Интерфейс графического устройства (Graphics Device Interface, **GDI**) – это часть **Windows** **API**, которая представляет собой библиотеку функций, обеспечивающих графический вывод на различные устрой-ства отображения.

С целью обеспечения возможности вывода на устройство (экран, принтер, память) одновременно от множества источников (процессов/приложений), в Windows был разработан специализированный механизм под названием **контекст устройства** (device context, **DC**) и подмножество обеспечивающих его работу функций. Указанные функции определяют размер клиентской области, шрифт, цвета и другие GDI-атрибуты и возвращают (предоставляют) приложению так называемый дескриптор контекста устройства (Device Context Descriptor).

В операционной системе Windows существует несколько типов контекстов устройства, а именно:

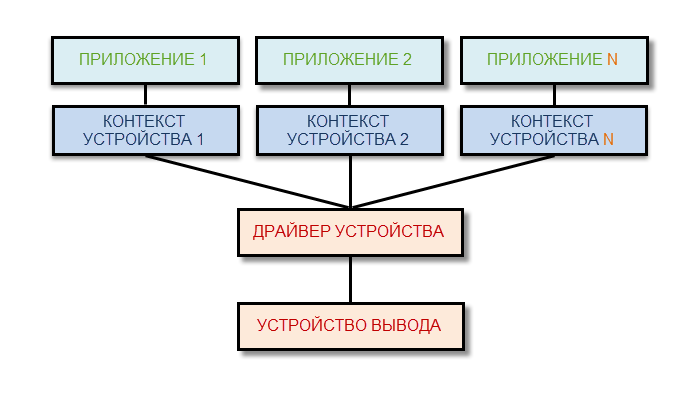
* **Контекст устройства отображения** (display);
* Контекст устройства печати (printer);
* Контекст устройства памяти (memory, CompatibleDC);
* Контекст информационного устройства (information);

Все виды контекстов устройств имеют идентичную структуру и содержат одни и те же атрибуты (характеристики).

Контекст устройства отображения - внутренняя структура данных ядра Windows, определяющая набор графических объектов и их атрибутов (режим отображения, начало координат, цвет пера, цвет кисти, тип шрифта, цвет/режим текста/фона, режим вывода/закрашивания и некоторые другие), используемая прикладной программой/системой для задания критериев вывода графических объектов (текст/графика) на экран видеоадаптера.

В GDI все операции рисования должны выполняться через контекст устройства [отображения].

Дескриптор (описатель) контекста отображения, возвращаемый функциями WinAPI, это своеобразный указатель, предоставляемый системой пользовательскому уровню приложения, посредством которого приложение получает доступ к устройству вывода (например, экран).



Каждое приложение в случае необходимости экранного вывода получает в свое распоряжение собственный (уникальный) контекст и осуществляет посредством него вывод изображения. Поэтому, более интуитивно дескриптор контекста устройства отображения воспринимается в качестве указателя на некую системную структуру в памяти ядра, через параметры которой системе задаются критерии вывода данных на физическое устройство отображения.

**Структура контекста устройства**

Надо учитывать, что контекст устройства является структурой, состоящей, в свою очередь, из описателей объектов графических примитивов (характеристик), используемых при работе с контекстом:

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут контекста | Значение по умолчанию | Функция для изменения | Функция для получения |
| Режим отображения (Mapping mode) | MM\_TEXT | SetMapMode | GetMapMode |
| Начало координат окна (Window origin) | (0,0) | SetWindowOrgEx | GetWindowOrgEx |
| Начало координат области вывода (Viewport Origin) | (0,0) | SetViewportOrgEx | GetViewportOrgEx |
| Протяженность окна (Window extent) | (1,1) | SetWindowExtEx | GetWindowExtEx |
| Протяженность области вывода (Viewport extent) | (1,1) | SetViewportExtEx, SetMapMode | GetViewportExtEx |
| Перо (Pen) | BLACK\_PEN | SelectObject | SelectObject |
| Кисть (Brush) | WHITE\_BRUSH | SelectObject | SelectObject |
| Шрифт (Font) | SYSTEM\_FONT | SelectObject | SelectObject |
| Битовый образ (Bitmap) | NOT | SelectObject | SelectObject |
| Текущая позиция пера (Current pen position) | (0,0) | MoveToEx, LineTo, PolylineTo, PolyBezierTo | GetCurrentPositionEx |
| Режим фона (Background mode) | OPAQUE | SetBkMode | GetBkMode |
| Цвет фона (Background color) | Обычно Белый (по Панели управления) | SetBkColor | GetBkColor |
| Цвет текста (TextColor) | Обычно Черный (по Панели управления) | SetTextColor | GetTextColor |
| Режим рисования (Drawing mode) | R2\_COPYPEN | SetROP2 | GetROP2 |
| Режим растяжения (Stretching mode) | BLACKONWHITE | SetStrethBltMode | GetStrethBltMode |
| Режим закрашивания многоугольников (Polygon filling mode) | ALTERNATE | SetPolyFillMode | GetPolyFillMode |
| Межсимвольный интервал (Intercharacter spacing) | 0 | SetTextCharacterExtra | GetTextCharacterExtra |
| Начало координат кисти (Brush origin) | (0,0) | SetBrushOrgEx | GetBrushOrgEx |
| Область отсечения (Clipping region) | Not. Окно (рабочая область) с обновляемым регионом обрезается. | SelectObject, SelectClipRgn | GetClipBox |

Каждый DC имеет текущие параметры: кисть, перо, шрифт и так далее, то есть текущий (рабочий) объект определенного типа.

Вы имеете возможность менять указанные атрибуты после того, как получили дескриптор контекста устройства отображения.

Эти действия выполняются через WinAPI, поскольку напрямую элементы структуры контекста устройства для изменения не доступны.

Контекст устройства отображения является критическим для системы ресурсом. Поэтому его необходимо освободить сразу по окончании использования.

### Рисование точки

Функция рисования точки SetPixel устанавливает цвет точки с заданными координатами:

COLORREF WINAPI SetPixel(

HDC hdc, // контекст отображения

int nXPos, // x-координата точки

int nYPos, // y-координата точки

COLORREF clrref); // цвет точки

Cконструировать цвет в формате COLORREF из отдельных компонент красного (r), зеленого (g) и голубого (b) цвета можно командой:

#define RGB(r,g,b) \

((COLORREF)(((BYTE)(r)|((WORD)(g)<<8)) | \

(((DWORD)(BYTE)(b))<<16)))

Например

SetPixel(hdc, 0, 0, RGB(0xff, 0, 0));

#### Рисование прямой линии

BOOL WINAPI LineTo(HDC hdc, int xEnd, int yEnd);

Эта функция рисует линию из текущей позиции пера, установленной ранее функцией MoveToEx, в точку с координатами (xEnd,yEnd). После того как линия будет нарисована, текущая позиция пера станет равной (xEnd,yEnd).

Функция LineTo возвращает TRUE при нормальном завершении или FALSE при ошибке.

Особенностью функции LineTo является то, что она немного не дорисовывает линию - эта функция рисует всю линию, не включая ее конец, т. е. точку (xEnd,yEnd).

#### Рисование ломаной линии

Функции Polyline , предназначенной для рисования ломаных линий, следует передать идентификатор контекста отображения hdc, указатель lppt на массив структур POINT, в котором должны находится координаты начала ломаной линии, координаты точек излома и координаты конца ломаной линии, а также размер этого массива cPoints:

BOOL WINAPI Polyline(

HDC hdc, // идентификатор контекста отображения

const POINT FAR\* lppt,// указатель на массив структур POINT

int cPoints); // размер массива

#### Рисование дуги эллипса

BOOL WINAPI Arc(

HDC hdc, // идентификатор контекста отображения

int nxLeft, int nyTop, // верхий левый угол

int nxRight, int nyBottom, // правый нижний угол

int nxStart, int nyStart, // начало дуги

int nxEnd, int nyEnd); // конец дуги

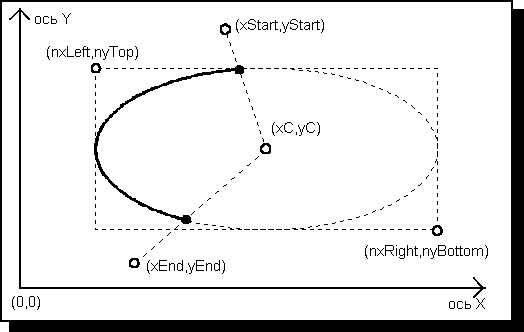


Рис. 2.13. Рисование дуги эллипса

Дуга рисуется в направлении против часовой стрелки.

Координаты центра эллипса (если это потребуется) можно вычислить следующим образом:

xC = (nxLeft + nxRight) / 2;

yC = (nyTop + nyBottom) / 2;

### Настройка атрибутов контекста отображения для рисования линий

Как это нетрудно заметить, функции, предназначенные для рисования линий, не имеют никаких параметров, определяющих толщину, цвет и стиль линии. Эти, а также другие характеристики (например, режим прозрачности), выбираются при установке соответствующих атрибутов контекста отображения.

#### Выбор пера

Для рисования линий приложения Windows могут выбрать одно из трех встроенных перьев, либо создать собственное перо.

Для выбора встроенного пера лучше всего воспользоваться макрокомандами GetStockPen и SelectPen , определенными в файле windowsx.h:

#define GetStockPen(i) ((HPEN)GetStockObject(i))

#define SelectPen(hdc, hpen) \

((HPEN)SelectObject((hdc), (HGDIOBJ)(HPEN)(hpen)))

Макрокоманда GetStockPen возвращает идентификатор встроенного пера, заданного параметром i. Вы можете выбрать для этого параметра одно из следующих значений:

|  |  |
| --- | --- |
| Значение | Описание |
| BLACK\_PEN | Перо, рисующее черную линию толщиной в один пиксел (для любого режима отображения). Это перо выбрано в контекст отображения по умолчанию |
| WHITE\_PEN | Перо белого цвета. Толщина пера также равна одному пикселу и не зависит от режима отображения |
| NULL\_PEN | Невидимое перо толщиной в один пиксел. Используется для рисования замкнутых закрашенных фигур (таких, как прямоугольник или эллипс) в тех случаях, когда контур фигуры должен быть невидимым |

После получения идентификатора пера его необходимо выбрать в контекст отображения при помощи макрокоманды SelectPen. Первый параметр этой макрокоманды используется для указания идентификатора контекста отображения, в который нужно выбрать перо, второй - для передачи идентификатора пера.

Макрокоманда SelectPen возвращает идентификатор пера, который был выбран в контекст отображения раньше. Вы можете сохранить этот идентификатор и использовать его для восстановления старого пера.

.

Если вас не устраивают встроенные перья, вы можете легко создать собственные. Для этого нужно воспользоваться функциями CreatePen или CreatePenIndirect.

Функция CreatePen позволяет определить стиль, ширину и цвет пера:

HPEN WINAPI CreatePen(

int fnPenStyle, // стиль пера

int nWidth, // ширина пера

COLORREF clrref); // цвет пера

Параметр fnPenStyle определяет стиль линии и может принимать одно из следующих значений, определенных в файле windows.h:

|  |  |  |
| --- | --- | --- |
| Стиль линии | Внешний вид | Описание |
| PS\_SOLID | **https://frolov-lib.ru/books/bsp/v14/img00021.gif** | Сплошная |
| PS\_DASH | **https://frolov-lib.ru/books/bsp/v14/img00022.gif** | Штриховая |
| PS\_DOT | **https://frolov-lib.ru/books/bsp/v14/img00023.gif** | Пунктирная |
| PS\_DASHDOT | **https://frolov-lib.ru/books/bsp/v14/img00024.gif** | Штрих-пунктирная, одна точка на одну линию |
| PS\_DASHDOTDOT | **https://frolov-lib.ru/books/bsp/v14/img00025.gif** | Штрих-пунктирная, две точки на одну линию |
| PS\_NULL |  | Невидимая |
| PS\_INSIDEFRAME | **https://frolov-lib.ru/books/bsp/v14/img00026.gif** | Линия, предназначенная для обводки замкнутых фигур |

Параметр nWidth определяет ширину пера. Используемая при этом единица длины зависит от режима отображения, поэтому вы можете задавать ширину пера не только в пикселах, но и в долях миллиметра или дюйма. Учтите, что для линий PS\_DASH, PS\_DOT, PS\_DASHDOT, PS\_DASHDOTDOT можно использовать только единичную или нулевую ширину, в обоих случаях ширина линии будет равна одному пикселу.

Если перо больше не нужно, его нужно удалить для освобождения памяти.

Прежде чем удалять созданное вами перо, следует выбрать в контекст отображения одно из встроенных перьев (например то, которое использовалось раньше). После этого для удаления вашего пера нужно вызвать макрокоманду DeleletePen , определенную в файле windowsx.h:

#define DeletePen(hpen) DeleteObject((HGDIOBJ)(HPEN)(hpen))

#### Выбор режима фона

Режим фона влияет на заполнение промежутков между штрихами и точками в штрих-пунктирных, штриховых и пунктирных линиях.

Напомним, что по умолчанию в контексте отображения установлен непрозрачный режим фона OPAQUE . В этом режиме промежутки закрашиваются цветом фона, определенным как атрибут контекста отображения. Приложение может установить прозрачный режим фона TRANSPARENT , в этом случае промежутки в линиях не будут закрашиваться (рис. 2.16).

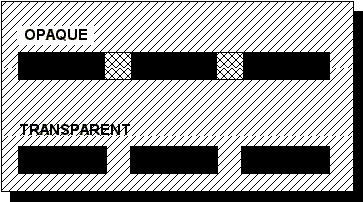


Рис. 2.16. Режимы фона OPAQUE и TRANSPARENT

Для установки режима фона предназначена функция SetBkMode :

int WINAPI SetBkMode(HDC hdc, int fnBkMode);

Эта функция устанавливает новый режим фона fnBkMode для контекста отображения hdc, возвращая в случае успеха код старого режима фона.

Для параметра fnBkMode вы можете использовать значения OPAQUE или TRANSPARENT, определенные в файле windows.h.

Приложение может определить текущий режим фона, вызвав функцию GetBkMode :

int WINAPI GetBkMode(HDC hdc);

С помощью функций SetBkColor и GetBkColor вы можете, соответственно, установить и определить текущий цвет фона, который используется для закраски промежутков между штрихами и точками линий:

COLORREF WINAPI SetBkColor(HDC hdc, COLORREF clrref);

COLORREF WINAPI GetBkColor(HDC hdc);

#### Выбор режима рисования

Для выбора режима рисования предназначена функция SetROP2 :

int WINAPI SetROP2(HDC hdc, int fnDrawMode);

Процесс рисования на экране монитора заключается в выполнении логической операции над цветами точек экрана и цветами изображения. Ниже в таблице мы привели возможные значения для параметра fnDrawMode. Для каждого режима рисования в этой таблице есть формула, с использованием которой вычисляется результат, и краткое описание режима рисования. В формулах цвет пера обозначается буквой P, цвет подложки - D.

|  |  |  |
| --- | --- | --- |
| Режим рисования | Формула | Цвет пиксела |
| R2\_COPYPEN | P | Соответствует (равен) цвету пера |
| R2\_BLACK | 0 | Черный |
| R2\_WHITE | 1 | Белый |
| R2\_NOP | D | Не меняется, т. е. перо ничего не рисует |
| R2\_NOT | ~D | Получается инвертированием цвета подложки, т. е. цвета пиксела до рисования |
| R2\_NOTCOPYPEN | ~P | Получается инвертированием цвета пера |
| R2\_MASKPEN | P&D | Комбинация компонент цветов, имеющихся как в цвете подложки, так и в цвете пера |
| R2\_NOTMASKPEN | ~(P&D) | Инверсия предыдущего значения |
| R2\_MERGEPEN | P|D | Комбинация компонент цветов подложки и пера |
| R2\_NOTMERGEPEN | ~(P|D) | Инверсия предыдущего значения |
| R2\_XORPEN | P^D | При определении цвета пиксела выполняется операция "ИСКЛЮЧАЮЩЕЕ ИЛИ" между компонентами цвета подложки и пера |
| R2\_NOTXORPEN | ~(P^D) | Инверсия предыдущего значения |
| R2\_MASKNOTPEN | ~P & D | Комбинация цвета подложки и инверсии цвета пера |
| R2\_MASKPENNOT | P & ~D | Комбинация двух цветов: инверсии цвета подложки и цвета пера |
| R2\_MERGENOTPEN | ~P | D | Комбинация компонент цветов подложки и инверсии цвета пера |
| R2\_MERGEPENNOT | P | ~D | Комбинация инверсии цвета подложки и цвета пера |

Если изображение и перо черно-белые, результат выполнения описанных выше операций (которые, кстати, называются растровыми операциями ) можно легко предсказать.

В режиме R2\_COPYPEN, который установлен в контексте отображения по умолчанию, цвет нарисованной линии будет такой же, как и цвет пера.

**Закрашивание прямоугольника**

Функция FillRect закрашивает прямоугольную область окна заданной кистью:

int WINAPI FillRect(

HDC hdc, // идентификатор контекста отображения

const RECT FAR\* lprc, // указатель на структуру RECT

HBRUSH hbrush); // идентификатор кисти для закрашивания

Параметр lprc должен указывать на структуру типа RECT, в которую следует записать координаты закрашиваемой прямоугольной области. Правая и нижняя граница указанной области не закрашивается.

Независимо от того, какая кисть выбрана в контекст отображения, функция FillRect будет использовать для закраски кисть, указанную параметром hbrush.

Учтите, что правильная работа функции FillRect гарантируется только в том случае, когда значение поля bottom структуры RECT больше значения поля top, а значение поля right больше значения поля left.

Для закрашивания границы прямоугольной области (т. е. для рисования прямоугольной рамки) можно использовать функцию FrameRect :

int WINAPI FrameRect(

HDC hdc, // идентификатор контекста отображения

const RECT FAR\* lprc, // указатель на структуру RECT

HBRUSH hbrush); // идентификатор кисти для закрашивания

Параметры этой функции аналогичны параметрам функции FillRect.

Ширина пера, используемого для рисования рамки, всегда равна одной логической единице. Структура RECT должна быть подготовлена таким же образом, что и для функции FillRect, т. е. значение поля bottom структуры RECT должно быть больше значения поля top, а значение поля right - больше значения поля left.

Значение, возвращаемое функциями FillRect и FrameRect не используется, приложения должны его игнорировать.

#### Рисование эллипса

Для рисования эллипса вы можете использовать функцию Ellipse :

BOOL WINAPI Ellipse(

HDC hdc, // идентификатор контекста отображения

int nxTL, // координата x верхнего левого угла

int nyTL, // координата y верхнего левого угла

int nxBR, // координата x правого нижнего угла

int nyBR); // координата y правого нижнего угла

Первый параметр этой функции указывает идентификатор контекста отображения, остальные - координаты верхнего левого и правого нижнего углов прямоугольника, в который должен быть вписан эллипс (рис. 2.19).

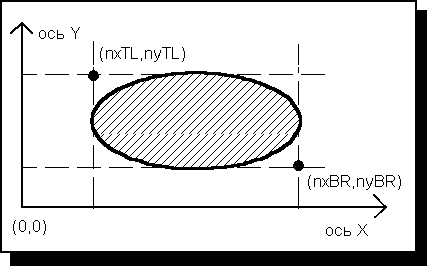


Рис. 2.19. Рисование эллипса

Рис. 2.21. Рисование сектора эллипса

#### Рисование многоугольников

Рисование многоугольников (рис. 2.22) выполняется функцией Polygon , аналогичной по своим параметрам функции Polyline, с помощью которой рисуются ломаные линии:

BOOL WINAPI Polygon(

HDC hdc, // идентификатор контекста отображения

const POINT FAR\* lppt,// указатель на массив структур POINT

int cPoints); // размер массива

Через параметр hdc передается идентификатор контекста отображения.

Параметр lppt указывает на массив структур POINT, в котором должны находится координаты вершин многоугольника. Параметр cPoints определяет размер этого массива.

Функция Polygon возвращает TRUE при нормальном завершении или FALSE при ошибке.

### Выбор кисти

Для закрашивания внутренней области замкнутых фигур вы можете использовать встроенные кисти, или кисти, созданные вашим приложением. Последние необходимо удалять после использования.

#### Использование встроенной кисти

Для выбора одной из встроенной кисти вы можете воспользоваться макрокомандой GetStockBrush , определенной в файле windowsx.h:

#define GetStockBrush(i) ((HBRUSH)GetStockObject(i))

В качестве параметра для этой макрокоманды можно использовать следующие значения:

|  |  |
| --- | --- |
| Значение | Описание |
| BLACK\_BRUSH | Кисть черного цвета |
| WHITE\_BRUSH | Кисть белого цвета |
| GRAY\_BRUSH | Серая кисть |
| LTGRAY\_BRUSH | Светло-серая кисть |
| DKGRAY\_BRUSH | Темно-серая кисть |
| NULL\_BRUSH | Бесцветная кисть, которая ничего не закрашивает |
| HOLLOW\_BRUSH | Синоним для NULL\_BRUSH |

Как видно из только что приведенной таблицы, в Windows есть только монохромные встроенные кисти.

Макрокоманда GetStockBrush возвращает идентификатор встроенной кисти.

Прежде чем использовать полученную таким образом кисть, ее надо выбрать в контекст отображения (так же, как и перо). Для этого проще всего воспользоваться макрокомандой SelectBrush :

#define SelectBrush(hdc, hbr) \

((HBRUSH)SelectObject((hdc), (HGDIOBJ)(HBRUSH)(hbr)))

Макрокоманда SelectBrush возвращает идентификатор старой кисти, выбранной в контекст отображения раньше.

#### Создание кисти

Если вам нужна цветная кисть, ее следует создать с помощью функции CreateSolidBrush :

HBRUSH WINAPI CreateSolidBrush(COLORREF clrref);

После использования созданной вами кисти ее следует удалить, не забыв перед этим выбрать в контекст отображения старую кисть. Для удаления кисти следует использовать макрокоманду DeleteBrush :

#define DeleteBrush(hbr) DeleteObject((HGDIOBJ)(HBRUSH)(hbr))

Приложение может заштриховать внутреннюю область замкнутой фигуры, создав одну из шести кистей штриховки функцией CreateHatchBrush :

HBRUSH WINAPI CreateHatchBrush(int fnStyle, COLORREF clrref);

С помощью параметра clrref вы можете определить цвет линий штриховки.

Параметр fnStyle задает стиль штриховки:

|  |  |
| --- | --- |
| Стиль штриховки | Внешний вид |
| HS\_BDIAGONAL | **https://frolov-lib.ru/books/bsp/v14/img00036.gif** |
| HS\_CROSS | **https://frolov-lib.ru/books/bsp/v14/img00037.gif** |
| HS\_DIAGCROSS | **https://frolov-lib.ru/books/bsp/v14/img00038.gif** |
| HS\_FDIAGONAL | **https://frolov-lib.ru/books/bsp/v14/img00039.gif** |
| HS\_HORIZONTAL | **https://frolov-lib.ru/books/bsp/v14/img00040.gif** |
| HS\_VERTICAL | **https://frolov-lib.ru/books/bsp/v14/img00041.gif** |

Вы можете использовать свой собственный стиль штриховки, создав кисть из битового изображения размером 8х8 пикселов (можно использовать только такой размер).

Если битовое изображение кисти определено в ресурсах приложения, его следует загрузить при помощи функции LoadBitmap . Эта функция возвратит идентификатор битового изображения. Затем для создания кисти этот идентификатор следует передать в качестве параметра функции CreatePatternBrush :

HBRUSH WINAPI CreatePatternBrush(HBITMAP hBitmap);

**Рисование прямоугольника**

Функция **Rectangle** вычерчивает прямугольник используя текущие значения пера и кисти

BOOL Rectangle(

[in] HDC hdc,

[in] int left,

[in] int top,

[in] int right,

[in] int bottom

);

**Получение дескриптора окна**

Функция **GetConsoleWindow** извлекает дескриптор окна, используемый консолью, связанной с вызывающим процессом

HWND hwnd = GetConsoleWindow();

**Получение ссылки на контекст устройства**

Функция **GetDC** извлекает дескриптор в контекст устройства (DC) для клиентской области указанного окна

HDC hdc = GetDC(hwnd);

**Освобождение контекста устройства**

Функция **ReleaseDC** освобождает контекст устройства (DC), освобождая его для использования другими приложениями.

ReleaseDC(hwnd, hdc);

**x=LOWORD(lParam);**

**y=HIWORD(lParam);**

<https://learn.microsoft.com/ru-ru/windows/win32/gdi/windows-gdi>

https://frolov-lib.ru/books/bsp/v14/ch2\_1.htm?ysclid=m7n927omf2419863666