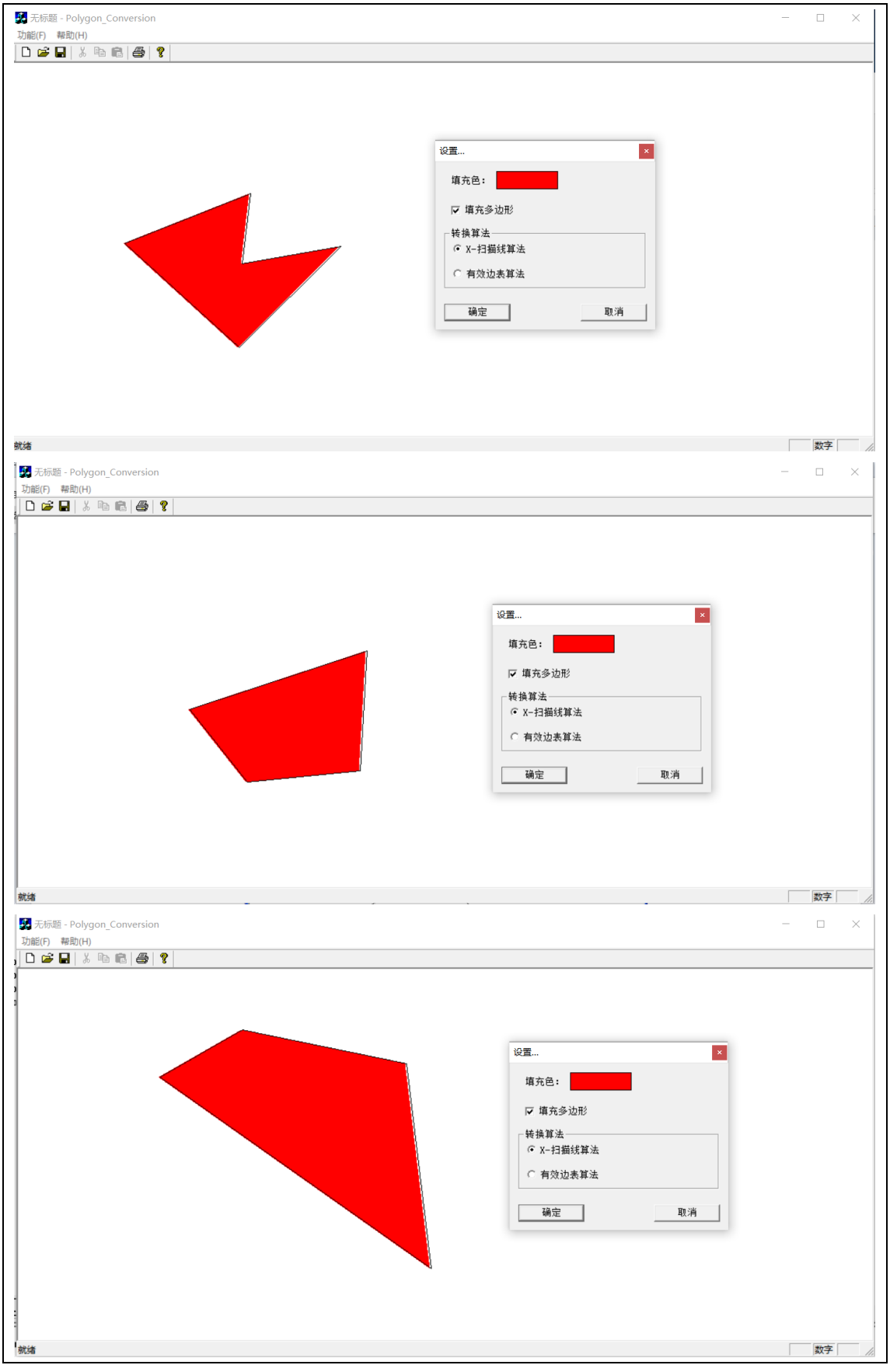
3.完成对所有扫描线的处理后，多边形的内部被填充完成。

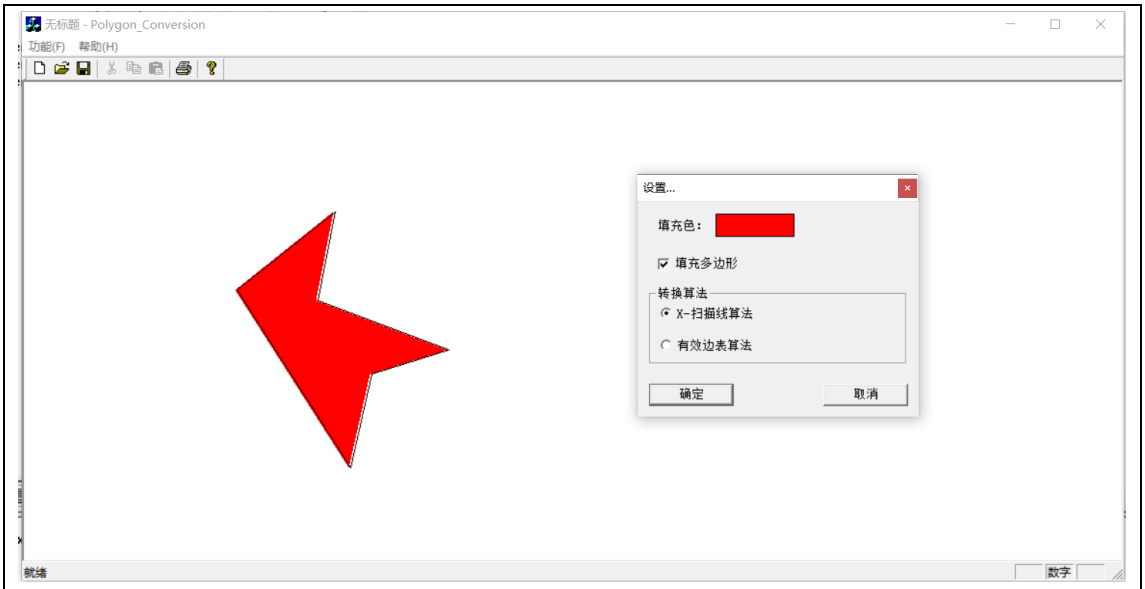
关键代码段：

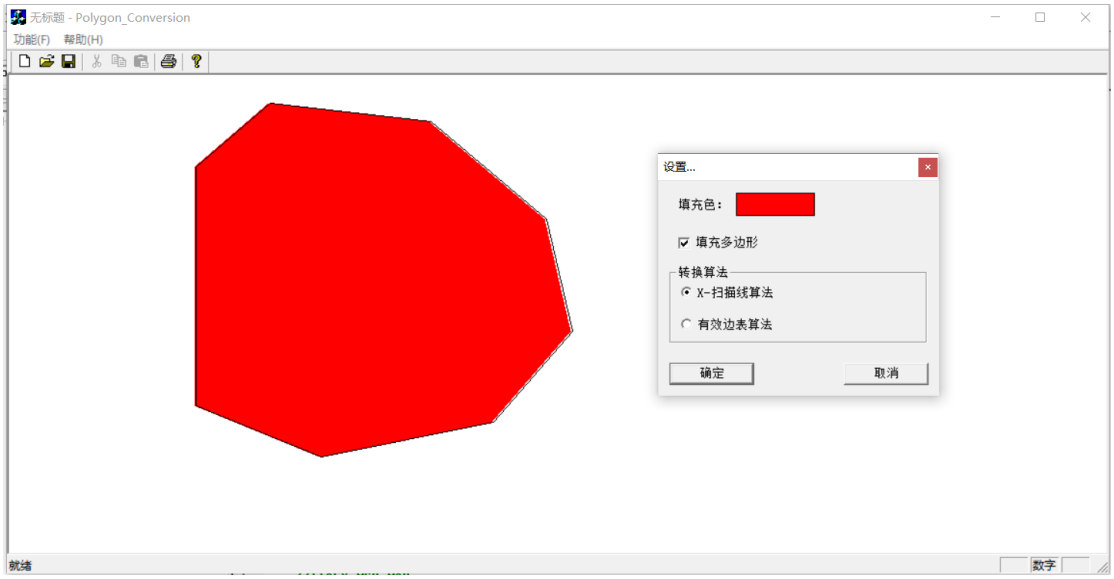
```
// 找到最小 Y 坐标和最大 Y 坐标
int minY = INT_MAX;
int maxY = INT_MIN;
for (int i = 0; i < VertexNum; ++i) {
    if (Vertices[i][1] < minY) {
        minY = Vertices[i][1];
    }
    if (Vertices[i][1] > maxY) {
        maxY = Vertices[i][1];
    }
}

// 对每条扫描线进行处理
for (int y = minY; y <= maxY; ++y) {
    // 存储与扫描线相交的边的 X 坐标
    std::vector<int> intersections;
```

```
        // 遍历每一条边，判断是否与当前扫描线相交
        for (int n = 0; n < VertexNum; ++n) {
            int nextIndex = (n + 1) % VertexNum; // 边的下一个顶点索引
            int x1 = Vertices[n][0];
            int y1 = Vertices[n][1];
            int x2 = Vertices[nextIndex][0];
            int y2 = Vertices[nextIndex][1];

            // 判断边是否与扫描线相交
            if ((y1 <= y && y2 > y) || (y2 <= y && y1 > y)) {
                // 计算交点的 X 坐标
                double xIntersect = (double)(x2 - x1) * (double)(y - y1) /
(double)(y2 - y1) + x1;
                intersections.push_back((int)xIntersect);
            }
        }

        // 对交点进行排序
        std::sort(intersections.begin(), intersections.end());

        // 绘制线段（边界）
        for (size_t i = 0; i < intersections.size(); i += 2) {
            int startX = intersections[i];
            int endX = intersections[i + 1];
            for (int x = startX; x < endX; ++x) {
                DrawPixel(x, y);
            }
        }
    }
```

结果截图：


无标题 - Polygon_Conversion
功能(F) 帮助(H)
设置...
填充色：
填充多边形
转换算法
X-扫描线算法
有效边表算法
确定
取消
就绪
数字





结果分析：

本次实验成功实现了 x 扫描算法，将图形绘制成功。

作业心得

我学到了以下几点心得体会：

1. 算法思想理解：扫描线填充算法的核心思想是通过扫描线与多边形边界的交点来确定多边形内部的像素填充顺序，从而实现多边形的填充。理解了这一思想后，就能够更好地理解和实现算法。

2. 代码实现细节：在编写代码的过程中，需要注意边界情况的处理以及数据结构的选择。比如，对于排序交点的向量，选择合适的数据结构能够更高效地实现排序操作。

3. 实验结果分析：在完成代码编写后，通过调试和测试可以观察算法的实际填充效果，从而分析算法的正确性和效率，并根据实际情况进行调整和优化。

思考：

Y-扫描线算法实现多边形填充的基本步骤：

1. 确定填充区域的范围：首先，找到多边形顶点集中的最小 Y 坐标和最大 Y 坐标，确定了需要处理的扫描线的范围。

2. 对每一条扫描线进行处理：从最小 Y 坐标开始，逐行扫描到最大 Y 坐标为止。

3.找到交点：在每一行扫描时，找出与该扫描线相交的多边形边界，计算出交点的 X 坐标。
4.填充区域:将每对相邻的交点之间的区域进行填充,即确定了该扫描线所覆盖的像素范围。
5.循环扫描：重复以上步骤，直到处理完所有的扫描线。
与 X-扫描线算法相比，Y-扫描线算法的主要异同点如下：
异同点：

- 扫描方向：X-扫描线算法按水平方向扫描，而 Y-扫描线算法按垂直方向扫描。
- 交点的计算：X-扫描线算法在每条扫描线上找出所有与多边形边界相交的点，而 Y-扫描线算法则是在每一行扫描时找出相交的点。
- 填充顺序：X-扫描线算法一般按照 X 坐标顺序对交点进行排序，而 Y-扫描线算法则按照 Y 坐标顺序进行填充。
- 适用范围：X-扫描线算法更适用于水平较长的填充区域，而 Y-扫描线算法更适用于垂直较长的填充区域。

虽然两种算法有所不同,但它们的核心思想都是通过扫描线与多边形的边界相交来确定填充区域，因此在实践中可以根据具体情况选择合适的算法进行实现。

提交日期：　　2024　年　4 月　6 日