

Mouse / Keyboard Events when using SimpleFramework on a windows form

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.directx.managed/2006-03/ms>

- *From:* "belliez" <belliez@xxxxxxxxxxxxxxxx>
 - *Date:* 24 Mar 2006 07:11:24 -0800
-

Hi,

Has anyone managed to get the latest version of the Sample Framework (Feb 2006 and Dec 2005) to work with full mouse and keyboard events. (I have added my code at the bottom of this post)

I have been trying now for 2 weeks with no luck. I can see the mouse messages on the form and by adding hooks to the mouse on `InitialiseGraphics()` I can also capture the mouse message in the `directx` application but I am not sure how I can pass this onto the framework to process so that when I press the left mouse button and move the mouse the model will move on screen. This works perfectly if I use `CreateWindow` but not if I use `SetWindow`.

I am using Simple Animation from the SDK browser and have removed the `CreateWindow(..)` and replaced it with:

```
sampleFramework.SetWindow(frm, frm, frm, true);
```

where `frm` is the form control (eventually it will be a panel) where I want the `directx` rendered to.

After stepping through the original Simple Animation and comparing with my new Simple Animation that uses an existing window instead of creating a new window I can see that I am not getting the `OnMsgProc()` event.

Can someone please help as to why I cannot see this being fired and maybe suggest a fix so that I can.

The code that calls this event is below and is unchanged from SDK Simple Animation apart from using `this.OnMsgProc` instead of `sample.OnMsgProc` (entire code for this method is at the end of the post).

Mouse / Keyboard Events when using SimpleFramework on a windows form

```
sampleFramework.SetWndProcCallback(new  
WndProcCallback(this.OnMsgProc));
```

What I have done so far is created a new project that shows a form called Form1.

I next referenced the simple animation project from the SDK and changed the outout type to class output in the project preferences.

I next renamed main() to InitialiseGraphics() in the Simple Animation sample so that there is only one Main() which resides in Form1.cs

On the constructor to Simple Animation I pass in the form reference, create a new Framework and pass the form reference to InitialiseGraphics() above.

InitialiseGraphics will now call SetWindow() using the frm reference passed in. I have also added some hooks to MouseDown and MouseMove but am not sure if these will be useful for passing messages to the sample framework.

I remove the last line:

```
sampleFramework.MainLoop()
```

and instead I call RunLoop() from form1 main (see below):

Thank you in advanced for your help / advice etc (esp if you think I am coding this incorrectly – I am from an embedded c background not an object oriented background).

Paul (belliez)

===== CODE BELOW =====

Form1 main() function looks like this:

```
static void Main()  
{  
Application.EnableVisualStyle();  
Guardian_GUI.GuardianGui frmGui = new GuardianGui();  
dxViewer.dxViewer viewer = new dxViewer.dxViewer(frmGui);  
viewer.RunLoop(frmGui);  
frmGui.Close();  
return;  
}
```

Simple Animation (was main) InitialiseGraphics() looks like this:

Mouse / Keyboard Events when using SimpleFramework on a windows form

```
public dxViewer(System.Windows.Forms.Form frm)
{
// Store framework
this.sampleFramework = new Framework();

// Create dialogs
hud = new Dialog(sampleFramework);

// Initialise Everything
InitialiseGraphics(frm);
}

public int InitialiseGraphics(System.Windows.Forms.Form frm)
{

// Set the callback functions. These functions allow the sample
framework to notify

// the application about device changes, user input, and windows
messages. The

// callbacks are optional so you need only set callbacks for events
you're interested

// in. However, if you don't handle the device reset/lost callbacks
then the sample

// framework won't be able to reset your device since the application
must first

// release all device resources before resetting. Likewise, if you
don't handle the

// device created/destroyed callbacks then the sample framework won't
be able to

// recreate your device resources.

sampleFramework.Disposing += new
EventHandler(this.OnDestroyDevice);
sampleFramework.DeviceLost += new EventHandler(this.OnLostDevice);
sampleFramework.DeviceCreated += new
DeviceEventHandler(this.OnCreateDevice);
sampleFramework.DeviceReset += new
DeviceEventHandler(this.OnResetDevice);
sampleFramework.SetWndProcCallback(new
WndProcCallback(this.OnMsgProc));
sampleFramework.SetCallbackInterface(this);

try
```

Mouse / Keyboard Events when using SimpleFramework on a windows form

```
{
// Show the cursor and clip it when in full screen
sampleFramework.SetCursorSettings(true, true);

// Initialize
this.InitializeApplication();

// Initialize the sample framework and create the desired
window and Direct3D
// device for the application. Calling each of these functions
is optional, but they
// allow you to set several options which control the behavior
of the sampleFramework.

sampleFramework.Initialize(true, true, true); // Parse the
command line, handle the default hotkeys, and show msgboxes

//sampleFramework.CreateWindow("dxViewer");
//sampleFramework.SetWinformWindow(frm, frm, frm);

sampleFramework.SetWindow(frm, frm, frm, true);

// Hook the keyboard event
sampleFramework.Window.KeyDown += new
System.Windows.Forms.KeyEventHandler(this.OnKeyEvent);

//sampleFramework.Window.MouseMove += new
System.Windows.Forms.MouseEventHandler(this.OnMouseMove);

//sampleFramework.Window.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.OnMouseDown);

sampleFramework.CreateDevice(0, true, frm.Width-100,
frm.Height-100, this);

// Pass control to the sample framework for handling the
message pump and
// dispatching render calls. The sample framework will call
your FrameMove
// and FrameRender callback when there is idle time between
handling window messages.

//sampleFramework.MainLoop();
}
#if(DEBUG)
catch (Exception e)
{

// In debug mode show this error (maybe – depending on settings)
sampleFramework.DisplayErrorMessage(e);
#else
```

Mouse / Keyboard Events when using SimpleFramework on a windows form

```
catch
{
// In release mode fail silently
#endif

// Ignore any exceptions here, they would have been handled by
other areas

return (sampleFramework.ExitCode == 0) ? 1 :
sampleFramework.ExitCode; // Return an error code here
}

// Perform any application-level cleanup here. Direct3D device
resources are released within the
// appropriate callback functions and therefore don't require any
cleanup code here.

return sampleFramework.ExitCode;
} //END FUNCTION
```

```
private void OnMouseDown(object sender,
System.Windows.Forms.MouseEventArgs e)
{
// Mouse Down captured here but I dont know how to pass this
message onto the framework for processing
}
```

```
private void OnMouseMove(object sender,
System.Windows.Forms.MouseEventArgs e)
{
// Mouse Move captured here but I dont know how to pass this
message onto the framework for processing
}
```

.