Microsoft Game API for Pocket PC

In this tutorial, we will use Microsoft Game API and learn
(1) how to access video frame buffer directly to speed up displaying
(2) how to manipulate the Hardware buttons.

1. Introduction to Game API

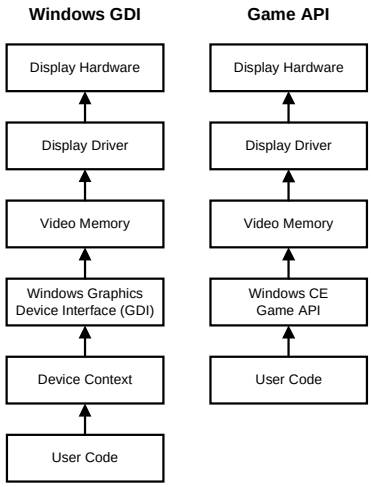
(a) Game API[1]

The figure below shows the difference between Windows GDI and Game API. The first approach is already used in *ENEE408G Multimedia Signal Processing Mobile Computing and Pocket PC Programming Manual*" Demo Project –1 and 2. When we want to draw on the screen, we need to call Device Context (DC) and Windows Graphics Device Interface (GDI). To satisfy the speed requirement for game developers, Microsoft provides Game API (GAPI), which is similar to DirectX used for Windows 32 system, to bypass the Window CE GDI entirely. There are three major features in GAPI.

(1) Full-screen mode: GAPI provides a standardized way for applications to move in and out of full-screen mode, including focus control. This means that if an appointment reminder pops up while you're in the game, the game will pause until focus is restored.

(2) Hardware mapping for game buttons: A simple way for developers to choose which buttons do which function in the application.

(3) Much faster resolution timers (5 milliseconds or better): This allows for real-time responsiveness and better handling of audio streams.



[1] GAPI (*gx.h* and *gx.dll*) is already in the Pocket PC 2002 SDK package. It can also be downloaded at http://www.microsoft.com/mobile/pocketpc/downloads/devdownloads.asp if you use older version SDK.

(b) Game API Structures

Two GAPI structures are used commonly in an application.

(1) GXDisplayProperties

This *GXDisplayProperties* provides information about the video frame buffer on a Pocket PC. This structure can be obtained by calling *GXGetDisplayProperties* function.

```
struct GXDisplayProperties {
        DWORD cxWidth;          // screen width
        DWORD cyHeight;         // screen height
        long cbxPitch;          // number of bytes to move right one x pixel - can be negative.
        long cbyPitch;          // number of bytes to move down one y pixel - can be negative.
        long cBPP;              // # of bits in each pixel
        DWORD ffFormat;         // bit depth properties.
};
```

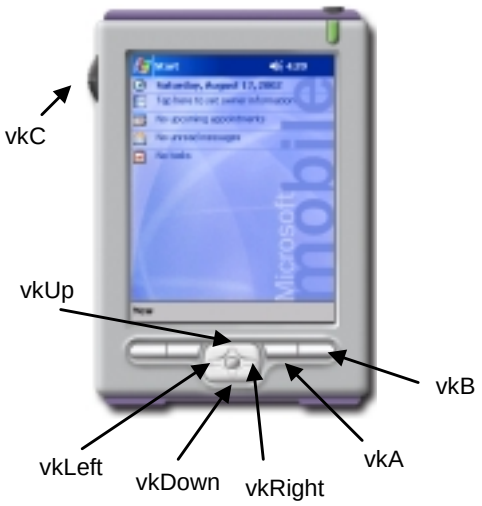
The value in ffFormat is relevant to bit depth which is defined as follows.

```
#define kfPalette     0x10     // Pixel values are indexes into a palette
#define kfDirect      0x20     // Pixel values contain actual level information
#define kfDirect555   0x40     // 5 bits each for red, green and blue values in a pixel.
#define kfDirect565   0x80     // 5 red bits, 6 green bits and 5 blue bits per pixel
#define kfDirect888   0x100    // 8 bits each for red, green and blue values in a pixel.
#define kfDirect444   0x200    // 4 red, 4 green, 4 blue
```

(2) GXKeyList

The *GXKeyList* structure provides information about the default hardware button assignments. *GXKeyList* variables are obtained by calling *GXGetDefaultKeys* function.

Structure	Corresponding Hardware Key

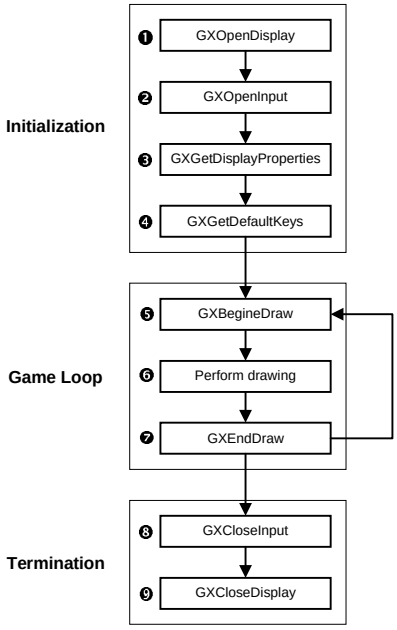
struct GXKeyList {
short vkUp;
POINT ptUp;
short vkDown;
POINT ptDown;
short vkLeft;
POINT ptLeft;
short vkRight;
POINT ptRight;
short vkA;
POINT ptA;
short vkB;
POINT ptB;
short vkC;
POINT ptC;
short vkStart;
POINT ptStart;
};
	

(c) Game API Functions

There are 12 functions defined in GAPI 1.2. The following table explains the description of each function.

Function Name	**Description**
GXOpenDisplay	Initializes GAPI. It can be called only once in an application.
GXCloseDisplay	Closes GAPI. Cleans up GAPI resources.
GXBeginDraw	Access the frame buffer for drawing.
GXEndDraw	Called when drawing is finished.
GXGetDisplayProperties	Obtain the information of the display device.
GXOpenInput	Captures the buttons for the game.
GXCloseInput	Release the buttons for normal use.
GXGetDefaultKeys	Provides information on the suggested buttons.
GXSuspend	Suspends GAPI subsystem to allow other applications to gain focus.
GXResume	Resumes GAPI operation when the game regains focus.
GXISDisplayDRAWBuffer	Determines whether a Pocket PC has a nonstandard display device. It is normally used with *GXSetViewPort* to handle nonstandard displays.
GXSetViewport	Allows GDI drawing and GAPI access to the same frame buffer.

The following figure shows the standard procedure how to use GAPI. We will use this flowchart as a guideline in the later example.



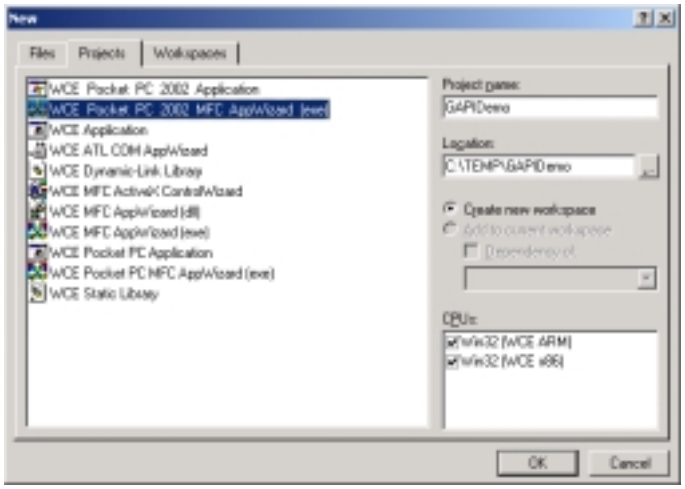
(d) Game API Video Mode

Most Pocket PC devices have a 16-bit display, which means we use two bytes to represent one pixel. Two modes are supported in GAPI, namely, *kfDirect555* and *kfDirect565*. Each color component (Red, Green, Blue) takes 5 bits in the first mode. The *kfDirect565* mode provides 5 red bits, 6 green bits, and 5 blue bits. We can pack a true color (24 bit, 8 bits for each color component) to *kfDirect565* in the following way.

```
struct COLOR{  BYTE Red;    BYTE Green;    BYTE Blue; };
Unsigned short Color;
Color = (unsigned short)  (   ((color.Red & 0xf8) <<8) |  ((color.Green & 0xfc) <<3)  |  ((color.Blue & 0xf8) >> 3)  );
```

2. Example

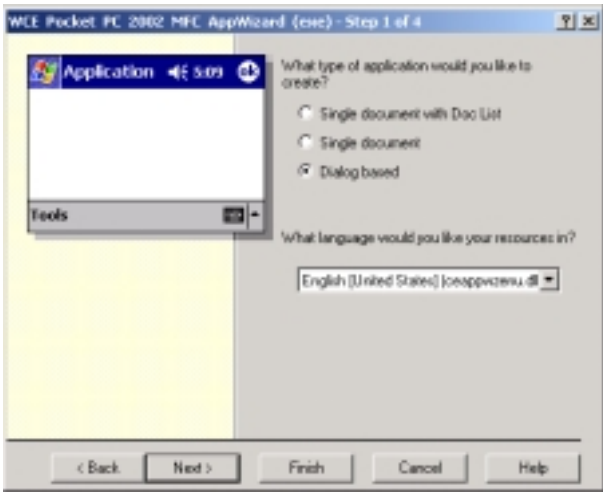
(a) **Create a new project** by New → Project from eVC menu bar



(1) Select "*WCE Pocket PC 2002 MFC AppWizard (exe)*" on the *Projects* tap.
(2) Key in "*GAPIDemo*" on *Project name* edit box and select a *Location* for this project. Also check "*Create new workspace*".
(3) Check "*Win32[WCE ARM]*" and "*Win32[WCE x86]*" on the *CPUs* box[2].
(4) Click "*OK*" and a *WCE Pocket PC 2002 MFC AppWizard* window will pop up.

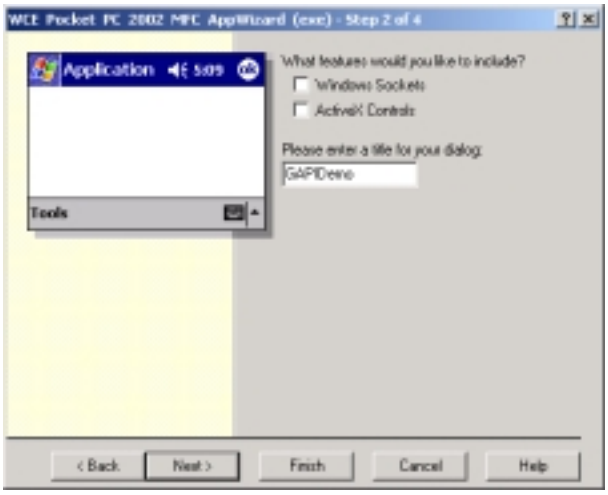
(b) WCE Pocket PC 2002 MFC AppWizard

(1) Choose "*Dialog Based*" and press "*Next*".

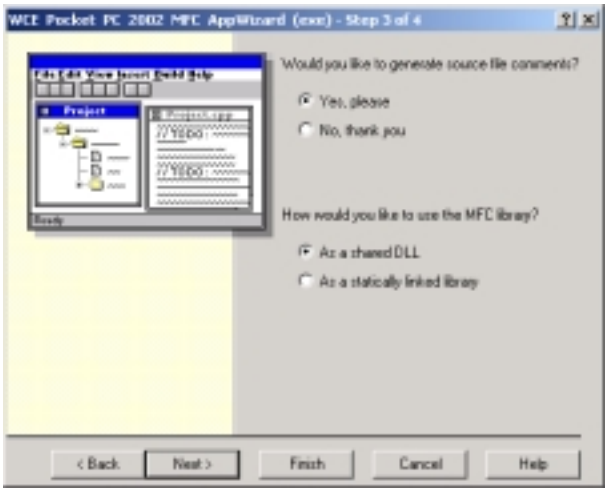


[2] If Pocket PC 2002 SDK is not available, you may create a new project by an older version of MFC AppWizard, such as *WCE MFC AppWizard* or *WCE Pocket PC MFC AppWizard*.

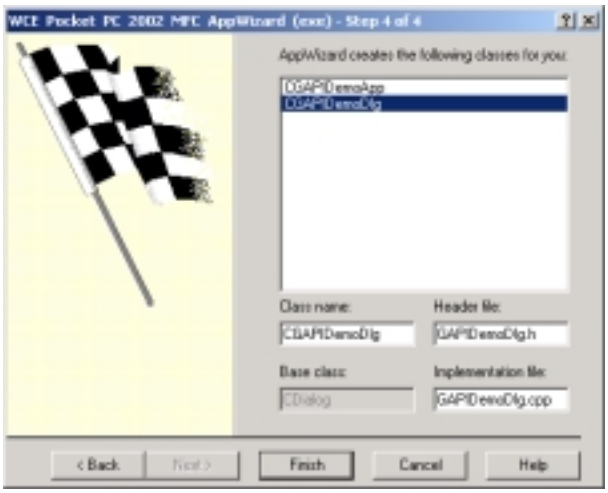
(2) Set title as "*GAPIDemo*" and disable other check items. Then click "*Next*".



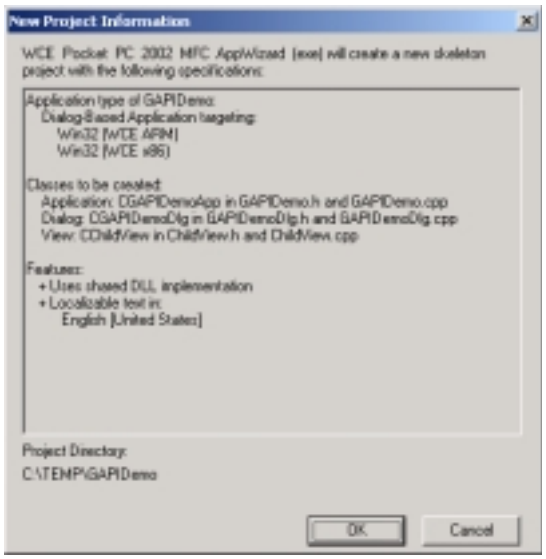
(3) Check "*Generate source file comment*" and "*As a shared DLL*". Click "*Next*".



(4) Click "*Finish*" to finish the initial setting.



(5) AppWizard will summarize the features and files of new project you created. Click "*OK*" to close the AppWizard.



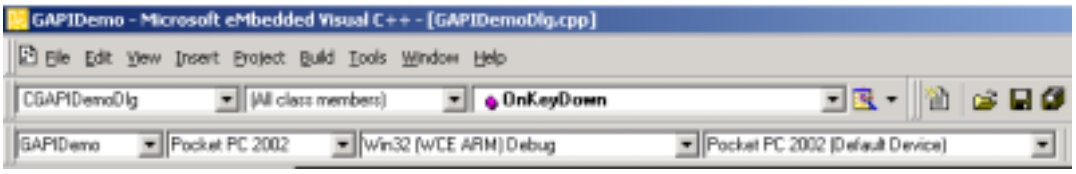
(c) Design Dialog Box

Since we use full screen mode in this demo, we do not need to design the GUI. Click on the *IDD_GAPIDEMO_DIALOG* on *Resource View* window. Delete "*TODO: Place dialog controls here*".

(d) Project Setting

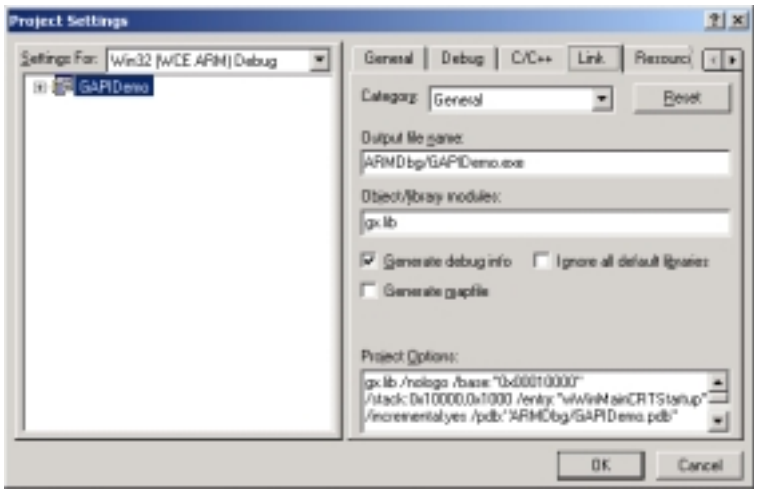
(1) Platform setting

We will test this demo directly in real Pocket PC device. In the eVC IDE, select the *Platform* as *Pocket PC 2002, Win32 [WCE x86] Debug*, and *Pocket PC 2002 Emulation*, which is shown as follows.



(2) Including Game API Library

To use GAPI, we need to include *gx.lib*. Click *Project* → *Settings* in the eVC IDE mean bar, you will see the following dialog box.



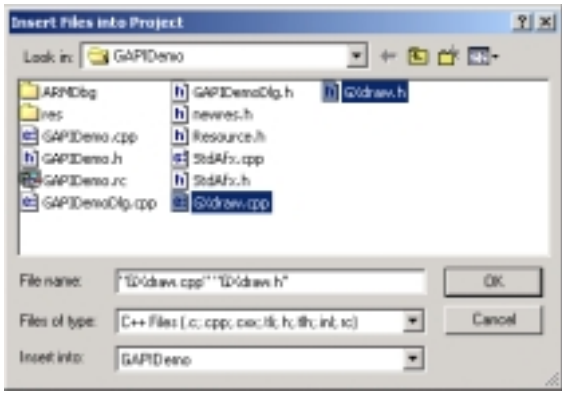
Choose "*Link*" tap. Type "*gx.lib*" in the "*Object/Library modules*" edit box.

(3) Add Existing modules to this project

There are several modules already provided for this demo.

(i). Copy and paste the following five files to your current working directory: *GXdraw.h* and *GXdraw.cpp*

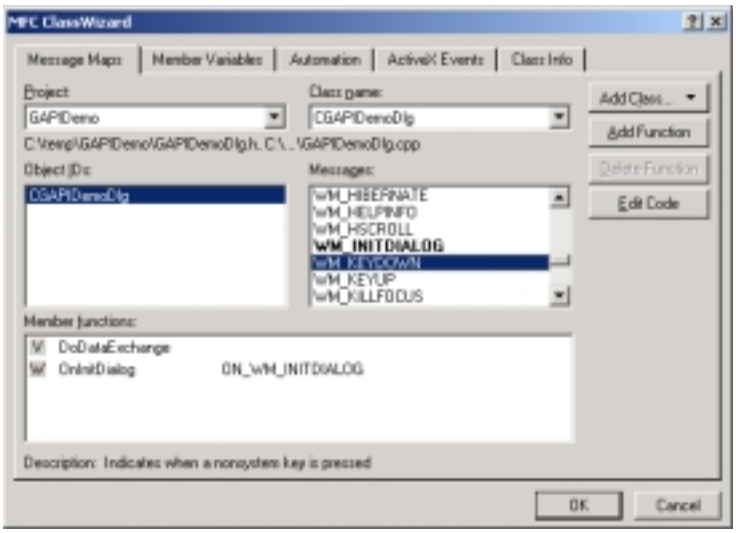
(ii). Click *Project* → *Add to Project* → *Files* from the eVC menu bar.



Add these five files into this project and click *OK*.

(4). Overriding Member functions of Dialog Window

To detect the actions of hardware keys, we need to override the member function, *OnKeyDown*, for our own usage. Also, we need to terminate this game application by calling *GXCloseInput* and *GXCloseDisplay* in *OnDestroy*. Click *View* → *ClassWizard* in the eVC IDE mean bar, a *MFC ClassWizard* dialog box will pop up.



Choose "*GAPIDemoDlg*" in the *Class name* list box and double click "*WM_KEYDOWN*" and "*WM_DESTROY*" in the *Messages* window. You can edit overriding codes by clicking "*Edit Code*" button.

(e) Edit *GAPIDemo.h*[3]

Double click *GAPIDemo.h* on the *File View* window to edit the header file.

```
/* GAPIDemoDlg.h : header file
   Game API Demo

   Author: Guan-Ming Su <gmsu@glue.umd.edu>
   Date: 01/17/03
*/

// include header -->
#include <gx.h>
#include "GXdraw.h"
// <--

#if !defined(AFX_GAPIDEMODLG_H__835169C8_C8CB_4BA1_A992_FCAB85936F4C__INCLUDED_)
#define AFX_GAPIDEMODLG_H__835169C8_C8CB_4BA1_A992_FCAB85936F4C__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

/////////////////////////////////////////////////////////////////////////////
// CGAPIDemoDlg dialog

class CGAPIDemoDlg : public CDialog
{
// Construction
public:
        CGAPIDemoDlg(CWnd* pParent = NULL);          // standard constructor
```

[3] Notice that "*// **Add your own code** →*" and "*//←*" indicates that you may put some codes in these areas. In this example, we use italic red font for the added codes (to distinguish from the codes already automatically generated by eVC).

```
        // add your own public variables here →
        GXDisplayProperties g_gxdp;   // structure contains information about the video frame buffer
        GXKeyList gxKeys;             // structure provides information about the hardware button

        unsigned int curPosX, curPosY;  // current (X,Y) position
        unsigned int DrawStep;          // distance between current pixel and previous pixel
        unsigned int MaxDrawStep;       // Maximal distance for two consecutive pixels
        int ColorIndex;                 // An index of color for drawing a pixel
        int maxColorNum;                // Number of color we define in this demo
        // ←

// Dialog Data
        //{{AFX_DATA(CGAPIDemoDlg)
        enum { IDD = IDD_GAPIDEMO_DIALOG };
                // NOTE: the ClassWizard will add data members here
        //}}AFX_DATA

        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CGAPIDemoDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:
        HICON m_hIcon;

        // Generated message map functions
        //{{AFX_MSG(CGAPIDemoDlg)
        virtual BOOL OnInitDialog();
        afx_msg void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);
        afx_msg void OnDestroy();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft eMbedded Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_GAPIDEMODLG_H__835169C8_C8CB_4BA1_A992_FCAB85936F4C__INCLUDED_)
```

(f) Edit *GAPIDemo.cpp*[4]

```
/* GAPIDemoDlg.cpp : implementation file
   Game API Demo

   Author: Guan-Ming Su <gmsu@glue.umd.edu>
   Date: 01/17/03
*/

#include "stdafx.h"
#include "GAPIDemo.h"
#include "GAPIDemoDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


// add your own global variables here →
COLOR GXCOLOR[]={   {0,0,0},        // black
```

[4] In this file, we use ❶, ❷, …, ❾ to illustrate the procedures listed in Section 1(c).

```
                                {255,0,0},      // red
                                {0,255,0},      // green
                                {0,0,255},      // blue
                                {255,0,255},    // magenta
                                {90,90,90},     // dark gray
                                {90,0,0},       // dark red
                                {0,90,0},       // dark green
                                {0,0,90}};      // dark blue
// ←

/////////////////////////////////////////////////////////////////////////////
// CGAPIDemoDlg dialog

CGAPIDemoDlg::CGAPIDemoDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CGAPIDemoDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CGAPIDemoDlg)
                // NOTE: the ClassWizard will add member initialization here
        //}}AFX_DATA_INIT
        // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
        m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CGAPIDemoDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CGAPIDemoDlg)
                // NOTE: the ClassWizard will add DDX and DDV calls here
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CGAPIDemoDlg, CDialog)
        //{{AFX_MSG_MAP(CGAPIDemoDlg)
        ON_WM_KEYDOWN()
        ON_WM_DESTROY()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CGAPIDemoDlg message handlers

BOOL CGAPIDemoDlg::OnInitDialog()
{
        CDialog::OnInitDialog();

        // Set the icon for this dialog.  The framework does this automatically
        //  when the application's main window is not a dialog
        SetIcon(m_hIcon, TRUE);                         // Set big icon
        SetIcon(m_hIcon, FALSE);                // Set small icon

        CenterWindow(GetDesktopWindow());       // center to the hpc screen

        // TODO: Add extra initialization
        // Add your own initialization here -->

        // get current window's handle
        CWnd *CCW = GetDesktopWindow();
        HWND m_hWnd = CCW->GetSafeHwnd();

        // ❶ Initialize GAPI
        if( GXOpenDisplay(m_hWnd, GX_FULLSCREEN) == 0 ){
                AfxMessageBox(_T("Cannot initialize GAME API with fullscreen")); }

        // ❷ Captures the buttons for the game
        GXOpenInput();
```

```
        // ❸ Obtain  information about the video frame buffer
        g_gxdp = GXGetDisplayProperties();
        if( !(g_gxdp.cBPP == 16) || !(g_gxdp.ffFormat | kfDirect565)){
                AfxMessageBox(_T("Full 16-bit color display is required!"));
                GXCloseDisplay();
                return FALSE;
        }

        //  ❹ Obtain information about the hardware button assignment
        gxKeys = GXGetDefaultKeys(GX_NORMALKEYS);


        // other initialization used in this demo
        curPosX = g_gxdp.cxWidth/2;
        curPosY = g_gxdp.cyHeight/2;

        MaxDrawStep = 20;
        DrawStep = 1;

        ColorIndex = 0;
        maxColorNum = 8;

        //<--

        return TRUE;  // return TRUE  unless you set the focus to a control
}



void CGAPIDemoDlg::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
        // TODO: Add your message handler code here and/or call default

        // Add the response to Hardware key here -->
        unsigned char *VidMem;

        // ❺ CAlled to access the video frame buffer for drawing
        VidMem = (unsigned char *)GXBeginDraw();

        // ❻ Perform drawing
        short vkKey;
        vkKey = (short)nChar;

        if( vkKey == gxKeys.vkLeft){          // Left navigation key
            if( curPosX >0){    // not reach the left border of window
                curPosX = curPosX - DrawStep;
                DrawPixel16(VidMem, curPosX, curPosY, GXCOLOR[ColorIndex], g_gxdp.cbxPitch,g_gxdp.cbyPitch);
            }}
        else if( vkKey == gxKeys.vkRight ){   // Right navigation key
            if( curPosX < g_gxdp.cxWidth-1){  // not reach the right border of window
                curPosX = curPosX + DrawStep;
                DrawPixel16(VidMem, curPosX, curPosY, GXCOLOR[ColorIndex], g_gxdp.cbxPitch,g_gxdp.cbyPitch);
            }}
        else if(vkKey == gxKeys.vkUp){       // Up navigation key
            if( curPosY >0){          // not reach the top
                curPosY = curPosY - DrawStep;
                DrawPixel16(VidMem, curPosX, curPosY, GXCOLOR[ColorIndex], g_gxdp.cbxPitch,g_gxdp.cbyPitch);
            }}
        else if(vkKey == gxKeys.vkDown) {     // Down navigation key
            if( curPosY < g_gxdp.cyHeight-1){    // not meet the bottom
                curPosY = curPosY + DrawStep;
                DrawPixel16(VidMem, curPosX, curPosY, GXCOLOR[ColorIndex], g_gxdp.cbxPitch,g_gxdp.cbyPitch);
            }}
        else if(vkKey == gxKeys.vkA) {       // Application Button A
            if(DrawStep < MaxDrawStep){      // increase the drawing step
                DrawStep = DrawStep + 1; }}
        else if(vkKey == gxKeys.vkB) {       // Application Button B
```

```
                if(DrawStep >0){            // decrease the drawing step
                        DrawStep = DrawStep - 1; }}
        else if(vkKey == gxKeys.vkC) {      // Application Button C
                if(ColorIndex < maxColorNum-1){   // change color
                        ColorIndex = ColorIndex + 1; }
                else{    // reset ColorIndex to 0
                        ColorIndex = 0;}}

        // ❼ drawing is complete
        GXEndDraw();

        //<--

        CDialog::OnKeyDown(nChar, nRepCnt, nFlags);
}

void CGAPIDemoDlg::OnDestroy()
{
        CDialog::OnDestroy();

        // TODO: Add your message handler code here
        // add your own procedures for destroying this window -->

        // ❽  Free the buttons for normal use
        GXCloseInput();

        // ❾  Close GAPI, Clear up GAPI resource
        GXCloseDisplay();

        // <--

}
```

(g) Compile and Execute

Compile you program by clicking  and run it by clicking  on the menu bar. You can use the navigation keys (*vkUp*, *vkDown*, *vkLeft*, and *vkRight*, which are shown in section 1(b)(2)) to draw lines. To change the color, press *vkC*, which is originally designed for recoder. You also can plot dotted lines with larger distance between colored pixel using *vkA* and smaller distance with *vkB*.

3. Other "Game" API

Several third party companies provide similar APIs to Microsoft GAPI.

(a) GapiDraw: http://www.gapidraw.com/
(b) GapiTools: http://www.gapitools.com/