// Pong.cpp : Defines the entry point for the application.

//

#include "stdafx.h"

#include "Pong.h"

#define MAX\_LOADSTRING 100

// Global Variables:

HINSTANCE hInst; // current instance

WCHAR szTitle[MAX\_LOADSTRING]; // The title bar text

WCHAR szWindowClass[MAX\_LOADSTRING]; // the main window class name

WCHAR w\_ball\_t[] = L"BALLCLASS";

WCHAR w\_paddle\_t[] = L"PADDLECLASS";

// Forward declarations of functions included in this code module:

ATOM MyRegisterClass(HINSTANCE hInstance);

ATOM MyRegisterClassB(HINSTANCE hInstance);

ATOM MyRegisterClassP(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

LRESULT CALLBACK BallProc(HWND, UINT, WPARAM, LPARAM);

LRESULT CALLBACK PaddleProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: Place code here.

// Initialize global strings

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_PONG, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

MyRegisterClassB(hInstance);

MyRegisterClassP(hInstance);

// Perform application initialization:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_PONG));

MSG msg;

// Main message loop:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

HWND hWnd, hBall, hPaddle;

//

// FUNCTION: MyRegisterClass()

//

// PURPOSE: Registers the window class.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW 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\_PONG));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = CreateSolidBrush(RGB(255, 255, 0));

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_PONG);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

ATOM MyRegisterClassB(HINSTANCE hInstance)

{

WNDCLASSEX ballex;

ballex.cbSize = sizeof(WNDCLASSEX);

ballex.style = CS\_HREDRAW | CS\_VREDRAW;

ballex.lpfnWndProc = BallProc;

ballex.cbClsExtra = 0;

ballex.cbWndExtra = 0;

ballex.hInstance = hInstance;

ballex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_PONG));

ballex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

ballex.hbrBackground = CreateSolidBrush(RGB(255, 0, 0));

ballex.lpszMenuName = MAKEINTRESOURCEW(IDC\_PONG);

ballex.lpszClassName = w\_ball\_t;

ballex.hIconSm = LoadIcon(ballex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&ballex);

}

ATOM MyRegisterClassP(HINSTANCE hInstance)

{

WNDCLASSEX paddleex;

paddleex.cbSize = sizeof(WNDCLASSEX);

paddleex.style = CS\_HREDRAW | CS\_VREDRAW;

paddleex.lpfnWndProc = WndProc;

paddleex.cbClsExtra = 0;

paddleex.cbWndExtra = 0;

paddleex.hInstance = hInstance;

paddleex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_PONG));

paddleex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

paddleex.hbrBackground = (HBRUSH)(COLOR\_WINDOW - 1);

paddleex.lpszMenuName = MAKEINTRESOURCEW(IDC\_PONG);

paddleex.lpszClassName = w\_paddle\_t;

paddleex.hIconSm = LoadIcon(paddleex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&paddleex);

}

//

// FUNCTION: InitInstance(HINSTANCE, int)

//

// PURPOSE: Saves instance handle and creates main window

//

// COMMENTS:

//

// In this function, we save the instance handle in a global variable and

// create and display the main program window.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Store instance handle in our global variable

hWnd = CreateWindow(szWindowClass, L"PONG", WS\_OVERLAPPEDWINDOW | WS\_EX\_CONTROLPARENT,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

//Set a window for a ball

hBall = CreateWindow(w\_ball\_t, L"BALL", WS\_CHILD | WS\_VISIBLE,

CW\_USEDEFAULT, 0, 50, 100, hWnd, nullptr, hInstance, nullptr);

//Set a window for a paddle

hPaddle = CreateWindow(w\_paddle\_t, L"PADDLE", WS\_CHILD | WS\_VISIBLE,

58, 226, 80, 15,hWnd, nullptr, hInstance, nullptr);

//Set window in the middle

RECT rc;

GetWindowRect(hWnd, &rc);

int xPos = (GetSystemMetrics(SM\_CXSCREEN) - rc.right) / 2;

int yPos = (GetSystemMetrics(SM\_CYSCREEN) - rc.bottom) / 2;

MoveWindow(hWnd, xPos, yPos, 200, 300, 0);

SetWindowLong(hWnd, GWL\_EXSTYLE, GetWindowLong(hWnd, GWL\_EXSTYLE) | WS\_EX\_LAYERED);

SetLayeredWindowAttributes(hWnd, 0, (255 \* 80) / 100, LWA\_ALPHA);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hBall, nCmdShow);

if (!hBall)

{

return FALSE;

}

if (!hPaddle)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

//

// FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// PURPOSE: Processes messages for the main window.

//

// WM\_COMMAND - process the application menu

// WM\_PAINT - Paint the main window

// WM\_DESTROY - post a quit message and return

//

//

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Parse the menu selections:

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\_GETMINMAXINFO:

{

MINMAXINFO \*minMaxInfo = (MINMAXINFO\*)lParam;

minMaxInfo->ptMaxSize.x = minMaxInfo->ptMaxTrackSize.x = 200;

minMaxInfo->ptMaxSize.x = minMaxInfo->ptMinTrackSize.x = 200;

minMaxInfo->ptMaxSize.y = minMaxInfo->ptMaxTrackSize.y = 300;

minMaxInfo->ptMaxSize.y = minMaxInfo->ptMinTrackSize.y = 300;

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

// TODO: Add any drawing code that uses hdc here...

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

LRESULT CALLBACK BallProc(HWND hBall, UINT message, WPARAM wParam, LPARAM lParam)

{

static int x = 10;

static int y = 10;

static int lx = 0;

static int rx = 20;

static int ty = 0;

static int by = 20;

switch (message)

{

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hBall, &ps);

// TODO: Add any drawing code that uses hdc here...

EndPaint(hBall, &ps);

}

break;

case WM\_CREATE:

{

MoveWindow(hBall, 80, 140, 20, 20, NULL);

SetTimer(hBall, 7, 50, NULL);

//Set region on window hBall

HRGN region = CreateEllipticRgn(0, 0, 20, 20);

SetWindowRgn(hBall, region, true);

break;

}

case WM\_TIMER:

{

if (rx >= 190)

x = -abs(x);

if (lx <= 0)

x = abs(x);

if (by >= 226)

y = -abs(y);

if (ty <= 0)

y = abs(y);

lx = lx + x;

ty = ty + y;

rx = rx + x;

by = by + y;

MoveWindow(hBall, lx, ty, rx - lx, by - ty, TRUE);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hBall, message, wParam, lParam);

}

return 0;

}

LRESULT CALLBACK PaddleProc(HWND hPaddle, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hPaddle, &ps);

// TODO: Add any drawing code that uses hdc here...

EndPaint(hPaddle, &ps);

}

break;

case WM\_MOUSEHOVER:

{

case WM\_MOUSEMOVE:

{

LPPOINT pos;

// GetCursorPos(pos);

break;

}

break;

}

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hPaddle, message, wParam, lParam);

}

return 0;

}

// Message handler for about box.

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;

}