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
```

Параметр hdc определяет контекст отображения, для которого необходимо изменить цвет точки. Параметры nXPos и nYPos определяют координаты точки в системе координат, которая зависит от установленного для контекста hdc режима отображения. Последний параметр clrref определяет новый цвет точки. В файле windows.h есть описание макрокоманды RGB , позволяющей сконструировать цвет в формате COLORREF из отдельных компонент красного (r), зеленого (g) и голубого (b) цвета:

```
#define RGB(r,g,b) \
  ((COLORREF)(((BYTE)(r)|((WORD)(g)<<8)) | \
  (((DWORD)(BYTE)(b))<<16)))
```

Вы можете использовать эту макрокоманду совместно с функцией SetPixel для установки, например, красного цвета точки, расположенной в начале системы координат (0,0), следующим образом:

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

Функция GetPixel позволяет узнать цвет точки, заданной идентификатором контекста отображения и координатами:

```
COLORREF WINAPI GetPixel(HDC hdc, int nXPos, int nYPos);
```

Рисование линий
Текущая позиция пера

Для рисования прямых линий (и только для этого) в контексте отображения хранятся координаты текущей позиции пера . Для изменения текущей позиции пера в Windows следует использовать функцию MoveToEx:

```
BOOL WINAPI MoveToEx(
  HDC hdc,        // идентификатор контекста отображения
  int x,          // x-координата
  int y,          // y-координата
  POINT FAR* lppt); // указатель на структуру POINT
```

Для контекста отображения hdc эта функция устанавливает текущую позицию пера, равную (x,y). В структуру типа POINT, на которую указывает параметр lppt, после возврата из функции будут записаны старые координаты пера.

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

Пример: MoveToEx(hdc, 100, 1300, NULL);

Чтобы узнать текущую позицию пера, приложение может использовать функцию GetCurrentPositionEx :

```
BOOL WINAPI GetCurrentPositionEx(HDC hdc, POINT FAR* lppt);
```

После вызова этой функции текущая позиция пера будет записана в структуру типа POINT, на которую указывает параметр lppt. Функция GetCurrentPositionEx возвращает TRUE при нормальном завершении или FALSE при ошибке.

Рисование прямой линии

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

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

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). Это иллюстрируется на рис. 2.11.

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
```

Функция Polyline возвращает TRUE при нормальном завершении или FALSE при ошибке. Она не использует текущую позицию пера и не изменяет ее. Если ломаная линия не замкнута, ее последняя точка не рисуется.

Пример: // Массив координат точек излома

POINT ptP[]={{350,10},{400,60},{300,60},{350,10}};
Polyline(hdc, ptP,4);

Рисование дуги эллипса

Функция Arc позволяет нарисовать дугу эллипса или окружности:

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

Первый параметр этой функции определяет контекст отображения, в котором будет нарисована дуга. Параметры (nxLeft,nyTop) и (nxRight,nyBottom) задают координаты, соответственно, верхнего левого и правого нижнего углов воображаемого прямоугольника, в который вписан эллипс. Начало дуги эллипса определяется пересечением эллипса с воображаемой прямой линией, проведенной из центра эллипса (xC,yC) в точку (xStart,yStart). Конец дуги определяется аналогично - как пересечение эллипса с воображаемой прямой линии, проведенной из центра эллипса в точку (xEnd,yEnd). Дуга рисуется в направлении против часовой стрелки.

Настройка атрибутов контекста отображения для рисования линий
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)))

Описание:

HGDIOBJ SelectObject(HDC hdc, // handle of device context HGDIOBJ hgdiobj // handle of object);
Выбирает логический объект для DC. В каждый момент времени может быть выбран только один объект, который должен удаляться сразу же, как только перестает использоваться.

Параметры:

DC: Идентификатор контекста устройства.
hObject: Карта бит, кисть, шрифт, перо или область.

Возвращаемое значение:

Заменяемый объект или не нуль, если DC метафайла, или 0, если ошибка.

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

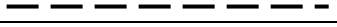
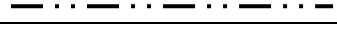
Значение	Описание
BLACK_PEN	Перо, рисующее черную линию толщиной в один пиксел (для любого режима отображения). Это перо выбрано в контекст отображения по умолчанию
WHITE_PEN	Перо белого цвета. Толщина пера также равна одному пикселу и не зависит от режима отображения
NULL_PEN	Невидимое перо толщиной 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.
Для создания собственного пера 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		Сплошная
PS_DASH		Штриховая
PS_DOT		Пунктирная
PS_DASHDOT		Штрих-пунктирная, одна точка на одну линию
PS_DASHDOTDOT		Штрих-пунктирная, две точки на одну линию
PS_NULL		Невидимая
PS_INSIDEFRAME		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. Поэтому вы не можете создать перо для рисования, например, пунктирной линии шириной 5 миллиметров.
Параметр clrref задает цвет пера. На первый взгляд линии PS_SOLID и PS_INSIDEFRAME похожи, однако между ними имеются различия, особенно заметные для широких линий. Широкая линия, имеющая стиль PS_SOLID, располагается по обе стороны оси, заданной координатами линии. Линии, имеющие стиль PS_INSIDEFRAME, располагаются внутри контура, определяющего размеры замкнутой фигуры.
Небольшое замечание относительно концов толстых линий. Концы толстых линий закруглены.
Другая возможность создать перо - вызвать функцию CreatePenIndirect :

HPEN WINAPI CreatePenIndirect(LOGPEN FAR* lplgpn);

Эта функция работает аналогично функции CreatePen, однако в качестве параметра ей необходимо передать указатель на структуру типа LOGPEN, в которой должны находиться характеристики создаваемого пера.
Структура LOGPEN и различные указатели на нее определены в файле windows.h:

```
typedef struct tagLOGPEN
{
  UINT    lopnStyle;   // стиль пера
  POINT   lopnWidth;   // ширина пера
  COLORREF lopnColor;  // цвет пера
} LOGPEN;
typedef LOGPEN*       PLOGPEN;
typedef LOGPEN NEAR* NPLOGPEN;
typedef LOGPEN FAR*  LPLOGPEN;
```

Заметим, что ширина пера в данном случае находится в поле x структуры POINT. Поле y не используется.
Если вы создали перо, его можно выбрать в контекст отображения при помощи макрокоманды SelectPen. После этого можно рисовать линии обычным образом, вызывая функции MoveToEx и LineTo.
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))

В качестве параметра этой макрокоманде необходимо передать идентификатор удаляемого пера.

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.
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);

Параметр hdc предназначен для указания контекста отображения, в котором необходимо установить новый режим рисования, определяемый параметром fnDrawMode. Функция SetROP2 возвращает код предыдущего режима рисования.

Процесс рисования на экране монитора заключается в выполнении логической операции над цветами точек экрана и цветами изображения. В таблице приведены 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

В режиме R2_COPYPEN, который установлен в контексте отображения по умолчанию, цвет нарисованной линии будет такой же, как и цвет пера. Для режимов R2_BLACK и R2_WHITE цвет линии будет, соответственно, черный и белый. В режиме R2_NOP вы не увидите нарисованную линию, так как цвет вдоль нее вообще не изменится. Более интересен режим R2_NOT, при использовании которого на черном фоне будет нарисована белая линия, а на белом фоне - черная.

С помощью функции GetROP2 приложение может определить режим рисования, установленный для контекста отображения hdc:

int WINAPI GetROP2(HDC hdc);

Рисование линий произвольного стиля

LineDDA – позволяет рисовать любые линии. Функция LineDDA имеет следующий прототип:

```
void WINAPI LineDDA(  int nxStart, int nyStart, // начальная точка
  int nxEnd, int nyEnd,     // конечная точка
  LINEDDAPROC lnddaprc,     // адрес функции для рисования
  LPARAM lParam);           // дополнительные параметры
```

Первые четыре параметра этой функции определяют координаты начальной и конечной точки, между которыми надо нарисовать линию. Через параметр lnddaprc передается указатель на функцию рисования, которая является функцией обратного вызова, определяемой программистом. Эта функция получает управление много раз, она вызывается для каждой точки рисуемой линии.

Для режима STRICT тип LINEDDAPROC определен в файле windows.h следующим образом:

typedef void (CALLBACK* LINEDDAPROC)(int, int, LPARAM);

Последний параметр предназначен для передачи дополнительных данных в функцию рисования.

Приведем прототип функции рисования (для функции можно использовать любое имя):

void CALLBACK _export LineProc(int xPos, int yPos, LPARAM lParam);

Первые два параметра представляют собой координаты точки, для рисования которых вызвана функция. Последний параметр соответствует последнему параметру функции LineDDA и содержит передаваемое этой функции значение.

Рисование замкнутых фигур

Для закрашивания внутренней области замкнутых фигур используется кисть, задаваемая как атрибут контекста отображения. Внешний контур фигуры обводится пером, которое также выбирается в контекст отображения.

Рисование прямоугольника

Простейшая функция, с помощью которой можно нарисовать прямоугольник, называется Rectangle :

```
BOOL WINAPI Rectangle(
  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
```

Функция Rectangle рисует прямоугольник для контекста отображения hdc, возвращая значение TRUE в случае успеха или FALSE при ошибке, последние четыре параметра функции задают координаты верхнего левого и нижнего правого угла прямоугольника.

В зависимости от стиля пера граница фигуры может находится полностью внутри прямоугольника, заданного координатами (nxTL, nyTL), (nxBR,nyBR) или выходить за его пределы. Если выбрать стиль пера PS_NULL, граница фигуры станет невидимой. В зависимости от кисти, выбранной в контекст отображения, внутренность прямоугольника может быть закрашенной в тот или иной цвет, заштрихована одним из нескольких способов или закрашена с помощью любого битового изображения размером 8x8 пикселов.

С помощью функции RoundRect можно нарисовать прямоугольник со скругленными углами.

По сравнению с функцией Rectangle функция RoundRect имеет два дополнительных параметра nxEllipse и nyEllipse, определяющих форму и радиус закругления:

```
BOOL WINAPI RoundRect(
  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
  int nxEllipse,  // ширина эллипса
  int nyEllipse); // высота эллипса
```

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.

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.

Используя функцию InvertRect , вы можете инвертировать содержимое прямоугольной области, заданной параметром lprc:

```
void WINAPI InvertRect(HDC hdc, const RECT FAR* lprc);
```

DrawFocusRect :

```
void WINAPI DrawFocusRect(HDC hdc, const RECT FAR* lprc);
```

Эта функция рисует прямоугольную рамку, предназначенную для выделения окна, имеющего фокус ввода.

Для рисования используется растровая операция "ИСКЛЮЧАЮЩЕЕ ИЛИ". Это приводит к тому, что для удаления нарисованной таким образом рамки ее достаточно нарисовать еще раз на том же месте. Для использования этой функции не нужно выбирать перо, рисующее пунктирную линию. Функция DrawFocusRect рисует пунктирную линию с нестандартным, очень близким расположением точек. Перед использованием этой функции необходимо установить режим отображения MM_TEXT. Первые две особенности позволяют

использовать ее для рисования рамки выделения произвольных участков изображения на экране монитора (при помощи мыши).

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.

Рисование сегмента эллипса

Сегмент эллипса можно нарисовать при помощи функции Chord :

```
BOOL WINAPI Chord(
  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
```

Параметры этой функции аналогичны параметрам рассмотренной нами ранее функции Arc.

Рисование сектора эллипса

Для рисования сектора эллипса следует использовать функцию Pie , аналогичную по своим параметрам функциям Arc и Chord:

```
BOOL WINAPI Pie(
  HDC hdc,   // идентификатор контекста отображения
  int nxLeft,  int nyTop,    // верхний левый угол
  int nxRight, int nyBottom, // правый нижний угол
  int nxStart, int nyStart,  // начало дуги
  int nxEnd,   int nyEnd);   // конец дуги
```

Рисование многоугольников

Рисование многоугольников 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. Она не использует текущую позицию пера и не изменяет ее.

В массиве структур POINT, определяющих вершины многоугольника, каждая вершина должна быть указана один раз. Функция Polygon автоматически замыкает ломаную линию, образующую многоугольник.

С помощью функции PolyPolygon можно нарисовать одновременно несколько многоугольников:

```
BOOL WINAPI PolyPolygon(
  HDC hdc,          // идентификатор контекста отображения
  const POINT FAR*lppt, // указатель на массив структур POINT
  int FAR* lpnPolyCounts, // адрес массива количества точек
                  //    в многоугольниках
  int cPolygons);   // количество многоугольников
```

Первый параметр hdc, как обычно, задает контекст отображения. Параметр cPolygons определяет количество многоугольников, которые нужно нарисовать. Параметр lppt должен содержать указатель на массив структур типа POINT, содержащий координаты вершин всех многоугольников. И, наконец, через параметр lpnPolyCounts передается указатель на массив целых чисел. Каждое число в этом массиве определяет количество точек в соответствующем многоугольнике. В отличие от функции Polygon, функция PolyPolygon не замыкает автоматически ломаную линию, образующую многоугольник.

В контексте отображения имеется атрибут, влияющий на способ закрашивания для самопересекающихся многоугольников. По умолчанию выбран режим ALTERNATE , в котором эти области не закрашиваются

(закрашиваются только те области, которые расположены между нечетными и четными сторонами многоугольника).
С помощью функции SetPolyFillMode вы можете изменить значение этого атрибута на WINDING . В этом режиме для того чтобы определить, надо ли закрашивать область многоугольника, учитывается направление, в котором был нарисован этот многоугольник. Каждая сторона многоугольника может быть нарисована в направлении либо по часовой стрелке, либо против часовой стрелки. Если воображаемая линия, нарисованная в направлении из внутренней области многоугольника в наружную, пересекает сегмент, нарисованный в направлении по часовой стрелке, содержимое некоторого внутреннего счетчика увеличивается на единицу. Если же эта линия пересекает сегмент, нарисованный против часовой стрелки, содержимое счетчика уменьшается на единицу. Область закрашивается только в том случае, если содержимое счетчика не равно нулю.
Приведем прототип функции SetPolyFillMode:
int WINAPI SetPolyFillMode(HDC hdc, int fnMode);
Параметр fnMode, определяющий режим закрашивания многоугольников, может принимать значения ALTERNATE или WINDING. Функция возвращает код старого режима закрашивания.
Вы можете определить используемый в данный момент режим закрашивания многоугольников с помощью функции GetPolyFillMode :
int WINAPI GetPolyFillMode(HDC hdc);

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

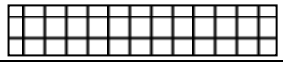
Для выбора одной из встроенной кисти вы можете воспользоваться макрокомандой GetStockBrush , определенной в файле windowsx.h:
#define GetStockBrush(i) ((HBRUSH)GetStockObject(i))
В качестве параметра для этой макрокоманды можно использовать следующие значения:

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

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);
В качестве параметра для этой функции необходимо указать цвет кисти. Для выбора цвета вы можете воспользоваться, например, макрокомандой RGB, позволяющей указать содержание отдельных цветовых компонент.
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	
HS_CROSS	


HS_DIAGCROSS	
HS_FDIAGONAL	
HS_HORIZONTAL	
HS_VERTICAL	

Закрашивание внутренней области окна

Напомним, что кисть можно использовать еще и для закрашивания внутренней области окна . Для этого идентификатор кисти следует записать в поле hbrBackground структуры типа WNDCLASS перед регистрацией класса окна:

wc.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);

Установка начальных координат кисти

Начальные координаты кисти (brush origin) - это атрибут контекста отображения. Он используются для определения координат точки внутри кисти, которая будет служить начальной при закраске внутренней области фигуры или окна. По умолчанию используются координаты (0,0), соответствующие верхнему левому углу кисти (в системе координат, выбранной в контекст отображения по умолчанию).

Если кисть используется для закраски внутренней области окна, верхний левый угол изображения кисти совмещается с верхним левым углом этой области. Затем изображение кисти многократно повторяется с шагом 8 пикселов.

При закраске фигур начальное расположение кисти привязывается не к фигуре, а по-прежнему к верхнему левому углу внутренней области окна. Поэтому при закраске, например, прямоугольника, верхний левый угол кисти может не совпадать с верхним левым углом прямоугольника.

Приложение может изменить начальные координаты кисти (сдвинуть кисть) при помощи функций SetBrushOrg и UnrealizeObject.

Прежде всего нужно вызвать функцию UnrealizeObject , передав ей в качестве параметра идентификатор сдвигаемой кисти (только если это не встроенная кисть):

BOOL WINAPI UnrealizeObject(HGDIOBJ hbrush);

В этом случае система сбросит координаты кисти после выбора ее в контекст отображения. После сброса надо установить новые значения координат кисти, вызвав функцию SetBrushOrg :

DWORD WINAPI SetBrushOrg(HDC hdc, int nx, int ny);

Параметры nx и ny определяют новые значения для начальных координат кисти пикселах (от 0 до 7).

В завершении следует снова выбрать кисть в контекст отображения при помощи макрокоманды SelectBrush.

Области

В интерфейсе GDI есть средства, позволяющие приложениям создавать области достаточно сложной формы из прямоугольных, многоугольных и эллиптических областей. Такие области можно закрашивать или использовать в качестве маски при выводе графического изображения. В последнем случае область называется областью ограничения. Она должна быть выбрана в контекст отображения.

Создание области

Приложение может создать область прямоугольной формы, область в виде многоугольника, область эллиптической формы. Можно комбинировать область из двух других, выполняя при этом над областями логические операции объединения, пересечения и т. д.

Прямоугольная область

Для создания прямоугольной области предназначены функции CreateRectRgn и CreateRectRgnIndirect :

HRGN WINAPI CreateRectRgn(
int nLeftRect, int nTopRect,
int nRightRect, int nBottomRect);
HRGN WINAPI CreateRectRgnIndirect(const RECT FAR* lprc);

Обе эти функции создают область прямоугольной формы, размеры которой заданы координатами верхнего левого и правого нижнего углов (для функции CreateRectRgn) или при помощи структуры типа RECT (для функции CreateRectRgnIndirect). Функции возвращают идентификатор созданной области.

Область - это объект, принадлежащий GDI, поэтому его следует удалить после использования. Лучше всего для этого воспользоваться макрокомандой DeleteRgn , определенной в файле windowsx.h следующим образом:

#define DeleteRgn(hrgn) DeleteObject((HGDIOBJ)(HRGN)(hrgn))

Вы можете создать область в виде прямоугольника со скругленными углами, если воспользуетесь функцией CreateRoundRectRgn :

HRGN WINAPI CreateRoundRectRgn(
int nLeftRect, int nTopRect,

int nRightRect, int nBottomRect,
int nWidth, int nHeight);

Первые четыре параметра аналогичны параметрам функции CreateRectRgn. Они определяют координаты углов прямоугольной области. Последние два параметра, nWidth и nHeight, предназначены, соответственно, для определения размеров воображаемого эллипса, формирующего скругленные углы.

Область в виде многоугольника

Можно создать область в виде произвольного многоугольника. Для этого следует воспользоваться функцией CreatePolygonRgn :

HRGN WINAPI CreatePolygonRgn(
const POINT FAR* lppt, // адрес массива точек
int cPoints, // размер массива
int fnPolyFillMode); // режим заполнения

Функция CreatePolyPolygonRgn создает область, состоящую из нескольких многоугольников:

HRGN WINAPI CreatePolyPolygonRgn(
const POINT FAR* lppt, // адрес массива точек
int FAR* lpnPolyCounts, // адрес массива количества точек
// в многоугольниках
int cPolygons); // количество многоугольников

Назначение параметров описанных выше двух функций аналогично назначению параметров функций рисования многоугольников Polygon и PolyPolygon (за исключением того, что при создании области не требуется указывать идентификатор контекста отображения).

Область эллиптической формы

Область эллиптической формы (или, как частный случай, круглой формы) можно создать при помощи функции CreateEllipticRgn :

HRGN WINAPI CreateEllipticRgn(int nLeftRect, int nTopRect, int nRightRect, int nBottomRect);

Параметры этой функции описывают координаты воображаемого прямоугольника, в который вписана область эллиптической формы.

Функция CreateEllipticRgnIndirect также используется для создания области в форме эллипса:

HRGN WINAPI CreateEllipticRgnIndirect(const RECT FAR* lprc);

В отличие от функции CreateEllipticRgn координаты прямоугольника задаются с помощью структуры типа RECT, указатель на которую передается через параметр lprc.

Комбинирование областей

Функция CombineRegion позволяет вам изменить существующую область, скомбинировав ее из двух других:

int WINAPI CombineRgn(
HRGN hrgnDest, // новая область
HRGN hrgnSrc1, // первая исходная область
HRGN hrgnSrc2, // вторая исходная область
int fnCombineMode); // режим комбинирования

Перед вызовом функции вам надо определить все три области. Функция объединит области hrgnSrc1 и hrgnSrc2, изменив соответствующим образом область hrgnDest.

Способ комбинирования областей зависит от значения параметра fnCombineMode:

Значение параметра	Способ образования области hrgnDest
RGN_AND	Пересечение областей hrgnSrc1 и hrgnSrc2
RGN_OR	Объединение областей hrgnSrc1 и hrgnSrc2
RGN_XOR	Объединение областей hrgnSrc1 и hrgnSrc2 с исключением перекрывающихся областей
RGN_DIFF	Область hrgnSrc1, которая не входит в область hrgnSrc2
RGN_COPY	Область hrgnSrc1

В зависимости от результата выполнения операции функция CombineRegion может вернуть одно из следующих значений:

Значение	Описание
ERROR	Ошибка
NULLREGION	Новая область пустая
SIMPLEREGION	Новая область не является самопересекающейся (т. е. граница созданной области не пересекает саму себя)
COMPLEXREGION	Создана самопересекающаяся область

Для облегчения комбинирования областей в файле windowsx.h определены макрокоманды, предназначенные для копирования, пересечения, объединения и вычитания областей. Все они созданы на базе только что описанной функции CombineRegion :

***#define CopyRgn (hrgnDst, hrgnSrc) ***
CombineRgn(hrgnDst, hrgnSrc, 0, RGN_COPY)

***#define IntersectRgn (hrgnResult, hrgnA, hrgnB) ***
CombineRgn(hrgnResult, hrgnA, hrgnB, RGN_AND)

***#define SubtractRgn (hrgnResult, hrgnA, hrgnB) ***
CombineRgn(hrgnResult, hrgnA, hrgnB, RGN_DIFF)

***#define UnionRgn (hrgnResult, hrgnA, hrgnB) ***
CombineRgn(hrgnResult, hrgnA, hrgnB, RGN_OR)

***#define XorRgn (hrgnResult, hrgnA, hrgnB) ***
CombineRgn(hrgnResult, hrgnA, hrgnB, RGN_XOR)

Перерисовка области

Вы можете отметить область как требующую перерисовки, вызвав функцию InvalidateRgn . В результате этого приложению будет передано сообщение WM_PAINT .
Приведем прототип функции InvalidateRgn:
void WINAPI InvalidateRgn(HWND hwnd, HRGN hrgn, BOOL fErase);
Через параметр hwnd следует передать идентификатор окна, содержащего обновленную область hrgn.
Параметр fErase определяет необходимость стирания фона окна перед перерисовкой. Если этот параметр имеет значение TRUE, фон стирается, если FALSE - нет.
Если ваше приложение обновило содержимое области, но не во время обработки сообщения WM_PAINT, оно может удалить область из списка областей, подлежащих перерисовке, вызвав функцию ValidateRgn :
void WINAPI ValidateRgn(HWND hwnd, HRGN hrgn);
Другие операции
Для изменения прямоугольной области можно воспользоваться функцией SetRectRgn , изменяющей координаты существующей прямоугольной области:
void WINAPI SetRectRgn(HRGN hrgn, int nLeftRect, int nTopRect, int nRightRect, int nBottomRect);
Изменение области выполняется быстрее ее удаления с последующим повторный созданием новой.
Для перемещения области предназначена функция OffsetRgn :
int WINAPI OffsetRgn(HRGN hrgn, int nX, int nY);
Эта функция сдвигает область на nX логических единиц по горизонтали и на nY логических единиц по вертикали, возвращая такие же значения, что и функция CombineRegion.
Вы можете сравнить две области при помощи функции EqualRgn :
BOOL WINAPI EqualRgn(HRGN hrgn1, HRGN hrgn2);
Эта функция возвращает TRUE, если области совпадают, и FALSE - если нет.
С помощью функции GetRgnBox можно узнать координаты воображаемого прямоугольника, в который вписана область:
int WINAPI GetRgnBox(HRGN hrgn, RECT FAR* lprc);
Эта функция возвращает такие же значения, что и функция CombineRegion
Иногда требуется определить, находится ли заданная точка или прямоугольная область внутри другой области. Это можно сделать при помощи функций PtInRegion и RectInRegion .
Функция PtInRegion возвращает TRUE, если точка (nX,nY) находится внутри области hrgn:
BOOL WINAPI PtInRegion(HRGN hrgn, int nX, int nY);
Аналогично для прямоугольной области:
BOOL WINAPI RectInRegion(HRGN hrgn, const RECT FAR* lprc);
Закрашивание области
Для закрашивания области кистью, выбранной в контекст отображения, предназначена функция PaintRgn :
BOOL WINAPI PaintRgn(HDC hdc, HRGN hrgn);
Функция FillRgn также закрашивает область, но, в отличие от функции PaintRgn, эта функция использует кисть, идентификатор которой передается ей в качестве параметра hbrush:
BOOL WINAPI FillRgn(HDC hdc, HRGN hrgn, HBRUSH hbrush);
С помощью функции FrameRgn вы можете обвести заданную область (закрасить ее границу), используя кисть hbrush:
BOOL WINAPI FrameRgn(HDC hdc, HRGN hrgn, HBRUSH hbrush, int nWidth, int nHeight);
Параметры nWidth и nHeight определяют, соответственно, ширину и высоту кисти в пикселах, используемой для рисования границы.
Функция InvertRgn инвертирует цвета в указанной области:
BOOL WINAPI InvertRgn(HDC hdc, HRGN hrgn);
Все эти функции возвращают TRUE при успешном завершении или FALSE при ошибке.
Графический редактор битовых изображений Paint Brush, который поставляется вместе с операционной системой Windows, умеет закрашивать внутренние области замкнутых фигур. Для закраски вам достаточно, находясь в соответствующем режиме редактора, указать мышью любую точку внутри фигуры.
Для реализации описанной выше операции в программном интерфейсе GDI предусмотрены функции FloodFill и ExtFloodFill.

Для функции FloodFill необходимо указать идентификатор контекста отображения hdc, координаты точки nX и nY, а также цвет контура clrref, ограничивающего область:

BOOL WINAPI FloodFill(HDC hdc, int nX, int nY, COLORREF clrref);

Функция возвращает TRUE при успешном завершении или FALSE при ошибке (ошибка возникает в том случае, когда указанная точка имеет цвет clrref или она лежит вне области ограничения данного контекста отображения).

Для закраски используется кисть, выбранная в контекст отображения.

Функция ExtFloodFill аналогична функции FloodFill:

BOOL WINAPI ExtFloodFill(HDC hdc, int nX, int nY, COLORREF clrref, UINT fuFillType);

Эта функция имеет дополнительный параметр fuFillType, определяющий способ закраски области. Параметр может принимать значения FLOODFILLBORDER или FLOODFILLSURFACE . В первом случае закрашивается внутренняя область фигуры, ограниченная контуром, имеющим цвет clrref (как и при использовании функции FloodFill). Во втором случае закрашивается вся область, имеющая цвет clrref.

Функция ExtFloodFill возвращает TRUE при успешном завершении или FALSE при ошибке. Если значение параметра fuFillType равно FLOODFILLBORDER, ошибка может возникнуть из-за тех же причин, что и при выполнении функции FloodFill. Если же значение параметра fuFillType равно FLOODFILLSURFACE, ошибка может возникнуть из-за того, что цвет точки (nX,nY) не равен clrref.

Область ограничения

Приложение может использовать область для маски, ограничивающей вывод. Для этого область следует выбрать в контекст отображения функцией SelectClipRgn :

int WINAPI SelectClipRgn(HDC hdc, HRGN hrgn);

В качестве значения параметра hrgn вы можете использовать значение NULL. В этом случае для ограничения вывода будет использована внутренняя область окна.

Сохранение и восстановление контекста отображения

Для сохранения атрибутов контекста отображения следует использовать функцию SaveDC :

int WINAPI SaveDC(HDC hdc);

Значение, возвращаемое этой функцией, необходимо использовать в качестве параметра nSavedDC для функции RestoreDC , восстанавливающей атрибуты контекста отображения:

BOOL WINAPI RestoreDC(HDC hdc, int nSavedDC);

Функция RestoreDC возвращает значение TRUE при успешном завершении или FALSE при ошибке.

В качестве значения параметра nSavedDC можно использовать -1. В этом случае будет восстановлен контекст, сохраненный при последнем вызове функции SaveDC.