GraphWorX64 Scripting
3D Viewport Scripting

APPLICATION NOTE

September 2013

Description: A scripting guide demonstrating how to interact with a 3D Viewport control.
OS Requirement: Vista x64/ Windows 7 x64/ Windows 8 x64/ Windows Server 2008 x64/ Windows Server 2008 R2 x64/ Windows Server 2012
General Requirement: General knowledge of GraphWorX64 and its scripting capabilities; familiarity with the 3D Viewport control.

Introduction

This guide will introduce you to writing scripts that interface with the 3D Viewport. It includes an example of adding and removing a pipe object.

This document assumes that you have read the GraphWorX64 Scripting – Quick Start application note.

Preparing the display

To create this example we will need a display containing a 3D Viewport control, plus four buttons to trigger our scripts. Start GraphWorX64, create a new display, then add a 3D Viewport and four buttons. Label the buttons like so:

- Add Pipe
- Remove Pipe
- View 1
- View 2

Add two named views to your 3D Viewport. Name them View 1 and View 2. Make sure you can see a difference between the two views.

NOTE: See the help files for how to create named views.

It is very important that the 3D Viewport control has a name. Our scripts will use this name to identify which 3D Viewport to interact with (in case there is more than one on a display). To name your 3D Viewport, select it and look at the Properties panel. In the Common section fill in "Scene3D" for the "(Name)" field.

Your display should look something like Figure 1 below.

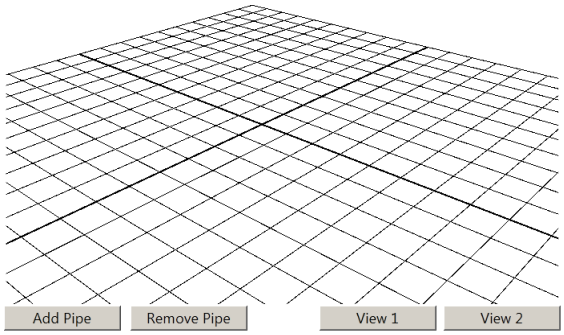
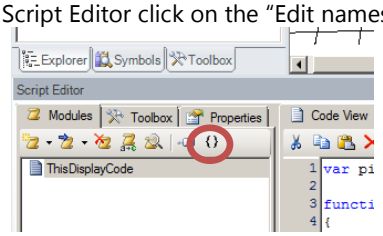


Figure 1 - Display with 3D Viewport and Buttons

Configure all four buttons to perform the Run Script command and create a new function for each of them from the Events page. (The procedure for this is described in the *GraphWorX64 Scripting – Quick Start* application note.)

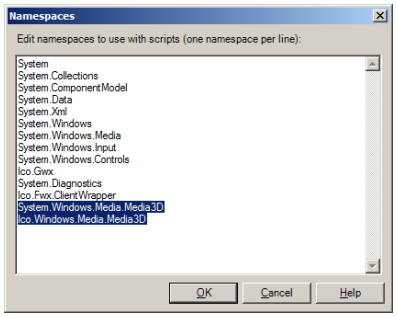
Writing the Add Pipe Script

First of all we will check the default namespaces. In your GraphWorX64 Script Editor click on the "Edit namespaces" icon



and add the following two lines if they are not on the list already:
System.Windows.Media.Media3D
Ico.Windows.Media.Media3D

The Namespaces list should look similar to



Back in the Scripting Editor, at the very top of the code window, outside of any functions, add this line:

```
var pipe : Pipe3D;
```

This will create a storage variable for our pipe object so that we can easily find it later in order to delete it.

Now, inside the function for the Add Pipe button we'll need to get a reference to our 3D Viewport object with this line of code:

```
var scene : Gwx3DControl =
ThisConfiguration.Document.GetElementByName("Sc
ene3D");
```

Then we'll need to create a new pipe object with these lines:

```
var points = new Point3DCollection();
points.Add(new Point3D(0.0, 0.0, 0.0));
points.Add(new Point3D(0.0, -1.0, 0.0));
points.Add(new Point3D(0.0, -1.0, -3.0));
points.Add(new Point3D(-4.0, -1.0, -3.0));
```

Copyright 2013 ICONICS, Inc.

And we'll add that pipe to the 3D Viewport with this line:

```
pipe = scene.Configuration3D.AddPipe(points);
```

Then we'll make sure we don't leave the pipe selected.

```
scene.SelectAllOfType(null);
```

NOTE: See the last page for the completed script.

Writing the Delete Pipe Script

The delete pipe script is very simple. We'll just need to get a reference to the 3D Viewport just like the first script, then we'll need to check if the pipe exists and delete it with these two lines:

```
if (pipe != null)
    scene.RemoveChild(pipe);
```

NOTE: See the last page for the completed script.

Writing the View Scripts

First we'll need to get a reference to the 3D viewport again. Then we'll set the animation duration for the camera:

```
scene.CameraAnimationDuration = 5;
```

Then we'll get one of the saved views and set it as our main view:

```
scene.MainCamera =
scene.FavoriteViews.Get("View 1");
```

The code for both view buttons will be the same, except one button will get "View 1" in the line above, and the other will get "View 2".

Before testing these views, you will probably want to add the pipe or some other objects to your viewer so you can tell when the view is changing.

NOTE: See the last page for the completed script.

Conclusion

This simple example shows some important concepts, like how to get a reference to a 3D Viewport control. This is very similar to the procedure for getting a reference to other controls as well. You've also learned about how to create an object and add it to the viewport.

Feel free to experiment with the other available methods. You can find these by using the Intellisense built into the Script Editor, or by browsing the GENESIS64 API Reference Help which can be launched from the start menu under ICONICS GENESIS64.

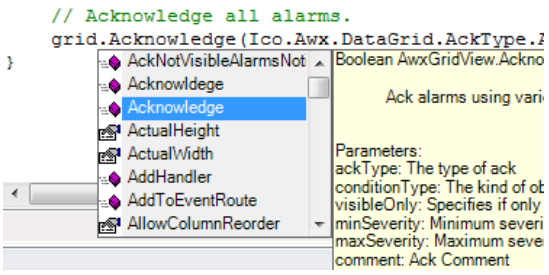


Figure 2 - Intellisense Showing Available Methods and Properties

ICONICS Customer Connection Portal hosts a library of automation examples that are very handy when you start scripting.

Complete code

Here is the full code of the functions we created above, along with some comments.

```
//Declare the pipe global variable
var pipe : Pipe3D;

function Pick1_CommandExecuted(sender : System.Object, cmdArgs : Ico.Gwx.CommandExecutionEventArgs)
{
    //Get a reference to the 3D Viewport
    var scene : Gwx3DControl = ThisConfiguration.Document.GetElementByName("Scene3D");

    //Create a new pipe object
    var points = new Point3DCollection();
    points.Add(new Point3D(0.0, 0.0, 0.0));
    points.Add(new Point3D(0.0, -1.0, 0.0));
    points.Add(new Point3D(0.0, -1.0, -3.0));
    points.Add(new Point3D(-4.0, -1.0, -3.0));

    //Add the pipe object to the viewport
    pipe = scene.Configuration3D.AddPipe(points);

    //Deselect the pipe
    scene.SelectAllOfType(null);
}

function Pick2_CommandExecuted(sender : System.Object, cmdArgs : Ico.Gwx.CommandExecutionEventArgs)
{
    //Get a reference to the 3D Viewport
    var scene : Gwx3DControl = ThisConfiguration.Document.GetElementByName("Scene3D");

    //If our pipe is there, remove it.
    if (pipe != null)
       scene.RemoveChild(pipe);
}

function Pick3_CommandExecuted(sender : System.Object, cmdArgs : Ico.Gwx.CommandExecutionEventArgs)
{
    //Get a reference to the 3D Viewport
    var scene : Gwx3DControl = ThisConfiguration.Document.GetElementByName("Scene3D");

    //Set the number of seconds for each camera change
    scene.CameraAnimationDuration = 5;

    //Change to the named view
    scene.MainCamera = scene.FavoriteViews.Get("View 1");

}


function Pick4_CommandExecuted(sender : System.Object, cmdArgs : Ico.Gwx.CommandExecutionEventArgs)
{
    var scene : Gwx3DControl = ThisConfiguration.Document.GetElementByName("Scene3D");
    scene.CameraAnimationDuration = 5;
    scene.MainCamera = scene.FavoriteViews.Get("View 2");
}
```