CHAPTER EIGHT
MOUSE AND KEY MESSAGS

8.1. Introduction

Each message sent by Windows 98 operating system is represented by unique 32-bit integer value and each message corresponds to some event. Each message has standard macro name.

For Instance:

WM_CHAR message is sent each time a key is pressed.
Some common windows 98 message macros are:

WM_CHAR	WM+_PAINT	WM_COMMAND
WM_DESTROY	WM_QUIT	WM_LABUTTONUP

Along with the message two more values are accompanied one of the value is of type WPARAM & the other is of type LPARAM. The values are called as wParam and lParam.
The wParam and lParam typically hold things like cursor or mouse coordinates, the value of keypress, or a system related value.

8.2. Responding to Mouse Message

Mouse Message:
We can say window operating system as a mouse based operating system, as all program respond to the mouse input.
Some mouse message that window 98 responds are:

WM_LBUUTONDOWN	WM_RBUTTONDOWN	WM_RBUTTONCLK
WM_LBUTTONUP	WM_RBUTTONUP	WM_LBUTTONCLK

Most computers use a two-button mouse.
Window 98 is capable of handling a mouse wire up to three buttons – left, middle, right.

8.3. PROCESSING WM_LBUTTONDOWN

When above message is received, the mouse current X,Y position is specified in LOWORD (lpram) and HIWORD(lparam), respectively.
The value of wParam supplies information about the state of the mouse and keyboard.
For instance:
MK_CONTROL along with mouse button, CTRL key is pressed.
MK_SHIFT along with mouse button, SHIFT key is pressed.
MK_MBUTTON middle button is down when right button is pressed.
MK_LBUTTON left button is down when left button is pressed.
MK_RBUTTON right button is down when right button is pressed.

Button up message:
Your program receives two messages whenever a mouse is clicked.
First is **Button-down** and second is **Button-up** message.

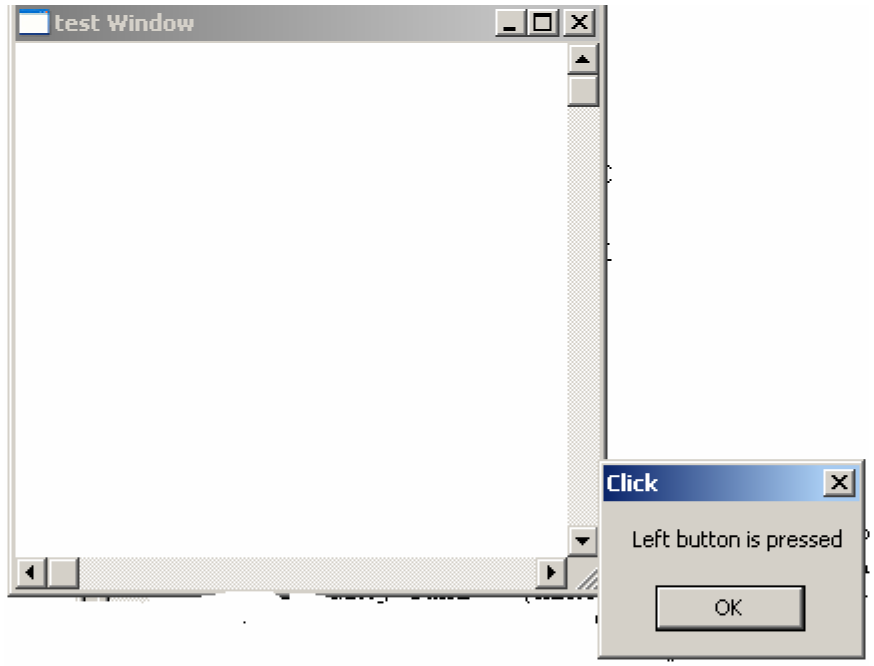
Double Click:
First enable your program for double click message.
For instance wc.style=CS_DBCLICKS;
Second, add message respond code for double click in your windowFunc().
WM_LBUTTONDBCLICK.

Following program handles mouse message.

Program 8.1: Handling mouse message

```
// mouse2.cpp : Defines the entry point for the application.
//

#include "stdafx.h"

#include <windows.h>

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

int WINAPI WinMain(HINSTANCE hInst,
                   HINSTANCE hPrev,
                   LPSTR     nCmd,
                   int       nShow)
{
        WNDCLASS wc;
   HWND hwnd;
   MSG msg;

   wc.cbClsExtra=NULL;
   wc.cbWndExtra    = 0;
   wc.hInstance     = hInst;
   wc.hIcon         = LoadIcon(hInst, IDI_APPLICATION);
   wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
   wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
        wc.lpfnWndProc   =WndProc;
   wc.lpszMenuName  = NULL;
   wc.lpszClassName = "MyWindow";
        wc.style=CS_DBLCLKS;

   if(!RegisterClass(&wc))
   {
      MessageBox(NULL, "Not Registered", "SEE THAT", MB_OK);
      return 0;
   }

        hwnd = CreateWindow(
      "MyWindow",
      "test Window",
      WS_OVERLAPPEDWINDOW|WS_VSCROLL|WS_HSCROLL,
      100, 100, 300,300,
      NULL, NULL, hInst, NULL);


   ShowWindow(hwnd, nShow);

   while(GetMessage(&msg, NULL, 0, 0) )
   {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
   };
        return msg.wParam;
}
LRESULT CALLBACK WndProc(HWND hWnd, UINT mesg, WPARAM
wParam, LPARAM lParam)
{
        HWND hwndl=NULL;

        switch(mesg)
        {
        case WM_LBUTTONDOWN:
                MessageBox(hWnd,"Left button is pressed","Click",MB_OK);
                break;
                case WM_LBUTTONUP:
                MessageBox(hWnd,"Left button is up","Click",MB_OK);
                break;
                case WM_RBUTTONDBLCLK:
                MessageBox(hWnd,"Left button is double
pressed","Click",MB_OK);
                break;

                case WM_DESTROY:
                MessageBox(hWnd,"Closed","Window is closed",MB_OK);
                PostQuitMessage(0);
                break;
        default:
                return DefWindowProc(hWnd, mesg,wParam,lParam);
        }
        return 0;
}
```

OUTPUT



Left button is pressed

8.4. RESPONDING TO KEY MESSAGE

The WM_CHAR message is generated when key is pressed.
Whenever key is pressed a WM_CHAR message is sent to the active window. wParam contains the ASCII value of the respective key pressed. LOWORD (lparam)contains the number of times the key is pressed. Window must establish a link between your program and screen. This can be done by acquiring **device context** (DC) by making a call to GetDC().
HDC GetDC(HWND hwnd);
Int ReleaseDC(HWND hwnd,HDC hdc);
BOOL TextOut(HDC DC,int x,int y,LPCSTR lpstr,int length);

In the Window function (WndProc), each time a WM_CHAR message is received, the character that is typed by the user is converted, using sprintf(), into a string that is one character long and then displayed using TextOut() at location 5,5.

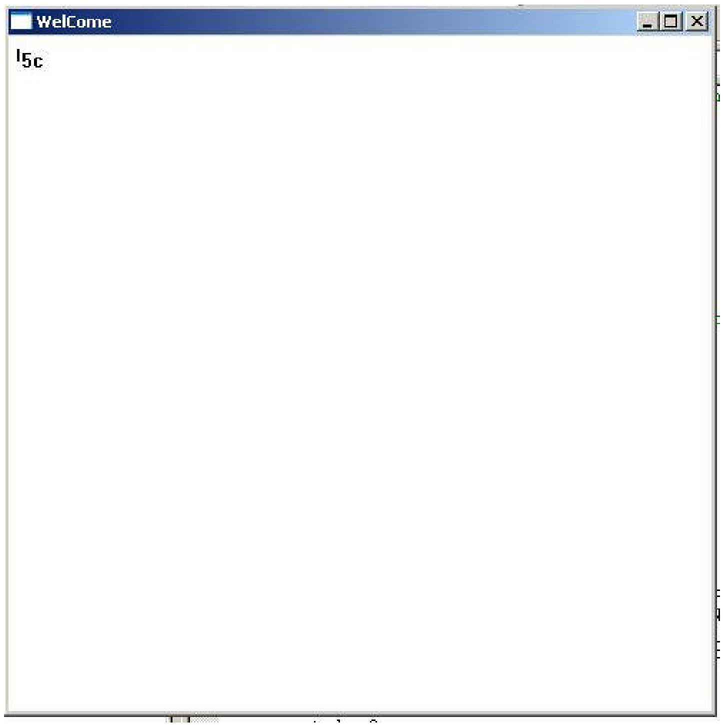
We have to erase the character before displaying another character, as windows is a graphic based system and each characters is of different shape & size, so we have used first TextOut for this purpose.

Following program creates handle key message.

8.2 ; handling key message WM_CHAR

```
        // key1.cpp : Defines the entry point for the application.
//
```

```
#include "stdafx.h"

#include <windows.h>
#include <stdio.h>
char str[40]=" ";
HDC hdc;

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

char szWinName[]="My Window";//name of window class

int WINAPI WinMain(HINSTANCE hInst,
                   HINSTANCE hPrev,
                   LPSTR     lpszargs,
                   int       hmode)
{
        WNDCLASSEX wc;//Define window class
   HWND hwnd;
   MSG msg;

   wc.cbSize=sizeof(WNDCLASSEX);
        wc.cbClsExtra=NULL;
   wc.cbWndExtra    = 0;
   wc.hInstance     = hInst;
   wc.hIcon         = LoadIcon(NULL, IDI_APPLICATION);
        wc.hIconSm       =LoadIcon(NULL, IDI_WINLOGO);
   wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
   wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
        wc.lpfnWndProc   =WndProc;
   wc.lpszMenuName  = NULL;
   wc.lpszClassName = szWinName;
        wc.style=0;

        //Register the window class

   if(!RegisterClassEx(&wc))
   {
      MessageBox(NULL, "Not Registered", "SEE THAT", MB_OK);
      return 0;
   }

        hwnd = CreateWindow(szWinName,
      "WelCome",
      WS_OVERLAPPEDWINDOW,
      20, 20, 500,500,
        HWND_DESKTOP, NULL, hInst, NULL);

   //dISPLAY WINDOW
   ShowWindow(hwnd, hmode);

   while(GetMessage(&msg, NULL, 0, 0) )
   {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
   };
        return msg.wParam;
}
LRESULT CALLBACK WndProc(HWND hWnd, UINT mesg, WPARAM
wParam, LPARAM lParam)
{
        switch(mesg)
        {
        case WM_CHAR:
                hdc=GetDC(hWnd);
                TextOut(hdc,5,5,"",3);
                sprintf(str,"5c",(char)wParam);//Stringsize character
                 TextOut(hdc,9,9,str,strlen(str));//Output char
                 ReleaseDC(hWnd,hdc);
                break;

                 case WM_DESTROY:
                 PostQuitMessage(0);
                break;
        default:
               return DefWindowProc(hWnd, mesg,wParam,lParam);
        }
        return 0;
}
```
OUTPUT



Messages are unique unsigned integer value.
Windows respond to each correspondence message.
Some Mouse messages are-WM_LBUTTONDOWN, WM_RBUTTONDOWN, WM_LBUTTONUP, WM_RBUTTONUP.
Key Messages can be handled by responding to WM_CHAR.

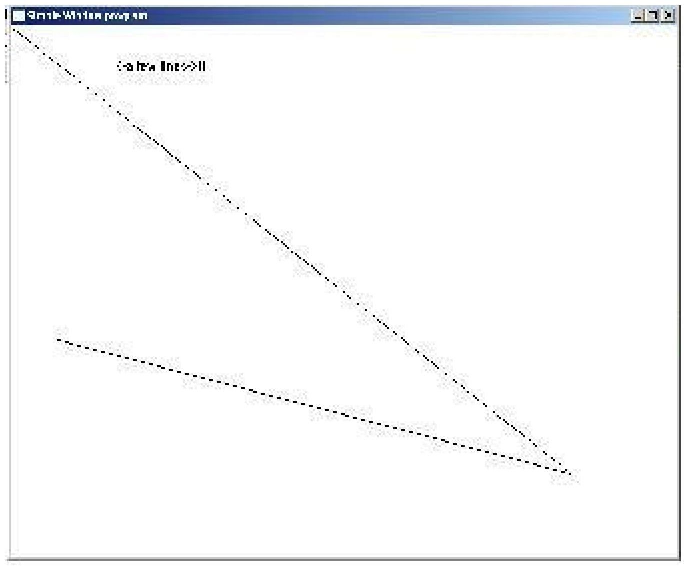
8.3 To draw straight line

```
// swp.cpp : Defines the entry point for the application.
//

#include "stdafx.h"
#include <windows.h>
//using namespace std;


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

char szProgName[]="ProgName";//name of window class

int WINAPI WinMain(HINSTANCE hInst,
                   HINSTANCE hPrev,
                   LPSTR     lpszCmdLine,
                   int       nShow)
{
        WNDCLASS wc;//Define window class
```

```
   HWND hwnd;
   MSG msg;

        wc.cbClsExtra=NULL;
   wc.cbWndExtra    = 0;
   wc.hInstance     = hInst;
   wc.hIcon         = 0;
        wc.hCursor      = LoadCursor(NULL, IDC_ARROW);
   wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
        wc.lpfnWndProc   =WndProc;
   wc.lpszMenuName  = NULL;
   wc.lpszClassName = szProgName;
        wc.style=CS_HREDRAW|CS_VREDRAW;

        //Register the window class

   if(!RegisterClass (&wc))
        return 0;

        hwnd = CreateWindow(szProgName,
      "Simple Window program",
      WS_OVERLAPPEDWINDOW,
      CW_USEDEFAULT, CW_USEDEFAULT,
CW_USEDEFAULT,CW_USEDEFAULT,
      NULL,NULL, hInst,NULL);

   //dISPLAY WINDOW
   ShowWindow(hwnd, nShow);
        UpdateWindow(hwnd);

   while(GetMessage(&msg, NULL, 0, 0) )
   {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
   };
        return msg.wParam;
}
LRESULT CALLBACK WndProc(HWND hWnd, UINT mesg, WPARAM
wParam, LPARAM lParam)
{
        HDC hdc;  //handle to Device Context
        /*A device context is a structure that defines a set of graphic objects
and their associated
        attributes, and the graphic modes that affect output. The graphic
objects include a pen for
```

```
        line drawing, a brush for painting and filling, a bitmap for copying or
scrolling parts of the
        screen, a palette for defining the set of available colors, a region for
clipping and other
        operations, and a path for painting and drawing operations. */


        PAINTSTRUCT ps;

        switch(mesg)
        {
        case WM_PAINT:
                hdc=BeginPaint(hWnd, &ps);
        MoveToEX(hdc, 0, 0,NULL);
                LineTo(hdc,639,429);

                MoveToEX(hdc,300,0,NULL);
                LineTo(hdc,50,300);


                 TextOut(hdc,120,30,"<-a few lines->",17);//Output char

                 ValidateRect(hWnd, NULL);
                 EndPaint(hWnd,&ps);
                 break;


        default:
               return DefWindowProc(hWnd, mesg,wParam,lParam);
        }
        return 0;
}
```

output


Simple Window program

```
<a few lines>>!!
```