1. Инструменты рисования и объекты GDI Windows.

GDI предоставляет набор инструментов для работы с графикой, таких как перья (Pen), кисти (Brush), шрифты (Font), и объекты для работы с растровой графикой (Bitmap). Эти инструменты используются для рисования линий, текста, геометрических фигур и изображений.

1. Контексты устройств. Способы получения контекстов устройств. Удаление контекстов.

Контекст устройства (Device Context, DC) — это структура данных, которая хранит информацию о состоянии графического вывода.

**Способы получения контекстов устройств:**

* GetDC(HWND hwnd) — получает контекст для окна.
* CreateCompatibleDC(HDC hdc) — создает контекст памяти.
* BeginPaint(HWND hwnd, PAINTSTRUCT\* ps) — используется при обработке WM\_PAINT.

**Удаление контекстов:**

* ReleaseDC(HWND hwnd, HDC hdc) — освобождает контекст устройства.
* DeleteDC(HDC hdc) — удаляет созданный контекст памяти.

1. Создание инструментов рисования GDI (Функции создания перьев, кистей и шрифтов).

**Функции для создания инструментов:**

* CreatePen() — создание пера.
* CreateSolidBrush() — создание кисти.
* CreateFont() — создание шрифта.

1. Загрузка инструментов в контекст. Свободное состояние инструментов. Удаление инструментов GDI.

Инструменты загружаются в контекст с помощью функции SelectObject(HDC hdc, HGDIOBJ h).

**Удаление инструментов GDI:**

* DeleteObject(HGDIOBJ hObject) — удаляет объект GDI.

1. Предопределенные (“встроенные”) инструменты с характеристиками по умолчанию.

Windows предоставляет встроенные инструменты, такие как GetStockObject(). Например:

* WHITE\_PEN, BLACK\_BRUSH, DEFAULT\_GUI\_FONT.

1. Логические и физические координаты. Режимы отображения (mapping mode).

GDI поддерживает режимы отображения (Mapping Modes), преобразующие логические координаты в физические:

* MM\_TEXT — 1 к 1 по пикселям.
* MM\_LOMETRIC, MM\_HIMETRIC — метрические единицы.

1. Отображение текста.

Для отображения текста используется TextOut(HDC hdc, int x, int y, LPCSTR lpString, int cchString).

1. Управление отображением текста.

**Выравнивание текста:** SetTextAlign().

**Цвет текста:** SetTextColor().

1. Задание цвета в Windows. Использование макроса RGB при создании кистей и перьев.

Цвет задается через макрос RGB(r, g, b) для кистей, перьев и текста.

Пример:

CreateSolidBrush(RGB(255, 0, 0)); // Красный цвет.

1. Определение характеристик шрифта. Пропорциональные шрифты True Type.

Шрифты создаются функцией CreateFont() или CreateFontIndirect().  
**TrueType-шрифты** поддерживают масштабируемость и антиалиасинг.

1. Контексты памяти. Создание контекста памяти совместимого с заданным контекстом устройства.

Контекст памяти создается с помощью CreateCompatibleDC(HDC hdc). Он используется для операций вне экрана.

1. Создание растров. Выбор растров в контекст памяти.

Растры создаются функцией CreateCompatibleBitmap() или из файла с помощью LoadBitmap().

1. Копирование блоков на примере копирования растра из контекста памяти в контекст устройства.

Функция BitBlt() копирует содержимое между контекстами:

BitBlt(hdcDest, x, y, width, height, hdcSrc, xSrc, ySrc, SRCCOPY);

1. Аппаратно-независимые растры. Структура аппаратно-независимых растров с палитрой и беспалитровых.

Формат BMP сохраняет изображение в независимом от устройств виде.

**Типы BMP:**

* **С палитрой:** использует таблицу цветов.
* **Без палитры:** содержит прямые значения цветов (24 бита на пиксель).

1. Информация в заголовках файлов BMP.

Заголовок BMP включает:

* Размер файла.
* Размеры изображения.
* Глубину цвета (1, 4, 8, 24 бита).

1. Создание растровых объектов GDI из аппаратно-независимых растров.

Используется CreateDIBSection() для создания аппаратно-независимого растра.

1. Копирование данных из растрового объекта в контекст памяти.

Функция GetDIBits() копирует пиксели растра в массив.

1. Типы Windows, используемые в GDI (POINT, RECT).

Основные структуры:

* POINT — координаты точки.
* RECT — координаты прямоугольника.

1. Функции для работы с данными GDI-типов.

PtInRect() — проверка, находится ли точка в прямоугольнике.

OffsetRect() — смещение прямоугольника.

1. Создание и использование таймеров в приложениях Windows для смены параметров изображения в контексте. Недостаток обновления, “ползущая графика”.

Для обновления изображения можно использовать SetTimer() и обработку WM\_TIMER.  
**Недостатки:** возможно мерцание из-за перерисовки.

1. Использование простоев приложения для обновления графических данных контекста устройства. Функция PeekMessage.

Функция PeekMessage() позволяет обрабатывать сообщения без блокировки основного потока.

1. Особенность в обработке сообщения WM\_QUIT при эффективном использовании ожиданий сообщений для обновления графических данных окна.

WM\_QUIT сигнализирует завершение приложения. Если ожидания сообщений используются для графики, нужно правильно обрабатывать это сообщение, чтобы предотвратить зацикливание.

Пример обработки:

while (PeekMessage(&msg, NULL, 0, 0, PM\_REMOVE)) {

if (msg.message == WM\_QUIT) break;

TranslateMessage(&msg);

DispatchMessage(&msg);

}