Лабораторная работа № 14

Тема: *Разработка программ с графическим выводом в окно, используя Win32 API*
Цель: *Освоить создание приложений с графическим интерфейсом с использованием функций Win32 API: перерисовка окна, работа с контекстом устройства отображения.*

Краткая теория

Графический вывод в окно обладает несколькими особенностями:

1. Для вывода в окно необходимо использовать специальные функции библиотеки Win32API. При этом, как правило, в окно стараются выводить данные в одном месте программы – при обработке сообщения WM_PAINT.
2. Интерфейс графических устройств (Graphics Device Interface - GDI) открывает доступ к большому количеству функций вывода.
3. Параметры вывода устанавливаются в контексте отображения с помощью функций GDI. Контекст отображения – это структура данных, которая содержит характеристики устройства ввода и указатели на выбранные инструменты рисования. Функции GDI используют только выбранные в контекст отображения параметры и инструменты рисования.
4. Дескриптор контекста отображения служит первым аргументом вызова всех функций, связанных с выводом в окно.

Способы обработки сообщения WM_PAINT

1 способ

```
case WM_PAINT:{
  PAINTSTRUCT ps;
  HDC hdc = BeginPaint(hwnd, &ps);
  //Настройка контекста и вывод в окно
  EndPaint(hwnd, &ps);
  break;
}
```

2 способ

```
case WM_PAINT:{
  HDC hdc = GetDC(hwnd);
  //Получение параметров контекста
  //Настройка контекста и вывод в окно
  ReleaseDC(hwnd, hdc);
  break;
}
```

Структура PAINTSTRUCT описана следующим образом:

```
typedef struct{
  HDC hdc;              //Дескриптор графического устройства
  BOOL fErase;          //Признак необходимости стирания фона
                        //области
  RECT rcPaint;         //Координаты прямоугольника перерисовки
  BOOL fRestore;              //Системные флаги
  BOOL fncUpdate;             //Системные флаги
  BYTE rgbReserved[16];       //Системные флаги
} PAINTSTRUCT;
```

Если fErase=TRUE, то фон окна стирается, иначе фон окна не изменяется.
Функции получения контекста:

```
HDC BeginPaint(HWND hwnd, PAINTSTRUCT *ps);
HDC GetDC(HWND hwnd);
```

Функции освобождения контекста:

```
BOOL EndPaint(HWND hwnd, CONST PAINTSTRUCT *ps);
int ReleaseDC(HWND hwnd, HDC hdc);
```

Пример: написать программу, которая в центре экрана выводит сообщение “Hello world!”

```
//Глобальные переменные:
char mess[] = "HELLO WORLD!";
int cX, cY;
…
//Фрагмент обработки события изменения размера окна
 case WM_SIZE:{
      cX = LOWORD(lParam)/2; cY = HIWORD(lParam)/2;
      break;
 }
```

```
//Вывод в окно (1 вариант)
 case WM_PAINT:{
  PAINTSTRUCT ps;
  HDC hdc = BeginPaint(hwnd,&ps);
  SetTextColor(hdc,RGB(255,0,0));
  SetBkColor(hdc,RGB(0,128,128));
  SetTextAlign(hdc,TA_CENTER);
  TextOut(hdc,cX,cY,mess,strlen(mess));
  EndPaint(hwnd,&ps);
  break;
}
```

```
//Вывод в окно (2 вариант)
case WM_PAINT:{
 HDC hdc = GetDC(hwnd);
 SetTextColor(hdc,RGB(255,0,0));
 SetBkColor(hdc,RGB(0,128,128));
 SetTextAlign(hdc,TA_CENTER);
 TextOut(hdc,cX,cY,mess,strlen(mess));
 ReleaseDC(hwnd,hdc);
 break;
}
```

Если в программе необходимо принудительно перерисовать окно, то для этого можно использовать функцию UpdateWindow, которая отсылает окну сообщение WM_PAINT. Функция перерисовки части окна:

BOOL InvalidateRect(HWND hwnd, CONST RECT *lpRect, BOOL bErase);

Функция добавляет прямоугольник в область перерисовки. Флаг bErase управляет режимом перекраски фона: если он равен TRUE, то фон перекрашивается, в противном случае – нет. Если в качестве lpRect передать значение NULL, то будет перерисовано все окно.

Функция удаления прямоугольника из списка прямоугольников перерисовки:

BOOL ValidateRect(HWND hwnd, CONST RECT *lpRect);

Пример: реализовать программу, в которой по центру экрана изначально выводится текст “Программа запущена”. Если пользователь нажмет левую клавишу мыши, то текст заменяется на “Нажата левая клавиша мыши”, а если правую – то на “Нажата правая клавиша мыши”.

```
//Глобальные переменные:
int cX, cY;    //Координаты центра окна
char mess[30] = "Программа запущена";  //Сообщение
…
//Обработчики нажатия клавиш мыши:
case WM_LBUTTONDOWN:{
   strcpy(mess,"Нажата левая клавиша мыши");
   InvalidateRect(hwnd,NULL,TRUE);
   break;
}
case WM_RBUTTONDOWN:{
   strcpy(mess,"Нажата правая клавиша мыши");
   InvalidateRect(hwnd,NULL,TRUE);
   break;
}
```

Существуют следующие виды контекста отображения:

- общий,
- для класса окон,
- личный,
- родительский,
- для окна.

Общий контекст отображения отличает высшая скорость работы доступа к нему. Для получения этого контекста необходимо вызвать одну из функций: BeginPaint или GetDC. Стиль окна с общим контекстом не может содержать значения CS_OWNDC, CS_PARENTDC или CS_CLASSDC.

Контекст отображения для класса окон создается для всех окон определенного класса. Для этого при регистрации класса окна указывают стиль CS_CLASSDC. Все окна этого класса будут пользоваться этим контекстом отображения. Приложение, однажды получив контекст отображения для класса окна, могут не освобождать его. Причем вызов функций EndPaint или ReleaseDC не освобождает такой контекст (хотя и не вредит). Если создано несколько окон такого класса, область ограничения и начало системы физических координат вывода автоматически настраиваются на то окно, которое использует контекст отображения. Такой контекст отображения повышает производительность вывода в окно за счет того, что не требует настройки параметров после каждого вызова функции BeginPaint или GetDC.

Личный контекст отображения определяется для окно со стилем CS_OWNDC. Его получают один раз и настраивают его атрибуты, а освобождают только при закрытии окна. Функции BeginPaint, GetDC, EndPaint, ReleaseDC не оказывают никакого влияния на этот контекст.

Родительский контекст отображения используют дочерние окна. Он позволяет дочерним окнам «унаследовать» атрибуты контекста отображения у родительского окна. С этой целью в стиле класса дочерних окон указывают CS_PARENTDC.

Контекст отображения для окна предназначен для реализации возможности рисования в любой части окна: заголовке, рабочей области, системного меню, рамок и т.д. Для получения этого контекста используют функцию GetWindowDC. Далее он используется аналогично общему контексту.

Функции установки атрибутов:

Установка цвета фона:

```
COLORREF SetBkColor(HDC hdc, COLORREF crColor);
```

COLORREF равноценен DWORD. Для формирования COLORREF используют функцию:

```
COLORREF RGB(BYTE clRed, BYTE clGreen, BYTE clBlue);
```

Функция установки режима фона:

```
int SetBkMode(HDC hdc, int iBkMode);
```

Параметр iBkMode может принимать два значения:

- OPAQUE – не прозрачный (по умолчанию),
- TRANSPARENT – прозрачный.

Функция установки режима рисования:

```
int SetROP2(HDC hdc, int fnDrawMode);
```

Значения fnDrawMode:

- R2_BLACK – черный;
- R2_COPYPEN – цвет пера;
- R2_MASKNOTPEN или R2_MERGENOTPEN – комбинация цвета пикселя до рисования и инверсии цвета пера;
- R2_MASKPEN или R2_MERGEPEN – комбинация цвета пикселя до рисования и цвета пера;

- R2_MASKPENNOT или R2_MERGEPENNOT– комбинация инверсии цвета пикселя до рисования и цвета пера;
- R2_NOP – остается неизменным;
- R2_NOT – инверсия цвета пикселя до рисования;
- R2_NOTCOPYPEN – инверсия цвета пера;
- R2_NOTMASKPEN или R2_NOTMERGEPEN – инверсия цвета пикселя до рисования;
- R2_WHITE – белый;
- R2_XORPEN – операция xor к цвета пикселя до рисования и пера.

Функция установки цвета текста:

```
COLORREF SetTextColor(HDC hdc, COLORREF crColor);
```

Функция устанвоки расстояния между буквами:

```
int SetTextCharasterExtra(HDC hdc, int nCharExtra);
```

Значение nCharExtra во всех режимах (кроме MM_TEXT) определяет расстояние в логических единицах, кратных пикселю.

Функция создания сплошной кисти:

```
HBRUSH CreateSolidBrush(COLORREF crColor);
```

Функция создания штриховой кисти:

```
HBRUSH CreateHatchBrush(int fnStyle, COLORREF crColor);
```

Значения fnStyle:

- HS_BDIAGONAL – линии под 45 градусов;
- HS_CROSS – в клетку без наклона;
- HS_DIAGCROSS – в клетку с наклоном 45 градусов;
- HS_FDIAGONAL – линии под 135 градусов;
- HS_HORIZONTAL – горизонтальные линии;
- HS_VERTICAL – вертикальные линии.

Функция выбора в контекст созданного объекта:
HGDIOBJ SelectObject(HDC hdc, HGDIOBJ hgdiobj);
Возвращает дескриптор предыдущего объекта.

Удаление объекта:
BOOL DeleteObject(HGDIOBJ hgdiobj);
Перед удалением возвращают старое значение объекта.

Функция создания пера:
HPEN CreatePen(int fnStyle, int iWidth, COLORREF crColor);

Значения fnStyle:

- PS_SOLID – сплошная;
- PS_DASH – штриховая;
- PS_DOT – пунктирная;
- PS_DASHDOT – штрихпунктирная с одной точкой;
- PS_DASHDOTDOT – штрихпунктирная с двумя точками;
- PS_NULL – невидимая;
- PS_INSIDEFRAME – линии для обводки замкнутых фигур.

Функция установки позиции пера:

```
BOOL MoveToEx(HDC hdc, int x, int y, LPPOINT lpPoint);
```

Структура POINT имеет следующее описание:

```
typedef struct{
  LONG x;
  LONG y;
} POINT;
```

Функция смены режима отображения:

```
int SetMapMode(HDC hdc, int fnMapMode);
```

Режимы fnMapMode:

MM_ANISOTROPIC	Произвольные направления для осей, произвольные логические единицы
MM_HIENGLISH	Логическая единица равна 0.001 дюйма, ось X – вправо, ось У – вверх.
MM_HIMECTRIC	Логическая единица равна 0.01 мм, ось X – вправо, ось У – вверх.
MM_ISOTROPIC	Логические единицы произвольны и одинаковы по обоим направлениям
MM_LOENGLISH	Логическая единица равна 0.01 дюйма, ось X – вправо, ось У – вверх.
MM_HIMECTRIC	Логическая единица равна 0.1 мм, ось X – вправо, ось У – вверх.
MM_TEXT	Логические единицы равны размерам пикселя, ось X – вправо, ось У – вниз
MM_TWIPS	Логическая единица равна 1/1440 дюйма, ось X – вправо, ось У – вверх.

Создание шрифта:

```
HFONT WINAPI CreateFontIndirect(const LOGFONT FAR *lpft);
```

В качестве параметра в функцию передается указатель на заполненную структуру LOGFONT.

Вывод текста:

```
BOOL TextOut(HDC hdc, int nXStart, int nYStart, LPCTSTR lpString,
             int cbString);
```

Функция выравнивания текста:

```
UINT SetTextAlign( HDC hdc, UINT fMode);
```

Режимы выравнивания:

- TA_LEFT, TA_RIGHT, TA_CENTER;
- TA_TOP, TA_BASELINE, TA_CENTER;
- TA_NOUPDATECP, TA_UPDATECP.

Функции рисования:

Вывод пикселя:

```
COLORREF SetPixel(HDC hdc, int x, int y, COLORREF crCol);
```

Чтение пикселя:

```
COLORREF GetPixel(HDC hdc, int x, int y);
```

Чтение текущей позиции:

```
BOOL GetCurrentPosition(HDC hdc, LPPOINT lpPnt);
```

Рисование линии от текущей позиции:

```
BOOL LineTo(HDC hdc, int x, int y);
```

Рисование линии:
```
BOOL DrawLine(HDC hdc, int x0, int y0, int x, int y);
```

Функция рисования дуги эллипса или окружности:
```
BOOL Arc(HDC hdc, int l, int t, int r, int b, int x0, int y0,
         int x, int y);
```

Дуга рисуется против хода часовой стрелки.

Функция рисования ломаной линии:
```
BOOL PolyLine(HDC hdc, CONST POINT *lppt, int nPoints);
```

Функция рисования прямоугольника:
```
BOOL Rectangle(HDC hdc, int l, int t, int r, int b);
```

Функция рисования прямоугольника с закругленными углами:
```
BOOL RoundRect(HDC hdc, int l, int t, int r, int b,
               int w, int h);
```

Функция рисования закрашенного прямоугольника:
```
int FillRect(HDC hdc, CONST RECT *lpRect, HBRUSH brush);
```

Функция рисования прямоугольной рамки:
```
int FrameRect(HDC hdc, CONST RECT *lpRect, HBRUSH brush);
```

Функция инвертирования цветов в заданной прямоугольной области:
```
BOOL InvertRect(HDC hdc, CONST RECT *lpRect);
```

Функция обозначения пунктиром границ заданной области:
```
BOOL DrawFocusRect(HDC hdc, CONST RECT *lpRect);
```

Функция рисования эллипса:
```
BOOL Ellipse(HDC hdc, int l, int t, int r, int b);
```

Функция рисования сегмента эллипса:
```
BOOL Chord(HDC hdc, int l, int t, int r, int b, int x0, int y0,
           int x, int y);
```

Функция рисования сектора эллипса:
```
BOOL Pie(HDC hdc, int l, int t, int r, int b, int x0, int y0,
         int x, int y);
```

Функция рисования полигона:
```
BOOL Polygon(HDC hdc, CONST POINT *lppt, int nPoints);
```

Ход работы

В рамках данной лабораторной работы необходимо, используя Win32API, разработать программу с графическим оконным интерфейсом согласно варианту задания. При разработке программы использовать мастера не рекомендуется. Выполнение лабораторной работы можно осуществлять в IDE Pelles C или Visual C++. Для разработки программы необходимо создать проект типа Win32 Application.

Варианты заданий:

1	В текстовом файле содержаться координаты точек на плоскости. Необходимо построить замкнутую ломаную линию по этим точкам. При этом каждый сегмент ломанной должен быть различного цвета. Цвет выбирается случайно.

2	В окне отображается квадрат, который плавно перемещается по горизонтали от одного края к другому. При этом если он двигается вправо, то он должен отображаться красным цветом. А если влево, то синим. Цвет фона произвольный, выбирается случайно при каждом запуске программы.
3	В центре окна нарисовать эллиптическую диаграмму. Диаграмму разбить на секторы 25, 65 и 10% красного, зеленого и голубого цветов и указать по центру дуги каждого сектора проценты. При всех изменениях размера окна диаграмма должна быть отображена полностью.
4	При нажатии левой кнопки мыши над рабочей областью окна в месте нажатия отображается зеленый кружок, а при нажатии правой кнопки мыши – красный квадрат. При повторном нажатии фигура в старом месте пропадает, а отображается в новом месте рабочей области.
5	В окне приложения отобразить рейтинг участников какого-либо события: для каждого участника нарисовать цилиндр, высота которого пропорциональна рейтингу участника, ниже цилиндра вывести фамилию участника, выше – рейтинг (в %). Для всех участников использовать разные цвета. Список участников загружается из текстового файла (не более 10 участников).
6	В центре рабочей области окна изображен квадрат, который плавно вращается вокруг своего центра. При нажатии любой клавиши мыши над квадратом направление вращения меняется на противоположное.
7	В окне отображается квадрат красного цвета. При приближении к нему курсора мыши он должен «убегать» в произвольном направлении, не выходя при этом за границы рабочей области окна.
8	В рабочей области окна располагаются прямоугольники. При нажатии левой клавиши мыши на прямоугольник, тот меняет цвет заливки согласно последовательности: красный – синий – желтый – зеленый – красный - … Координаты прямоугольников задаются в текстовом файле. В каждой строке по 4 числа: координаты левого верхнего и правого нижнего углов.
9	При нажатии левой кнопки мыши в месте нажатия появляется красный кружок, который начинает плавно «падать» к нижней границе окна. Когда он ее достигает, то исчезает. Допускается присутствие нескольких кружков на экране одновременно.
10	В центре она нарисовать мишень из 10 полей и в каждом поле вывести его значение (от 1 с краю, до 10 в центре). При любых изменениях окна мишень должна полностью отображаться в окне. Поля раскрасить случайно выбранными различными цветами. Пользователь может нажимать мышью на мишень, при этом в заголовке окна выводится сумма «нажатий».
11	В окне отобразить черным цветом оси координат (0XУ) и синим цветом график функции c*cos(x), где X принимает значения от 0 до 10π и C≠0. Размах вывода по осям ординат и абсцисс – 90% от размеров окна при любых изменениях размеров. Значение C берется из текстового файла.
12	Центр рабочей области занимает малиновый прямоугольник с вписанным голубым эллипсом размером в половину области по осям. После нажатия левой клавиши мыши это место занимает синий прямоугольник, а после правой клавиши мыши – зеленый эллипс. При изменении размера окна вернуться к исходному состоянию.
13	В окне нарисовать шахматную доску и обозначить поля. При нажатии левой клавиши мыши над полем доски сообщить, используя MessageBox, имя поля.
14	В рабочей части окна при движении мыши с нажатой левой клавишей мыши рисовать траекторию движения курсора мыши. При нажатии правой клавиши мыши поменять цвет для рисования. Линия не должна пропадать при перетаскивании экрана.
15	В окне отобразить клетки для игры в крестики – нолики. При нажатии левой клавиши мыши в клетке нарисовать крестик, а правой клавиши мыши – нолик. Запретить заполнять клетку более одного раза.

16	Нарисовать светофор, в котором цвет «зажигается» при нажатии на эту лампу левой клавишей мыши. Одновременно может гореть только один цвет.
17	В окне с помощью мыши выделить прямоугольник и создать эффект мигания этого прямоугольника. Прямоугольник выделяется путем нажатия левой клавиши мыши в одном углу и отпускания левой клавиши мыши в противоположном углу.
18	Нарисовать несколько геометрических фигур. При нажатии над любой из них показать, что выбрана эта фигура. В качестве фигур использовать прямоугольники и эллипсы. Показывать выбор фигуры осуществлять изменением цвета и толщины границы фигуры.
19	Нарисовать графики функций: 2*a*cos(kt)*exp(-nt) и 2*a*sin(kt)*exp(-nt). A, K и N – константы, T – аргумент функции. Значения A, K, N берутся из текстового файла. T изменяется от 0 до текущей ширины экрана. График по оси ординат должен масштабироваться по высоте экрана.
20	Пользователь с помощью мыши рисует на экране треугольник: тремя щелками левой клавиши мыши задает позиции вершин треугольника. Построить окружность, описанную вокруг треугольника. При нажатии правой клавиши мыши очистить окно.
21	Пользовать с помощью мыши рисует линии на экране. Рисование начинается при нажатии левой клавиши мыши над рабочей областью экрана (в эту точку помещается начало линии), а заканчивается при отпускании левой клавиши мыши (в эту точку помещается конец линии). Пока левая клавиша мыши нажата линия должна рисоваться от начала, до текущего положения мыши.
22	При запуске программы на экране в произвольном месте отображается квадрат. Реализовать эффект перетаскивания квадрата: при нажатии левой клавиши мыши над квадратом тот начинает перемещаться туда, куда перемещается указатель мыши, пока пользователь не отпустит левую клавишу мыши.
23	Реализовать программу, в которой в произвольном месте рабочей области окна отображается закрашенный круг. Пользователь может перемещать этот круг, нажимая клавиши «вверх», «вниз», «вправо», «влево» на клавиатуре. Круг не должен выходить за границы рабочей области.
24	Написать программу «Часы»: на экране отображается циферблат часов с часовой, минутной и секундной стрелкой. Секундная стрелка самая длинная и тонкая, а часовая – самая короткая и толстая. Изначально часы устанавливаются по текущему системному времени.
25	Реализовать программу с бегущей строкой. По окну (по центру вертикали) справа налево перемещается текст, длина которого больше ширины окна. После того как последний символ текста «уезжает» за левый край окна, начало текста снова появляется справа.

Контрольные вопросы

1. Какими особенностями обладает вывод в окно в Win32 API?
2. Что такое контекст отображения?
3. При обработке какого сообщения следует перерисовывать окно?
4. Опишите типовую последовательность действий при перерисовке окна.
5. Как программно вызвать перерисовку окна?
6. Какие виды контекстов отображения бывают, и в каких случаях они используются?
7. Перечислите основные функции установки атрибутов отображения.
8. Перечислите основные функции рисования графических примитивов.