1. **GDI是什么的英文缩写？**

Graphics Device Interface

**2、 什么是设备的无关性？**

设备的无关性是指应用程序或操作系统能够不受硬件设备类型、品牌或型号的限制而运行。它是一种能够在不同的设备上运行的能力，而不需要针对每个设备编写独立的代码。在计算机图形学中，设备的无关性通常是指图形设备无关性，即应用程序能够使用相同的代码生成可在多种图形设备上渲染的图像。这样，开发人员就可以编写只需一次编译和调试，就能在不同的平台和设备上运行的程序。

**3、 GDI的主要任务是什么？起到什么样的作用？**

GDI的主要任务是管理Windows系统中的图形输出，提供绘制图形和文本、处理字体、维护设备的无关性等功能。

作用：

1. 提供绘图API：GDI提供了一系列函数和结构体，可以使开发人员轻松地绘制图形、文本、曲线和其他图像效果。

2. 处理字体：GDI提供了大量与字体相关的API，包括字体的选择、创建和渲染等功能。

3. 管理绘图表面：GDI负责将绘图操作映射到正确的设备上，确保图像在不同的显示器和打印机上都能正确呈现。

4. 提供设备无关性：GDI提供了一些函数和结构体，可以使应用程序在不考虑底层硬件设备类型的情况下绘图。这样，应用程序只需要编写一次代码，在任何支持Windows GDI的设备上都可以运行。

**4、 什么是设备描述表？它的作用是什么？**

设备描述表是一种数据结构，用于描述硬件设备的属性和特性。在Windows系统中，设备描述表通常包含在驱动程序中，可以通过操作系统提供的API来访问。

设备描述表的作用是告诉操作系统如何与硬件设备进行交互。它包含了硬件设备的各种信息，例如设备类型、厂商ID、设备序列号、支持的功能和性能参数等。根据这些信息，操作系统可以选择适当的驱动程序来管理硬件设备，并向应用程序提供正确的接口。在Windows系统中，设备描述表通常使用一种名为“设备信息集”（Device Information Set）的结构来组织，每个设备都有一个唯一的标识符。应用程序可以使用基于设备描述表的API来遍历设备信息集，查找并枚举系统中可用的设备，并获取有关这些设备的详细信息。

**5、 图形刷新包括什么？**

图形刷新是指在屏幕上更新显示的图像。它包括：

1. 清空屏幕：在绘制新的图形之前，需要清空原来的图像内容，否则新绘制的图形与原来的图形会重叠在一起。

2. 绘制图形：这是刷新图像最重要的步骤。应用程序需要调用GDI函数或其他图形库函数来绘制所需的图形。这些函数将创建各种形状、颜色和纹理的图形，在屏幕上进行显示。

3. 更新屏幕：当一个新的图形被绘制出来后，需要将它更新到屏幕上。这通常通过操作系统提供的API完成，例如windows中的InvalidateRect() 和UpdateWindow() 函数等。

4. 双缓冲处理：为了避免图形闪烁和不流畅的效果，一些高级的图形应用程序使用双缓冲技术。其基本思想是将整个图像绘制到离屏缓冲区（Off-Screen Buffer）中，然后再将整个图像一次性地复制到屏幕上，从而消除图形的闪烁问题。

**6、 刷新请求有哪些？**

1. 重绘（Redraw）请求：当窗口被暴露在显示器上、大小发生变化、覆盖或其他窗口遮挡时，操作系统会向应用程序发送重绘请求，要求应用程序重新绘制窗口内容。

2. 定时器（Timer）请求：应用程序可以创建定时器，以一定的时间间隔来更新图像。当定时器到期时，操作系统会向应用程序发送定时器请求，要求应用程序更新屏幕上的图像。

3. 用户输入（User Input）请求：当用户执行某种操作时（例如鼠标单击、键盘输入等），操作系统会向应用程序发送用户输入请求，要求应用程序更新屏幕上的图像以响应用户操作。

4. 系统事件（System Event）请求：例如屏幕分辨率、颜色模式、设备连接状态等发生变化时，操作系统会向应用程序发送系统事件请求，要求应用程序更新屏幕上的图像。

**7、 应用程序情况下会发生刷新请求？同时发出什么消息？**

刷新请求通常是由操作系统发送给应用程序的消息。根据不同的请求类型，消息名称和消息参数也会有所不同，应用程序需要相应地处理这些消息，以便及时更新屏幕上的图像。

1. 重绘请求（WM\_PAINT）：当窗口需要重新绘制时（例如创建、移动、调整大小等），操作系统会发送WM\_PAINT消息给应用程序。

2. 定时器请求（WM\_TIMER）：应用程序可以使用SetTimer函数创建定时器，在指定的时间间隔内更新屏幕上的图像。当定时器到期时，操作系统会发送WM\_TIMER消息给应用程序。

3. 鼠标和键盘输入请求：当用户点击鼠标或输入键盘时，操作系统会向应用程序发送相应的消息（例如WM\_LBUTTONDOWN、WM\_KEYDOWN等），应用程序可以通过处理这些消息来更新屏幕上的图像。

4. 自定义消息：应用程序还可以定义自己的消息类型，并在需要时向自己发送这些消息。例如，一个绘图应用程序可能会定义一个名为“WM\_DRAW”的消息类型，用于触发绘图操作。

**8、 什么是无效区域？**

无效区域（Invalid Region）指的是屏幕上需要被重新绘制的部分区域。当窗口需要重绘时，操作系统会将需要更新的区域标记为无效区域，并向应用程序发送重绘消息。应用程序可以根据无效区域来决定哪些部分需要进行重新绘制。

在Windows中，无效区域通常表示为一个矩形区域，由左上角和右下角坐标确定。操作系统使用一种称为“无效矩形列表”（Invalid Rectangles List）的数据结构来跟踪窗口的无效区域。该列表包含了所有需要重新绘制的矩形区域。

当窗口收到重绘请求时，应用程序可以通过调用GetUpdateRect或GetUpdateRgn函数来获取当前的无效区域。应用程序可以使用这些函数返回的矩形或区域信息，确定需要重新绘制的图像区域，以便提高绘制效率。然后，应用程序可以使用GDI函数或其他图形库函数来绘制新的图像，并将其复制到无效区域上。

**9、 有效的刷新方法有哪些？大家觉得哪种比较好？在代码中若主动要求刷新窗口，应该怎么办？**

有效的刷新方法包括以下几种：

1. WM\_PAINT消息：当窗口需要重绘时，操作系统会向应用程序发送WM\_PAINT消息，应用程序可以处理这个消息，并在其中使用GDI函数或其他图形库函数来重新绘制窗口内容。

2. WM\_ERASEBKGND消息：当窗口被暴露在显示器上、大小发生变化、被覆盖或其他窗口遮挡时，操作系统会向应用程序发送WM\_ERASEBKGND消息，应用程序可以在这个消息中清空窗口背景。

3. 定时器：应用程序可以使用定时器来周期性地更新屏幕上的图像，例如实现动画效果或实时数据的更新。

4. 事件响应：当用户进行某些操作时（例如鼠标单击、键盘输入等），应用程序可以根据相应的事件响应适当地更新屏幕上的图像。

使用WM\_PAINT消息最好。它与Windows系统紧密集成，在应用程序中使用方便，效率高，而且具有很好的兼容性。

当应用程序需要主动刷新窗口时，可以调用InvalidateRect或InvalidateRgn函数来设置窗口的无效区域，然后等待操作系统向应用程序发送WM\_PAINT消息，进行重新绘制。如果应用程序需要立即更新窗口，可以调用UpdateWindow函数，强制操作系统发送WM\_PAINT消息，立即更新窗口内容。

**10、 获取设备环境常用的两种方法是什么？**

1. GetDC函数：GetDC函数是最常用的获取设备环境的方法之一。它用于获取指定窗口或屏幕区域的设备环境句柄（HDC），可以通过该句柄来进行图形输出操作。

2. BeginPaint/EndPaint函数：BeginPaint和EndPaint函数是一对用于获取设备环境的API函数。通常用于在窗口上进行绘图操作，它们会自动处理WM\_PAINT消息，并获取相应的设备环境句柄。

**11、 调用BeginPaint函数和GetDc有什么联系与区别？在实际编程中怎样选择使用？**

BeginPaint和GetDC函数都可以用于获取设备环境（Device Context），用于在窗口上进行绘图操作。它们的联系和区别如下：

联系：

1. 它们都返回一个设备环境句柄（HDC），用于后续的绘图操作。

2. 它们都要求在使用完毕后，调用相应的释放函数（EndPaint或ReleaseDC）来释放设备环境句柄。

区别：

1. BeginPaint函数主要用于处理WM\_PAINT消息，自动对绘图区域进行设置，并返回包含绘图信息的PAINTSTRUCT结构体指针，方便程序员直接使用。而GetDC函数则需要手动指定窗口或者屏幕中的区域来获取设备环境句柄。

2. 在使用BeginPaint函数获取设备环境时，只能在WM\_PAINT消息处理函数中才能使用，而GetDC函数可以在任何时候使用，不受限制。

3. 由于BeginPaint自动处理了一些与WM\_PAINT消息相关的操作，因此在使用BeginPaint获取设备环境时，可以减少程序员编写的代码量，同时更加便捷。

在实际编程中，选择使用BeginPaint还是GetDC，取决于特定场景的具体需求：

1. 如果需要在处理WM\_PAINT消息时输出图形，则建议使用BeginPaint函数，以简化代码并提高效率。

2. 如果需要在其他时候输出图形，或需要在程序的多个位置使用同一个设备环境，则建议使用GetDC函数，并在使用完毕后立即释放设备环境，以防止资源泄漏和其他问题。

**12、 什么是映像模式？为什么要用映像模式**

映像模式（Bitmap Mode）是一种绘图模式，它使用内存中的位图（Bitmap）作为绘图缓冲区，通过修改位图中的像素值来进行绘图操作。通常情况下，映像模式使用GDI函数或其他图形库函数来操作位图，最后将位图复制到设备环境中显示。

映像模式和直接在设备环境中绘图相比，有以下几个优点：

1. 可以在内存中进行绘图，避免了频繁地访问昂贵的外部设备。

2. 可以预先生成位图，并在需要时进行输出，节省了重复计算的时间和资源。

3. 可以在位图上进行各种高级的处理，例如变换、过滤等，提供了更加丰富的图形效果。

**13、 了解几种常用的映像模式，默认是哪种映像模式。**

常用的映像模式有以下几种：

1. DIB（Device Independent Bitmap）：DIB是一种独立于设备的位图格式，它可以用于在不同的设备环境中显示图片。DIB最大的优点是具有很高的兼容性和可移植性，因此它被广泛应用于跨平台图形开发和图像格式转换等领域。

2. GDI+：GDI+是微软公司开发的一套高级绘图API库，它提供了丰富的图形处理功能，包括渐变、阴影、透明度、混合等效果，并支持多种图像格式。使用GDI+可以方便地实现各种炫酷的图形效果，并为开发人员节省了大量的时间和精力。

3. OpenGL：OpenGL是一种基于图形硬件加速的跨平台3D图形 API，它提供了强大的3D渲染功能，并支持多种图形文件格式。OpenGL主要应用于游戏开发、科学可视化、CAD等领域，并被广泛应用于桌面、移动和嵌入式系统等平台。

在Windows平台上，默认的映像模式是GDI（Graphics Device Interface），它是Windows操作系统的核心图形库，提供了各种绘图函数和设备驱动程序，能够支持多种设备环境和图像格式。GDI是Windows GUI应用程序的基本绘图方式，同时也是其他图形库的基础。

**14、 了解映像模式中函数的使用（不需要书面写），可以写部分重点函数的理解。**

1. CreateCompatibleBitmap函数：CreateCompatibleBitmap函数用于创建与指定设备环境兼容的位图对象。它可以用于创建一个内存中的位图对象，并将其用作绘图缓冲区，以便在位图上进行各种高级的处理和操作。

2. SelectObject函数：SelectObject函数用于向设备环境中选择一个GDI对象，例如位图、画笔、画刷等，以便后续的绘图操作使用该对象。它返回被替换的对象的句柄，方便程序员在需要时恢复该对象。

3. BitBlt函数：BitBlt函数用于从源设备环境中复制一个矩形区域的像素数据到目标设备环境中，以实现图像的复制、移动、缩放等效果。它支持多种位图格式，并提供了丰富的参数选项，可以灵活地控制图像输出的方式和效果。

4. StretchBlt函数：StretchBlt函数是BitBlt函数的扩展版，它可以实现更加灵活的缩放和变换效果。它通过对源图像进行拉伸或缩小，生成目标图像，从而实现图像的变换和处理。

**15、 画笔的操作有哪些？**

1. 创建画笔：使用CreatePen函数创建一个画笔对象，并指定相应的属性，例如颜色、线条宽度、样式等。

HPEN hPen = CreatePen(PS\_SOLID, 1, RGB(255, 0, 0));

2. 选择画笔：使用SelectObject函数将画笔对象选入设备环境中，以便后续的绘图操作使用该画笔。同时，SelectObject会返回原始的画笔对象，方便程序员在需要时进行恢复。

HDC hdc = GetDC(NULL);

HPEN hOldPen = (HPEN)SelectObject(hdc, hPen);

// ...

SelectObject(hdc, hOldPen); // 恢复原始的画笔

ReleaseDC(NULL, hdc);

3. 销毁画笔：使用DeleteObject函数销毁一个画笔对象，释放其占用的内存资源。这个操作通常在不再需要画笔对象时进行，以避免内存泄漏和其他问题。

DeleteObject(hPen);

4. 设置画笔属性：使用SetDCPenColor函数、SetDCBrushColor函数和SetPenWidth函数等设置画笔的颜色、宽度等属性。

HDC hdc = GetDC(NULL);

SetDCPenColor(hdc, RGB(0, 0, 255));

// ...

ReleaseDC(NULL, hdc);

**16、 了解画笔中使用的函数名字的英文意思，便于大家记忆函数，（不需要书面写出）**

**17、 了解常用的绘图函数，尤其是饼图函数，椭圆弧线函数。想一想怎么画圆，有没有画圆的函数？**

常用的绘图函数：

1. MoveToEx：将当前绘图点移动到指定位置。

2. LineTo：从当前绘图点画一条直线到指定点。

3. Rectangle：绘制一个矩形。

4. RoundRect：绘制一个圆角矩形。

5. Ellipse：绘制一个椭圆形。

6. Arc：绘制一个圆弧线。

7. Pie：绘制一个饼图。

饼图函数Pie和椭圆弧线函数Arc：

1. Pie函数：Pie函数用于绘制一个填充的扇形。它接受四个参数：设备环境句柄、左上角坐标、右下角坐标、起始角度和结束角度。

//红色的填充扇形：

HDC hdc = GetDC(NULL);

HPEN hPen = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

HBRUSH hBrush = CreateSolidBrush(RGB(255, 0, 0));

SelectObject(hdc, hPen);

SelectObject(hdc, hBrush);

Pie(hdc, 100, 100, 500, 500, 0, 90);

DeleteObject(hPen);

DeleteObject(hBrush);

ReleaseDC(NULL, hdc);

1. Arc函数：Arc函数用于绘制一个未填充的弧形。它接受五个参数：设备环境句柄、左上角坐标、右下角坐标、起始角度和结束角度。

//蓝色的未填充弧形

HDC hdc = GetDC(NULL);

HPEN hPen = CreatePen(PS\_SOLID, 1, RGB(0, 0, 255));

SelectObject(hdc, hPen);

Arc(hdc, 100, 100, 500, 500, 0, 90);

DeleteObject(hPen);

ReleaseDC(NULL, hdc);

绘制圆形可以使用Ellipse函数，该函数接受四个参数：设备环境句柄、左上角坐标、右下角坐标。对于正方形的圆，可以使用相等的左上角和右下角坐标。

//黑色的填充圆形

HDC hdc = GetDC(NULL);

HPEN hPen = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

HBRUSH hBrush = CreateSolidBrush(RGB(0, 0, 0));

SelectObject(hdc, hPen);

SelectObject(hdc, hBrush);

Ellipse(hdc, 100, 100, 500, 500);

DeleteObject(hPen);

DeleteObject(hBrush);

ReleaseDC(NULL, hdc);