**《计算机图形学》作业报告**

专业: **计算机科学与技术** 姓名: **侯腾跃** 学号: **2022217477**

|  |
| --- |
| 作业报告要求   1. 绘制程序流程图，给出核心程序代码段； 2. 展示运行结果（至少包括凹多边形、凸多边形的5个例子）并分析结果； 3. 算法拓展分析：是否可以用Y-扫描线算法实现多边形的扫描转换和区域填充？如何实现？比较与X-扫描线算法的异同； 4. 实现心得。 |
| **1.程序流程图**    **2.核心程序代码段**  void CPolygon\_ConversionView::X\_Scan\_Line\_Conersion(int Vertices[][2], int VertexNum)  {  // 求最小和最大y值  int minY = Vertices[0][1];  int maxY = Vertices[0][1];  for (int i = 1; i < VertexNum; i++)  {  minY = std::min(minY, Vertices[i][1]);  maxY = std::max(maxY, Vertices[i][1]);  }  // 对每一条扫描线进行处理  for (int y = minY; y <= maxY; y++)  {  // 求交点  int num = 0;  int pointX[20];  for (int i = 0; i < VertexNum; i++)  {  int j = (i + 1) % VertexNum;  if (Vertices[i][1] == Vertices[j][1] ||  (y < Vertices[i][1] && y < Vertices[j][1]) ||  (y >= Vertices[i][1] && y >= Vertices[j][1]))  {  continue;  }  pointX[num++] = static\_cast<int>(Vertices[i][0] + (y - Vertices[i][1]) \* (Vertices[j][0] - Vertices[i][0]) / (Vertices[j][1] - Vertices[i][1]));  }  // 对交点进行排序  std::sort(pointX, pointX + num);  // 绘制扫描线  for (int i = 0; i < num; i += 2)  {  for (int x = pointX[i]; x <= pointX[i + 1]; x++)  {  DrawPixel(x, y);  }  }  }  }  **3. 运行结果**    凸多边形    凸多边形    凹多边形  **不规则图形：**    不规则形    心形  **4. 运行结果分析：**  从实验截图可以看出，该函数能够正确地填充凸多边形和凹多边形的内部。  **5.是否可以用Y-扫描线算法实现多边形的扫描转换和区域填充？如何实现？比较与X-扫描线算法的异同。**  可以。  **1.初始化：**  创建两个表：全局边表（Global Edge Table, GET）和活性边表（Active Edge Table, AET）。全局边表用于存储所有边的信息，活性边表用于存储当前扫描线上相关的边。  **2.构建全局边表（GET）：**  遍历多边形的所有边，将每条边的信息（如起点、终点、斜率、最小 Y 值和最大 Y 值）存储到全局边表中。  将全局边表按边的最小 Y 值排序。  **3.扫描线处理：**  从最小的 Y 值开始，逐行扫描。在每条扫描线上，执行以下操作：  更新活性边表（AET）：从全局边表中移除当前扫描线对应的边，加入到活性边表中。更新活性边表中每条边的交点坐标（根据斜率计算）。按交点的 X 值对活性边表中的边进行排序。移除活性边表中最大 Y 值小于当前扫描线的边。  填充当前扫描线：遍历活性边表，按交点的 X 值配对填充区域。  **4.结束条件：**当扫描线超过多边形的最大 Y 值时，算法结束。   1. **扫描线算法与 X-扫描线算法的异同**   **相同点**  基本原理相同：两者都是通过逐行扫描多边形，计算扫描线与多边形边的交点来确定填充区域。  数据结构类似：都需要维护全局边表和活性边表，用于存储边的信息和当前扫描线相关的边。  核心步骤相似：构建边表。更新边表。计算交点并填充。  **不同点**  与X-扫描线算法相比，Y-扫描线算法的主要不同在于扫描线的方向。另外，Y-扫描线算法更适合于垂直于Y轴的图形，而X-扫描线算法更适合于垂直于X轴的图形。值得注意的是，Y-扫描线算法需要对多边形上的点进行排序，而X-扫描线算法则需要对线段进行排序。  **6.作业心得**  通过这次多边形扫描转换作业，我对计算机图形学中的填充算法有了更深入的理解。在实现 X-扫描线算法时，我遇到了许多挑战，尤其是处理凹多边形时，交点排序和配对问题让我颇费周折。经过多次调试和优化，我学会了如何通过引入标志位和调整逻辑来正确处理复杂情况。这个过程让我深刻体会到算法细节的重要性，也让我明白了理论与实践之间的差距。  在尝试实现有效边表转换算法时，我进一步认识到数据结构对算法效率的显著影响。通过优化边表的维护和更新逻辑，我成功提高了程序的运行效率。这让我意识到，合理设计数据结构是解决复杂问题的关键。  在测试过程中，我发现不同类型的多边形对算法的性能和正确性提出了不同的挑战。凸多边形相对简单，而凹多边形则需要更细致的处理。这让我明白，算法的通用性和鲁棒性是需要重点关注的方面。  这次作业也让我对 Y-扫描线算法产生了兴趣。通过对比分析，我了解到不同扫描方向的算法各有优劣，选择合适的算法需要根据具体问题的特征来决定。这次作业不仅提升了我的编程能力，也让我对算法的优化和选择有了更深刻的认识。未来，我将继续深入学习计算机图形学，探索更多高效且实用的算法。 |
| **提交日期： 年 月 日** |

**表. 作业成绩评定表**

|  |  |  |  |
| --- | --- | --- | --- |
| 评价内容 | | 权重 | 得分 |
| 算法实现完整性与正确性 | | 0.3 |  |
| 多边形类型多样性与运行效果 | | 0.2 |  |
| 核心代码与程序流程图 | | 0.2 |  |
| 结果总结分析 | | 0.15 |  |
| 算法扩展分析 | | 0.15 |  |
| 合计 |  | | |
| 指导教师（签章）： 年 月 日 | | | |