# 学习大纲

|  |
| --- |
|  |

## Rectangle函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Rectangle** 函数绘制一个矩形。 矩形使用当前笔轮廓，并使用当前画笔填充。 语法 C++复制  BOOL Rectangle(  [in] HDC hdc,  [in] int left,  [in] int top,  [in] int right,  [in] int bottom  ); 参数 [in] hdc  设备上下文的句柄。  [in] left  矩形左上角的 x 坐标（以逻辑坐标为单位）。  [in] top  矩形左上角的 y 坐标（以逻辑坐标为单位）。  [in] right  矩形右下角的 x 坐标（以逻辑坐标为单位）。  [in] bottom  矩形右下角的 y 坐标（以逻辑坐标为单位）。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 **矩形**既不使用当前位置，也不更新当前位置。  绘制的矩形不包括下边缘和右边缘。  如果使用PS\_NULL笔，矩形的尺寸高度减少 1 像素，宽度减少 1 像素。 示例 有关示例，请参阅 [使用填充形状](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/using-filled-shapes)。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [填充形状函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shape-functions)  [填充形状概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shapes)  [RoundRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-roundrect) |

## RoundRect函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **RoundRect** 函数绘制一个带有圆角的矩形。 矩形使用当前笔轮廓，并使用当前画笔填充。 语法 C++复制  BOOL RoundRect(  [in] HDC hdc,  [in] int left,  [in] int top,  [in] int right,  [in] int bottom,  [in] int width,  [in] int height  ); 参数 [in] hdc  设备上下文的句柄。  [in] left  矩形左上角的 x 坐标（以逻辑坐标表示）。  [in] top  矩形左上角的 y 坐标（以逻辑坐标表示）。  [in] right  矩形右下角的 x 坐标（以逻辑坐标表示）。  [in] bottom  矩形右下角的 y 坐标（以逻辑坐标表示）。  [in] width  用于绘制圆角的椭圆的宽度（以逻辑坐标表示）。  [in] height  用于绘制圆角的椭圆的高度（以逻辑坐标表示）。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 此函数既不使用也不更新当前位置。 示例 有关示例，请参阅 [使用填充的形状](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/using-filled-shapes)。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [填充的形状函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shape-functions)  [填充形状概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shapes)  [矩形](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-rectangle) |

### 圆角矩形的绘制原理

|  |
| --- |
|  |

## Ellipse函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **椭圆**函数绘制一个椭圆。 椭圆的中心是指定边界矩形的中心。 椭圆形是使用当前笔轮廓的，并使用当前画笔填充。 语法 C++复制  BOOL Ellipse(  [in] HDC hdc,  [in] int left,  [in] int top,  [in] int right,  [in] int bottom  ); 参数 [in] hdc  设备上下文的句柄。  [in] left  边界矩形左上角的 x 坐标（以逻辑坐标表示）。  [in] top  边界矩形左上角的 y 坐标（以逻辑坐标表示）。  [in] right  边界矩形右下角的 x 坐标（以逻辑坐标表示）。  [in] bottom  边界矩形右下角的 y 坐标（以逻辑坐标表示）。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 **Ellipse** 既不使用也不更新当前位置。 示例 有关示例，请参阅 [使用填充的形状](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/using-filled-shapes)。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [弧](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-arc)  [ArcTo](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-arcto)  [填充的形状函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shape-functions)  [填充形状概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shapes) |

## Chord函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **和弦**函数绘制一个和弦 (一个由椭圆和线段（称为正交) ）交集的区域。 和弦使用当前笔轮廓，并使用当前画笔填充。 语法 C++  BOOL Chord(  [in] HDC hdc,  [in] int x1,  [in] int y1,  [in] int x2,  [in] int y2,  [in] int x3,  [in] int y3,  [in] int x4,  [in] int y4  ); 参数 [in] hdc  显示和弦的设备上下文的句柄。  [in] x1  边界矩形左上角的 x 坐标（以逻辑坐标为单位）。  [in] y1  边界矩形左上角的 y 坐标（以逻辑坐标为单位）。  [in] x2  边界矩形右下角的 x 坐标（以逻辑坐标为单位）。  [in] y2  边界矩形右下角的 y 坐标（以逻辑坐标为单位）。  [in] x3  定义和弦开始的径向端点的 x 坐标（以逻辑坐标为单位）。  [in] y3  定义和弦开头的径向端点的 y 坐标（以逻辑坐标为单位）。  [in] x4  定义和弦终点的径向端点的 x 坐标（以逻辑坐标为单位）。  [in] y4  定义和弦终点的径向端点的 y 坐标（以逻辑坐标为单位）。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 和弦的曲线由适合指定边框的椭圆定义。 曲线从椭圆与第一个径向相交的点开始，并逆时针延伸至椭圆与第二个径向相交的点。 通过绘制一条从第一条径向与第二条径向曲线的交集到第二条径向曲线的交集的直线，来闭合和弦。  如果曲线的起点和终点相同，则绘制一个完整的椭圆。  **Chord** 既不使用当前位置，也不更新当前位置。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [AngleArc](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-anglearc)  [弧](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-arc)  [ArcTo](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-arcto)  [填充形状函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shape-functions)  [填充形状概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shapes)  [派](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-pie) |

## Pie函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Pie** 函数绘制一个由椭圆和两个径向的交集绑定的饼形楔形。 饼图使用当前笔轮廓，并使用当前画笔填充。 语法 C++复制  BOOL Pie(  [in] HDC hdc,  [in] int left,  [in] int top,  [in] int right,  [in] int bottom,  [in] int xr1,  [in] int yr1,  [in] int xr2,  [in] int yr2  ); 参数 [in] hdc  设备上下文的句柄。  [in] left  边界矩形左上角的 x 坐标（以逻辑坐标为单位）。  [in] top  边界矩形左上角的 y 坐标（以逻辑坐标为单位）。  [in] right  边界矩形右下角的 x 坐标（以逻辑坐标为单位）。  [in] bottom  边界矩形右下角的 y 坐标（以逻辑坐标为单位）。  [in] xr1  第一个径向的端点的 x 坐标（以逻辑坐标为单位）。  [in] yr1  第一个径向的端点的 y 坐标（以逻辑坐标为单位）。  [in] xr2  第二个径向的端点的 x 坐标（以逻辑坐标为单位）。  [in] yr2  第二个径向的端点的 y 坐标（以逻辑坐标为单位）。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 饼图的曲线由适合指定边框的椭圆定义。 曲线从椭圆与第一个径向相交的点开始，并逆时针延伸至椭圆与第二个径向相交的点。  **Pie** 函数既不使用当前位置，也不更新当前位置。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [AngleArc](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-anglearc)  [弧](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-arc)  [ArcTo](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-arcto)  [和弦](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-chord)  [填充形状函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shape-functions)  [填充形状概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shapes) |

## Polygon函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Polygon** 函数绘制一个由两个或多个直线连接的顶点组成的多边形。 多边形使用当前笔进行轮廓，并使用当前画笔和多边形填充模式进行填充。 语法 C++复制  BOOL Polygon(  [in] HDC hdc,  [in] const POINT \*apt,  [in] int cpt  ); 参数 [in] hdc  设备上下文的句柄。  [in] apt  指向 [POINT](https://learn.microsoft.com/zh-cn/windows/win32/api/windef/ns-windef-point) 结构的数组的指针，该数组以逻辑坐标形式指定多边形的顶点。  [in] cpt  数组中的顶点数。 此值必须大于或等于 2。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 多边形通过绘制从最后一个顶点到第一个顶点的线条自动关闭。  **Polygon** 函数既不使用也不更新当前位置。  忽略任何额外的点。 若要绘制具有更多点的线条，请将数据划分为多个组，其中每个组的点数都小于最大点数，并为每个点组调用 函数。 请记得连接线段。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [填充的形状函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shape-functions)  [填充形状概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shapes)  [GetPolyFillMode](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getpolyfillmode)  [点](https://learn.microsoft.com/zh-cn/windows/win32/api/windef/ns-windef-point)  [PolyPolygon](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-polypolygon)  [折线](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-polyline)  [PolylineTo](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-polylineto)  [SetPolyFillMode](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setpolyfillmode) |

## PolyPolygon函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PolyPolygon** 函数绘制一系列闭合多边形。 每个多边形使用当前笔进行轮廓，并使用当前画笔和多边形填充模式进行填充。 此函数绘制的多边形可以重叠。 语法 C++复制  BOOL PolyPolygon(  [in] HDC hdc,  [in] const POINT \*apt,  [in] const INT \*asz,  [in] int csz  ); 参数 [in] hdc  设备上下文的句柄。  [in] apt  指向 [POINT](https://learn.microsoft.com/zh-cn/windows/win32/api/windef/ns-windef-point) 结构的数组的指针，这些结构定义多边形的顶点（以逻辑坐标表示）。 多边形是连续指定的。 通过绘制从最后一个顶点到第一个顶点的线条，自动关闭每个多边形。 每个顶点应指定一次。  [in] asz  指向整数数组的指针，其中每个整数指定相应多边形中的点数。 每个整数必须大于或等于 2。  [in] csz  多边形的总数。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 此函数既不使用也不更新当前位置。  将忽略任何额外的点。 若要绘制具有更多点的多边形，请将数据划分为多个组，其中每个组的点数都小于最大点数，并为每个点组调用 函数。 请注意，最好仅在其中一个组中有一个多边形。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [填充形状函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shape-functions)  [填充形状概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shapes)  [GetPolyFillMode](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getpolyfillmode)  [点](https://learn.microsoft.com/zh-cn/windows/win32/api/windef/ns-windef-point)  [多边形](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-polygon)  [折线](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-polyline)  [PolylineTo](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-polylineto)  [SetPolyFillMode](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setpolyfillmode) |

# 演练

## 1.新建一个win32项目，取名：Lesson44-draw-shapes

|  |
| --- |
|  |

## 2.定位到WndProc函数的WM\_PAINT消息处理代码，先绘制有矩形，代码如下

|  |
| --- |
|  |

### 效果如下

|  |
| --- |
|  |

## 3然后我们练习一下RoundRect函数，请你好好理解圆角矩形的绘制原理相当于在四个角落分别绘制四个小椭圆或者是四个小圆。

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## 4.然后我们来练习一下椭圆

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## 5.绘制弓形，它好圆弧非常相似

|  |
| --- |
|  |

### 效果：

|  |
| --- |
|  |

## 6.绘制Pie也就是饼形图

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## 7.如果此时你用一样的参数再绘制一个Chord，

|  |
| --- |
|  |

### 效果如下

|  |
| --- |
|  |

### 这就是这两个函数的区别

## 8.下面我们学习绘制多边形，首先需要定义一个数组存放多边形的顶点

|  |
| --- |
|  |

### 然后绘制代码如下

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## 9.可以绘制五角星，使用下面的点数组pts2

|  |
| --- |
|  |
|  |

### 效果

|  |
| --- |
|  |

## 10.PolyPolygon的绘制练习，需要一个顶点多一点的数组，还需要一个保存多少个图形的数组

|  |
| --- |
|  |
|  |

### 效果

|  |
| --- |
|  |

# 这一节的学习到此为止，下面是完整代码

## Lesson44-draw-shapes.cpp

|  |
| --- |
| // Lesson44-draw-shapes.cpp : 定义应用程序的入口点。  //  #include "stdafx.h"  #include "Lesson44-draw-shapes.h"  #define MAX\_LOADSTRING 100  // 全局变量:  HINSTANCE hInst; // 当前实例  TCHAR szTitle[MAX\_LOADSTRING]; // 标题栏文本  TCHAR szWindowClass[MAX\_LOADSTRING]; // 主窗口类名  // 此代码模块中包含的函数的前向声明:  ATOM MyRegisterClass(HINSTANCE hInstance);  BOOL InitInstance(HINSTANCE, int);  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);  INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);  int APIENTRY \_tWinMain(HINSTANCE hInstance,  HINSTANCE hPrevInstance,  LPTSTR lpCmdLine,  int nCmdShow)  {  UNREFERENCED\_PARAMETER(hPrevInstance);  UNREFERENCED\_PARAMETER(lpCmdLine);  // TODO: 在此放置代码。  MSG msg;  HACCEL hAccelTable;  // 初始化全局字符串  LoadString(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);  LoadString(hInstance, IDC\_LESSON44DRAWSHAPES, szWindowClass, MAX\_LOADSTRING);  MyRegisterClass(hInstance);  // 执行应用程序初始化:  if (!InitInstance (hInstance, nCmdShow))  {  return FALSE;  }  hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LESSON44DRAWSHAPES));  // 主消息循环:  while (GetMessage(&msg, NULL, 0, 0))  {  if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  }  return (int) msg.wParam;  }  //  // 函数: MyRegisterClass()  //  // 目的: 注册窗口类。  //  // 注释:  //  // 仅当希望  // 此代码与添加到 Windows 95 中的“RegisterClassEx”  // 函数之前的 Win32 系统兼容时，才需要此函数及其用法。调用此函数十分重要，  // 这样应用程序就可以获得关联的  // “格式正确的”小图标。  //  ATOM MyRegisterClass(HINSTANCE hInstance)  {  WNDCLASSEX wcex;  wcex.cbSize = sizeof(WNDCLASSEX);  wcex.style = CS\_HREDRAW | CS\_VREDRAW;  wcex.lpfnWndProc = WndProc;  wcex.cbClsExtra = 0;  wcex.cbWndExtra = 0;  wcex.hInstance = hInstance;  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LESSON44DRAWSHAPES));  wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);  wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_LESSON44DRAWSHAPES);  wcex.lpszClassName = szWindowClass;  wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));  return RegisterClassEx(&wcex);  }  //  // 函数: InitInstance(HINSTANCE, int)  //  // 目的: 保存实例句柄并创建主窗口  //  // 注释:  //  // 在此函数中，我们在全局变量中保存实例句柄并  // 创建和显示主程序窗口。  //  BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)  {  HWND hWnd;  hInst = hInstance; // 将实例句柄存储在全局变量中  hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,  CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);  if (!hWnd)  {  return FALSE;  }  ShowWindow(hWnd, nCmdShow);  UpdateWindow(hWnd);  return TRUE;  }  //  // 函数: WndProc(HWND, UINT, WPARAM, LPARAM)  //  // 目的: 处理主窗口的消息。  //  // WM\_COMMAND - 处理应用程序菜单  // WM\_PAINT - 绘制主窗口  // WM\_DESTROY - 发送退出消息并返回  //  //  LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)  {  int wmId, wmEvent;  PAINTSTRUCT ps;  HDC hdc;  POINT pts[] = {{50,80},{100,30},{180,50},{220,80}};  POINT pts2[] = {{0,200},{600,200},{100,600},{300,0},{500,600}};  //绘制多个多边形需要2个数组  POINT pts3[] = {{0,200},{600,200},{100,600},{300,0},{500,600},{400,50},{500,50},{500,150},{400,150}};//顶点总数  int pn[2] = {5,4};//需要绘制的图形的总数  switch (message)  {  case WM\_COMMAND:  wmId = LOWORD(wParam);  wmEvent = HIWORD(wParam);  // 分析菜单选择:  switch (wmId)  {  case IDM\_ABOUT:  DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);  break;  case IDM\_EXIT:  DestroyWindow(hWnd);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  break;  case WM\_PAINT:  hdc = BeginPaint(hWnd, &ps);  // TODO: 在此添加任意绘图代码...  //Rectangle(hdc,100,100,320,200);  //圆角矩形  //RoundRect(hdc,100,100,320,200,30,20);  //椭圆  //Ellipse(hdc,100,100,320,200);  //弦形或者说是拱形,弓形，它可以直接使用圆弧函数的参数，它只是比圆弧多绘制了一根直线而已  //Chord(hdc,100,100,320,200,290,50,80,300);  //绘制Pie  //Pie(hdc,100,100,320,200,200,50,80,260);  //Chord(hdc,100,100,320,200,200,50,80,260);  //绘制多边形  //Polygon(hdc,pts,4);  //绘制多个多边形,这里绘制2个图形，一个五角星一个是矩形  PolyPolygon(hdc,pts3,pn,2);  Polygon(hdc,pts2,5);  EndPaint(hWnd, &ps);  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  return 0;  }  // “关于”框的消息处理程序。  INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)  {  UNREFERENCED\_PARAMETER(lParam);  switch (message)  {  case WM\_INITDIALOG:  return (INT\_PTR)TRUE;  case WM\_COMMAND:  if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)  {  EndDialog(hDlg, LOWORD(wParam));  return (INT\_PTR)TRUE;  }  break;  }  return (INT\_PTR)FALSE;  } |