4 Графическая подсистема. GDI и GDI API

4.1 Основные концепции и сущности

Логическое графическое устройство (GD) – абстракция реального или виртуальное
Обеспечение унифицированного доступа для ввода, вывода, управления (аналогично прочим логическим устройствам). Использование драйверов (иерархии драйверов).

GDI – интерфейс графических устройств.

Объектный подход: набор системных объектов (GDI-объектов) и системных вызовов («методов») для работы с ними (GDI API).

Основные объекты:

– Device Context (DC) – «контекст» графического устройства, абстракция «поверхности рисования», соответствует логическому устройству, окну или его части, причем «поверхность рисования» может быть доступна для чтения и/или записи в зависимости от типа и особенностей этого устройства

– Pen, Brush, Font – основные инструменты для формирования изображения («перо», «кисть», «шрифт»)

– Bitmap, Metafile – содержимое контекста в растровом или векторном виде

– Region, Palette и др. – вспомогательные объекты, участвующие в формировании изображений

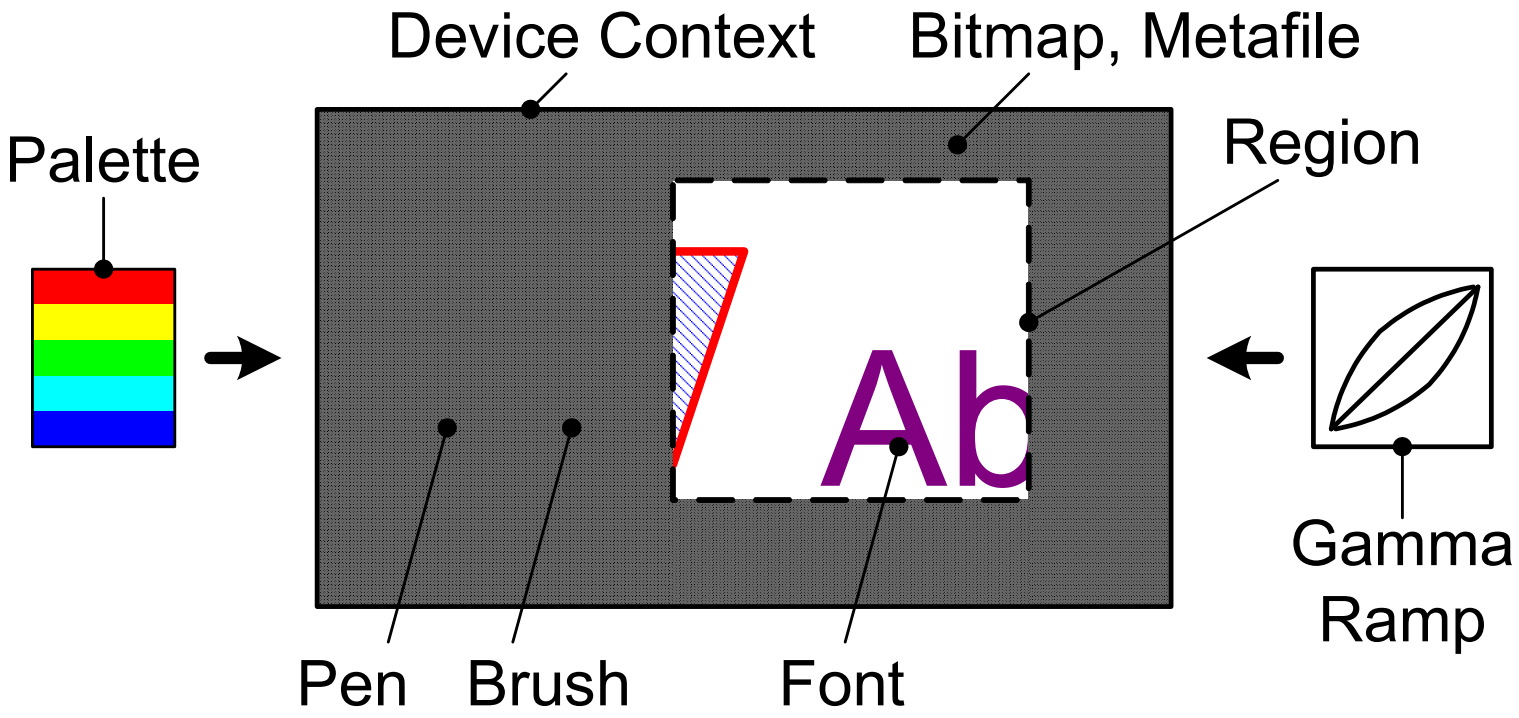

Рис. – Объекты графической подсистемы GDI

Идентификация всех объектов/инструментов – **`HANDLE`** или переопределение от него **`HGDIOBJ`**, а также **`HPEN`**, **`HBITMAP`**, **`HPALETTE`** и т.п. для конкретных видов объектов.

С каждым контекстом DC одновременно связано не более одного объекта каждого вида (некоторые могут быть «пустыми»).

Выбор нового объекта вместо текущего:

```
HGDIOBJ SelectObject( HDSC hDC, HGDIOBJ hNewObj)
```

Функция возвращает описатель прежнего

После завершения использования объекта желательно возвращать предыдущую установку.

4.2 Контексты графических устройств

Объект Device Context (DC) – «контекст» графического устройства, абстракция «поверхности рисования», соответствует логическому устройству, окну или его части; описатель «экземпляра» логического устройства. Все действия над изображением в логическом устройстве выполняются через контекст.
(Удачное название для надстройки над DC в VCL – Canvas.)

Идентификация – Handle **`HDC`**.

Виды контекстов:
– контекст ***окна*** – связан с окнами графического интерфейса Windows
– контекст ***устройства*** – связан с внешними устройствами ввода-вывода (могут также быть виртуальными)
– контекст ***в памяти*** – не имеет устройства отображения, роль «поверхности» играет Bitmap или Metafile (обязательно должны присутствовать в контексте)

Получение контекста:
– **`GetDC()`**, **`GetDCEx()`**, **`GetWindowDC()`** – для контекстов окна
– **`CreateDC()`**, **`CreateCompatibleDC()`** – для прочих

Прекращение действия контекста (вместо **`CloseHandle()`**!):
– **`ReleaseDC()`** – для контекстов окна
– **`DeleteDC()`** – для прочих

(Формирование и вывод изображения приходится выполнять значительно чаще, чем его считывание, поэтому далее будет обсуждаться в основном запись в контекст, а не чтение из него.)

4.3 Параметры отображения

Система координат:
– направление осей
– начало отсчета (***origin***)
– единица измерений (***logical unit***)

Совокупность параметров – ***режим отображения*** (***Mapping Mode***)

Предопределенные режимы:

Режим	Единца измерения	Направления осей
MM_TEXT	1 пиксель	вправо-вниз
MM_LOMETRIC	0,1 мм	вправо-вверх
MM_HIMETRIC	0,01 мм	вправо-вверх
MM_LOENGLISH	0,01" (дюйм, inch)	вправо-вверх
MM_HIENGLISH	0,001" (дюйм, inch)	вправо-вверх
MM_TWIPS	1 twips = 1/20 «точки» = 1/1440" (1 «точка» = 1/72")	вправо-вверх
MM_ISOTROPIC	настраиваемая (по обе-им осям синхронно)	настраиваемые
MM_ANISOTROPIC	настраиваемая (по обе-им осям независимо)	настраиваемые

Управление режимом отображения:

– **`SetMapMode()`**, **`GetMapMode()`**

После установки режима **`MM_ISOTROPIC`** или **`MM_ANISOTROPIC`** обычно следует настройка единицы измерения. Также это может потребоваться при точной калибровке конкретного графического устройства.

Управление единицей измерений (***extent***) и (косвенно) направлением осей:

– **`GetWindowExtEx()`**, **`SetWindowExtEx()`**

– **`GetViewportExtEx()`**, **`GetViewportExtEx()`**

При этом значение имеют не абсолютные величины параметров, а соотношение их для «окна» (Window) и «области вывода» (Viewport):

Режим Maping Mode	Значения (видеорежима 1280×800, Windows 7)				
	Extent		Origin		
	Window	Viewport	Window	Viewport	
MM_TEXT	1×1	1×1	(0;0)	(0;0)	фиксированные
MM_LOMETRIC	4516×4516	1280×-800	(0;0)	(0;0)	
MM_HIMETRIC	45156×45156	1280×-800	(0;0)	(0;0)	
MM_LOENGLISH	1778×1778	1280×-800	(0;0)	(0;0)	
MM_HIENGLISH	17778×17778	1280×-800	(0;0)	(0;0)	
MM_TWIPS	25600×25600	1280×-800	(0;0)	(0;0)	
MM_ISOTROPIC	4516×4516	1280×-800	(0;0)	(0;0)	начальные
MM_ANISOTROPIC	4516×4516	1280×-800	(0;0)	(0;0)	

Управление точкой начала отсчета:

– **`GetWindowOrgEx()`**, **`SetWindowOrgEx()`**

– **`GetViewportOrgEx()`**, **`SetViewportOrgEx()`**

Принцип управления аналогичен «размерности»: соотношение параметров, заданных для «окна» и «области вывода». По умолчанию (и в ненастраиваемых режимах) это точка (0;0).

4.4 Формирование цвета и управление цветом

Тип **COLORREF** – «логический» цвет, 24-разрядный, прямое кодирование трех цветовых компонент (логических!). Размещение в двойном слове (**DWORD**).

Функции (макросы) для «сборки» цвета из компонент и для выделения компонент из цвета:
– **RGB**(***r***,***g***,***b***) – объединение компонент в логический цвет **COLORREF**
– **GetRValue**(***cr***), **GetGValue**(***cr***), **GetBValue**(***cr***) – выделение компонент из **COLORREF**

Объект Palette (логическая палитра) – табличное преобразование и расширение цветов.

Более низкий уровень преобразования цвета – т.н. «гамма-преобразование» с помощью «рампы» (***Gamma Ramp***):

массивы значений цветов

`GetDeviceGammaRamp()`
`SetDeviceGammaRamp()`

4.5 Инструменты графики – векторные изображения

Различие между векторным и растровым способом формирования и описания изображений.

В Win GDI основной подход – ***векторный***: изображение состоит из ряда графических примитивов, каждый из которых прорисован соответствующим инструментом с заданными параметрами.

Объекты (инструменты):

– Pen («перо») – линии и контуры примитивов

– Brush («кисть») – закраска внутренних областей примитивов

– Font («шрифт») – текст (символы и строки символов)

4.6 Инструменты графики – растровые изображения

Простейшие функции попиксельного ввода-вывода:

```
COLORREF GetPixel( hDC, nX, nY);
COLORREF SetPixel( hDC, nX, nY, COLORREF crColor);
BOOL SetPixelV( hDC, nX, nY, COLORREF crColor);
```

Функция **SetPixelV()** приводит значение цвета к ближайшему представимому в данном контексте.

Кроме отдельных пикселей, можно манипулировать областями (не путать с объектами GDI Region), перемещая их содержимое в пределах контекста или между контекстами.

Простой перенос без преобразования:

```
BitBlt();
```

Перенос в пярмоугольную область с маштабированием:

```
StretchBlt();
```

Перенос в непрямоугольную область с искажением формы:

```
PlgBlt();
```

Перенос с маскированием части изображения (маска задается растром – битовой картой):

```
MaskBlt();
```

Использование этих функций может быть эффективно также и для сложных векторных изображений, если переносом фрагментов растра можно заменить неоднократный повторный вывод множества элементов.

4.7 Другие возможности работы с изображениями

4.7.1 Использование контекста в памяти

Работа с изображением – обычно трудоемкая задача из-за большого количества данных (пикселей), к которым необходимо обратиться, особенно если это требует привилегированного доступа к памяти физического устройства.

Один из приемов повышения эффективности – использование контекста в памяти («теневого контекста» в качестве промежуточного «буфера»:

1) Создание/получение рабочего контекста – «видимого»

2) Создание совместимого с ним контекста в памяти (compatible DC) – «теневого»

3) Создание и привязка для «теневого» контекста объекта Bitmap или Metafile в качестве виртуальной «поверхности рисования»

4) Формирование в «теневом» контексте изображения любым обычным способом

5) Перенос по мере надобности изображения или его фрагментов (функции `***Blt()`) из «теневого» контекста в «видимый»

Таким образом достигается выигрыш:

– обращения к контексту в памяти требуют значительно меньше времени

– можно повторно использовать однажды сформированное сложное изображение вместо полного повторного построения

– можно заранее заготовить и затем использовать набор изображений (например, последовательность кадров анимации)

– перерисовка изображения или его фрагмента единым блоком обычно визуально предпочтительнее, чем последовательная перерисовка всех отдельных элементов (примитивов)

4.7.2 Использование регионов (областей отсечения)

Объект GDI Region – обычно для представления ***области отсечения***: при применении к данному контексту любых графических примитивов модифицируются только пиксели, входящие в область отсечения, за ее пределами действие примитивов «отсекается».

Возможности:

– базовая (элементарная) форма региона – прямоугольник

– регионы могут комбинироваться друг с другом с применение операций OR, AND, XOR

– избирательное включение в регион неклиентской области окна, дочерних окон и т.п.

– регион сложной формы представляется комбинацией элементарных регионов

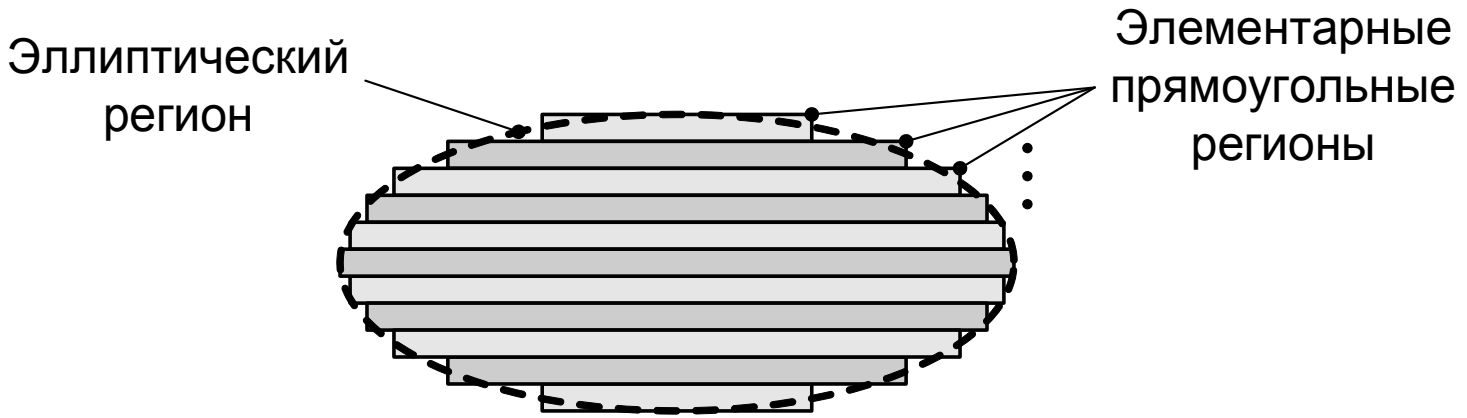

Рис. – Область отсечения сложной формы

Эффект:

– дополнительные возможности формирования сложного изображения во взаимодействии с основными инструментами

– автоматическое применение системой при восстановлении частично «поврежденного» окна (после перекрытия другим окном)

– сокращение времени перерисовки при включении в область отсечения только той части изображения, которая действительно нуждается в модификации (решение о попадании/непопадании в область отсечения намного быстрее, чем реальное обращение к пикселю)