STK Engine Tutorial - C#

In this exercise you will gain hands-on experience using STK X to embed STK functionality in a container application created with C#.

Tutorial Source code

Completed project and C# source code can be found in the STK/STK Engine install at the following location:

<STK Install Folder>/CodeSamples/CustomApplications/CSharp/Tutorial

Create the project

1. Start Visual Studio.

2. From the File menu, select New Project….

3. Under Installed>Templates, select Visual C# as the Project type and the Windows Forms Application template. Enter a project name and location of your choice.

4. The **Form1.cs** form is opened in Design mode.

Add References

1. In the Solution Explorer window on the top right of the screen, right-click References and select Add Reference…

2. In the popup Reference Manager window, select the Browse tab on the left and then click the Browse… button on the bottom right of the window.

3. A File Explorer window will open. Navigate to the same directory as earlier, "C:\Program Files\AGI\STK 12\bin\Primary Interop Assemblies", and ctrl + click to add AGI.STKObjects.Interop.dll, AGI.STKUtil.Interop.dll, AGISTKVgt.Interop.dll, and AGI.STKX.Interop.dll.

4. Click Add to close the File Explorer and then click OK to close the Reference Manager window. You should now see these DLLs listed under References.

5.   Select the AGI.STKX.Interop reference, and then in the Properties window, change Embed Interop Types to "False".

Add the STK X controls to the Toolbox

1. If the Toolbox is not already displayed, select it from the View menu.

2. Right click inside the toolbox's General section and select "Choose Items..." from the context menu.

3. On the ".NET Framework Components" tab, click the browse button and browse to the following location: <STK Install Folder>/bin/Primary Interop Assemblies.

4. Select the AGI.STKX.Controls.Interop.dll and click the Open button.

5. Click the OK button. The controls are now ready to be drag and dropped onto your Windows form.

Send commands to STK X

1. Drag the AxAgUiAxVOCntrl onto **Form1.cs**. Place and size the control it so that it approximately fills the right-hand portion of the form. Then drag the AxAgUiAx2DCntrl onto the left-hand portion of the form, sizing it so that it leaves some space for other controls to be added.
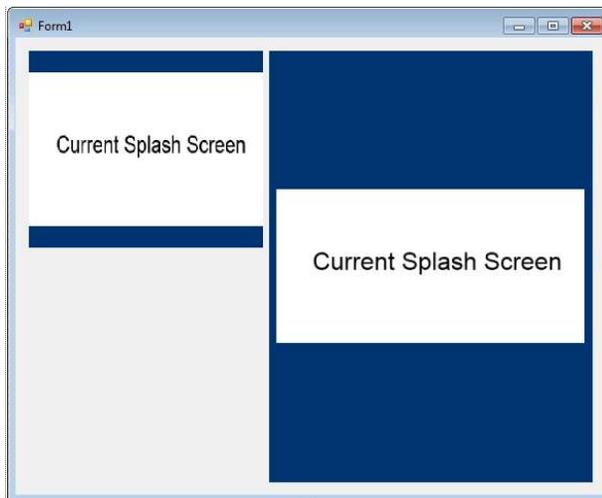
2. Add three buttons and a label to the form, arranged as shown here:
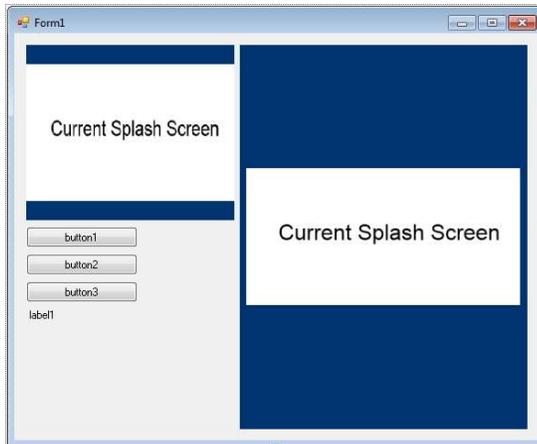
3. Change the Text property for the top button to New Scenario.

4. Double-click the New Scenario button in the Designer to expose the code for its Click event handler.

5. Modify the **button1_Click** method by adding the following line:

6. this.axAgUiAx2DCntrl1.Application.ExecuteCommand("New / Scenario Test");

7. Build and run the application.

8. Click the New Scenario button. It may take a few minutes for the results to display.
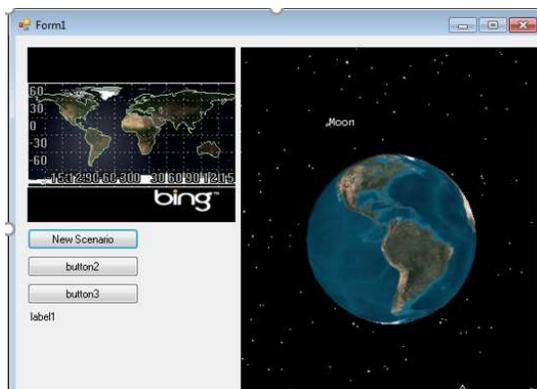
9. Move the mouse in the **Globe** window to rotate the globe (holding down the left mouse button) or zoom in and out on it (holding down the right mouse button).

Add Zoom In/Out to the Map control

1. After closing the Form1 window, change the Text properties of the remaining 2 buttons to Zoom In and Zoom Out.

2. Double-click on the Zoom In button to expose the code for the click event handler.

3. Add the following line to the event handler:

4. this.axAgUiAx2DCntrl1.ZoomIn();

5. Similarly, modify the click event handler for the Zoom Out button by adding the following:

6. this.axAgUiAx2DCntrl1.ZoomOut();

7. Build and run the application, and click the New Scenario button.

8. Click the Zoom In button. Use the mouse to define the zoom area in the **Map** window.

9. You can zoom out by clicking the Zoom Out button.

10. Close the Form1 window.

Respond to events raised by STK X

1. In the Visual Studio IDE, switch to the Design view.

2. Select the Map control, and display its **Properties** dialog.

3. In the **Properties** dialog, click the (lightning bolt) icon.

4. On the line for the DblClick event, enter On2DMapDblClick and press Enter.
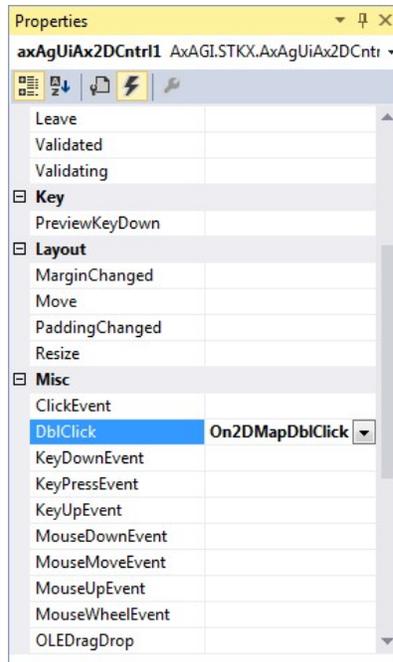
5. This exposes the code for the new event handler that you added. Modify that code as follows, add the following line to the event handler:

6. MessageBox.Show("2D Map double click");

7. Build and run the application.

8. Double-click in the **Map** window. The message box will be displayed.

9. You can now use the same approach to hook up to Globe control events.

10. You will now respond to STK X application events. Add the line below to the form code at the beginning of the Form1 class definition (see note below). This adds the **stkxApp** member variable to your form. Through this variable you can access the STK X application object.

```
private AGI.STKX.AgSTKXApplication stkxApp;
```

NOTE: Strictly speaking, Form1 is handled as 2 partial classes in Visual Studio, appearing in two separate source files. The intent of the above instruction is that you add the stkxApp declaration to the beginning of the partial class defined in Form1.cs, though you could instead have added it to the list of global variable declarations found in Form1.Designer.cs.

11. You will now subscribe for notification on the STK X application object. As this object is not a control, the procedure is a little bit different. Add the following lines to the form constructor, after the call to InitializeComponent():

12. this.stkxApp = new AGI.STKX.AgSTKXApplication();

```
this.stkxApp.OnScenarioNew += this.OnNewScenario;
```

13. Add the following callback function to the Form1 class:

```
private void OnNewScenario(String path)
{
        MessageBox.Show("New scenario created: " + path);
}
```

14. You will now have to make sure that your stkxApp object is released properly. Go to Design view, display the **Properties** dialog for the form, click the ⚡ (lightning bolt) icon, and select the FormClosing; type in "FinalRelease" and press Enter.

15. Add the following code to your new event handler:

System.Runtime.InteropServices.Marshal.ReleaseComObject(stkxApp);

```
GC.WaitForPendingFinalizers();
```

16. Build and run the application.

17. Click the New Scenario button. A message box will be displayed as a result of this event.

Add map picking

1. Return to the Design view of Form1 in Visual Studio. Bring up the **Properties** dialog for the label, and clear the Text property (so that no text displays initially).

2. Select the Globe control. In its **Properties** dialog, click the ⚡ (lightning bolt) icon.

3. On the line for the MouseMoveEvent event, enter `On3DMouseMove` and press Enter.

4. Enter the following code for your new event handler:

```
AGI.STKX.IAgPickInfoData pickInfoData =
        this.axAgUiAxVOCntrl1.PickInfo(e.x, e.y);
if (pickInfoData.IsLatLonAltValid)
{
        this.label1.Text= "Lat: " + pickInfoData.Lat+ "\r\nLon: "  +
                pickInfoData.Lon;
}
else
{
        this.label1.Text="";
}
```

5. Build and run the application.

6. Click the New Scenario button. Move the mouse over the Globe control. The mouse move event will be called. The mouse coordinates are converted to latitude/longitude and displayed in the label.

Set STK X properties

1. Click on the Globe control in the Properties window you can now change the 'BackColor' property. Click on the dropdown box in the right column. The **Color** dialog box allows you to set the background color of the Globe control's viewing square. Set the color to a color or your choice. The background color on the Globe object will change.
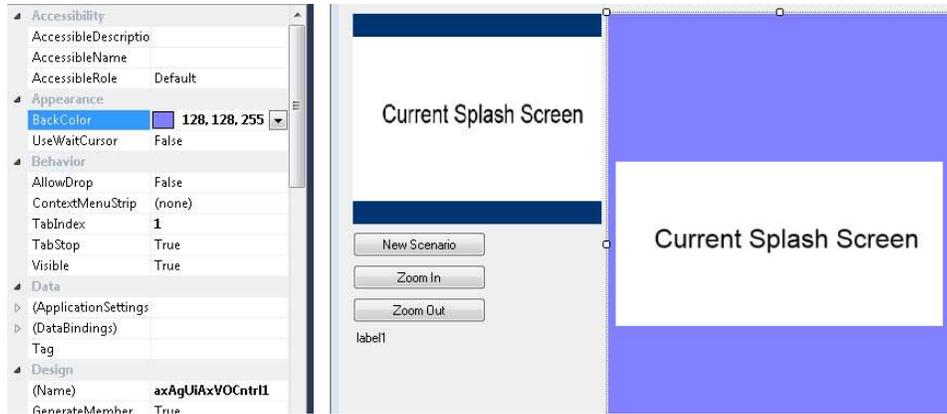
2. Click on the 'Picture' property in the properties window. Click on the [...] (expand button) in the right column to bring up the "Select Resources" dialog box which shows picture properties. These properties allow you to change or remove the splash panel displayed in the STK X control.

3. Click on Import… and navigate to an alternate picture.

4. Select the picture, and then click OK. Your picture will now be shown as the splash panel when no scenario is loaded.