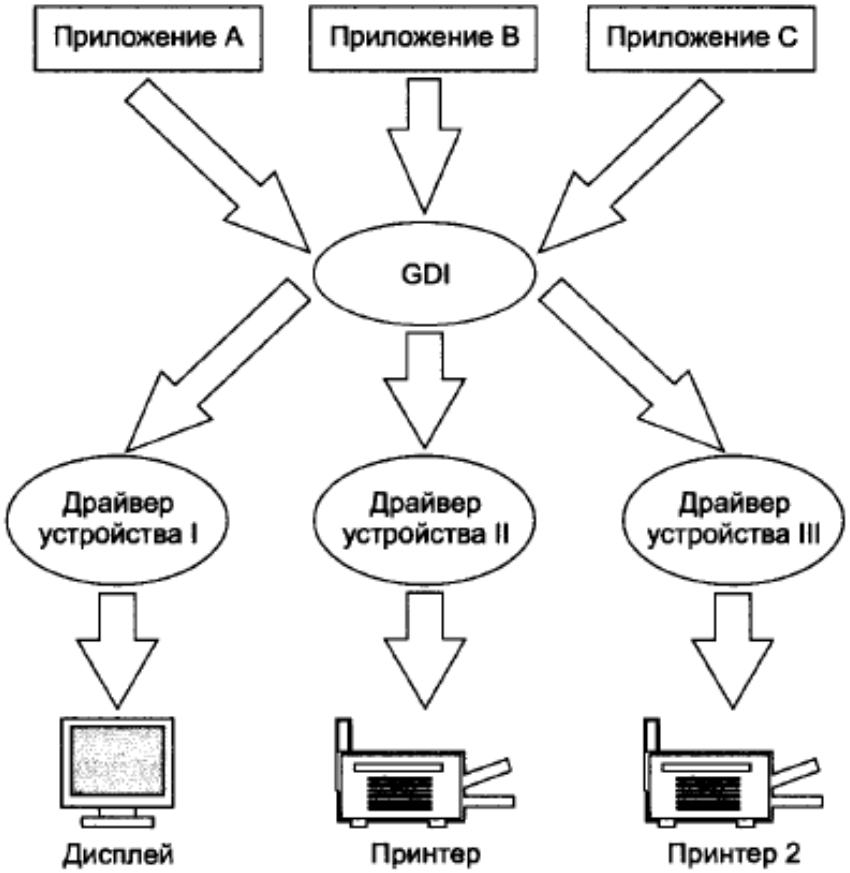
Тема 3. Графический интерфейс устройств

Концепция GDI

Графический интерфейс устройств (GDI) – совокупность программных средств Windows, обеспечивают рисование графических примитивов аппаратно-независимым образом.



Группы графических устройств

- растровые (*raster*) устройства: в растровой графике изображения описываются прямоугольными массивами пикселов, которые называют *растровыми изображениями* или *битовыми картами* (*bitmap*).

  Число цветов, которые могут быть представлены в битовом образе, равно 2^n, где n — количество битов на пиксел.

- векторные (*vector*) устройства: векторная графика — это построение линий, кривых и заполненных областей с применением аналитической геометрии.

Контекст устройства

Контекст устройства (Device Context - DC) – это внутренняя структура данных, которая определяет набор графических объектов и ассоциированных с ними атрибутов, а также графических режимов, влияющих на вывод.

Основные графические объекты:

- *Перо (pen)* для рисования линий.
- *Кисть (brush)* для заполнения фона или заливки фигур.
- *Растровое изображение (bitmap)* для отображения в указанной области окна.
- *Палитра (palette)* для определения набора доступных цветов.
- *Шрифт (font)* для вывода текста.
- *Регион (region)* для отсечения области вывода.

Контекст устройства (продолжение)

Контекст устройства содержит много атрибутов, определяющих поведение функций GDI. Благодаря этому списки параметров функций GDI содержат только самую необходимую информацию. Все остальное система извлекает из контекста устройства.

Контекст устройства относится к числу системных ресурсов, количество которых в системе может быть ограничено; работа с такого рода ресурсами всегда протекает одинаково: сначала надо получить у системы требуемый ресурс, а закончив работу с ним, вернуть его системе.

Функции GDI

- функции, которые получают (или создают) и освобождают (или уничтожают) контекст устройства;
- функции, которые получают информацию о контексте устройства;
- функции, которые устанавливают и получают атрибуты контекста устройства;
- функции рисования;
- функции, которые работают с объектами GDI.

Примитивы GDI

- прямые (отрезки) и кривые;
- закрашенные области;
- битовые шаблоны (растровые шаблоны, растровые образы);
- текст.

Типы контекста устройства

- контекст дисплея обеспечивает работу с дисплеем;
- контекст принтера обеспечивает работу с принтером;
- контекст в памяти (совместимый контекст) моделирует в памяти устройство вывода;
- метафайловый контекст также моделирует устройство вывода, сохраняя векторные и растровые команды в файле на диске;
- информационный контекст служит для получения данных от устройства;

Дескриптор контекста устройства

Дескриптор (описатель) контекста устройства (handle to a device context - HDC) – число, которое Windows использует для внутренней ссылки на объект.

Возвращая этот дескриптор после вызова соответствующих функций, Windows тем самым предоставляет разработчику право на использование данного устройства. После этого дескриптор контекста устройства передается как параметр в функции GDI, чтобы идентифицировать устройство, на котором должно выполняться отображение.

Во время обработки каждого отдельного сообщения программа должна получить и освободить дескриптор. После освобождения он становится недействительным и не должен далее использоваться.

Получение дескриптора. 1-й метод

```
HDC BeginPaint(
    HWND hWnd,                      // дескриптор окна
    LPPAINTSTRUCT lpPaint);         // указатель на структуру типа
                                    //PAINTSTRUCT

BOOL EndPaint(HWND hWnd, const LPPAINTSTRUCT *lpPaint);
```

Типовой процесс обработки сообщения WM_PAINT:

```
        HDC hDC;
        PAINTSTRUCT ps;
…
        case WM_PAINT:
            hdc = BeginPaint (hWnd, &ps);
            [инициализация атрибутов контекста устройства]
            [рисование в клиентской области окна]
            EndPaint (hWnd, &ps);
            return 0;
```

Получение дескриптора. 2-й метод

```
HDC GetDC(HWND hWnd); // дескриптор окна
HDC GetWindowDC(HWND hWnd);

int ReleaseDC(HWND hWnd,
              HDC hDC); // дескриптор контекста устройства
```

Типовой процесс обработки:

```
HDC hdc;
…
hDC = GetDC(hWnd);
[ использование функций GDI ]
ReleaseDC(hWnd, hDC);
```

Контекст принтера

При необходимости вывода на принтер программа должна создать контекст устройства с помощью функции

```
HDC CreateDC(
      LPCTSTR lpszDriver, // имя драйвера
      LPCTSTR lpszDevice, // имя устройства
      LPCTSTR lpszOutput, // не используется, =NULL
      CONST DEVMODE* lpInitData // данные о принтере
);
```

После распечатки прикладная программа должна удалить контекст принтера с помощью функции

```
BOOL DeleteDC(HDC hDC); // дескриптор контекста устройства
```

Информационный контекст

Для того, чтобы получить, например, характеристики принтера, программа создает информационный контекст, используя для этого функцию

```
HDC CreateIC(
      LPCTSTR lpszDriver,   // имя драйвера
      LPCTSTR lpszDevice,   // имя устройства
      LPCTSTR lpszOutput,   // имя порта или файла (=NULL
                            // для Win32)
      CONST DEVMODE* lpDvm // указатель на структуру, куда
);                          // записывается извлекаемая
                            //информация
```

После того, как надобность в информационном контексте миновала, программа должна удалить его с помощью функции `DeleteDC()`.

Контекст в памяти

Алгоритм работы с контекстом в памяти состоит из нескольких шагов:

1) Получения дескриптора контекста устройства (hDC) для окна, в которое будет осуществляться вывод изображения.
2) Получения дескриптора bitmap'а, который будет отображаться в окне.
3) Получения совместимого с hDC контекста в памяти (для хранения изображения) с помощью функции

```
HDC CreateCompatibleDC(HDC hDC);
```

4) Выбора изображения (hBitmap) как текущего для контекста в памяти (hCompatibleDC).
5) Копирования изображения контекста в памяти (hCompatibleDC) на контекст устройства (hDC).
6) Удаления совместимого контекста (hCompatibleDC).
7) Принятия мер для того, чтобы замещенный bitmap из контекста в памяти не остался в памяти.
8) Освобождения контекста устройства (hDC).

Установка атрибутов контекста

Пиксель — минимальный по размерам изобразительный элемент, которым может управлять приложение.

Цвет пикселя — некоторая точка в трехмерном RGB-пространстве, образованном тремя цветовыми осями: *Red* (красная цветовая составляющая), *Green* (зеленая составляющая) и *Blue* (синяя составляющая):

черный цвет — все три составляющие = 0 (0x00);

белый цвет — все три составляющие = 255 (0xFF).

Всего комбинаций — 2^{24} или 16 777 216 цветов.

Управление цветом

Функция преобразования в формат **COLORREF** (эквивалент **DWORD**):

```
COLORREF RGB(BYTE byRed, BYTE byGreen, BYTE byBlue);
```

В случае неудачи **RGB** возвращает **CLR_INVALID**.

Функции разделения данных **COLORREF** на составляющие RBG-модели:

```
BYTE GetRValue(COLORREF rgb); // число от 0 до 255
BYTE GetGValue(COLORREF rgb):
BYTE GetBValue(COLORREF rgb);
```

Цвет фона

Некоторые графические примитивы GDI содержат пиксели двух видов:

- основные *(foreground),*
- фоновые *(background).*

Фон графического элемента — совокупность его фоновых пикселей.

Атрибуты контекста:

- цвет фона графических элементов *(background color),*
- режим смешивания фона (*background mix mode*) .

Функция для фона

Цвет фона графических элементов *(background color)* устанавливает функция:

```
COLORREF SetBkColor( HDC hdc, // дескриптор контекста
                     COLORREF crColor // новый цвет фона
                   );
```

Возвращаемое значение - предыдущее значение цвета RGB; **CLR_INVALID** – в случае неудачи.

Режимы фона

- непрозрачный (OPAQUE - значение по умолчанию),
- прозрачный (TRANSPARENT).

Режим фона устанавливает функция:

```
int SetBkMode( HDC hdc, // дескриптор контекста
               int  iBkMode     // режим фона
 );
```

Возвращаемое значение - предыдущее значение режима;

0 – в случае неудачи.

Определение текущего режим фона:

```
int GetBkMode( HDC hdc ); // дескриптор контекста
```

Возвращаемое значение - OPAQUE или TRANSPARENT;

0 – в случае неудачи.

Типы координатных систем

1) Физическая система координат представляет собой матрицу пикселов фиксированной ширины и высоты. Начало координат находится в левом верхнем углу. Ось X направлена слева направо, а ось Y— сверху вниз. Максимальный размер физического устройства равен 2^{27} x 2^{27} пикселов.

2) Система координат устройства используются при работе с контекстами устройств. В общем случае система координат устройства является под множеством соответствующей физической системы координат.

3) Логическая система координат — собственная система координат приложения, приближенная к его геометрической модели. В таких системах координат удобнее работать, и они гораздо меньше зависят от оборудования.

Функции преобразования

1) между физическими экранными координатами и координатами устройств (клиентскими координатами):

```
BOOL ClientToScreen(HWND hWnd, LPOINT lpPoint);
BOOL ScreenToClient(HWND hWnd, LPOINT lpPoint);
```

2) между координатами устройства и логическими координатами:

```
BOOL DPtoLP(
  HDC hdc,              // дескриптор контекста
  LPPOINT lpPoints,     // массив точек
  int nCount            // количество точек в массиве
);
```

Режим отображения

Режим отображения	Логические единицы	Направление увеличения	
		ось x	ось y
MM_TEXT	Пиксели	вправо	вниз
MM_LOMETRIC	0.1 мм	вправо	вверх
MM_HIMETRIC	0.01 мм	вправо	вверх
MM_LOENGLISH	0.01 дюйма	вправо	вверх
MM_HIENGLISH	0.001 дюйма	вправо	вверх
MM_TWIPS	1/1440 дюйма	вправо	вверх
MM_ISOTROPIC	Произвольные (x=y)	выбирается	выбирается
MM_ANISOTROPIC	Произвольные (x!=y)	выбирается	выбирается

Установить режим отображения:

```
int SetMapMode(
    HDC hdc,
    int fnMapMode); //  новый режим отображения
```

Определить текущий режим:

```
    int iMapMode = GetMapMode (hdc);
```

Системы координат

Переместить систему координат (логические единицы):

```
BOOL SetWindowOrgEx( HDC hdc,
          int x, int y,  // новые координаты
          LPPOINT lpPoint);  // старые координаты
```

Новое начало системы координат (физические единицы):

```
BOOL SetViewportOrgEx( HDC hdc,
          int xView, int yView, // новые координаты
          LPPOINT lpPoint); // старые координаты
```

Изменение масштаба:

```
BOOL SetWindowExtEx( HDC hdc, (логические единицы)
    int nxWinExt, int nyWinExt, //новые гор. и верт.
                                // размеры
    LPSIZE lpSize); //старые значения
BOOL SetViewportExtEx(HDC hdc, (физические единицы):
          int nxViewExt, int nyViewExt,
          LPSIZE lpSize);
```

Системы координат

Пример.

Создать систему координат с началом отсчета в левом нижнем углу окна. Ось x направить слева направо, а ось y - снизу вверх. Логические значения высоты и ширины изменять от 0 до 1000. Установить одинаковый масштаб по осям x и y.

```
SetMapMode(hdc, MM_ISOTROPIC);
SetWindowExtEx((hdc, 1000, 1000);
SetViewportExtEx(hdc, cxClient, -cyClient);
SetViewportOrgEx(hdc, 0, cyClient);
```

Спасибо за внимание!