# 这一节我们来学习单选按钮控件

# 学习大纲

|  |
| --- |
|  |

## SendMessage函数用法

|  |
| --- |
| 将指定的消息发送到一个或多个窗口。 **SendMessage** 函数调用指定窗口的窗口过程，在窗口过程处理消息之前不会返回 。  若要发送消息并立即返回，请使用 [SendMessageCallback](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-sendmessagecallbacka) 或 [SendNotifyMessage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-sendnotifymessagea) 函数。 若要将消息发布到线程的消息队列并立即返回，请使用 [PostMessage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-postmessagea) 或 [PostThreadMessage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-postthreadmessagea) 函数。 语法 C++复制  LRESULT SendMessage(  [in] HWND hWnd,  [in] UINT Msg,  [in] WPARAM wParam,  [in] LPARAM lParam  ); 参数 [in] hWnd  类型：**HWND**  窗口的句柄，其窗口过程将接收消息。 如果此参数 **HWND\_BROADCAST** ( (HWND) 0xffff) ，则消息将发送到系统中的所有顶级窗口，包括禁用或不可见的无所有者窗口、重叠窗口和弹出窗口;但消息不会发送到子窗口。  消息发送受 UIPI 约束。 进程线程只能将消息发送到完整性级别较低或相等进程的线程的消息队列。  [in] Msg  类型： **UINT**  要发送的消息。  有关系统提供的消息的列表，请参阅 [系统定义的消息](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/about-messages-and-message-queues)。  [in] wParam  类型：WPARAM  其他的消息特定信息。  [in] lParam  类型：LPARAM  其他的消息特定信息。 返回值 类型： **LRESULT**  返回值指定消息处理的结果;这取决于发送的消息。 注解 当消息被 UIPI 阻止时，使用 [GetLastError](https://learn.microsoft.com/zh-cn/windows/desktop/api/errhandlingapi/nf-errhandlingapi-getlasterror) 检索的最后一个错误设置为 5 (拒绝访问) 。  需要使用 **HWND\_BROADCAST** 进行通信的应用程序应使用 [RegisterWindowMessage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-registerwindowmessagea) 函数获取应用程序间通信的唯一消息。  系统仅对系统消息执行封送处理， (范围为 0 到 ([WM\_USER-1](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/wm-user)) ) 。 若要将其他消息 (那些 >= **WM\_USER**) 发送到另一个进程，必须执行自定义封送处理。  如果指定的窗口是由调用线程创建的，则窗口过程将立即作为子例程调用。 如果指定的窗口是由其他线程创建的，则系统会切换到该线程并调用相应的窗口过程。 仅在接收线程执行消息检索代码时，才会处理线程之间发送的消息。 在接收线程处理消息之前，将阻止发送线程。 但是，发送线程将在等待处理其消息时处理传入的非排队消息。 若要防止这种情况，请将 [SendMessageTimeout](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-sendmessagetimeouta) 与SMTO\_BLOCK集结合使用。 有关非排队消息的详细信息，请参阅 [非排队消息](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/about-messages-and-message-queues)。  辅助功能应用程序可以使用 **SendMessage** 将 [WM\_APPCOMMAND](https://learn.microsoft.com/zh-cn/windows/desktop/inputdev/wm-appcommand) 消息发送到 shell 以启动应用程序。 此功能不保证适用于其他类型的应用程序。 |

## SendDlgItemMessage函数用法

|  |
| --- |
| 将消息发送到对话框中的指定控件。 语法 LRESULT SendDlgItemMessageA(  [in] HWND hDlg,  [in] int nIDDlgItem,  [in] UINT Msg,  [in] WPARAM wParam,  [in] LPARAM lParam  ); 参数 [in] hDlg  类型：**HWND**  包含控件的对话框的句柄。  [in] nIDDlgItem  类型： **int**  接收消息的控件的标识符。  [in] Msg  类型： **UINT**  要发送的消息。  有关系统提供的消息的列表，请参阅 [系统定义的消息](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/about-messages-and-message-queues)。  [in] wParam  类型：WPARAM  其他的消息特定信息。  [in] lParam  类型：LPARAM  其他的消息特定信息。 返回值 类型： **LRESULT**  返回值指定消息处理的结果，取决于发送的消息。 注解 在处理消息之前 **，SendDlgItemMessage** 函数不会返回 。  使用 **SendDlgItemMessage** 与检索指定控件的句柄并调用 [SendMessage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-sendmessage) 函数相同。 示例 有关示例，请参阅 [创建无模式对话框](https://learn.microsoft.com/zh-cn/windows/desktop/dlgbox/using-dialog-boxes)。  **备注**  winuser.h 标头将 SendDlgItemMessage 定义为别名，该别名根据 UNICODE 预处理器常量的定义自动选择此函数的 ANSI 或 Unicode 版本。 将非特定编码别名与非非特定编码的代码混合使用可能会导致不匹配，从而导致编译或运行时错误。 有关详细信息，请参阅 [**函数原型的约定**](https://learn.microsoft.com/zh-cn/windows/win32/intl/conventions-for-function-prototypes)。 |

# 演练

## 1.新建一个c++常规空项目，取名：Lesson30-RadioButton，然后新建一个cpp文件，取名radiobuttondemo.cpp,然后我们添加一个对话框资源，给确定按钮添加代码，然后用这个对话框资源来创建基于对话框的应用程序

|  |
| --- |
|  |

### 然后我们添加一些基本代码

|  |
| --- |
| #include<Windows.h>  #include"resource.h"  INT\_PTR CALLBACK DlgProc(HWND hDlg,UINT uMsg,WPARAM wParam,LPARAM lParam)  {  int cmdId;  BOOL ret = TRUE;  switch(uMsg)  {  case WM\_COMMAND:  cmdId = LOWORD(wParam);  switch(cmdId)  {  case IDOK:  if(IDOK==MessageBox(hDlg,TEXT("退出程序？"),TEXT("退出确认"),MB\_OKCANCEL))  {  EndDialog(hDlg,FALSE);  }  break;  default:  ret = FALSE;  break;  }  break;  case WM\_CLOSE:  EndDialog(hDlg,FALSE);  break;  default:  ret = FALSE;  break;  }  return ret;  }  int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE hPrev,LPTSTR lpCmd,int nShow)  {  DialogBox(hInstance,MAKEINTRESOURCE(IDD\_DLGMAIN),NULL,DlgProc);  return 0;  } |

## 2.我们在对话框里面添加一个组框，标题改为选择颜色，然后在里面添加8个单选按钮，代表不同的颜色，然后再添加一个组框，添加2给单选按钮，代表不同形状，然后在右边添加一个静态文本，作为绘图区域，设置边框，不要文字，ID为IDC\_STATIC\_DRAW，然后在他的上面添加一个静态文本，内容为：在下面的绘图区域作图

|  |
| --- |
|  |

## 3.此时你运行程序会有一个问题，就是即使在两个不同的组框里面，还是只能选择一个单选按钮，我们需要做的就是把每一个组框里面的第一个单选按钮的Group属性设置为True，其余的都是False，而且还需要设置Tab顺序，点击格式-》tab顺序可以查看，tab顺序是连续的，都是在一组的。那么如何修改tab顺序？点击就可以了

|  |
| --- |
|  |

### 可以看到1-8是一个组，14-15是另外一个组

## 4.然后我们来修改这些单选按钮的文本属性和ID属性

|  |
| --- |
|  |

### 5.所有的颜色的ID命名为IDC\_RADIO\_颜色名称，如IDC\_RADIO\_BLACK,所有的形状的id为IDC\_RADIO\_形状，如IDC\_RADIO\_RECT,

## 老师的绘制代码

|  |
| --- |
|  |

## 6.我们先定义一个函数来用指定的颜色和形状绘图，叫做DrawGraph，代码如下

|  |
| --- |
| void DrawGraph(HWND hwnd)  {  HDC hdc;  HBRUSH hBGBrush,hbrush,hOldBrush;  RECT rc;  GetClientRect(hwnd,&rc);  hbrush = CreateSolidBrush(color);  hBGBrush = CreateSolidBrush(RGB(255,255,255));  hdc = GetDC(hwnd);  //1.填充白色背景  FillRect(hdc,&rc,hBGBrush);  hOldBrush = (HBRUSH)SelectObject(hdc,hbrush);  //绘图  if(shape){  Rectangle(hdc,rc.left,rc.top,rc.right,rc.bottom);  }  else  {  Ellipse(hdc,rc.left,rc.top,rc.right,rc.bottom);  }  DeleteObject(SelectObject(hdc,hOldBrush));  ReleaseDC(hwnd,hdc);  } |

## 7.当我们选择颜色或者选择选择的时候，都需要定义这个函数来绘图

|  |
| --- |
|  |
|  |

## 8.因为现在建议使用全局变量我们把上面的代码修改如下

|  |
| --- |
| #include<Windows.h>  #include"resource.h"  void DrawGraph(HWND hwnd,int shape,COLORREF color)  {  HDC hdc;  HBRUSH hBGBrush,hbrush,hOldBrush;  RECT rc;  GetClientRect(hwnd,&rc);  hbrush = CreateSolidBrush(color);  hBGBrush = CreateSolidBrush(RGB(255,255,255));  hdc = GetDC(hwnd);  //1.填充白色背景  FillRect(hdc,&rc,hBGBrush);  hOldBrush = (HBRUSH)SelectObject(hdc,hbrush);  //绘图  if(shape){  Rectangle(hdc,rc.left,rc.top,rc.right,rc.bottom);  }  else  {  Ellipse(hdc,rc.left,rc.top,rc.right,rc.bottom);  }  DeleteObject(SelectObject(hdc,hOldBrush));  ReleaseDC(hwnd,hdc);  }  INT\_PTR CALLBACK DlgProc(HWND hDlg,UINT uMsg,WPARAM wParam,LPARAM lParam)  {  int cmdId;  BOOL ret = TRUE;  static COLORREF color;  static int shape;  static HWND hwndPaint;//绘图窗口句柄    switch(uMsg)  {  case WM\_INITDIALOG:  hwndPaint = GetDlgItem(hDlg,IDC\_STATIC\_DRAW);//获取绘图窗口  shape = 1;  color = RGB(255,255,0);  break;  case WM\_PAINT:  UpdateWindow(hDlg);  DrawGraph(hwndPaint,shape,color);  ret = FALSE;//在对话框的WM\_PAINT消息响应响应加上这一句，否则有问题  break;  case WM\_COMMAND:  cmdId = LOWORD(wParam);  switch(cmdId)  {  case IDOK:  if(IDOK==MessageBox(hDlg,TEXT("退出程序？"),TEXT("退出确认"),MB\_OKCANCEL))  {  EndDialog(hDlg,FALSE);  }  break;  case IDC\_RADIO\_BLACK:  color = RGB(0,0,0);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_BLUE:  color = RGB(0,0,255);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_GREEN:  color = RGB(0,255,0);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_CYAN:  color = RGB(0,255,255);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_RED:  color = RGB(255,0,0);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_MAGENTA:  color = RGB(255,0,255);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_YELLOW:  color = RGB(255,255,0);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_WHITE:  color = RGB(255,255,255);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_RECT:  shape = 1;  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_ELLIPSE:  shape =0;  DrawGraph(hwndPaint,shape,color);  break;  default:  ret = FALSE;  break;  }  break;  case WM\_CLOSE:  EndDialog(hDlg,FALSE);  break;  default:  ret = FALSE;  break;  }  return ret;  }  int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE hPrev,LPTSTR lpCmd,int nShow)  {  DialogBox(hInstance,MAKEINTRESOURCE(IDD\_DLGMAIN),NULL,DlgProc);  return 0;  } |

## 注意：我们只是处理WM\_COMMAND消息是不够的，我们还需要处理WM\_PAINT消息，我们在初始化对话框的信息响应代码里面设置了默认值，所以我们在响应WM\_PAINT消息时把默认图形绘制了，注意这里的写法：ret = FALSE这一句话非常重要，如果没有它，程序会有问题，他的意思是把我们不感兴趣的东西交给默认对话框过程处理。这部分的代码如下

|  |
| --- |
|  |

## 9.不过有一点问题，我们的程序默认没有形状然后按钮，我们需要他选中我们的默认图形和默认色，我们可以在对话框初始化信息响应代码里面对这两个默认的控件发送BM\_SETCHECK信息，把wParam设置为1.

|  |
| --- |
|  |

### 效果：

|  |
| --- |
|  |

### 注意：静态文本控件也是一个窗口，所以我们可以用它来绘图

# 这一节的学习到此结束，这一节的完整代码如下

## radiobuttondemo.cpp

|  |
| --- |
| #include<Windows.h>  #include"resource.h"  void DrawGraph(HWND hwnd,int shape,COLORREF color)  {  HDC hdc;  HBRUSH hBGBrush,hbrush,hOldBrush;  RECT rc;  GetClientRect(hwnd,&rc);  hbrush = CreateSolidBrush(color);  hBGBrush = CreateSolidBrush(RGB(255,255,255));  hdc = GetDC(hwnd);  //1.填充白色背景  FillRect(hdc,&rc,hBGBrush);  hOldBrush = (HBRUSH)SelectObject(hdc,hbrush);  //绘图  if(shape){  Rectangle(hdc,rc.left,rc.top,rc.right,rc.bottom);  }  else  {  Ellipse(hdc,rc.left,rc.top,rc.right,rc.bottom);  }  DeleteObject(SelectObject(hdc,hOldBrush));  ReleaseDC(hwnd,hdc);  }  INT\_PTR CALLBACK DlgProc(HWND hDlg,UINT uMsg,WPARAM wParam,LPARAM lParam)  {  int cmdId;  BOOL ret = TRUE;  static COLORREF color;  static int shape;  static HWND hwndPaint;//绘图窗口句柄    switch(uMsg)  {  case WM\_INITDIALOG:  hwndPaint = GetDlgItem(hDlg,IDC\_STATIC\_DRAW);//获取绘图窗口  shape = 1;  color = RGB(255,255,0);  SendMessage(GetDlgItem(hDlg,IDC\_RADIO\_YELLOW),BM\_SETCHECK,1,0);//这里默认选中黄色  //SendMessage(GetDlgItem(hDlg,IDC\_RADIO\_RECT),BM\_SETCHECK,1,0);//这里默认选中矩形  SendDlgItemMessage(hDlg,IDC\_RADIO\_RECT,BM\_SETCHECK,1,0);//也可以这么写  break;  case WM\_PAINT:  UpdateWindow(hDlg);  DrawGraph(hwndPaint,shape,color);  ret = FALSE;//在对话框的WM\_PAINT消息响应需要加上这一句，否则有问题  break;  case WM\_COMMAND:  cmdId = LOWORD(wParam);  switch(cmdId)  {  case IDOK:  if(IDOK==MessageBox(hDlg,TEXT("退出程序？"),TEXT("退出确认"),MB\_OKCANCEL))  {  EndDialog(hDlg,FALSE);  }  break;  case IDC\_RADIO\_BLACK:  color = RGB(0,0,0);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_BLUE:  color = RGB(0,0,255);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_GREEN:  color = RGB(0,255,0);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_CYAN:  color = RGB(0,255,255);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_RED:  color = RGB(255,0,0);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_MAGENTA:  color = RGB(255,0,255);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_YELLOW:  color = RGB(255,255,0);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_WHITE:  color = RGB(255,255,255);  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_RECT:  shape = 1;  DrawGraph(hwndPaint,shape,color);  break;  case IDC\_RADIO\_ELLIPSE:  shape =0;  DrawGraph(hwndPaint,shape,color);  break;  default:  ret = FALSE;  break;  }  break;  case WM\_CLOSE:  EndDialog(hDlg,FALSE);  break;  default:  ret = FALSE;  break;  }  return ret;  }  int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE hPrev,LPTSTR lpCmd,int nShow)  {  DialogBox(hInstance,MAKEINTRESOURCE(IDD\_DLGMAIN),NULL,DlgProc);  return 0;  } |