ใบงานที่ 2 Keyboard and Mouse

# **วัตถุประสงค์**

1. ตรวจจับการทำงานของ Keyboard และ Mouse

2. ใช้เครื่องมือต่างๆ ในการสร้าง Application

# **เนื้อหาเบื้องต้น**

https://github.com/Desktop-Programming-Lab-2559/LAB-02/blob/master/LabIntro.md

# **ลำดับการทดลอง**

1. เรียกโปรแกรม Microsoft Visual Studio

2. สร้าง Project ใหม่ โดยเลือกเมนู File >> New >> Project… (Ctrl+Shift+N)

· ช่อง Templates: ให้เลือก Visual C++ และเลือกชนิด project เป็น Empty Project

· ช่อง Name: ให้ใส่ชื่อของ Project เป็น EasyWin32

· ช่อง Location: ให้เลือกตำแหน่งที่จะสร้าง Project (D:\Student\_Code\LAB\_01\_03)

· ส่วนที่เหลือ ให้คงไว้ตามที่ปรากฏ กด OK

3. เพิ่ม source code ให้กับ project โดยการเลือกเมนู PROJECT >> Add New Item…

· ตั้งชื่อไฟล์เป็น main.cpp

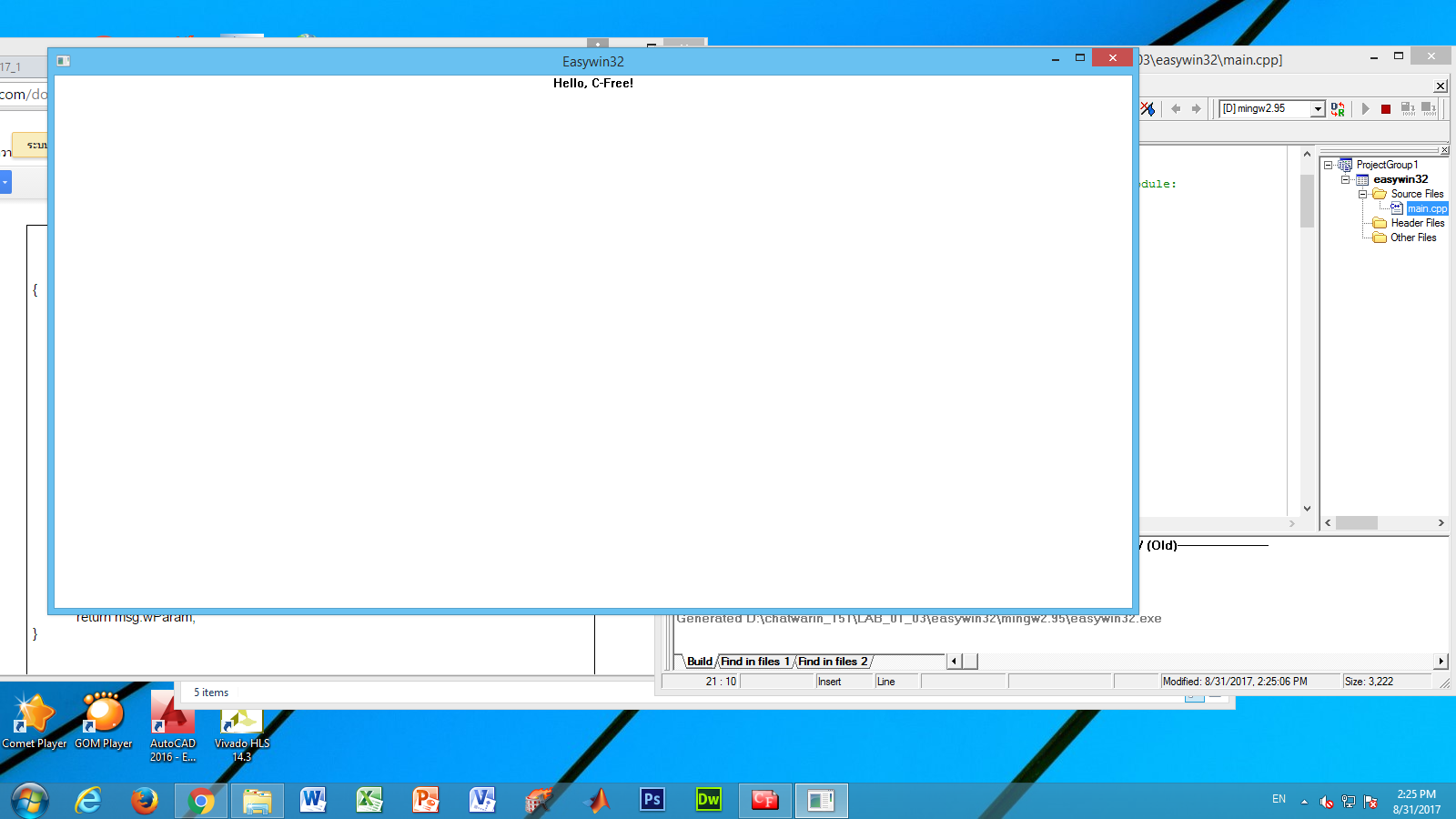
· กด Add เพื่อเพิ่มไฟล์

4. พิมพ์โปรแกรมดังต่อไปนี้ลงในไฟล์ main.cpp

|  |
| --- |
| #include <windows.h>  #include <tchar.h>  // Global Variables:  HINSTANCE hInst; // current instance  TCHAR szTitle[]="Easywin32"; // The title bar text  TCHAR szWindowClass[]="WinApp"; // the class name  // Foward declarations of functions included in this code module:  ATOM MyRegisterClass(HINSTANCE hInstance);  BOOL InitInstance(HINSTANCE, int);  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);  int APIENTRY WinMain(HINSTANCE hInstance,  HINSTANCE hPrevInstance,  LPSTR lpCmdLine,  int nCmdShow)  {  // TODO: Place code here.  MSG msg;  // Register Class  MyRegisterClass(hInstance);  // Perform application initialization:  if (!InitInstance (hInstance, nCmdShow))  {  return FALSE;  }  // Main message loop:  while (GetMessage(&msg, NULL, 0, 0))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  return msg.wParam;  }  //  // FUNCTION: MyRegisterClass()  //  // PURPOSE: Registers the window class.  //  // COMMENTS:  //  // This function and its usage is only necessary if you want this code  // to be compatible with Win32 systems prior to the 'RegisterClassEx'  // function that was added to Windows 95.  //  ATOM MyRegisterClass(HINSTANCE hInstance)  {  WNDCLASSEX wcex;  wcex.cbSize = sizeof(WNDCLASSEX);  wcex.style = CS\_HREDRAW | CS\_VREDRAW;  wcex.lpfnWndProc = (WNDPROC)WndProc;  wcex.cbClsExtra = 0;  wcex.cbWndExtra = 0;  wcex.hInstance = hInstance;  wcex.hIcon = LoadIcon (NULL, IDI\_APPLICATION);  wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);  wcex.lpszMenuName = NULL;  wcex.lpszClassName = szWindowClass;  wcex.hIconSm = LoadIcon (NULL, IDI\_APPLICATION);  return RegisterClassEx(&wcex);  }  //  // FUNCTION: InitInstance(HANDLE, 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)  {  HWND hWnd;  hInst = hInstance; // Store instance handle in our global variable  hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,  CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);  if (!hWnd)  {  return FALSE;  }  ShowWindow(hWnd, nCmdShow);  UpdateWindow(hWnd);  return TRUE;  }  //  // FUNCTION: WndProc(HWND, unsigned, WORD, LONG)  //  // 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)  {  PAINTSTRUCT ps;  HDC hdc;  RECT rt;  char szHello[]="Hello, C-Free!";    switch (message)  {  case WM\_PAINT:  hdc = BeginPaint(hWnd, &ps);  // TODO: Add any drawing code here...    GetClientRect(hWnd, &rt);  DrawText(hdc, szHello, strlen(szHello), &rt, DT\_CENTER);  EndPaint(hWnd, &ps);  break;  case WM\_CLOSE:  DestroyWindow(hWnd);  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  return 0;  } |

5. ทำการทดสอบโปรแกรม/บันทึกผล

Ans. อันดับแรกเริ่มจากการประกาศตัวแปรส่วนกลาง ต่อมาคือในส่วนของการเรียกใช้ฟังก์ชั่น จัดรูปแบบของ window สร้าง window และเช็คว่าถ้ามีอยู่แล้วไม่ต้องเรียก แต่ถ้าไม่มีก็สร้างมันขึ้น ส่วนของอัพเดท window เพื่อเผื่อมีการเปลี่ยนแปลง จากนั้นต้องการให้ window ขึ้นแสดงว่า hello c-free โดยใช้ฟังก์ชั่น drawtext



6. เพิ่ม Function Marker() ในโปรแกรม

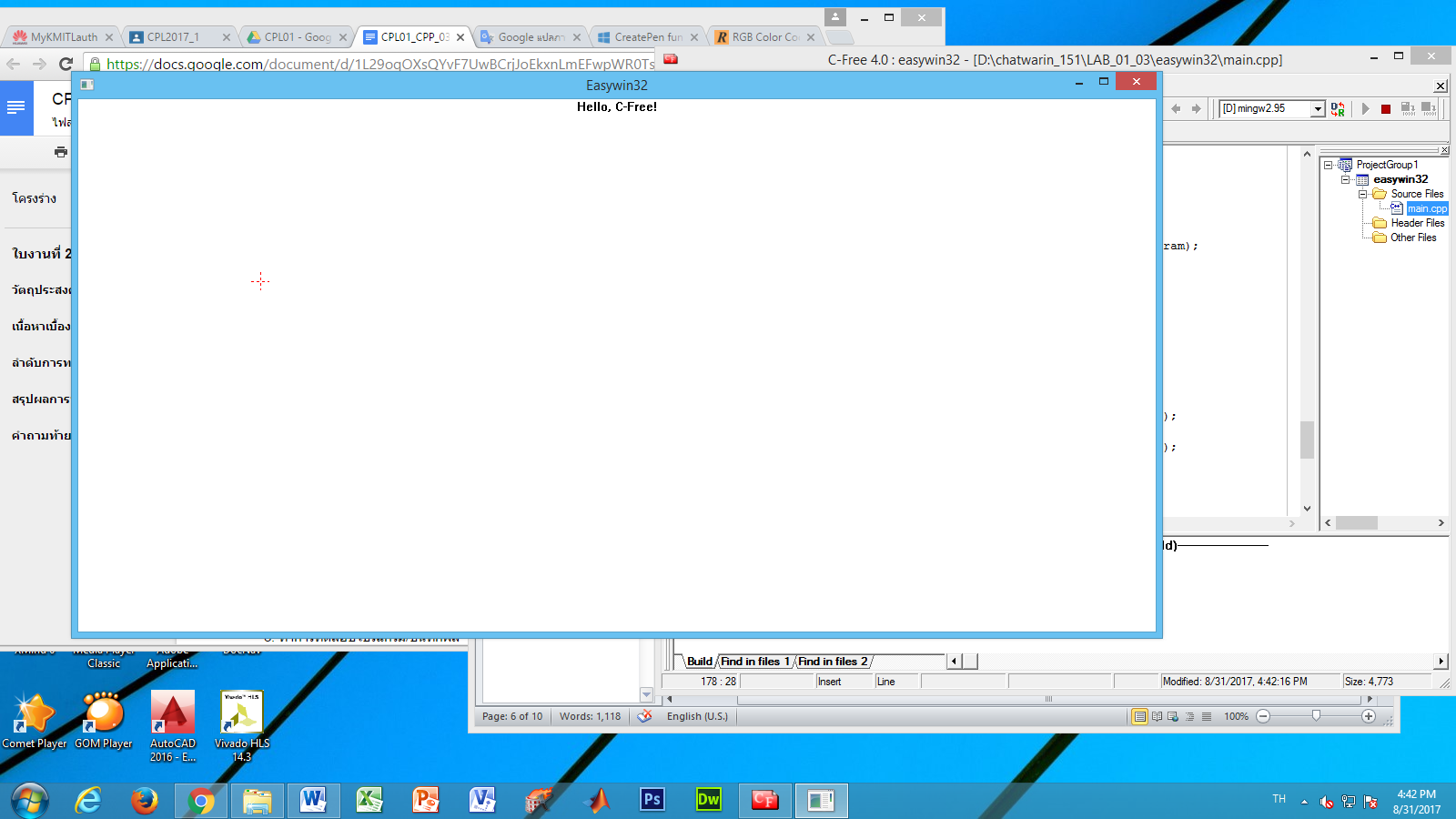
|  |
| --- |
| void Marker(LONG x, LONG y, HWND hwnd)  {  HDC hdc;  HPEN hPen;  hdc = GetDC(hwnd);  hPen = CreatePen(PS\_DOT,1,RGB(255,0,0));    SelectObject(hdc, hPen);  MoveToEx(hdc, (int) x - 10, (int) y, (LPPOINT) NULL);  LineTo(hdc, (int) x + 10, (int) y);  MoveToEx(hdc, (int) x, (int) y - 10, (LPPOINT) NULL);  LineTo(hdc, (int) x, (int) y + 10);  DeleteObject(hPen);  ReleaseDC(hwnd, hdc);  } |

7. แก้ไขโปรแกรม

|  |  |
| --- | --- |
| case WM\_PAINT:  hdc = BeginPaint(hWnd, &ps);  // TODO: Add any drawing code here...  GetClientRect(hWnd, &rt);  DrawText(hdc, szHello, strlen(szHello), &rt, DT\_CENTER);  EndPaint(hWnd, &ps);  break; | case WM\_PAINT:  hdc = BeginPaint(hWnd, &ps);  // TODO: Add any drawing code here...  GetClientRect(hWnd, &rt);  DrawText(hdc, szHello, strlen(szHello), &rt, DT\_CENTER);  Marker(50,50,hWnd)  EndPaint(hWnd, &ps);  break; |

8. ทำการทดสอบโปรแกรม/บันทึกผล

Ans. เพิ่มฟังก์ชั่น marker เข้ามา โดยกำหนดตัวแปรที่ตัวแปรส่วนกลางด้วย จากนั้นแก้ไขหลังคำสั่งฟังก์ชั่น drawtext เพิ่มคำสั่งฟังก์ชั่น marker เพื่อให้มีเส้นมาร์คเพิ่มเข้ามาในหน้าต่าง window ตามตำแหน่งแกน x,y ที่กำหนด



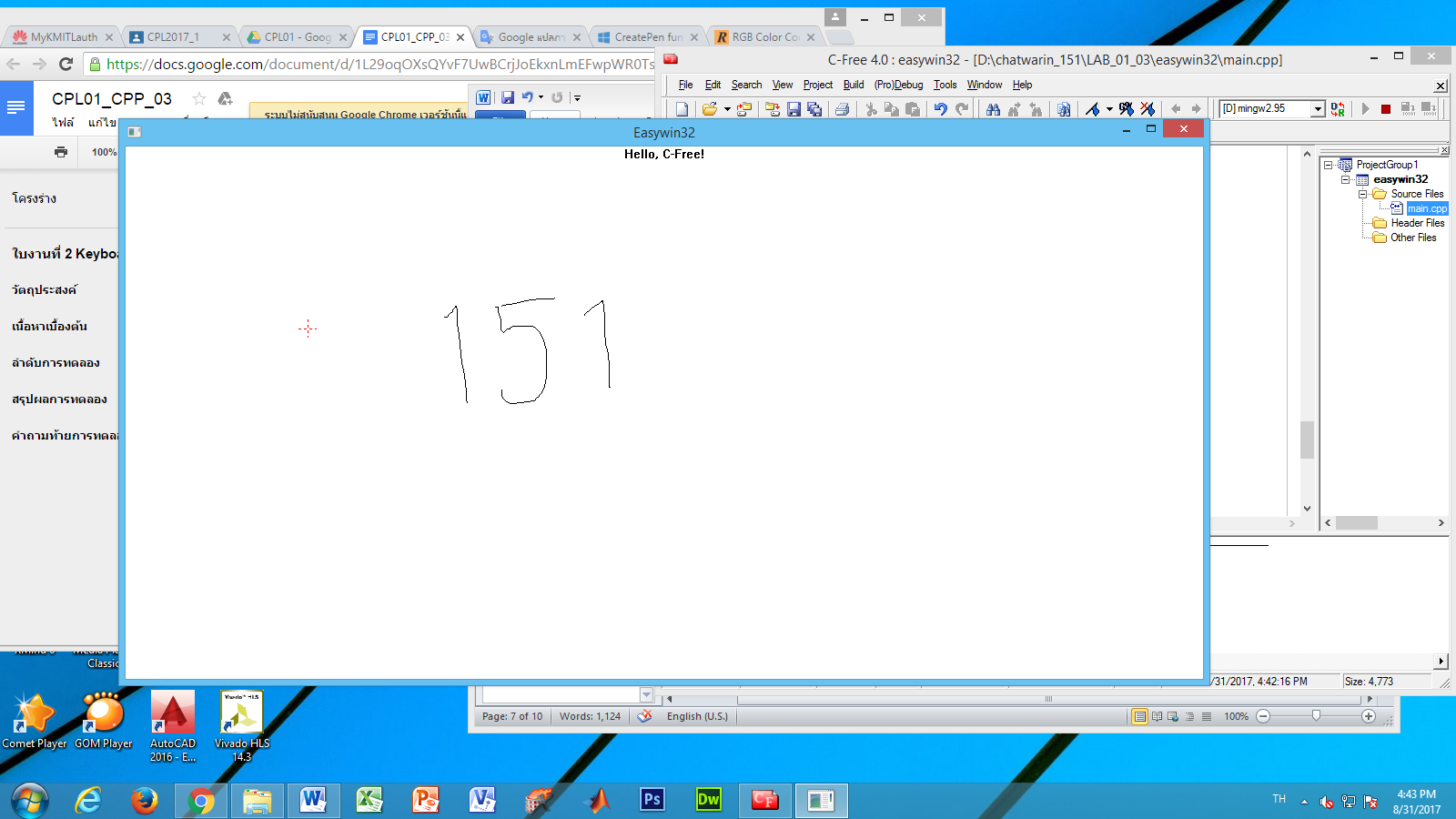
9. เพิ่มโปรแกรมตามตาราง

|  |
| --- |
| case WM\_LBUTTONDOWN:  fDraw = TRUE;  ptPrevious.x = LOWORD(lParam);  ptPrevious.y = HIWORD(lParam);  return 0L;    case WM\_LBUTTONUP:  if (fDraw)  {  hdc = GetDC(hWnd);  MoveToEx(hdc, ptPrevious.x, ptPrevious.y, NULL);  LineTo(hdc, LOWORD(lParam), HIWORD(lParam));  ReleaseDC(hWnd, hdc);  }  fDraw = FALSE;  return 0L;    case WM\_MOUSEMOVE:  if (fDraw)  {  hdc = GetDC(hWnd);  MoveToEx(hdc, ptPrevious.x, ptPrevious.y, NULL);  LineTo(hdc, ptPrevious.x = LOWORD(lParam),  ptPrevious.y = HIWORD(lParam));  ReleaseDC(hWnd, hdc);  }  return 0L; |

10. เพิ่มตัวแปร BOOL fDraw = FALSE; POINT ptPrevious; ไว้ที่ Global Area

11. ทำการทดสอบโปรแกรม/บันทึกผล

Ans. เพิ่มคำสั่ง case WM\_LBUTTONDOWN: ถ้ามีการกดเม้าส์เกิดขึ้นจะเก็บค่าเป็น true แล้วจะจำตำแหน่งแรกไว้ case WM\_LBUTTONUP: ถ้ามีการกดเม้าส์เกิดขึ้นแล้วมีการเคลื่อนที่เกิดขึ้นจะเกิดเส้นตามตำแหน่งที่เราลากจากตำแหน่งแรกที่ได้มีการจำไว้ case WM\_MOUSEMOVE: คือคำสั่งเม้าส์เลื่อน ถ้าหากเม้าส์มีการขยับจะให้เขียนได้ไหม จากนั้นอย่าลืมเพิ่มการกำหนดตัวแปรไว้ที่ตัวแปรส่วนกลางเพื่อไว้เรียกใช้ตัวแปรนั้นๆได้



12. เพิ่มตัวแปร COLORREF Color[]={RGB(255,0,0),RGB(0,255,0),RGB(0,0,255),RGB(0,0,0)}; ไว้ที่ Global Area

13. แก้ไข Function Marker() ตามตาราง

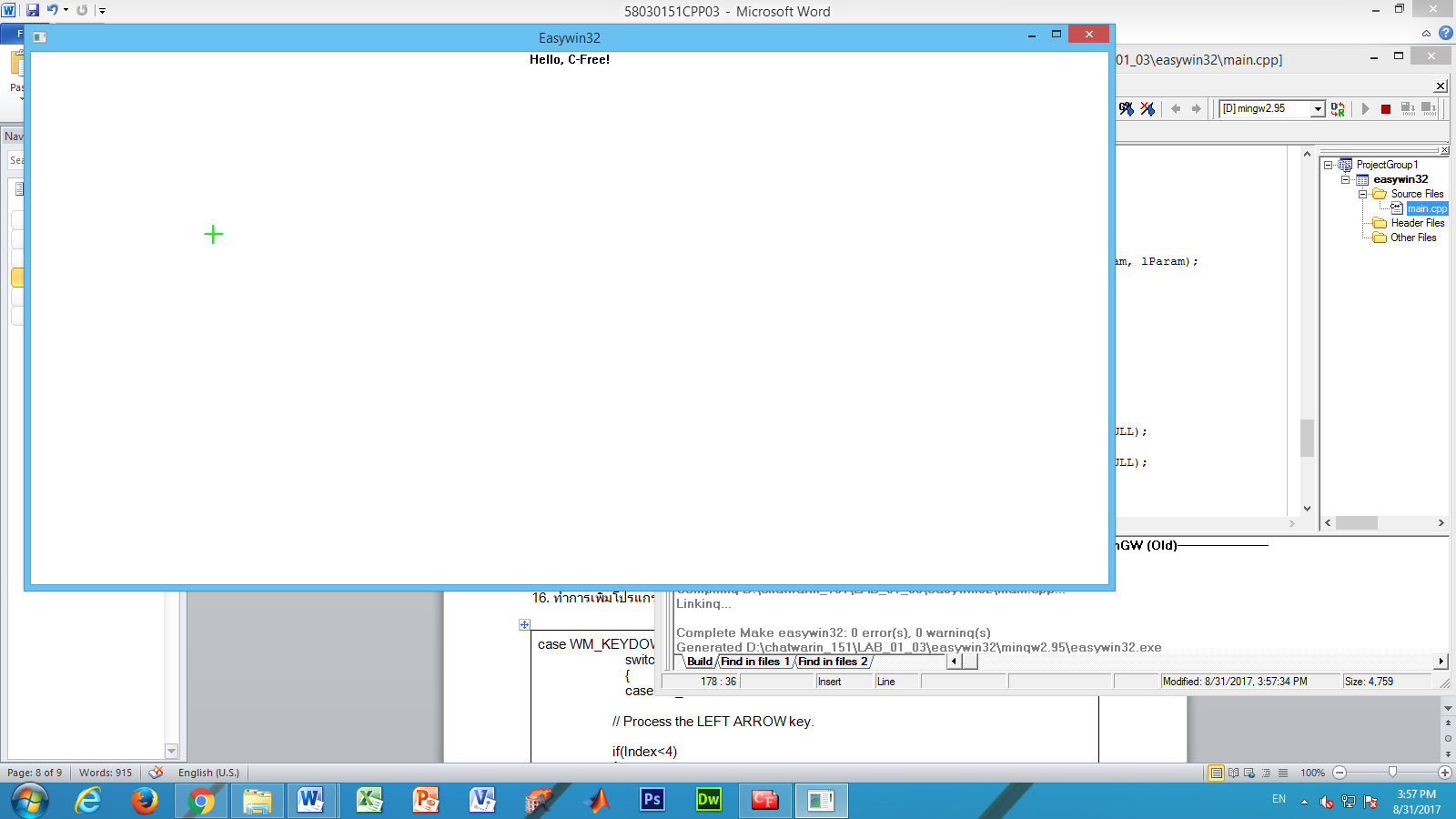
|  |
| --- |
| void Marker(LONG x, LONG y,int Index, HWND hwnd)  {  HDC hdc;  HPEN hPen;  hdc = GetDC(hwnd);  hPen = CreatePen(PS\_DOT,1,Color[Index]);    SelectObject(hdc, hPen);  MoveToEx(hdc, (int) x - 10, (int) y, (LPPOINT) NULL);  LineTo(hdc, (int) x + 10, (int) y);  MoveToEx(hdc, (int) x, (int) y - 10, (LPPOINT) NULL);  LineTo(hdc, (int) x, (int) y + 10);  DeleteObject(hPen);  ReleaseDC(hwnd, hdc);  } |

14. แก้ไขโปรแกรมตามตาราง

|  |
| --- |
| case WM\_PAINT:  hdc = BeginPaint(hWnd, &ps);  // TODO: Add any drawing code here...    GetClientRect(hWnd, &rt);  DrawText(hdc, szHello, strlen(szHello), &rt, DT\_CENTER);  Marker(50,50,Index,hWnd);  EndPaint(hWnd, &ps);  break; |

15. ทำการทดสอบโปรแกรม/บันทึกผล

Ans. แก้ไขประโยคคำสั่งในฟังก์ชั่น marker เพิ่มตัวแปรใหม่อีกตัว คือ Index เพื่อเป็นการเปลี่ยนแปลงค่าได้ง่ายขึ้น ประกาศตัวแปร Index ในส่วนของตัวแปรส่วนกลางด้วยเพื่อไว้เรียกใช้ได้อีก โดยเปลี่ยนแปลงค่าที่เดียว

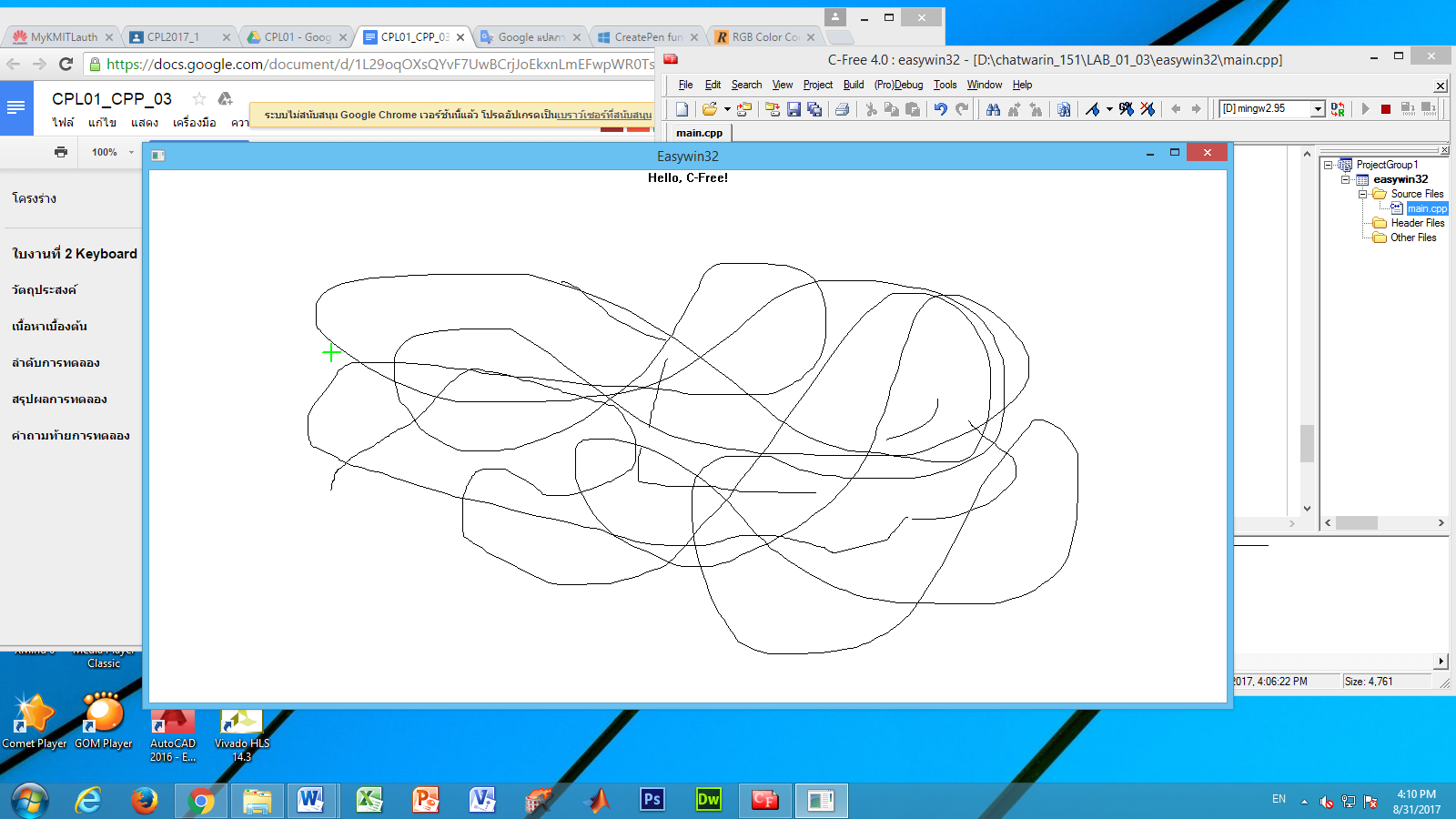


16. ทำการเพิ่มโปรแกรมตามตาราง

|  |
| --- |
| case WM\_KEYDOWN:  switch (wParam)  {  case VK\_LEFT:    // Process the LEFT ARROW key.    if(Index<4)  {  Index++;  }else  {  Index=0;  }  InvalidateRect(hWnd, NULL, TRUE);  default:  break;  }  break; |

17. ทำการทดสอบโปรแกรม/บันทึกผล

Ans. เป็นคำสั่งฟังก์ชั่นการกดเม้าส์เมื่อไหร่จะมีอีเวนท์เกิดขึ้นมา ตามคำสั่งคือถ้ามีการกดน้อยกว่า4เมื่อไหร่ให้มีการลากเส้นแล้วถ้ากดปุ่ม Alt 2 ครั้ง เส้นจะลากต่อเนื่องโดยไม่ต้องกดคลิกเม้าส์



# สรุปผลการทดลอง

Ans. จากการทดลองจะเห็นได้ว่า จากในตัวโปรแกรมเราจะรู้เพิ่มว่าสามารถออกแบบรูปแบบการใช้ปากกาวาดเขียน การเปลี่ยนสีของปากกา การจัดการหน้าต่าง window เป็นต้น ได้รู้คำสั่งต่าง ๆ เพิ่มเติมสำหรับตัวโปรแกรม คำสั่งพิเศษ หรือหลักไวทยากรณ์ของภาษาโปรแกรมที่ใช้

# คำถามท้ายการทดลอง

1. เขียนโปรแกรมวาดรูปที่สามารถเปลี่ยนสีปากกาได้

#include <windows.h>

#include <tchar.h>

// Global Variables:

HINSTANCE hInst; // current instance

TCHAR szTitle[]="Easywin32"; // The title bar text

TCHAR szWindowClass[]="WinApp"; // the class name

// BOOL fDraw = FALSE; POINT ptPrevious;

COLORREF Color[]={RGB(255,0,0),RGB(0,255,0),RGB(0,0,255),RGB(0,0,0)};

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

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

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

void Marker(LONG x, LONG y,int Index, HWND hWnd);

BOOL fDraw = FALSE;

POINT ptPrevious;

int Index=1;

int HIndex = 1;

int APIENTRY WinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPSTR lpCmdLine,

int nCmdShow)

{

// TODO: Place code here.

MSG msg;

// Register Class

MyRegisterClass(hInstance);

// Perform application initialization:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

// Main message loop:

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

// FUNCTION: MyRegisterClass()

// PURPOSE: Registers the window class.

// COMMENTS:

// This function and its usage is only necessary if you want this code

// to be compatible with Win32 systems prior to the 'RegisterClassEx'

// function that was added to Windows 95.

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

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

wcex.lpfnWndProc = (WNDPROC)WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon (NULL, IDI\_APPLICATION);

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

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = NULL;

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon (NULL, IDI\_APPLICATION);

return RegisterClassEx(&wcex);

}

// FUNCTION: InitInstance(HANDLE, 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)

{

HWND hWnd;

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

hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

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

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

// FUNCTION: WndProc(HWND, unsigned, WORD, LONG)

// 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)

{

PAINTSTRUCT ps;

HDC hdc;

RECT rt;

char szHello[]="Hello, C-Free!";

switch (message)

{

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

// TODO: Add any drawing code here...

GetClientRect(hWnd, &rt);

DrawText(hdc, szHello, strlen(szHello), &rt, DT\_CENTER);

Marker(50,50,Index,hWnd);

EndPaint(hWnd, &ps);

break;

case WM\_LBUTTONDOWN:

fDraw = TRUE;

ptPrevious.x = LOWORD(lParam);

ptPrevious.y = HIWORD(lParam);

return 0L;

case WM\_LBUTTONUP:

if (fDraw)

{

hdc = GetDC(hWnd);

MoveToEx(hdc, ptPrevious.x, ptPrevious.y, NULL);

LineTo(hdc, LOWORD(lParam), HIWORD(lParam));

ReleaseDC(hWnd, hdc);

}

fDraw = FALSE;

return 0L;

case WM\_KEYDOWN:

switch (wParam)

{

case VK\_LEFT: // Process the LEFT ARROW key.

if(Index<4)

{

Index++;

}else

{

Index=0;

}

// InvalidateRect(hWnd, NULL, TRUE);

default:

break;

}

break;

case WM\_MOUSEMOVE:

if (fDraw)

{

HDC hdc;

HPEN hPen;

hdc = GetDC(hWnd);

hPen = CreatePen(PS\_DOT,15,Color[Index]);

SelectObject(hdc,hPen);

MoveToEx(hdc, ptPrevious.x, ptPrevious.y, NULL);

LineTo(hdc, ptPrevious.x = LOWORD(lParam),

ptPrevious.y = HIWORD(lParam));

ReleaseDC(hWnd, hdc);

}

return 0L;

case WM\_CLOSE:

DestroyWindow(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

void Marker(LONG x, LONG y,int Index, HWND hWnd)

{

HDC hdc;

HPEN hPen;

hdc = GetDC(hWnd);

hPen = CreatePen(PS\_DOT,3,Color[Index]);

SelectObject(hdc, hPen);

MoveToEx(hdc, (int) 200 - 10, (int) 100, (LPPOINT) NULL);

LineTo(hdc, (int) 200 + 10, (int) 100);

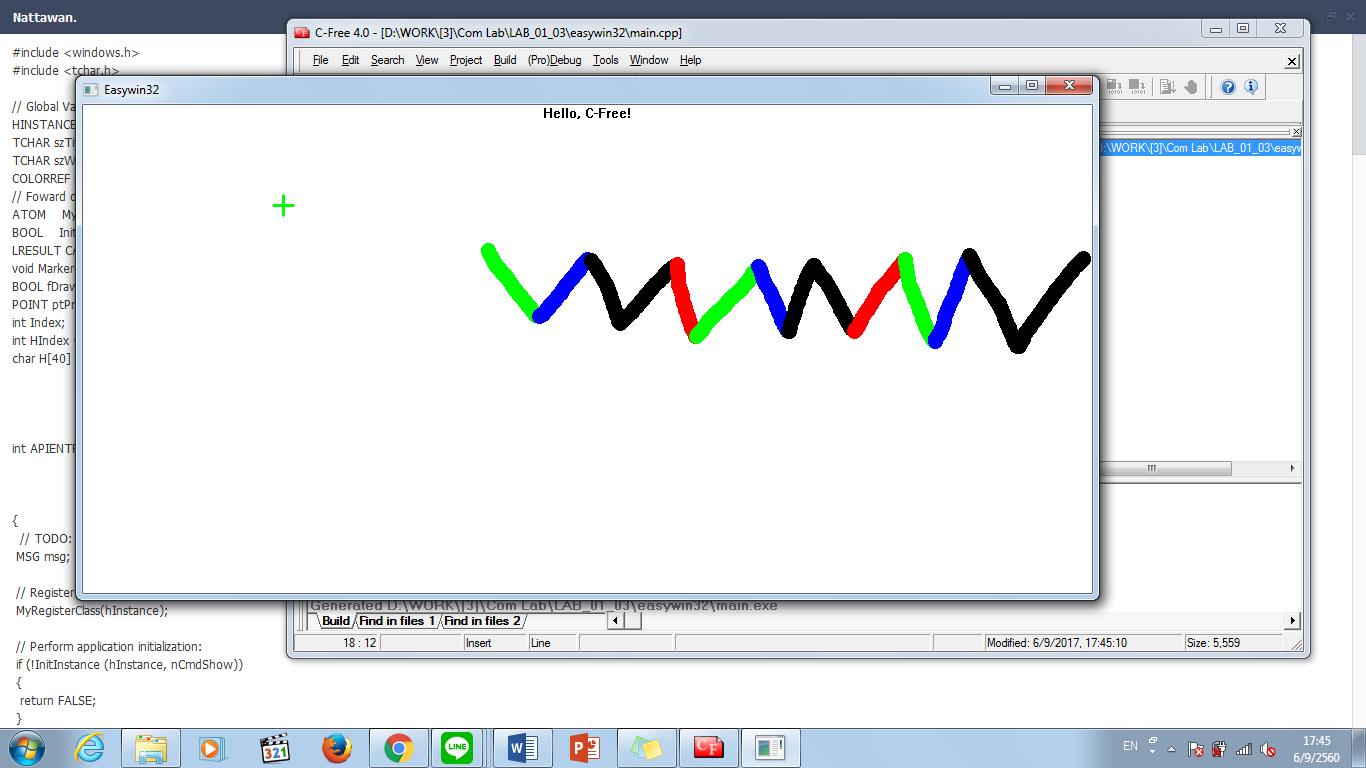
MoveToEx(hdc, (int) 200, (int) 100 - 10, (LPPOINT) NULL);

LineTo(hdc, (int) 200, (int) 100 + 10);

DeleteObject(hPen);

ReleaseDC(hWnd, hdc);

}



1. เขียนโปรแกรมพิมพ์ข้อมูลจากคีย์บอร์ด แล้วแสดงผลที่หน้าจอ

#include <windows.h>

#include <tchar.h>

// Global Variables:

HINSTANCE hInst; // current instance

TCHAR szTitle[]="Easywin32"; // The title bar text

TCHAR szWindowClass[]="WinApp"; // the class name

// BOOL fDraw = FALSE; POINT ptPrevious;

COLORREF Color[]={RGB(255,0,0),RGB(0,255,0),RGB(0,0,255),RGB(0,0,0)};

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

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

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

void Marker(LONG x, LONG y,int Index, HWND hWnd);

BOOL fDraw = FALSE;

POINT ptPrevious;

int Index=1;

int HIndex = 1;

char H[40] = {0};

int APIENTRY WinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPSTR lpCmdLine,

int nCmdShow)

{

// TODO: Place code here.

MSG msg;

// Register Class

MyRegisterClass(hInstance);

// Perform application initialization:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

// Main message loop:

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

// FUNCTION: MyRegisterClass()

// PURPOSE: Registers the window class.

// COMMENTS:

// This function and its usage is only necessary if you want this code

// to be compatible with Win32 systems prior to the 'RegisterClassEx'

// function that was added to Windows 95.

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

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

wcex.lpfnWndProc = (WNDPROC)WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon (NULL, IDI\_APPLICATION);

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

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = NULL;

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon (NULL, IDI\_APPLICATION);

return RegisterClassEx(&wcex);

}

// FUNCTION: InitInstance(HANDLE, 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)

{

HWND hWnd;

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

hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

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

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

// FUNCTION: WndProc(HWND, unsigned, WORD, LONG)

// 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)

{

PAINTSTRUCT ps;

HDC hdc;

RECT rt;

char szHello[]="Hello, C-Free!";

char ch[50];

switch (message)

{

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

// TODO: Add any drawing code here...

GetClientRect(hWnd, &rt);

DrawText(hdc, szHello, strlen(szHello), &rt, DT\_CENTER);

Marker(50,50,Index,hWnd);

EndPaint(hWnd, &ps);

break;

case WM\_LBUTTONDOWN:

fDraw = TRUE;

ptPrevious.x = LOWORD(lParam);

ptPrevious.y = HIWORD(lParam);

return 0L;

case WM\_LBUTTONUP:

if (fDraw)

{

hdc = GetDC(hWnd);

MoveToEx(hdc, ptPrevious.x, ptPrevious.y, NULL);

LineTo(hdc, LOWORD(lParam), HIWORD(lParam));

ReleaseDC(hWnd, hdc);

}

fDraw = FALSE;

return 0L;

case WM\_KEYDOWN:

switch (wParam)

{

case VK\_LEFT: // Process the LEFT ARROW key.

if(Index<4)

{

Index++;

}else

{

Index=0;

}

// InvalidateRect(hWnd, NULL, TRUE);

default:

break;

}

break;

case WM\_CHAR:

hdc = GetDC(hWnd);

H[HIndex] =(TCHAR)wParam;;

HIndex++;

TextOut(hdc, 100, 200, H, 40);

ReleaseDC(hWnd, hdc);

break;

case WM\_MOUSEMOVE:

if (fDraw)

{

HDC hdc;

HPEN hPen;

hdc = GetDC(hWnd);

hPen = CreatePen(PS\_DOT,15,Color[Index]);

SelectObject(hdc,hPen);

MoveToEx(hdc, ptPrevious.x, ptPrevious.y, NULL);

LineTo(hdc, ptPrevious.x = LOWORD(lParam),

ptPrevious.y = HIWORD(lParam));

ReleaseDC(hWnd, hdc);

}

return 0L;

case WM\_CLOSE:

DestroyWindow(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

void Marker(LONG x, LONG y,int Index, HWND hWnd)

{

HDC hdc;

HPEN hPen;

hdc = GetDC(hWnd);

hPen = CreatePen(PS\_DOT,3,Color[Index]);

SelectObject(hdc, hPen);

MoveToEx(hdc, (int) 200 - 10, (int) 100, (LPPOINT) NULL);

LineTo(hdc, (int) 200 + 10, (int) 100);

MoveToEx(hdc, (int) 200, (int) 100 - 10, (LPPOINT) NULL);

LineTo(hdc, (int) 200, (int) 100 + 10);

DeleteObject(hPen);

ReleaseDC(hWnd, hdc);

}

