|  |  |  |  |
| --- | --- | --- | --- |
| **计算机图形学实验报告（一）** | | | |
| 学 号 | 姓 名 | 班 级 | 报告日期 |
| 180400715 | 杨卓宸 | 1804102 | 2020/10/16 |
| 实验内容 | 要求：  1.创建一个单文档程序  2.能够交互绘制折线  3.窗口刷新自动重绘。  思路：  左键按下时，确定折线的一个顶点，鼠标移动的过程中，绘制当前的线段，右键按下时，结束折线的绘制。 | | |
| 实验目的 | 1.熟练掌握MFC编程方法  2.能够绘制简单的二维图形（该实验无曲线）  3.掌握橡皮筋直线的画法  4.理解掌握图形刷新重绘的意义及方法  5.掌握交互式绘图的方法 | | |
| 实验预备知识 | 1.绘图工具类：  CPen类：GDI 画笔，用于画线。默认的画笔用于绘制与一个像素等宽的黑色实线。  CPalette类：GDI调色板，包含系统可用的色彩信息，是应用程序和彩色输出设备环境(如显示器)的接口。  CRgn类：GDI 区域，用于设备环境(通常是窗口)内的区域操作，通常和CDC类中与裁剪(clipping)有关的成员函数配合使用。  CPoint类：CPoint是对Windows结构POINT的封装，CPoint表示屏幕上的一个二维点。  2.绘图常用函数的使用方法：  画矩形 BOOL Rectangle( int x1, int y1, int x2, int y2 ); BOOL Rectangle( LPCRECT lpRect );  3.交互式绘图的方法：  通过添加消息函数onLButtonDown，onMouseMove，onRButtonDown，通过获取鼠标指针的坐标信息来获取折线的顶点及绘制方向信息。通过修改onDraw函数重绘并保持折线。  4.如何添加消息处理函数：  打开类视图，右键选中视类，点击属性，打开属性窗口，点击消息按钮，在相应的响应处添加消息处理函数。  5.重绘的意义与方法：  将已完成的图信息保存下来，重新绘制。 | | |
| 实验过程描述 | 1.首先在view类头文件中定义：  public:  CPoint startPoint; //线段起始点  CPoint oldPoint; //线段中止点  BOOL isDrawing = FALSE; //绘图状态标识  std::vector<std::pair<CPoint, CPoint>> lines;  //线段点存储集合  public:  afx\_msg void OnLButtonDown(UINT nFlags, CPoint point);  //鼠标左键按下消息处理函数  afx\_msg void OnMouseMove(UINT nFlags, CPoint point);  //鼠标移动绘图跟随处理函数  afx\_msg void OnRButtonDown(UINT nFlags, CPoint point);  //鼠标右键按下消息处理函数   1. 补充添加3个消息函数的源码：   void Ctest1View::OnLButtonDown(UINT nFlags, CPoint point)  {  // TODO: 在此添加消息处理程序代码和/或调用默认值  if (isDrawing) {  CDC\* pDC = GetDC();  //起点  pDC->MoveTo(startPoint);  //划线，终点  pDC->LineTo(point);  ReleaseDC(pDC);  lines.push\_back(std::make\_pair(startPoint, point));  }  else {  isDrawing = TRUE;  }  startPoint = point;  oldPoint = point;  CView::OnLButtonDown(nFlags, point);  }  void Ctest1View::OnMouseMove(UINT nFlags, CPoint point)  {  // TODO: 在此添加消息处理程序代码和/或调用默认值  if (isDrawing) {  CDC\* pDC = GetDC();  //起点  pDC->SetROP2(R2\_NOT);  pDC->MoveTo(startPoint);  pDC->LineTo(oldPoint);  pDC->MoveTo(startPoint);  //划线，终点  pDC->LineTo(point);  oldPoint = point;  ReleaseDC(pDC);  }  CView::OnMouseMove(nFlags, point);  }  void Ctest1View::OnRButtonDown(UINT nFlags, CPoint point)  {  // TODO: 在此添加消息处理程序代码和/或调用默认值  CPoint pt;  GetCursorPos(&pt);  lines.push\_back(std::make\_pair(startPoint, point));  isDrawing = FALSE;  CView::OnRButtonDown(nFlags, point);  }   1. 添加OnDraw函数中重绘的代码：   void Ctest1View::OnDraw(CDC\* pDC)  {  Ctest1Doc\* pDoc = GetDocument();  ASSERT\_VALID(pDoc);  if (!pDoc)  return;  // TODO: 在此处为本机数据添加绘制代码  for (auto line : lines)  {  //起点  pDC->MoveTo(line.first);  //划线，终点  pDC->LineTo(line.second);  }  }   1. 最后进行调试运行。 | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验结果 | 1. 首先按下左键确定起始点，之后依次按下四次左键确定中间四个顶点，最后按下右键结束绘画的同时，将该点作为终点，因此共6个顶点。   捕获   1. 第一次绘制完成后，可再次绘制下一系列线段。并且可以保持所有线段都可以显示。   捕获2   1. 窗口拉伸刷新后，线段仍能保持重绘。   捕获4 | | | |
| 实验当中问题  及解决方法 | 1. 起初不熟悉MFC的操作：上网查找简单的操作例子后，较为熟悉MFC的操作。 2. VS2019默认无法找到类视图，导致无法进行实验：通过不断摸索，发现应该在视图模块里面勾选显示视图类。   3、上网查询相关重绘信息后发现晦涩难懂：仔细阅读课程PPT，最后成功使用一个简单的画折线函数完成重绘。  4、右键单击时最后一段折线会消失，因为并没有进行存储，加入存储语句后，最后一段折线可以正常显示。  5、鼠标跟随绘图的功能比较困难，通过查找，了解到要通过不断反色处理上条线段来达到跟随效果，最终通过pDC->SetROP2(R2\_NOT)实现。 | | | |
| 成绩（教师打分） | 优秀 | 良好 | 及格 | 不及格 |