Minesweeper based on Win32 Programming with C++

Developer Intent

This program was designed as a game like Microsoft Minesweeper and the interface should possess all functions a Minesweeper game should has, including

1. Display with N\*N squares
2. Open a square with the left mouse click. If it turns out to be a mine then game over. If not, the surrounding squares will be opened randomly and the corresponding figures will be shown on squares, meaning the game goes on. Mark a square you think is a mine with the right mouse click to make sure that the square is checked and the game goes on.
3. Different difficulty levels will be set to meet demands of various players, meaning the variety of the number N and the amount of mines.

Scopes

C++ language Windows programming Visual Studio IDE

Input

Mouse click function

case WM\_LBUTTONDOWN// change the status of the square

case WM\_LBUTTONUP// Open a square. If it turns out to be a mine then game over. If not, the surrounding squares will be opened randomly and the corresponding figures will be shown on squares, meaning the game goes on.

case WM\_RBUTTONUP// mark the square

Output

LRESULT CALLBACK WndProc{

case WM\_CREATE// initiate variables

case WM\_COMMAND// choose levels and to start or exit

case WM\_PAINT// draw elements on the window

}

User guide

Just the same as the classical minesweeper game or you can refer to the developer intent.

Code

**1.UI design—draw the window**

**Draw the interface of Minesweeper using GDI in windows programming by C++ (including different colors and effects of shadows to show 3D effects). The main task is to display the number on the square of surrounding mines after open it.**

case WM\_PAINT:

{

hdc = BeginPaint(hWnd, &ps);

// TODO: 在此处添加使用 hdc 的任何绘图代码...

GetClientRect(hWnd, &rect);

hBrush = CreateSolidBrush(RGB(200, 200, 200));

SetBkColor(hdc, RGB(200, 200, 200));

SelectObject(hdc, hBrush);

SelectObject(hdc, GetStockObject(NULL\_PEN));

Rectangle(hdc, 0, 0, rect.right, rect.bottom);

//绘制外边界

DrawBorder(hdc, 0, 0, rect.right, rect.bottom, SHADOW\_WIDTH3, color1, color2, true);

//绘制保留区

DrawBorder(hdc, SHADOW\_WIDTH3 + BORDER, SHADOW\_WIDTH3 + BORDER,

BOX\_WIDTH \* COLS + 2 \* SHADOW\_WIDTH2, RESERVED\_AREA + 2 \* SHADOW\_WIDTH2, SHADOW\_WIDTH2, color1, color2, false);

//绘制内边界

DrawBorder(hdc, SHADOW\_WIDTH3 + BORDER, SHADOW\_WIDTH3 + 2 \* BORDER + RESERVED\_AREA + 2 \* SHADOW\_WIDTH2,

BOX\_WIDTH \* COLS + 2 \* SHADOW\_WIDTH2, BOX\_HEIGHT\*ROWS + 2 \* SHADOW\_WIDTH2, SHADOW\_WIDTH2, color1, color2, false);

//绘制雷区的box

xStart = mineRect.left;

yStart = mineRect.top;

RECT rect0;

for (i = 0; i < ROWS; i++)

{

for (j = 0; j < COLS; j++)

{

if (!box[i][j].isOpened)

DrawBorder(hdc, xStart, yStart, BOX\_WIDTH - 2, BOX\_HEIGHT - 2, SHADOW\_WIDTH2,

color1, color2, true);

else

{

DrawBorder(hdc, xStart, yStart, BOX\_WIDTH - 2, BOX\_HEIGHT - 2, SHADOW\_WIDTH2,

color3, color3, true);

DrawBorder(hdc, xStart, yStart, BOX\_WIDTH, BOX\_HEIGHT, SHADOW\_WIDTH1,

color4, color4, true);

if (box[i][j].roundMineNumber != 0)

{

rect0.left = mineRect.left + BOX\_WIDTH \* j;

rect0.right = rect0.left + BOX\_WIDTH;

rect0.top = mineRect.top + BOX\_HEIGHT \* i;

rect0.bottom = rect0.top + BOX\_HEIGHT;

wsprintf(szBuffer, TEXT("%d"), box[i][j].roundMineNumber);

//输出周围雷的个数

DrawText(hdc, szBuffer, -1, &rect0, DT\_SINGLELINE | DT\_CENTER | DT\_VCENTER);

}

}

xStart += BOX\_WIDTH;

}

yStart += BOX\_HEIGHT;

xStart = SHADOW\_WIDTH3 + BORDER + SHADOW\_WIDTH2;

}

EndPaint(hWnd, &ps);

}

return 0;

**2.Mine area design—specific function**

**The entire area is designed as a N\*N array structure to save data and all the functions below are based this framework.**

//draw the border of squares

void DrawBorder(HDC hdc, int x, int y, int width, int height, int BorderWidth, COLORREF &color1, COLORREF &color2, bool WTop)

{

int i;

HPEN hpen1, hpen2;

hpen1 = CreatePen(PS\_SOLID, 1, color1);

hpen2 = CreatePen(PS\_SOLID, 1, color2);

if (WTop)

SelectObject(hdc, hpen2);

else

SelectObject(hdc, hpen1);

for (i = 0; i < BorderWidth; i++)

{

MoveToEx(hdc, x + i, y + i, NULL);

LineTo(hdc, x + width - i, y + i);

MoveToEx(hdc, x + i, y + i, NULL);

LineTo(hdc, x + i, y + height - i);

}

if (WTop)

SelectObject(hdc, hpen1);

else

SelectObject(hdc, hpen2);

for (i = 0; i < BorderWidth; i++)

{

MoveToEx(hdc, x + width - i, y + height - i, NULL);

LineTo(hdc, x + width - i, y + i);

MoveToEx(hdc, x + width - i, y + height - i, NULL);

LineTo(hdc, x + i, y + height - i);

}

}

//initiate the status of squares in a self-defined structure including all functions needed

void InitBox()

{

numberOpened = 0;

for (i = 0; i<ROWS; i++)

for (j = 0; j<COLS; j++)

{

box[i][j].containMine = false;

box[i][j].isOpened = false;

box[i][j].roundMineNumber = 0;

box[i][j].isMarked = false;

}

}

//count the number of surrounding mines

void ComputeRoundMineNo()

{

for (i = 0; i<ROWS; i++)

for (j = 0; j<COLS; j++)

{

if (i>0 && box[i - 1][j].containMine == true)

{

box[i][j].roundMineNumber++;

}

if (i>0 && j<COLS - 1 && box[i - 1][j + 1].containMine == true)

{

box[i][j].roundMineNumber++;

}

if (j<COLS - 1 && box[i][j + 1].containMine == true)

{

box[i][j].roundMineNumber++;

}

if (i<ROWS - 1 && j<COLS && box[i + 1][j + 1].containMine == true)

{

box[i][j].roundMineNumber++;

}

if (i<ROWS - 1 && box[i + 1][j].containMine == true)

{

box[i][j].roundMineNumber++;

}

if (i<ROWS - 1 && j>0 && box[i + 1][j - 1].containMine == true)

{

box[i][j].roundMineNumber++;

}

if (j>0 && box[i][j - 1].containMine == true)

{

box[i][j].roundMineNumber++;

}

if (i>0 && j>0 && box[i - 1][j - 1].containMine == true)

{

box[i][j].roundMineNumber++;

}

}

}

//set mines randomly using the rand() function

void RandomSetMines()

{

int row, col;

srand((UINT)GetCurrentTime());

for (i = 0; i<mineNumber;)

{

row = rand() % ROWS;

col = rand() % COLS;

if (box[row][col].containMine != true)

{

box[row][col].containMine = true;

i++;

}

}

}

//update the appearance of squares after they are opened automatically to differ from those still closed

void AutoOpenBoxes(HDC &hdc, int x, int y)

{

if (x >= 0 && y >= 0 && x < ROWS && y < COLS &&

box[x][y].roundMineNumber == 0 && !allRoundIsOpened(x, y))

{

drawRect(hdc, x - 1, y);

drawRect(hdc, x - 1, y + 1);

drawRect(hdc, x, y + 1);

drawRect(hdc, x + 1, y + 1);

drawRect(hdc, x + 1, y);

drawRect(hdc, x + 1, y - 1);

drawRect(hdc, x, y - 1);

drawRect(hdc, x - 1, y - 1);

AutoOpenBoxes(hdc, x - 1, y);

AutoOpenBoxes(hdc, x - 1, y + 1);

AutoOpenBoxes(hdc, x, y + 1);

AutoOpenBoxes(hdc, x + 1, y + 1);

AutoOpenBoxes(hdc, x + 1, y);

AutoOpenBoxes(hdc, x + 1, y - 1);

AutoOpenBoxes(hdc, x, y - 1);

AutoOpenBoxes(hdc, x - 1, y - 1);

}

}

//update the status of the game(window)

void UpdateTheWindow(HWND &hwnd, int rows, int cols, int mineNO)

{

ROWS = rows;

COLS = cols;

mineNumber = mineNO;

mineRect.left = SHADOW\_WIDTH3 + BORDER + SHADOW\_WIDTH2;

mineRect.right = mineRect.left + BOX\_WIDTH \* COLS;

mineRect.top = SHADOW\_WIDTH3 + 2 \* BORDER + RESERVED\_AREA + 3 \* SHADOW\_WIDTH2;

mineRect.bottom = mineRect.top + BOX\_HEIGHT \* ROWS;

box = new Box\*[COLS];

for (int k = 0; k< COLS; k++)

box[k] = new Box[ROWS];

InitBox();

RandomSetMines();

ComputeRoundMineNo();

int cx = BOX\_WIDTH \* COLS + 2 \* SHADOW\_WIDTH3 +

2 \* SHADOW\_WIDTH2 + 2 \* iSM\_BorderX + 2 \* BORDER;

int cy = BOX\_HEIGHT \* ROWS + 2 \* SHADOW\_WIDTH3 +

4 \* SHADOW\_WIDTH2 + 2 \* iSM\_BorderY + 3 \* BORDER + RESERVED\_AREA + iSM\_CaptionY + iSM\_MenuY;

MoveWindow(hwnd, 0, 0, cx, cy, true);

}

//draw rectangles as squares are opened and display the text to show if you win the game

void drawRect(HDC &hdc, int x, int y)

{

RECT rect1;

COLORREF color1 = RGB(200, 200, 200),

color2 = RGB(120, 120, 200);

TCHAR szBuffer[2] = { "0" };

if (x<0 || y<0 || x >= ROWS || y >= COLS)

return;

rect1.left = mineRect.left + BOX\_WIDTH \* y;

rect1.right = rect1.left + BOX\_WIDTH;

rect1.top = mineRect.top + BOX\_HEIGHT \* x;

rect1.bottom = rect1.top + BOX\_HEIGHT;

if (!box[x][y].isOpened)

{

DrawBorder(hdc, rect1.left, rect1.top,

BOX\_WIDTH - 2, BOX\_HEIGHT - 2, SHADOW\_WIDTH2, color1, color1, true);

DrawBorder(hdc, rect1.left, rect1.top,

BOX\_WIDTH, BOX\_HEIGHT, SHADOW\_WIDTH1, color2, color2, true);

if (!box[x][y].containMine && box[x][y].roundMineNumber != 0)

{

wsprintf(szBuffer, TEXT("%d"), box[x][y].roundMineNumber);

DrawText(hdc, szBuffer, -1, &rect1, DT\_SINGLELINE | DT\_CENTER | DT\_VCENTER);

}

box[x][y].isOpened = true;

numberOpened++;

if (numberOpened == ROWS \* COLS - mineNumber)

{

MessageBox(NULL, TEXT("祝贺你赢了!"), TEXT("扫雷"), MB\_ICONWARNING);

}

}

}

//calculate if all surrounding squares are opened and return a bool value to a certain square

bool allRoundIsOpened(int x, int y)

{

bool flag = true;

if (x > 0 && !box[x - 1][y].isOpened)

flag = false;

if (x > 0 && y < COLS - 1 && !box[x - 1][y + 1].isOpened)

flag = false;

if (y < COLS - 1 && !box[x][y + 1].isOpened)

flag = false;

if (x < ROWS - 1 && y< COLS - 1 && !box[x + 1][y + 1].isOpened)

flag = false;

if (x < ROWS - 1 && !box[x + 1][y].isOpened)

flag = false;

if (x < ROWS - 1 && y > 0 && !box[x + 1][y - 1].isOpened)

flag = false;

if (y > 0 && !box[x][y - 1].isOpened)

flag = false;

if (x > 0 && y > 0 && !box[x - 1][y - 1].isOpened)

flag = false;

return flag;

}

**3.Mouse interaction design—mouse respond function**

**Specific input of the game and it can be divided into several parts including left button up, left button down, right button up, right button down and mouse move.**

case WM\_LBUTTONDOWN:

hdc = GetDC(hWnd);

isLButtonDown = true;

pt.x = LOWORD(lParam);

pt.y = HIWORD(lParam);

grid.left = mineRect.left;

grid.right = mineRect.right;

grid.top = mineRect.top;

grid.bottom = mineRect.bottom;

if (PtInRect(&grid, pt))

{

j = (pt.x - grid.left) / BOX\_WIDTH;

i = (pt.y - grid.top) / BOX\_HEIGHT;

grid.left += BOX\_WIDTH \* j;

grid.top += BOX\_HEIGHT \* i;

grid.right = grid.left + BOX\_WIDTH;

grid.bottom = grid.top + BOX\_HEIGHT;

hdc = GetDC(hWnd);

DrawBorder(hdc, grid.left, grid.top, BOX\_WIDTH - 2, BOX\_HEIGHT - 2, SHADOW\_WIDTH2, color3, color3, false);

DrawBorder(hdc, grid.left, grid.top, BOX\_WIDTH, BOX\_HEIGHT, SHADOW\_WIDTH1, color4, color4, false);

}

DeleteDC(hdc);

return 0;

case WM\_LBUTTONUP:

static HPEN hPen;

static RECT rect1;

hdc = GetDC(hWnd);

hPen = CreatePen(PS\_SOLID, 2, RGB(255, 255, 0));

SelectObject(hdc, hPen);

SetBkColor(hdc, RGB(200, 200, 200));

isLButtonDown = false;

rect1.left = mineRect.left + BOX\_WIDTH \* j;

rect1.right = rect1.left + BOX\_WIDTH;

rect1.top = mineRect.top + BOX\_HEIGHT \* i;

rect1.bottom = rect1.top + BOX\_HEIGHT;

if (!box[i][j].isOpened)

{

if (!box[i][j].containMine)

{

if (box[i][j].roundMineNumber != 0)

{

wsprintf(szBuffer, TEXT("%d"), box[i][j].roundMineNumber);

//输出周围雷的个数

DrawText(hdc, szBuffer, -1, &rect1, DT\_SINGLELINE | DT\_CENTER | DT\_VCENTER);

}

box[i][j].isOpened = true;

numberOpened++;

if (numberOpened == ROWS \* COLS - mineNumber)

{

MessageBox(hWnd, TEXT("祝贺你赢了!"), TEXT("扫雷"), MB\_ICONWARNING);

}

AutoOpenBoxes(hdc, i, j);

}

else

{

for (int k = 0; k<ROWS; k++)

for (int l = 0; l<COLS; l++)

{

if (box[k][l].containMine)//点到雷后,使所有的雷显现

{

rect.left = mineRect.left + BOX\_WIDTH \* l;

rect.top = mineRect.top + BOX\_HEIGHT \* k;

rect.right = rect.left + BOX\_WIDTH;

rect.bottom = rect.top + BOX\_HEIGHT;

DrawBorder(hdc, rect.left, rect.top, BOX\_WIDTH - 2, BOX\_HEIGHT - 2, SHADOW\_WIDTH2, color3, color3, false);

DrawBorder(hdc, rect.left, rect.top, BOX\_WIDTH, BOX\_HEIGHT, SHADOW\_WIDTH1, color4, color4, false);

DrawText(hdc, "\*", -1, &rect, DT\_SINGLELINE | DT\_CENTER | DT\_VCENTER);

}

}

MessageBox(hWnd, TEXT("游戏结束"), TEXT("扫雷"), MB\_ICONWARNING);

}

}

DeleteDC(hdc);

return 0;

case WM\_RBUTTONDOWN:

hdc = GetDC(hWnd);

isRButtonDown = true;

pt.x = LOWORD(lParam);

pt.y = HIWORD(lParam);

grid.left = mineRect.left;

grid.right = mineRect.right;

grid.top = mineRect.top;

grid.bottom = mineRect.bottom;

if (PtInRect(&grid, pt))

{

j = (pt.x - grid.left) / BOX\_WIDTH;

i = (pt.y - grid.top) / BOX\_HEIGHT;

grid.left += BOX\_WIDTH \* j;

grid.top += BOX\_HEIGHT \* i;

grid.right = grid.left + BOX\_WIDTH;

grid.bottom = grid.top + BOX\_HEIGHT;

hdc = GetDC(hWnd);

}

DeleteDC(hdc);

return 0;

case WM\_RBUTTONUP:

//static HPEN hPen;

hdc = GetDC(hWnd);

static RECT rect2;

hPen = CreatePen(PS\_SOLID, 2, RGB(255, 255, 0));

SelectObject(hdc, hPen);

SetBkColor(hdc, RGB(200, 200, 200));

isRButtonDown = false;

if (box[i][j].isMarked==false&&!box[i][j].isOpened)

{

rect2.left = mineRect.left + BOX\_WIDTH \* j;

rect2.right = rect2.left + BOX\_WIDTH;

rect2.top = mineRect.top + BOX\_HEIGHT \* i;

rect2.bottom = rect2.top + BOX\_HEIGHT;

DrawText(hdc, "?", -1, &rect2, DT\_SINGLELINE | DT\_CENTER | DT\_VCENTER);

box[i][j].isMarked = true;

}

else if (box[i][j].isMarked == true && box[i][j].isOpened == true)

{

rect.left = mineRect.left + BOX\_WIDTH \* j;

rect.right = rect.left + BOX\_WIDTH;

rect.top = mineRect.top + BOX\_HEIGHT \* i;

rect.bottom = rect.top + BOX\_HEIGHT;

//DrawBorder(hdc, grid.left, grid.top, BOX\_WIDTH - 2, BOX\_HEIGHT - 2, SHADOW\_WIDTH2, color1, color2, true);

//DrawBorder(hdc, grid.left, grid.top, BOX\_WIDTH, BOX\_HEIGHT, SHADOW\_WIDTH1, color3, color3, true);

DrawText(hdc, " ", -1, &rect, DT\_SINGLELINE | DT\_CENTER | DT\_VCENTER);

box[i][j].isMarked = false;

}

else if (box[i][j].isMarked == true && !box[i][j].isOpened)

{

rect.left = mineRect.left + BOX\_WIDTH \* j;

rect.right = rect.left + BOX\_WIDTH;

rect.top = mineRect.top + BOX\_HEIGHT \* i;

rect.bottom = rect.top + BOX\_HEIGHT;

//DrawBorder(hdc, grid.left, grid.top, BOX\_WIDTH - 2, BOX\_HEIGHT - 2, SHADOW\_WIDTH2, color1, color2, true);

//DrawBorder(hdc, grid.left, grid.top, BOX\_WIDTH, BOX\_HEIGHT, SHADOW\_WIDTH1, color3, color3, true);

DrawText(hdc, " ", -1, &rect, DT\_SINGLELINE | DT\_CENTER | DT\_VCENTER);

box[i][j].isMarked = false;

}

else if (box[i][j].isMarked == false && box[i][j].isOpened == true)

{

break;

}

DeleteDC(hdc);

return 0;

case WM\_MOUSEMOVE:

hdc = GetDC(hWnd);

if (isLButtonDown)

{

pt.x = LOWORD(lParam);

pt.y = HIWORD(lParam);

rect.left = mineRect.left;

rect.right = mineRect.right;

rect.top = mineRect.top;

rect.bottom = mineRect.bottom;

if (PtInRect(&rect, pt))//判断是否在雷区

{

m = i; n = j;

j = (pt.x - rect.left) / BOX\_WIDTH;

i = (pt.y - rect.top) / BOX\_HEIGHT;

rect.left += BOX\_WIDTH \* j;

rect.top += BOX\_HEIGHT \* i;

rect.right = rect.left + BOX\_WIDTH;

rect.bottom = rect.top + BOX\_HEIGHT;

if (grid.left == rect.left && grid.right == rect.right

&&grid.bottom == rect.bottom && grid.top == rect.top)

return 0;

hdc = GetDC(hWnd);

if (!box[m][n].isOpened)//鼠标移走后,恢复未按下

{

DrawBorder(hdc, grid.left, grid.top, BOX\_WIDTH - 2, BOX\_HEIGHT - 2, SHADOW\_WIDTH2, color1, color2, true);

DrawBorder(hdc, grid.left, grid.top, BOX\_WIDTH, BOX\_HEIGHT, SHADOW\_WIDTH1, color3, color3, true);

}

//使当前的box成为按下状态

grid.left = rect.left;

grid.right = rect.right;

grid.bottom = rect.bottom;

grid.top = rect.top;

DrawBorder(hdc, grid.left, grid.top, BOX\_WIDTH - 2, BOX\_HEIGHT - 2, SHADOW\_WIDTH2, color3, color3, true);

DrawBorder(hdc, grid.left, grid.top, BOX\_WIDTH, BOX\_HEIGHT, SHADOW\_WIDTH1, color4, color4, true);

}

DeleteDC(hdc);

}

return 0;

Illustrations

