**《Windows应用程序原理》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  | | |  |  |
|  | |  | | |  |  |
| **实验名称** | 实验二 编写Windows窗口应用程序实现图形输出 | | | | | |
| **实验时间** | 2024.4.9 | | **实验地点** | DS3304 | | |
| **实验性质** | | | **□验证性 ☑设计性 □综合性** | | | |
| 一、实验目的  1.熟悉Windows窗口应用程序开发环境和工具。  2.掌握在窗口应用程序中进行图形输出的方法。  3.学习使用键盘输入来控制图形的移动。  4.加深对图形绘制和用户交互的理解。  5.培养对Windows应用程序开发的兴趣和技能。  6.编写Windows窗口应用程序实现图形输出。 | | | | | | |
| 二、实验项目内容  1.编程实现简易俄罗斯方块，单矩形块。  2.从顶部中间出现，出现是矩形块填充色为蓝色。  3.由键盘方向键控制矩形左右移动，落到客户区底部时填充色为红色，自行定义每次移动的距离。 | | | | | | |
| 1. 实验过程或算法（源程序）   #include <windows.h>  #include <stdbool.h>  #define CELL 20  #define ROWS 25  #define COLS 15  static int g\_x = 0;  static int g\_y = 7;//初始位置  int prev\_x = 0;  int prev\_y = 0;  UINT timer\_id = 0;  byte g\_panel[ROWS][COLS] = { 0 };//正在移动的方块  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);  void DrawPanel(HDC hdc);//绘制表格  void FreeDown(HDC HDC);//下移  void LeftShift(HDC hdc);//左移  void RightShift(HDC hdc);//右移  void Eliminate(HDC hdc);//消行  int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,  PSTR szCmdLine, int iCmdShow)  {  static TCHAR szAppName[] = TEXT("Demo");  MSG msg;  HWND hwnd;  WNDCLASS wndclass;  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.lpszMenuName = NULL;  wndclass.lpszClassName = szAppName;  if (!RegisterClass(&wndclass)) {  MessageBox(NULL, TEXT("Program requires Windows NT!"),  szAppName, MB\_ICONERROR);  return 0;  }  hwnd = CreateWindow(szAppName, TEXT("RussianBox"),  WS\_OVERLAPPEDWINDOW,  0, 0,  0, 0,  NULL, NULL, hInstance, NULL);  ShowWindow(hwnd, iCmdShow);  UpdateWindow(hwnd);  while (GetMessage(&msg, NULL, 0, 0)){  TranslateMessage(&msg);  DispatchMessage(&msg);  }  return msg.wParam;  }  void DrawPanel(HDC hdc) {  int x, y;  RECT rect;  for (y = 0; y < ROWS; y++) {  for (x = 0; x < COLS; x++) {  //计算方块的边框范围  rect.top = y \* CELL + 1;  rect.bottom = (y + 1) \* CELL - 1;  rect.left = x \* CELL + 1;  rect.right = (x + 1) \* CELL - 1;  FrameRect(hdc, &rect, (HBRUSH)GetStockObject(BLACK\_BRUSH));  }  }  }  void RefreshPanel(HDC hdc)//刷新面板{  int x, y;  RECT rect;  HBRUSH h\_bSolid = CreateSolidBrush(RGB(0, 0, 255)),  h\_rSolid = CreateSolidBrush(RGB(255, 0, 0)),  h\_bEmpty = (HBRUSH)GetStockObject(WHITE\_BRUSH);  //if(NULL==block) return;  //刷屏  for (x = 0; x < ROWS; x++) {  for (y = 0; y < COLS; y++) {  //为避免刷新掉方块的边框，对rect的范围设置比边框范围小1  rect.top = x \* CELL + 2;  rect.bottom = (x + 1) \* CELL - 2;  rect.left = y \* CELL + 2;  rect.right = (y + 1) \* CELL - 2;  if (g\_panel[x][y]) {  if (x != g\_x ||y != g\_y)  FillRect(hdc, &rect, h\_rSolid);  else  FillRect(hdc, &rect, h\_bSolid);  }  else  FillRect(hdc, &rect, h\_bEmpty);  }  }  }  void FreeDown(HDC hdc) {  prev\_x = g\_x;  prev\_y = g\_y;  g\_panel[prev\_x][prev\_y] = 0;  g\_x++;  g\_panel[g\_x][g\_y] = 1;//到达底部  if (g\_x == ROWS - 1 || g\_panel[g\_x + 1][g\_y]) {  g\_x = 0;  g\_y = 7;  prev\_x = 0;  prev\_y = 0;  Eliminate(hdc);  }  }  void LeftShift(HDC hdc) {  if (g\_y && g\_panel[g\_x][g\_y - 1] != 1) {  prev\_x = g\_x;  prev\_y = g\_y;  g\_panel[prev\_x][prev\_y] = 0;  g\_y--;  g\_panel[g\_x][g\_y] = 1;  }  else return;  }  void RightShift(HDC hdc) {  if (g\_y < 14 && g\_panel[g\_x][g\_y + 1] != 1) {  prev\_x = g\_x;  prev\_y = g\_y;  g\_panel[g\_x][g\_y] = 0;  g\_y++;  g\_panel[g\_x][g\_y] = 1;  }  else  return;  }  void Eliminate(HDC hdc) {  int i;  int x, y;  bool isEliminate;  for (i = 0; i < COLS; i++) {  if (g\_panel[ROWS - 1][i])  if (i == COLS - 1)  isEliminate = true;  else  isEliminate = false;  else {  isEliminate = false;  break;  }  }  if (isEliminate) {  for (y = 0; y < COLS; y++)  g\_panel[ROWS - 1][y] = 0;  for (x = ROWS - 2; x >= 0; x--) {  for (y = 0; y < COLS; y++) {  if (g\_panel[x][y]) {  g\_panel[x + 1][y] = 1;  g\_panel[x][y] = 0;  }  }  }  }  RefreshPanel(hdc);  }  LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam){  HWND hdc;  PAINTSTRUCT ps;  switch (message)  {//函数调用  case WM\_CREATE:  MoveWindow(hwnd, 400, 10, CELL \* COLS + 16, CELL \* ROWS + 39, FALSE);  SetTimer(hwnd, 1, 1000, NULL);//计时器  return 0;  case WM\_TIMER:  hdc = GetDC(hwnd);  FreeDown(hdc);  RefreshPanel(hdc);  ReleaseDC(hwnd, hdc);  return 0;  case WM\_KEYDOWN:  hdc = GetDC(hwnd);  switch (wParam) {  case VK\_LEFT:  LeftShift(hdc);  break;  case VK\_RIGHT:  RightShift(hdc);  break;  case VK\_DOWN:  FreeDown(hdc);  break;  }  RefreshPanel(hdc);  ReleaseDC(hwnd, hdc);  case WM\_PAINT:  hdc = BeginPaint(hwnd, &ps);  g\_panel[g\_x][g\_y] = 1;  DrawPanel(hdc);  RefreshPanel(hdc);  EndPaint(hwnd, &ps);  return 0;  case WM\_DESTROY:  PostQuitMessage(0);  return 0;  }  return DefWindowProc(hwnd, message, wParam, lParam);  } | | | | | | |
| 1. 实验结果及分析和（或）源程序调试过程     图1 窗口弹出并从中间落下蓝色单矩形块    图2 落到客户区底部时填充色为红色    图3 矩形块可在客户区底部堆积    图4 当一行矩形堆积满后消除整行，同时上方矩形下落补上消除区域    图5 矩形块可通过键盘方向向右移动    图6 矩形块可通过键盘方向向左移动 | | | | | | |