

GG1003

Windows编程

李俊琴

李俊琴

实验3-1 GDI绘图实验

余庆祥

2220631136

BDP

2022/9/22

**内页写作格式**

**一、实验名称：**要用最简练的语言反映实验的内容。

**二、实验日期：**写明做实验的具体年、月、日及组別。

**三、实验目的：**使用简洁的文字或关键字来敘述，是以怎样的目的作此实验的。

**四、实验环境：**实验的操作系统和软件等。

**五、实验的步骤和方法：**这是实验报告极其重要的内容。这部分要写明经过哪几个步骤。

**六、数据记录和计算：**指从实验中测到的数据以及计算结果。

**七、实验结果或结论：**即根据实验过程中所见到的现象和测得的数据，得出结论。

**八、备注或说明**：可写上实验成功或失败的原因，实验后的心得体会、建议等。

|  |  |
| --- | --- |
| **1、实验名称** | 实验3-1 GDI绘图实验 |
| **2、实验日期** | **2022/9/22** |
| **3、实验目的** | |
| 1．理解设备环境在绘图中的作用  2．掌握绘图工具的创建，理解绘图工具和设备环境之间的关系  3．掌握绘图步骤，掌握绘图函数的使用 | |
| **4、实验环境** | |
| 1、vs2019 | |
| **5、实验的步骤和方法** | |
| 1. 将实验二中的窗口代码修改，在窗口处理函数中添加绘图代码：响应WM\_PAINT消息，在其中按照绘图步骤，用BeginPaint方法获取设备环境句柄，创建彩色的、具有某种样式的画笔和画刷，选入设备环境，在窗口上画椭圆、矩形等图形，这时创建并选入设备环境的画笔、画刷将作用在绘制图形上，最后删除画笔画刷、用EndPaint释放设备环境。调试运行，观察程序效果。 2. 在第1题的基础上，修改代码，在键盘按键消息（WM\_KEYDOWN）中，使用GetDC/ReleaseDC方法获取/释放设备环境，实现原来的画图操作。编译运行，把窗口拖动、最小化之后再还原窗口，看看出现什么问题，结合大课的ppt考虑原因。 3. 用GDI绘制一个游戏界面，如下图所示。其中，左边绿色部分，是游戏控制面板，上面有两个按钮或文本框，记录玩家姓名、分数等，右边是个类似棋盘的游戏内容面板，其中分布了很多条线，构成棋盘的样子。窗口不可以最大化、不可以调整大小。注意：游戏面板中的多条线段，可以循环多次绘制；窗口的类型，可以查询MSDN中CreateWindow的style参数。    思考题 查询MSDN，学习Polygon函数的使用，画出黄色五角星，然后尝试画五星红旗。注意观察五星红旗的样子，其中每个五角星每个角都严格相同，每个小五角星都朝向大五角星，所以在你的程序中应该有严格的计算来控制大小、位置，而不是随意设置一些值。 | |
| **6、数据记录和计算** | |
| **（代码编写思路及关键代码）**   1. 代码：   LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)  {  HDC hdc;//  PAINTSTRUCT ps;//  HBRUSH hBrush, hOldBrush;//  HPEN hPen, hOldPen;//  switch (message)  {  case WM\_PAINT:  hdc = BeginPaint(hwnd, &ps);//  hBrush = CreateHatchBrush(HS\_DIAGCROSS, RGB(255, 100, 100));//  hPen = CreatePen(PS\_DASHDOTDOT, 2, RGB(0, 0, 200));//  hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);//  hOldPen = (HPEN)SelectObject(hdc, hPen);//  Rectangle(hdc, 40, 220, 230, 350);//  Ellipse(hdc, 40 + 300, 220, 230 + 300, 350);//  SelectObject(hdc, hOldBrush);//  SelectObject(hdc, hOldPen);//  DeleteObject(hBrush);//  DeleteObject(hPen);//  EndPaint(hwnd, &ps);//  return 0;  break;  case WM\_KEYDOWN:  if (wParam == VK\_ESCAPE)  DestroyWindow(hwnd);  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hwnd, message, wParam, lParam);  }  return 0;  }   1. 代码   LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)  {  switch (message)  {  case WM\_PAINT:  return 0;  case WM\_KEYDOWN:  hdc = BeginPaint(hwnd, &ps);//  Game\_Paint(hwnd);//  EndPaint(hwnd, &ps);//  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hwnd, message, wParam, lParam);  }  return 0;  }  BOOL Game\_Init(HWND hwnd)  {  hdc = GetDC(hwnd);  Game\_Paint(hwnd);  ReleaseDC(hwnd, hdc);  return TRUE;  }  VOID Game\_Paint(HWND hwnd)  {  hBrush = CreateHatchBrush(HS\_DIAGCROSS, RGB(255, 100, 100));//  hPen = CreatePen(PS\_DASHDOTDOT, 2, RGB(0, 0, 200));//  hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);//  hOldPen = (HPEN)SelectObject(hdc, hPen);//  Rectangle(hdc, 40, 220, 230, 350);//  Ellipse(hdc, 40 + 300, 220, 230 + 300, 350);//  SelectObject(hdc, hOldBrush);//  SelectObject(hdc, hOldPen);//  DeleteObject(hBrush);//  DeleteObject(hPen);//  }   1. 代码：   HWND hwnd = CreateWindow(  L"xiang",  WINDOW\_TITLE,  WS\_OVERLAPPEDWINDOW ^ WS\_MAXIMIZEBOX ^ WS\_THICKFRAME,//  CW\_USEDEFAULT,  CW\_USEDEFAULT,  WINDOW\_WIDTH,  WINDOW\_HEIGHT,  NULL,  NULL,  hInstance,  NULL  );  BOOL Game\_Init(HWND hwnd)  {  hdc = GetDC(hwnd);  Game\_Paint(hwnd);  ReleaseDC(hwnd, hdc);  return TRUE;  }  VOID Game\_Paint(HWND hwnd)  {  int Round = 15; // 圆角角度  // 左边绿色框  hBrush = CreateSolidBrush(RGB(19, 197, 123));//  hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);//  hPen = CreatePen(PS\_SOLID, 4, RGB(19 + 30, 197 + 30, 123 + 30));//  hOldPen = (HPEN)SelectObject(hdc, hPen);//  RoundRect(hdc, 10, 10, 180, 560, Round, Round);//  // 左边俩小框  hPen = CreatePen(PS\_SOLID, 2, RGB(171 + 30, 93 + 30, 93 + 30));//  hOldPen = (HPEN)SelectObject(hdc, hPen);//  hBrush = CreateSolidBrush(RGB(171, 93, 93));//  hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);//  RoundRect(hdc, 40, 400, 140 + 10, 400 + 30, Round, Round);//  // 左边俩小框  hBrush = CreateSolidBrush(RGB(171, 93, 93));//  hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);//  RoundRect(hdc, 40, 400 + 50, 140 + 10, 400 + 80, Round, Round);//  // 右边红色框  hBrush = CreateSolidBrush(RGB(221, 82, 82));//  hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);//  hPen = CreatePen(PS\_SOLID, 4, RGB(221 + 30, 82 + 30, 82 + 30));//  hOldPen = (HPEN)SelectObject(hdc, hPen);//  RoundRect(hdc, 185, 10, 185 + 600, 560, Round, Round);//  // 右边线条  hPen = CreatePen(PS\_SOLID, 2, RGB(0, 0, 0));//  hOldPen = (HPEN)SelectObject(hdc, hPen);//  for (int x = 185 + 50; x < (185 + 600); x += 50)//  {  MoveToEx(hdc, x, 10, NULL);//  LineTo(hdc, x, 560);//  }  for (int y = 10 + 50; y < 560; y += 50)//  {  MoveToEx(hdc, 185, y, NULL);//  LineTo(hdc, 185 + 600, y);//  }  SelectObject(hdc, hOldBrush);//  SelectObject(hdc, hOldPen);//  DeleteObject(hBrush);//  DeleteObject(hOldPen);//  }  思考题：   1. 代码：   LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)  {  HDC hdc;//  PAINTSTRUCT ps;//  HBRUSH hBrush, hOldBrush;//  HPEN hPen, hOldPen;//  switch (message)  {  case WM\_PAINT:  hdc = BeginPaint(hwnd, &ps);//  hBrush = (HBRUSH)CreateSolidBrush(RGB(255, 0, 0));//  hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);//  // 红色背景  Rectangle(hdc, 0, 0, 900, 600);//    // 交叉两条线  MoveToEx(hdc, 450, 0, NULL);//  LineTo(hdc, 450, 600);//  MoveToEx(hdc, 0, 300, NULL);//  LineTo(hdc, 900, 300);//  // 左上角的线  for (int x = 0; x < 900 / 2; x += (900 / 2 / 15))//  {  MoveToEx(hdc, x, 0, NULL);//  LineTo(hdc, x, 600 / 2);//  }  for (int y = 0; y < 600 / 2; y += (600 / 2 / 10))//  {  MoveToEx(hdc, 0, y, NULL);//  LineTo(hdc, 900 / 2, y);//  }  // 画出大五角星的圆心  hBrush = (HBRUSH)CreateSolidBrush(RGB(250, 244, 8));//  hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);//  Ellipse(hdc, 5 \* (900 / 2 / 15) - 5, 5 \* (600 / 2 / 10) - 5, 5 \* (900 / 2 / 15) + 5, 5 \* (600 / 2 / 10) + 5);//  SelectObject(hdc, hOldBrush);//  DeleteObject(hBrush);//  EndPaint(hwnd, &ps);//  return 0;  case WM\_KEYDOWN:  if (wParam == VK\_ESCAPE)  DestroyWindow(hwnd);  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hwnd, message, wParam, lParam);  }  return 0;  }  二、代码：  #include <Windows.h>  #include <cmath>////  #define WINDOW\_WIDTH 950  #define WINDOW\_HEIGHT 650  #define WINDOW\_TITLE L"实验4"  #define PI 3.14159265358979323846////  // 全局变量声明部分  int nWidth = 900 / 2 / 15;////  int nHeight = 600 / 2 / 10;////  POINT ptOrg;////  // 全局函数声明部分  LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam);  void SetFivePoints(POINT\* pts, int nCount, int r, int angle, POINT pOrg);////  void DrawFiveStart(HDC hdc);////  void DrawSmallFiveStart(HDC hdc, int angle, POINT ptOrg);////  int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)  {  WNDCLASS wndClass = { 0 };  wndClass.style = CS\_HREDRAW | CS\_VREDRAW;  wndClass.lpfnWndProc = WndProc;  wndClass.cbClsExtra = 0;  wndClass.cbWndExtra = 0;  wndClass.hInstance = hInstance;  wndClass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);  wndClass.hCursor = LoadCursor(NULL, IDC\_ARROW);  wndClass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);  wndClass.lpszClassName = L"xiang";  wndClass.lpszMenuName = NULL;  if (!RegisterClass(&wndClass))  {  MessageBox(NULL, L"窗口类创建失败", L"wndClass created fail!", 0);  }  HWND hwnd = CreateWindow(  L"xiang",  WINDOW\_TITLE,  WS\_OVERLAPPEDWINDOW,  CW\_USEDEFAULT,  CW\_USEDEFAULT,  WINDOW\_WIDTH,  WINDOW\_HEIGHT,  NULL,  NULL,  hInstance,  NULL  );    if (!hwnd)  {  MessageBox(hwnd, L"窗口创建失败", L"window created fail!", 0);  return 0;  }    ShowWindow(hwnd, nShowCmd);  UpdateWindow(hwnd);  MSG msg = { 0 };  while (msg.message != WM\_QUIT)  {  if (PeekMessage(&msg, 0, 0, 0, PM\_REMOVE))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  }  UnregisterClass(L"窗口注销", wndClass.hInstance);  return 0;  }  LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)  {  HDC hdc;//  PAINTSTRUCT ps;//  HBRUSH hBrush, hOldBrush;//  HPEN hPen, hOldPen;//  switch (message)  {  case WM\_PAINT:  hdc = BeginPaint(hwnd, &ps);//  hBrush = (HBRUSH)CreateSolidBrush(RGB(255, 0, 0));//  hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);//  // 红色背景  Rectangle(hdc, 0, 0, 900, 600);//    // 画出五角星  hBrush = (HBRUSH)CreateSolidBrush(RGB(250, 244, 8));//  hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);//  hPen = CreatePen(PS\_NULL, 0, RGB(0, 0, 0));////  hOldPen = (HPEN)SelectObject(hdc, hPen);////  DrawFiveStart(hdc);////  // 画出小的五角星  DrawSmallFiveStart(hdc, 70, { nWidth \* 10, nHeight \* 2 });////  DrawSmallFiveStart(hdc, 120, { nWidth \* 12, nHeight \* 4 });////  DrawSmallFiveStart(hdc, 90, { nWidth \* 12, nHeight \* 7 });////  DrawSmallFiveStart(hdc, 70, { nWidth \* 10, nHeight \* 9 });////  SelectObject(hdc, hOldBrush);//  DeleteObject(hBrush);//  SelectObject(hdc, hOldPen);////  DeleteObject(hPen);////  EndPaint(hwnd, &ps);//  return 0;  case WM\_KEYDOWN:  if (wParam == VK\_ESCAPE)  DestroyWindow(hwnd);  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hwnd, message, wParam, lParam);  }  return 0;  }  void SetFivePoints(POINT\* pts, int nCount, int r, int angle, POINT pOrg)////  {  for (int i = 0; i < nCount; i++)////  {  pts[i].x = pOrg.x + (LONG)(r \* cos((angle + i \* 72) \* PI / 180));////  pts[i].y = pOrg.y - (LONG)(r \* sin((angle + i \* 72) \* PI / 180));////  }  }  void DrawFiveStart(HDC hdc)  {  // 画出五角星  POINT pts[5] = { 0 };////  POINT pOrg = { 0 };////  pOrg.x = 5 \* nWidth;////  pOrg.y = 5 \* nHeight;////    SetPolyFillMode(hdc, WINDING);  SetFivePoints(pts, 5, 35 \* 3, 90, pOrg);////  POINT pNewPts[5] = { pts[0], pts[2], pts[4], pts[1], pts[3] };////  Polygon(hdc, pNewPts, 5);////  }  void DrawSmallFiveStart(HDC hdc, int angle, POINT ptOrg)////  {  POINT pts[5] = { 0 };////  SetFivePoints(pts, 5, nHeight, angle, ptOrg);////  POINT pNewPts[5] = { pts[0], pts[2], pts[4], pts[1], pts[3] };////  Polygon(hdc, pNewPts, 5);////  } | |
| **7、实验结果或结论（**实验结果怎么样？你从这个实验你学会了什么？得出了什么结论？） | |
| **（实验结果截图+文字说明）**   1. 效果图      1. 效果图：   拖动后：    最小化后再还原窗口：    两种情况都出现了渲染的图不同程度的被隐藏掉，是由于渲染函数放到了WM\_KEYDOWN里，而不在WM\_PAINT里导致的。每次窗口变动时，系统都会调用WM\_PAINT函数来重新渲染，由于函数移到了WM\_KEYDOWN，所以无法随着窗口的改动而事实渲染，只能靠按下键盘来实现渲染。  3、效果图：    思考题：   1. 效果图：     二、效果图： | |
| **8、备注或说明**  **）** | |
|  | |