Тема № 9. Графика в Windows (API – функции), обработка сообщений от клавиатуры, мыши, меню, полос прокрутки

9.1. Контексты устройств

Функции для получения контекста устройств (в программе используются хэндлы контекстов устройств - HDC):

- HDC GetDC(HWND hWnd);
- HDC BeginPaint(HWND hWnd, LPPAINTSTRUCT lpPaint);

Освобождение контекста устройств:

- int ReleaseDC(HWND hWnd, HDC hDC);
- BOOL EndPaint(HWND hWnd, CONST PAINTSTRUCT *lpPaint);

9.2. Графические «перья» и «кисти»

Функция для получения предопределенных «перьев» или «кистей»:

- HGDIOBJ GetStockObject(int);

Возможные параметры:

Для «перьев»:

- WHITE_PEN, BLACK_PEN, NULL_PEN

Для «кистей»:

- WHITE_BRUSH, LTGRAY_BRUSH, GRAY_BRUSH, DKGRAY_BRUSH, BLACK_BRUSH, NULL_BRUSH

Функция для создания «пера»:

- HPEN CreatePen(int, int, COLORREF);

Первый параметр определяет стиль, возм. значения:

- PS_SOLID /* _____ */
- PS_DASH /* ------- */
- PS_DOT /* ..... */
- PS_DASHDOT /* _._._._ */
- PS_DASHDOTDOT /* _.._.._ */
- PS_NULL

Второй параметр определяет толщину.

Третий параметр определяет цвет, для получения цвета можно использовать макрос:

RGB(r,g,b) он возвращает тип объявленный как

typedef DWORD COLORREF;

Создать сплошную кисть:

- HBRUSH CreateSolidBrush(COLORREF);

Создать кисть заданного стиля:

- HBRUSH CreateHatchBrush(int, COLORREF);

Первый параметр определяет стиль, возм. значения:

- HS_HORIZONTAL /* ----- */
- HS_VERTICAL /* ||||| */
- HS_FDIAGONAL /* \\\\\ */
- HS_BDIAGONAL /* ///// */
- HS_CROSS /* +++++ */
- HS_DIAGCROSS /* xxxxx */

Загрузить перо или кисть в контекст устройства:

- HGDIOBJ SelectObject(HDC, HGDIOBJ);

После работы с объектами их необходимо удалить:

BOOL DeleteObject(HGDIOBJ);

9.3. Функции для вывода графических изображений

- COLORREF SetPixel(HDC, int, int, COLORREF);
- COLORREF SetTextColor(HDC, COLORREF);
- COLORREF SetBkColor(HDC, COLORREF);
- BOOL TextOutA(HDC, int, int, LPCSTR, int);
- BOOL MoveToEx(HDC, int, int, LPPOINT);
- BOOL LineTo(HDC, int, int);
- BOOL Rectangle(HDC, int, int, int, int);
- BOOL Ellipse(HDC, int, int, int, int);
- BOOL Polygon(HDC, CONST POINT *, int);

9.4. Примеры вывода графических изображений. Сообщения от мыши и от клавиатуры

Сообщение от клавиатуры (WM_CHAR)

Код символа содержится в младшей части wParam.

Сообщения от мыши:

Координаты курсора: x – младшее слово lParam, y – старшее слово lParam, wParam содержит дополнительную информацию (например, какие клавиши нажаты при перемещении мыши).

Для получения младшего и старшего слов можно использовать макросы:

x=LOWORD(lParam); y=HIWORD(lParam);

9.5. Перерисовка окна. Сообщение WM_PAINT

Основные способы сохранения изображения в памяти (перерисовки):

- сохранение изображения в виде некоторых данных (массивов, списков и т.п.) в памяти или файле, при перерисовке восстановление изображения по сохраненным данным, или для построения изображения (перерисовки используются функциональные зависимости);
- вывод изображения в виртуальное окно, при перерисовке копирование виртуального окна в контекст устройства окна (клиентской области).

При перерисовке для получения и освобождения контекста устройства используются функции BeginPaint и EndPaint.

Послать сообщение на перерисовку (WM_PAINT)

BOOL InvalidateRect(HWND *hWnd*,
// handle to window
CONST RECT* *lpRect*,
// rectangle coordinates
BOOL *bErase* // erase state
);

9.6. Использование меню

Описание меню в файле ресурсов:

```
MENU1 MENU DISCARDABLE
BEGIN
    POPUP "Пункт меню1"
    BEGIN
        MENUITEM "Пункт 2",                 ID_1_2
        MENUITEM "&Пункт 3",                  ID_1_3,
    CHECKED
    END
END
```

Подключение к окну:

```
wcl.lpszMenuName="MENU1"
```

Обработка сообщения от меню (сообщение WM_COMMAND)

```
case WM_COMMAND:
     int nItem;
// Получаем идентификатор выбранного элемента
     nItem=LOWORD(wParam);
        switch(nItem)
     {
// Обработка элемента с идентификатором ID_1_2
     case ID_1_2:
……….
          return 0;
// Обработка элемента с идентификатором ID_1_3
     case ID_1_3:
……….
          return 0;
     }
```

9.7. Вывод графических файлов в окно

9.8. Полосы прокрутки

При создании окна определяет стиль окна (третий параметр функции CreateWindow) как

WS_OVERLAPPEDWINDOW |
WS_VSCROLL | WS_HSCROLL

Функции для работы с полосами прокрутки:

Установить диапазон ПП:

SetScrollRange(HWND hWnd, int nBar, int nMinPos, int nMaxPos, BOOL bRedraw);

Возможные значений параметра nBar:

SB_HORZ - гориз. ПП, SB_VERT - верт. ПП

Установить значение ползунка:

```
SetScrollPos( HWND hWnd,  int nBar,
    int nPos,  BOOL bRedraw);
```

Сообщения полос прокрутки:

- WM_HSCROLL - для горизонт ПП;
- WM_VSCROLL - для вертикальной ПП;

Младшее слово параметра wParam несет информацию о действии, которое выполнено с ПП

Возможные значения младшего слова wParam:

- SB_LINEUP Нажатие на стрелку вверх
- SB_LINELEFT Нажатие на стрелку влево
- SB_LINEDOWN …. на стрелку вниз
- SB_LINERIGHT …. на стрелку вправо
- SB_PAGEUP Нажатие выше ползунка
- SB_PAGELEFT Нажатие левее ползунка
- SB_PAGEDOWN …. ниже ползунка

- SB_PAGERIGHT … правее ползунка
- SB_THUMBPOSITION Перемещение ползунка закончено
- SB_THUMBTRACK Перемещение ползунка
- SB_TOP Нажата клавиша “Home”
- SB_LEFT Нажата клавиша “Home”
- SB_BOTTOM Нажата клавиша “End”
- SB_RIGHT Нажата клавиша “End”

Старшее слово wParam содержит текущее значение ползунка.

Пример обработки сообщений

```
case WM_VSCROLL:
 switch(LOWORD(wParam))
 {
 case SB_LINEDOWN:
     if (Pos<maxRange) Pos++;
     break;
 case SB_LINEUP:
     if (Pos>0) Pos--;
     break;
```

```
case SB_THUMBTRACK:
        Pos=HIWORD(wParam);
}
    SetScrollPos(hwnd, SB_VERT, Pos,
   TRUE);
    InvalidateRect(hwnd, NULL, TRUE);
    return 0;
```