# Using VC++.NET for MicroStation v8 programming

Stanislav Sumbera

## Summary

VC++.NET offers perfect platform for developing traditional MDL application together with innovative .NET technology. This research is focused on integration .NET Forms with MDL applications.

Date: February 2003, Programming languages : VC++.NET, Published in PenBrush's CAD III/Q 2003, pp 30-33

Download test code

see discussion on newsgroup news:N5y9V4E2CHA.2260@prdweb004.viecon.com

## Intro'dust'ion

Nowadays we may here .NET everywhere. Recently I have published article on .NET and VBA interoperability where you may find basic information. However you as core MDL programmer you would like to get something more natural to get smoothly into .NET. I mean we need direct access to MDL API without any P/Invoke. The ultimate solution is VC++.NET. It offers mixing managed (pure .NET code) with unmanaged (traditional native code) into one piece of code. Oh yeah ! this allows nearly miracles, specially through technology called IJW -It Just Work! In VC++.NET you are a king of .NET as well as native code and of course of MDL API. Moreover with beta release of Visual Studio 2003 it is possible to visually design .NET Forms GUI even for VC++. OK, still reading ? today is not 1.April...

## Designing one little .NET box

VisualStudio.NET is very comfortable for developing VC++ application. After visual design of dialog box, it is necessary to add some support for MicroStation native window handling and for mixing native data with managed garbage collected class:

## lets create a class for handling pointers to registered MicroStation native window and for hook

```
class HwndWrap
{
```

```
public: GuiWindowP pWindow;    // handler for MicroStation native window
public: HHOOK hHook;            // hook handler for routing messages
};
```

here is a snippet of dllForm.h - the file is originally made from visual design of GUI but here it must be slightly modified for MicroStation

```
#pragma once
#include "stdafx.h"
#include "windows.h"
#undef MessageBox

#define winNT
#define NO_BOOLEAN_TYPE
#include "mssystem.fdf"
#include "msnativewindow.h"
#include "HwndWrap.h"

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;

namespace NETDLL{

        public __gc class dllForm : public System::Windows::Forms::Form {

        public:

                HwndWrap *wrapper;    // handler of native pointers

                dllForm(void);
                ~dllForm();

                // GUI definition follows here
                private: System::Windows::Forms::TextBox * textBox1;
                private: System::Windows::Forms::MonthCalendar * monthCalendar1;


                // EVENTS..button and size changed
                private: System::Void button1_Click(System::Object * sender,
                   System::EventArgs * e){
                   //... do something from MDL on button press
                        mdlSystem_newDesignFile("nic.dgn");
                }

                private: System::Void dllForm_SizeChanged(System::Object * sender,
                  System::EventArgs * e){
                // let MS adjust window min and max size
                  if (this->WindowState == FormWindowState::Minimized)
                                mdlNativeWindow_minimize(wrapper->pWindow);
                   else if (this->WindowState == FormWindowState::Maximized)
                               mdlNativeWindow_maximize(wrapper->pWindow);
                }
        }
```

## let's create a storage for our form managed object

```
using namespace NETDLL;
__gc class ManObj {
        public:   static dllForm *form;
 };
```

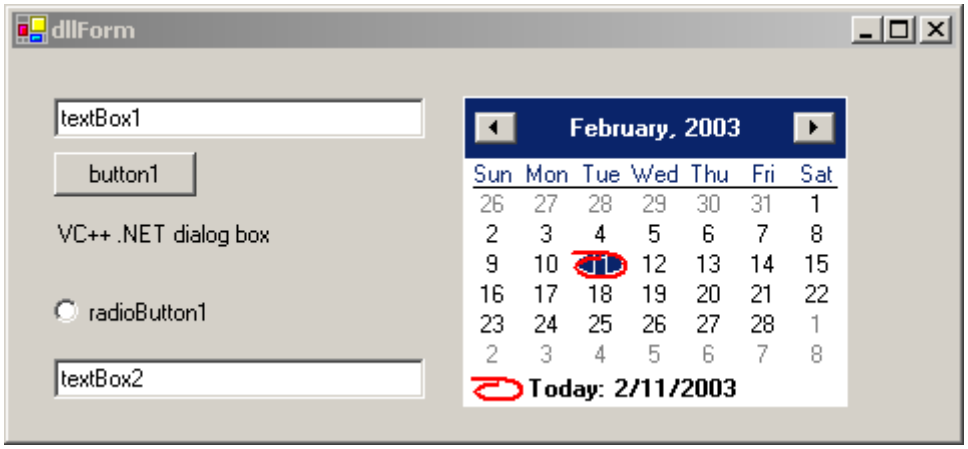
## constructor and destructor of .NET form

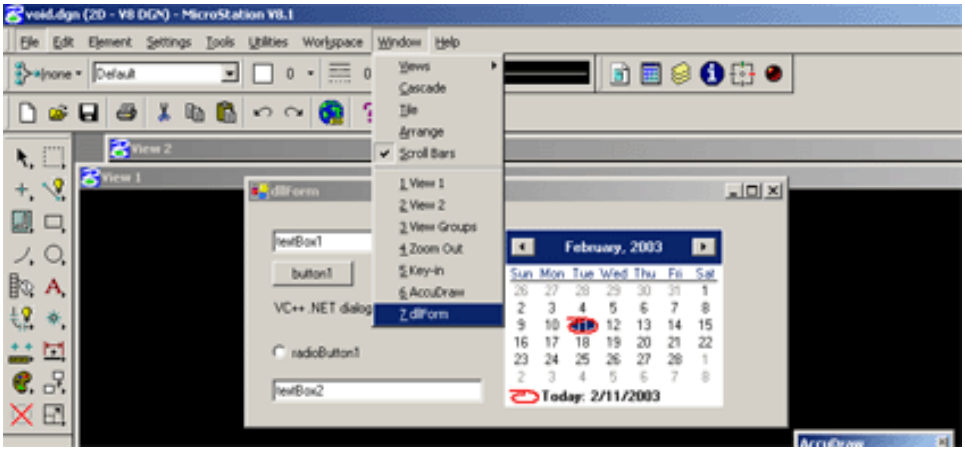
```
dllForm::dllForm(void) {
 wrapper = new HwndWrap();
 InitializeComponent();
 this->Show();
 HWND formWin = FindWindow(NULL,"DllForm"); // not perfect...
 mdlNativeWindow_initialize("NETDLL"); // init native window for MS
 mdlNativeWindow_createMSWindow (&(wrapper->pWindow), formWin, FALSE, TRUE,1);
 mdlNativeWindow_addToWindowList(wrapper->pWindow); // add to window list in MS
 mdlNativeWindow_setAsChild(wrapper->pWindow,0,FALSE); // set parent to MS
 // set redirection for hooks ..still same old way..
 wrapper->hHook = SetWindowsHookEx( WH_GETMESSAGE, GetMsgProc, NULL,
GetCurrentThreadId());
}

dllForm::~dllForm(){
 mdlNativeWindow_removeFromWindowList(wrapper->pWindow);
 mdlNativeWindow_destroyMSWindow(wrapper->pWindow,FALSE);
 UnhookWindowsHookEx(wrapper->hHook);
delete wrapper;
}
```

## finally export function for MDL and create dialog

```
extern "C"
__declspec(dllexport) void loadNETbox(){
    ManObj::form = new dllForm();
}
```

## .NET VC++ dialog box inside MicroStation v8.1






## Conclusion

VC++.NET is perfect and straightforward for use in mixed environments where traditional native as well as .NET code must be used. Transparency of native and .NET code is amazing. Even MFC is still alive, it is more or less deprecated object oriented technology so use .NET forms instead.