**暨南大学本科实验报告专用纸**

课程名称 windows编程实验 成绩评定

实验项目名称 Windows标准图形界面程序 指导教师 刘晓翔

实验项目编号 实验三 实验项目类型 设计型 实验地点 304

学生姓名 陈 伟 学号 2022100092

学院 智能科学与工程学院 系 人工智能 专业 人工智能

实验时间 2025 年6月4日上午～6月17日上午温度 14 ℃湿度

1. **实验目的**

①了解MFC消息映射机制；②掌握如何在窗口中绘图和输出文本；③掌握菜单编程（静态菜单与动态菜单）；④掌握界面元素（标题栏、窗口背景、鼠标形状、浮动工具栏、状态栏、启动界面）的编程方法。

1. **实验内容和要求**

利用Visual C++6.0软件开发工具编写程序，在实验二生成的VC工程基础上，完成一个具备图形/文本输出、菜单、自制工具栏、定制状态栏的Windows标准图形界面程序。

1. **主要仪器设备**

仪器：计算机

实验环境： Windows XP + Visual C++6.0

1. **实验原理**

MFC 文档/视图架构​​

文档（Document）：负责数据存储和管理（如绘图数据）。

视图（View）：负责数据显示和用户交互（如绘制图形、响应鼠标事件）。

框架窗口（Frame Window）：包含菜单、工具栏、状态栏等界面元素。

2. 消息映射机制​​

MFC 采用消息映射（Message Map）机制处理 Windows 消息（如菜单点击、鼠标移动）：

消息分类：

命令消息（WM\_COMMAND）：菜单项、工具栏按钮触发。

窗口消息（WM\_XXX）：鼠标、键盘、绘图事件。

消息路由：消息通过 BEGIN\_MESSAGE\_MAP 和 ON\_COMMAND 宏绑定到对应的处理函数。

3. 图形/文本输出原理​​

设备上下文（Device Context, DC）：

通过 CDC 类（CClientDC）获取绘图上下文。

使用 Rectangle()、TextOut() 等方法绘制图形和文本。

重绘机制：

调用 Invalidate() 触发 OnDraw() 重绘，确保窗口缩放时内容不丢失。

4. 菜单与工具栏​​

菜单：

在资源编辑器中设计菜单项，通过 ON\_COMMAND 绑定处理函数。

共享命令 ID（如 ID\_DRAW\_RECT）使菜单和工具栏按钮触发同一函数。

工具栏：

使用 CToolBar 类加载工具栏资源（IDR\_MYTOOLBAR）。

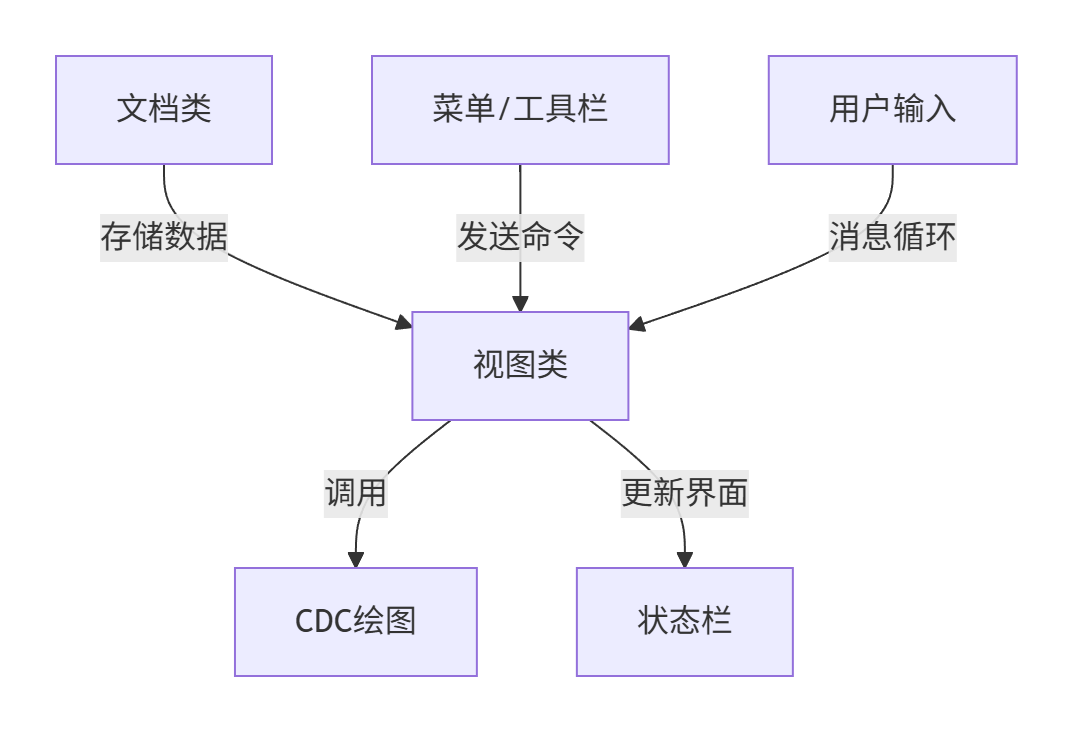
通过 EnableDocking() 和 DockControlBar() 实现停靠功能。

5. 状态栏定制​​

状态栏面板：

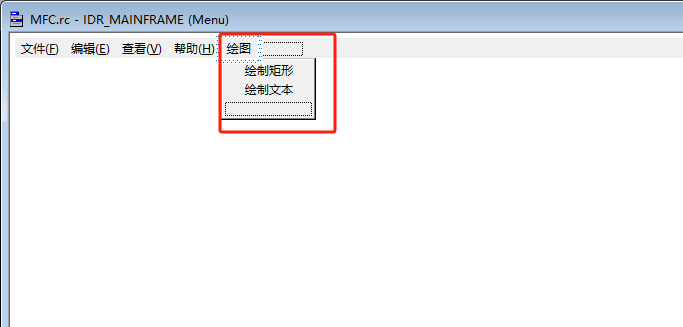
修改 indicators 数组定义面板数量和用途。

使用 CStatusBar::SetPaneText() 动态更新显示内容



1. **实验步骤与调试**

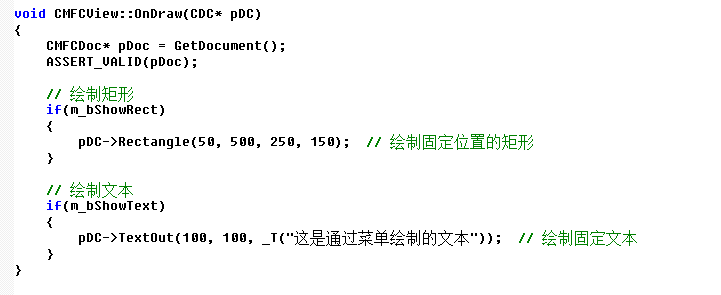
首先实现菜单栏绘制图形和文本，添加一个菜单项，这个菜单名称为绘图，他有两个功能分别为绘制矩形和绘制文本

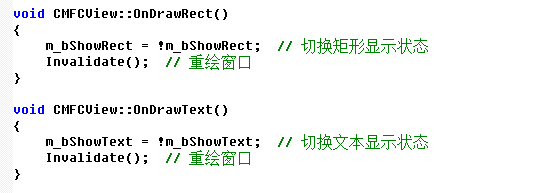


点击"绘制矩形"菜单会在窗口(50,50)-(250,150)位置绘制一个矩形

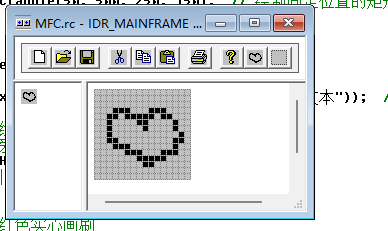
点击"绘制文本"菜单会在窗口(100,100)位置显示文本"这是通过菜单绘制的文本"

再次点击相应菜单会清除对应的图形或文本

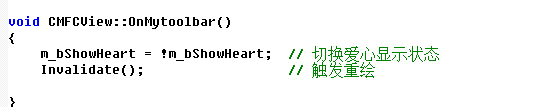




第二步实现自制工具栏首先添加一个新的工具栏，将他的ID设为IDR\_MYTOOLBAR，然后对他添加类向导



添加完成之后修改MFCVIEW.cpp的代码



第三步实现定制状态栏

1. 修改状态栏指示器数组

在MainFrm.cpp中找到indicators数组，修改为：

static UINT indicators[] =

{

ID\_SEPARATOR, // 状态行面板（显示操作提示）

ID\_INDICATOR\_CAPS, // Caps Lock状态

ID\_INDICATOR\_NUM, // Num Lock状态

};

2. 在视图中更新状态栏

void CMFCView::OnMouseMove(UINT nFlags, CPoint point)

{

// 获取主框架指针

CMainFrame\* pFrame = (CMainFrame\*)AfxGetApp()->m\_pMainWnd;

if (pFrame)

{

CString strCoord;

strCoord.Format(\_T("X=%d, Y=%d"), point.x, point.y);

// 更新面板（操作提示）

CString strTip = \_T("就绪");

if (m\_bShowRect) strTip = \_T("矩形模式");

if (m\_bShowText) strTip = \_T("文本模式");

pFrame->m\_wndStatusBar.SetPaneText(1, strTip);

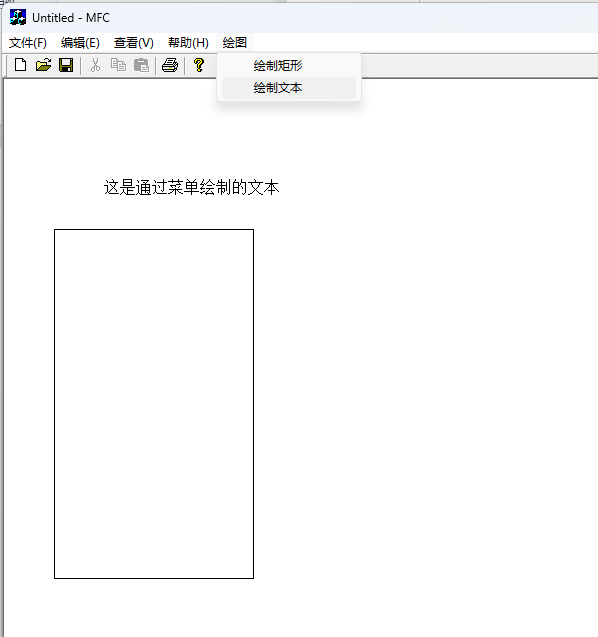
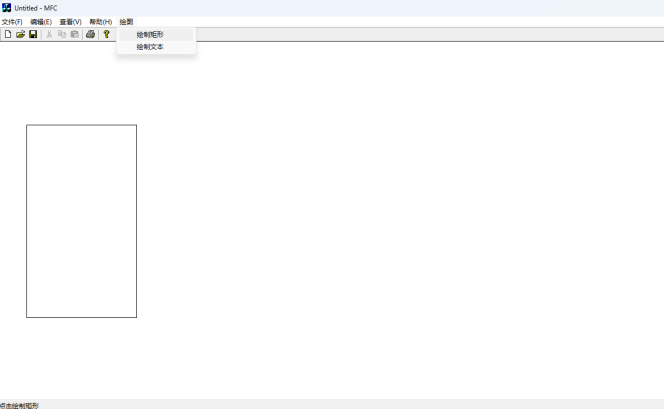
}

CView::OnMouseMove(nFlags, point);

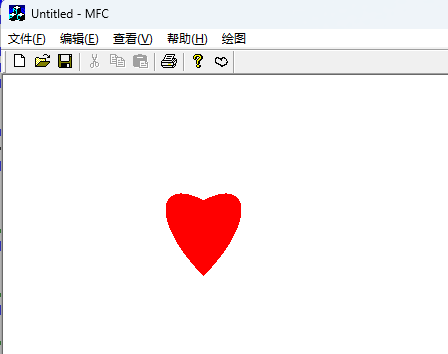
}

1. **实验结果与分析**

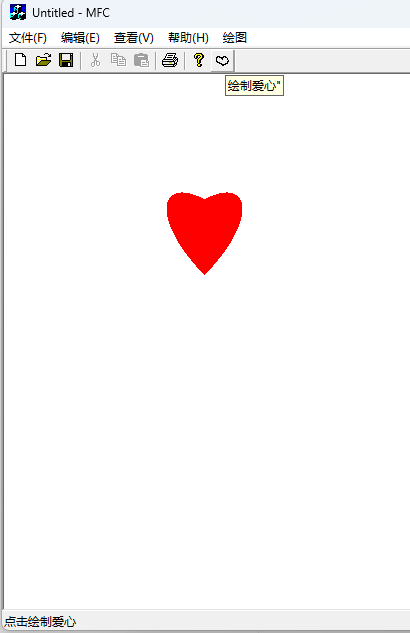
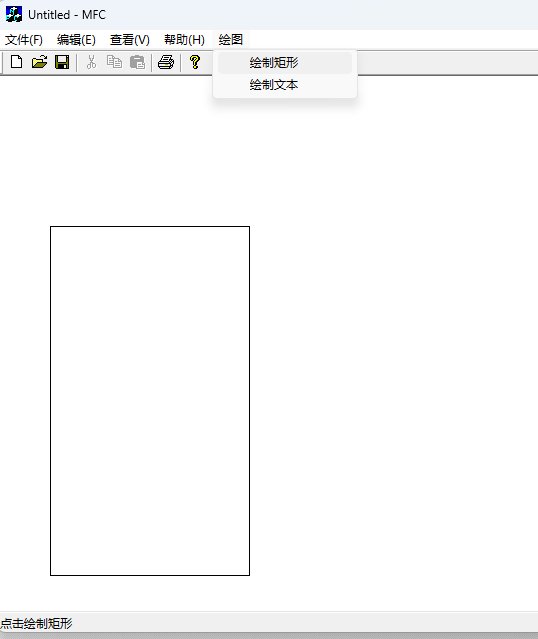
使用菜单实现图形和文本的绘制



实现自制工具栏绘制爱心



再实现状态栏



当切换模式时，可以看到左下角状态栏分别变成“点击绘制矩形”和“点击绘制爱心”

1. **源程序**

// MainFrm.cpp : implementation of the CMainFrame class

//

#include "stdafx.h"

#include "MFC.h"

#include "MainFrm.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/////////////////////////////////////////////////////////////////////////////

// CMainFrame

IMPLEMENT\_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN\_MESSAGE\_MAP(CMainFrame, CFrameWnd)

//{{AFX\_MSG\_MAP(CMainFrame)

// NOTE - the ClassWizard will add and remove mapping macros here.

// DO NOT EDIT what you see in these blocks of generated code !

ON\_WM\_CREATE()

//}}AFX\_MSG\_MAP

END\_MESSAGE\_MAP()

static UINT indicators[] =

{

ID\_SEPARATOR, // status line indicator

ID\_INDICATOR\_CAPS,

ID\_INDICATOR\_NUM,

ID\_INDICATOR\_SCRL,

};

/////////////////////////////////////////////////////////////////////////////

// CMainFrame construction/destruction

CMainFrame::CMainFrame()

{

// TODO: add member initialization code here

}

CMainFrame::~CMainFrame()

{

}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)

{

if (CFrameWnd::OnCreate(lpCreateStruct) == -1)

return -1;

if (!m\_wndToolBar.CreateEx(this, TBSTYLE\_FLAT, WS\_CHILD | WS\_VISIBLE | CBRS\_TOP

| CBRS\_GRIPPER | CBRS\_TOOLTIPS | CBRS\_FLYBY | CBRS\_SIZE\_DYNAMIC) ||

!m\_wndToolBar.LoadToolBar(IDR\_MAINFRAME))

{

TRACE0("Failed to create toolbar\n");

return -1; // fail to create

}

if (!m\_wndStatusBar.Create(this) ||

!m\_wndStatusBar.SetIndicators(indicators,

sizeof(indicators)/sizeof(UINT)))

{

TRACE0("Failed to create status bar\n");

return -1; // fail to create

}

// TODO: Delete these three lines if you don't want the toolbar to

// be dockable

m\_wndToolBar.EnableDocking(CBRS\_ALIGN\_ANY);

EnableDocking(CBRS\_ALIGN\_ANY);

DockControlBar(&m\_wndToolBar);

return 0;

}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)

{

if( !CFrameWnd::PreCreateWindow(cs) )

return FALSE;

// TODO: Modify the Window class or styles here by modifying

// the CREATESTRUCT cs

return TRUE;

}

/////////////////////////////////////////////////////////////////////////////

// CMainFrame diagnostics

#ifdef \_DEBUG

void CMainFrame::AssertValid() const

{

CFrameWnd::AssertValid();

}

void CMainFrame::Dump(CDumpContext& dc) const

{

CFrameWnd::Dump(dc);

}

#endif //\_DEBUG

/////////////////////////////////////////////////////////////////////////////

// CMainFrame message handlers