# IMAT3606: Lab 3

In this lab you will be familiarised with one of the platforms for event driven programming and concepts of messaging (events) and message handlers (event handlers). Microsoft windows API has been chosen for these examples due to its wide range of applications in tool development for games but it is just an example and you do not have to use it for your coursework 9alternatives are SFML, SDL, FreeGLUT, etc).

**Windows API questions:**

Please write the following programs on your own. Please submit your zipped project folder and completed word file to the following link by the end of the session and an updated version before the next lab session.

In weeks 3 and 4, event driven programming and specially Microsoft Windows API programming will be examined. You should be able to solve these problems without reference to the lecture notes and by only reading the instructions and the hints, but lectures will definitely help to get a better understanding of why the code is the way it is.

1. A dialogue based windows application: The easiest type of windows application is a dialogue based application. Here some controls are added on a dialogue window and actions will be executed based on choices by the user from the dialogue box.

Create an empty win32 application project in Visual studio and copy the following code into an added new C++ file.

#include <windows.h>   
int WINAPI WinMain(HINSTANCE hinstance, HINSTANCE hprevinstance, LPSTR lpcmdline, int ncmdshow)

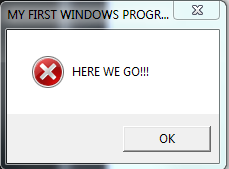
{

MessageBox(NULL, TEXT("HERE WE GO!!!"), TEXT("MY FIRST WINDOWS PROGRAM"), MB\_OK | MB\_ICONEXCLAMATION );

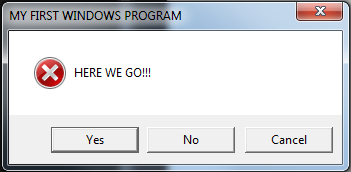
return(0);

}

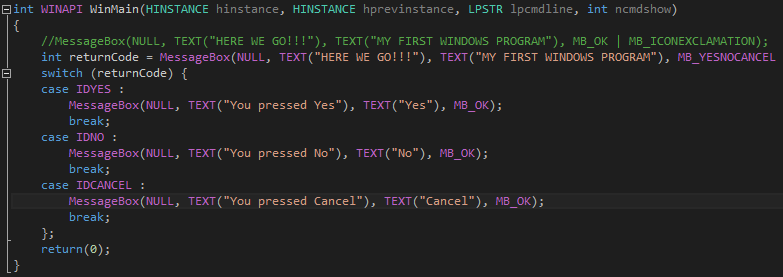
1. Change the exclamation mark on the window to an error icon; what other options are available? (use autocomplete by using ctrl+Space after typing few characters)



1. Change the type of the dialogue box from having only an OK button to the one with Yes, No and Cancel buttons (read the helps file at: <http://msdn.microsoft.com/en-us/library/windows/desktop/ms645505(v=vs.85).aspx> for MB\_ constants you will need to use)



1. Using the example in the help file or by following these steps define some actions for each button:
   1. Record the return value of the MessageBox function as an int;
   2. Define switch statement on the return value and create cases for IDCANCEL, IDYES and IDNO. In each case define a new message box to confirm what has been pressed.



1. Consider the following program from Charles Petzold’s book on Programming windows (1998) and answer the following questions:
2. #include <windows.h>
3. #include < ------ >
4. #define NUM 1000
5. #define TWOPI (2 \* 3.14159)
6. LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
7. int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,PSTR szCmdLine, int iCmdShow)
8. {
9. static TCHAR szAppName[] = TEXT ("SineWave") ;
10. HWND hwnd ;
11. MSG msg ;
12. WNDCLASS wndclass ;
13. wndclass.style = CS\_HREDRAW | CS\_VREDRAW ;
14. wndclass.lpfnWndProc = ------- ;
15. wndclass.cbClsExtra = 0 ;
16. wndclass.cbWndExtra = 0 ;
17. wndclass.hInstance = hInstance ;
18. wndclass.hIcon = LoadIcon (NULL, IDI\_APPLICATION) ;
19. wndclass.hCursor = LoadCursor (NULL, IDC\_ARROW) ;
20. wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE\_BRUSH) ;
21. wndclass.lpszMenuName = NULL ;
22. wndclass.lpszClassName = szAppName ;
23. if (!RegisterClass (&wndclass))
24. {
25. MessageBox (NULL, TEXT("Program requires Windows NT!"),szAppName, MB\_ICONERROR) ;
26. return 0 ;
27. }
28. hwnd = CreateWindow (szAppName, TEXT ("Sine Wave Using Polyline"),
29. WS\_OVERLAPPEDWINDOW,
30. CW\_USEDEFAULT, CW\_USEDEFAULT,
31. CW\_USEDEFAULT, CW\_USEDEFAULT,
32. NULL, NULL, hInstance, NULL) ;
33. --------------------;
34. UpdateWindow (hwnd) ;
35. while (GetMessage (&msg, NULL, 0, 0))
36. {
37. TranslateMessage (&msg) ;
38. DispatchMessage (&msg) ;
39. }
40. return msg.wParam ;
41. }
42. LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
43. {
44. static int cxClient, cyClient, prevX, prevY ;
45. HDC hdc ;
46. int i ;
47. PAINTSTRUCT ps ;
48. POINT apt [NUM] ;
49. switch (message)
50. {
51. case WM\_SIZE:
52. cxClient = LOWORD (lParam) ;
53. cyClient = HIWORD (lParam) ;
54. return 0 ;
55. case WM\_PAINT:
56. hdc = BeginPaint (hwnd, &ps) ;
57. MoveToEx (hdc, 0, cyClient / 2, NULL) ;
58. LineTo (hdc, cxClient, cyClient / 2) ;
59. for (i = 0 ; i < NUM ; i++)
60. {
61. apt[i].x = i \* cxClient / NUM ;
62. apt[i].y = (int) (cyClient / 2 \* (1 - sin (TWOPI \* i / NUM))) ;
63. }
64. Polyline (hdc, apt, NUM) ;
65. return 0 ;
66. case WM\_DESTROY:
67. PostQuitMessage (0) ;
68. return 0 ;
69. }
70. return DefWindowProc (hwnd, message, wParam, lParam) ;
71. }

* 1. Which library is needed to be included in line 2? (hint: just read the code and see what functions are used. Alternatively, just comment the relevant #include line and see what complaints you receive from the compiler)

**The maths library cmath**

* 1. Personalise the window by changing its title; (hint: again read the code and try your guess)

**Line 8 “static TCHAR szAppName[] = TEXT ("SineWave");” change the string value of TEXT();**

* 1. Fill the gap in line 13; Hint: lpfnWndProc is a long pointer pointing at windows procedure function. This function is where we define what happens with each key-press or other messages from the user or the system. This line of code defines a call back to that function and simply specifies where these actions are defined.

**WndProc**

* 1. What function needs to be called in line 32 before UpdateWindow(hwnd)? (hint: run the program without the line and you will see! Or rather not see! ;-) ). If this hint was not enough create a new non-empty win32 project in visual studio and find the relevant line before updating the window.

**ShowWindow(hwnd, iCmdShow);**

* 1. Change the graph to draw only the half above the x-axis (there are many ways of doing this, try to find the easiest!)

**int yValue = (int)(cyClient / 2 \* (1 - sin(TWOPI \* i / NUM)));**

**if (yValue < cyClient / 2) //If yValue is above midpoint**

**apt[i].y = yValue;**

**else //Else just set to midpoint**

**apt[i].y = cyClient / 2;**

* 1. What is the role of NUM=1000? Change this value and compile/run the program for NUM=10, 50, 100, 500 to see the effect of NUM. If you were renaming this variable what would you call it?

**Defines the number of partitions or steps along the x-axis, changing this value will change the resolution of the curve being drawn with higher values producing a smoother curve.**

**I would rename it to maxXPartitions**

* 1. What are cxClient and cyClient? You can debug the program and insert a breakpoint at the lines these variable are assigned to find out. Do they have anything to do with the size of the window?

**They are the window’s new width and height values on a WM\_SIZE message.**

* 1. Change the size of the window and find its exact width and height using the same breakpoints.

**Width = 1903, Height = 1112**

* 1. In the following lines:  
     for (i = 0 ; i < NUM ; i++) {

apt[i].x = i \* cxClient / NUM ;

apt[i].y = (int) (cyClient / 2 \* (1 - sin (TWOPI \* i / NUM))) ;

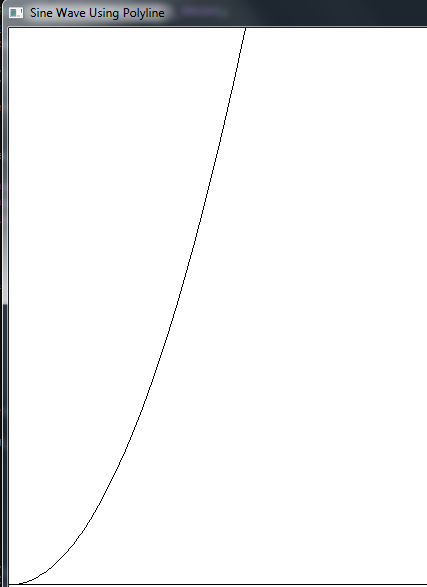
}

Explain the role of apt[] and write the mathematical formulas y in terms of x the form y=f(x); Hint: replace the values of NUM, cxClient and cyClient with constants and find I in terms of x and replace it in the second formula. Draw the first 5 values of x and y in the loop graphically.

**apt[] is the array of points the Polyline function will draw lines between.**

**y = f(a / 2(1-sin(2piX/b)))**

* 1. Use this program to draw the following functions:
     1. Y=x2



* + 1. Y=Log(x)



1. In the following example Explain what affect the following code will create? Try doing this only by reading the code and understanding the loop). Also note the functions used in this example as you may need them in next week’s problems:

hdc = GetDC(hwnd);

for (int index=0; index < 100; index++)

{

int x = rand()%1000;

int y = rand()%700;

COLORREF color = RGB(rand()%255,rand()%255,rand()%255);

SetPixel(hdc, x,y, color);

}

ReleaseDC(hwnd, hdc);

**This loop will select 100 random pixels (Between 1000x700) and assign a random RGB colour on every WM\_PAINT message**

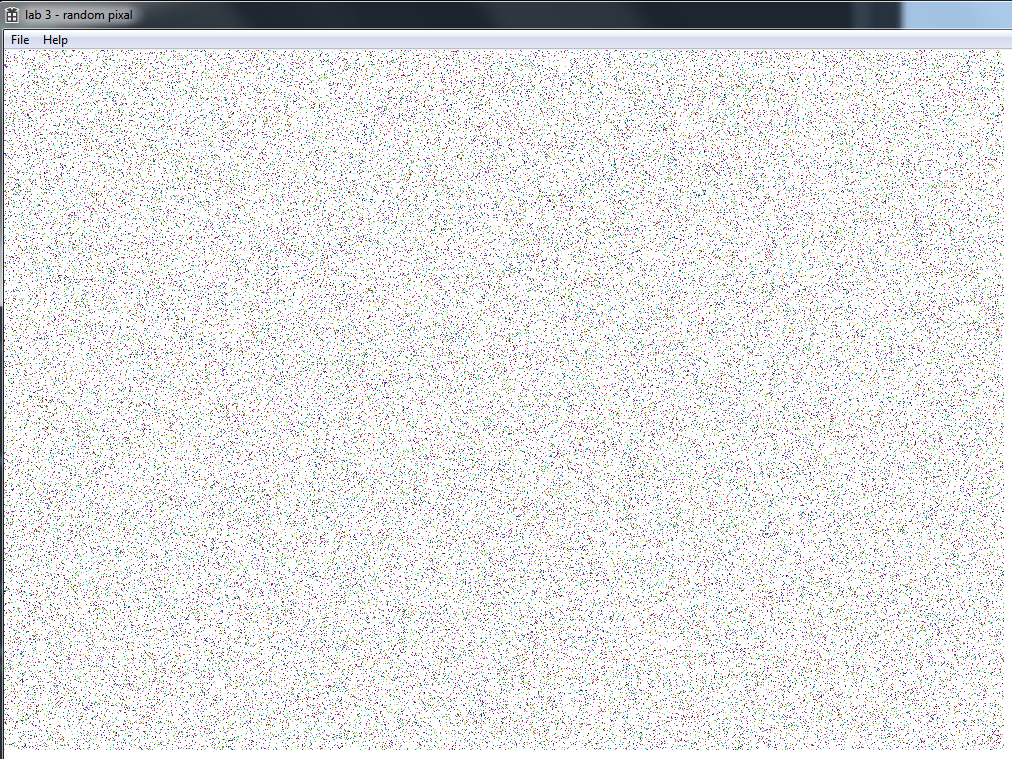
* 1. Create a non-empty Win32 project and insert the loop inside WM\_PAINT message handler in the windows procedure function (Hint: you will have to change the way you receive your device context)

Run on every paint call



* 1. Use the original code in another message (for example WM\_KEYDOWN)

Example of holding down the space key for a while



* 1. Print some text on windows on the position of a left click of mouse button. What would happen if you try to create the text using double click? (note that you will need to change the style of the window to accept double clicks)