Windows API Code Pack – Show Mesh

Contents

Introduction

Software Libraries Needed

To use Direct X in managed programs, a wrapper is needed. There are several wrappers available. Some of them are listed below.

- Windows® API Code Pack for Microsoft® .NET Framework. See http://archive.msdn.microsoft.com/WindowsAPICodePack
- SharpDX. See http://sharpdx.org/
- SlimDX. See http://slimdx.org/

This sample uses Windows API Code Pack to display 3D scene.

Your computer needs to install Direct X SDK:

http://www.microsoft.com/en-us/download/details.aspx?id=6812

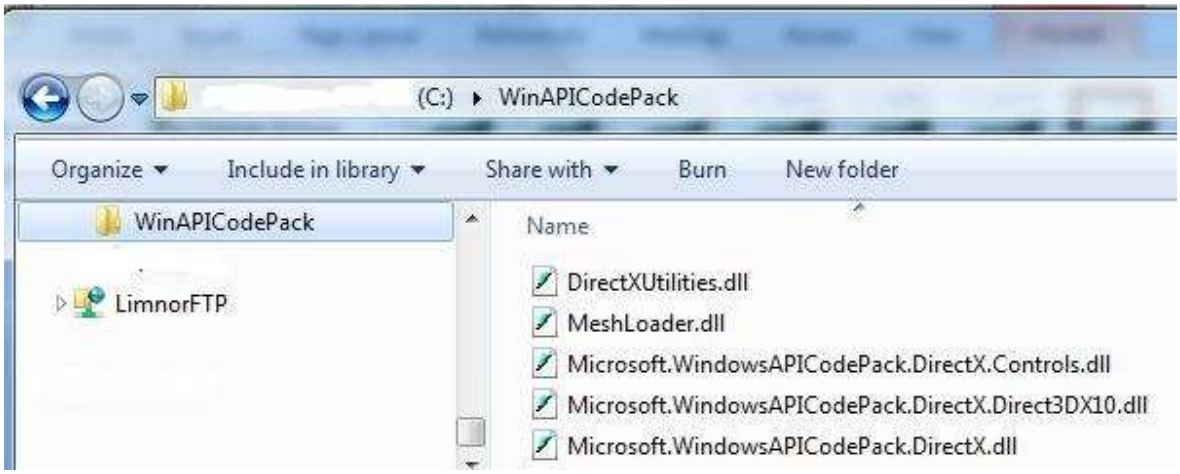
Your computer also needs to have Windows API Code Pack. You may download it from Microsoft web site:

http://archive.msdn.microsoft.com/WindowsAPICodePack/Release/ProjectReleases.aspx?ReleaseId=4906

Or, you may get the binaries from the following zip file:

http://www.limnor.com/studio/WinAPICodePackBin.zip

For this sample, we assume the above zip file is unzipped into folder C:\WinAPICodePack:



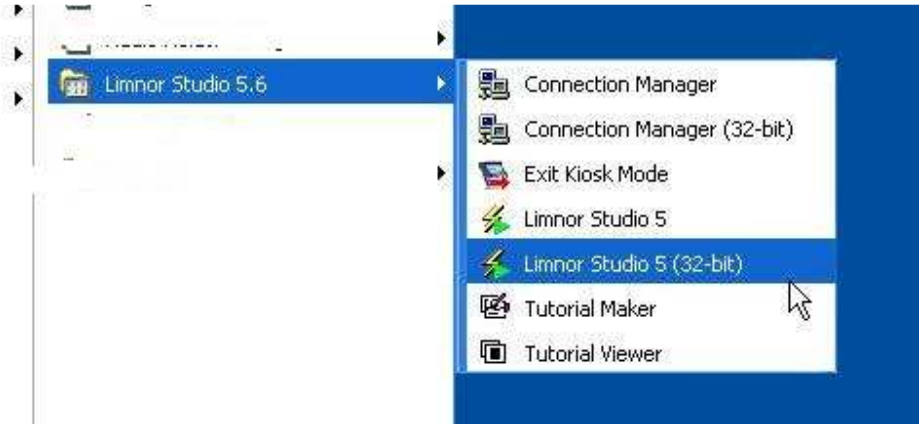
This sample project can be downloaded from

http://www.limnor.com/studio/WinAPICodePackDX10ShowMesh.zip

Target Platform

The zip file, http://www.limnor.com/studio/WinAPICodePackBin.zip, contains Windows API Code Pack targeting platform x86. Pay attention to following issues:

- If you use Limnor Studio 5.6 then use Limnor Studio 5.6 (32-bit).

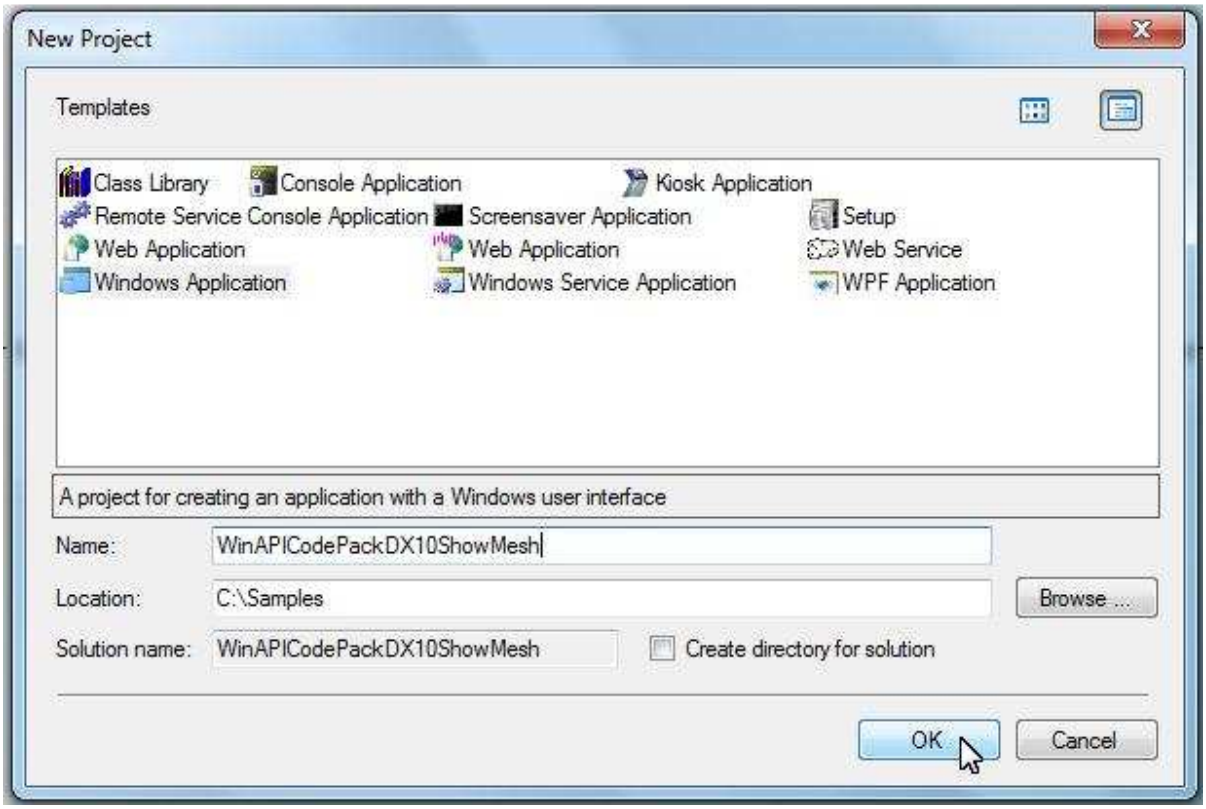


- Your projects needs to be targeting x86 platform. See http://www.limnor.com/studio_x86.html

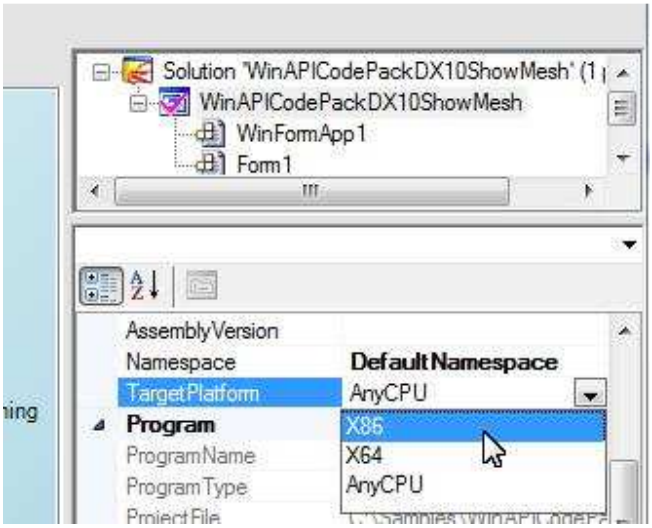
Create a Windows Form Application

This sample is a Windows Form Application.

Name the new project “WinAPICodePackDX10ShowMesh”



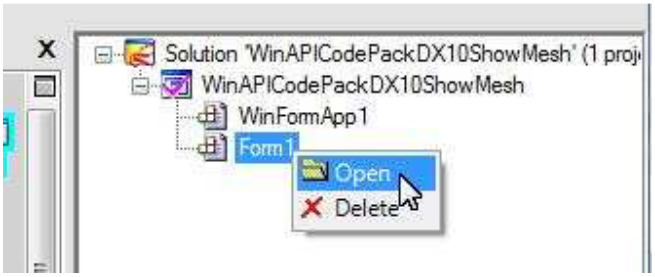
For this sample, set target platform to x86:

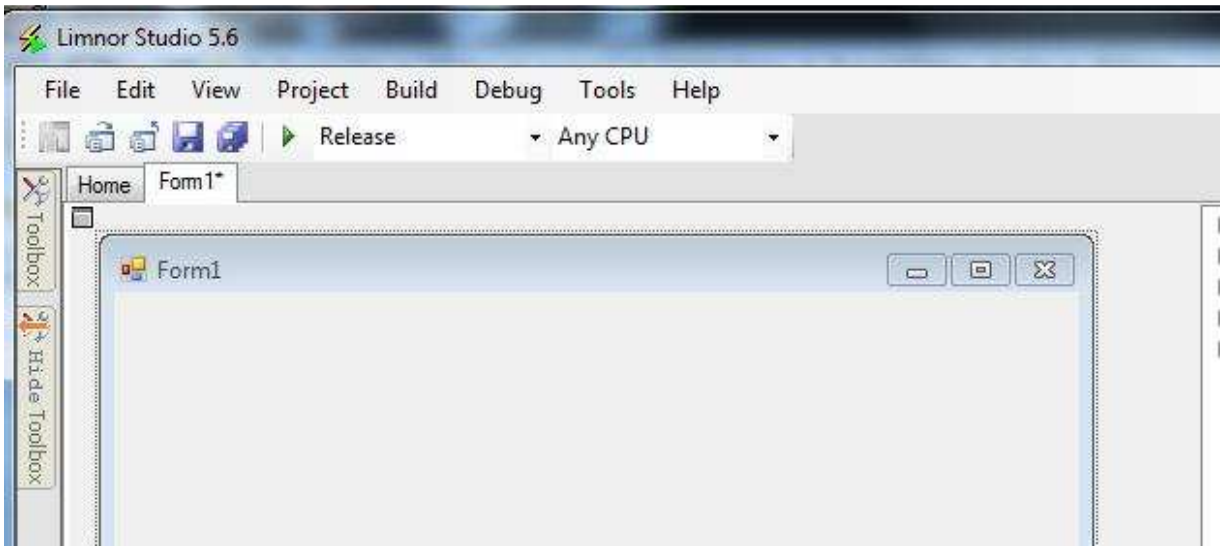


Use DirectControl

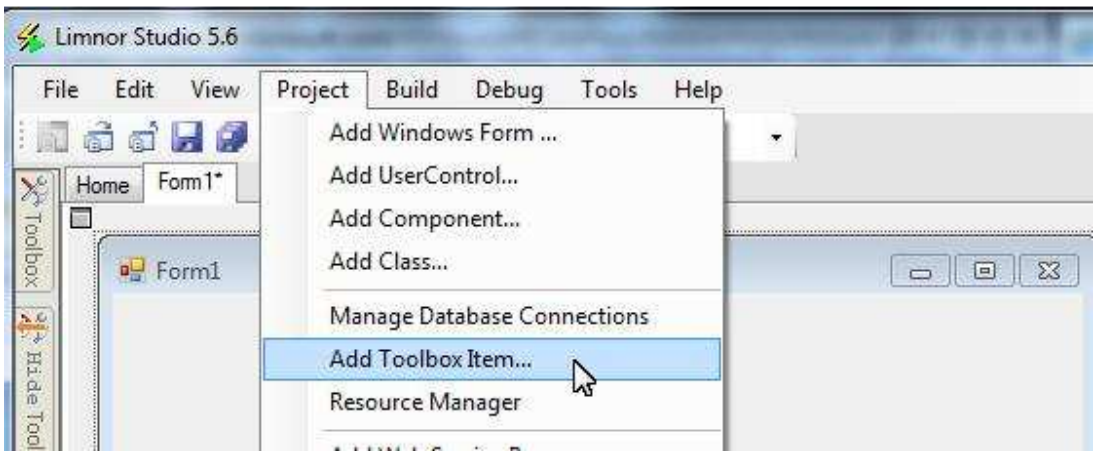
DirectControl is a class provided by the Windows API Code Pack, which can be used to display 3D objects. This sample uses it to display rotating 3D scene.

Open the form into designer:



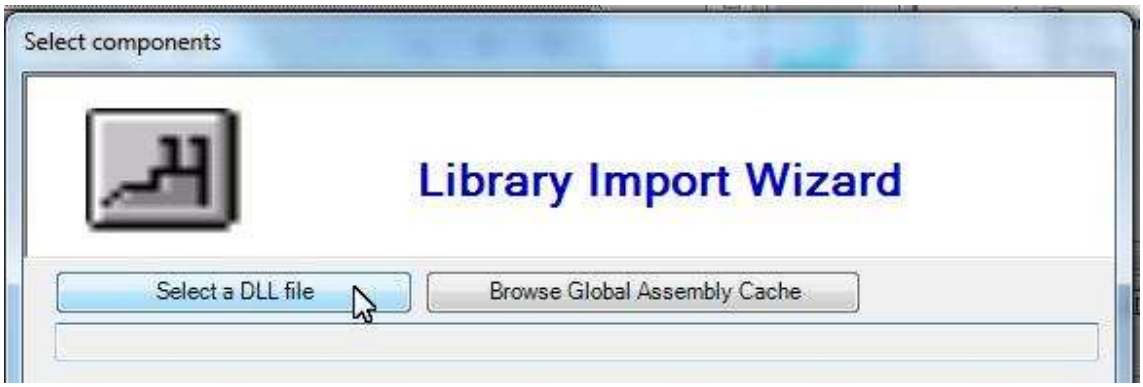


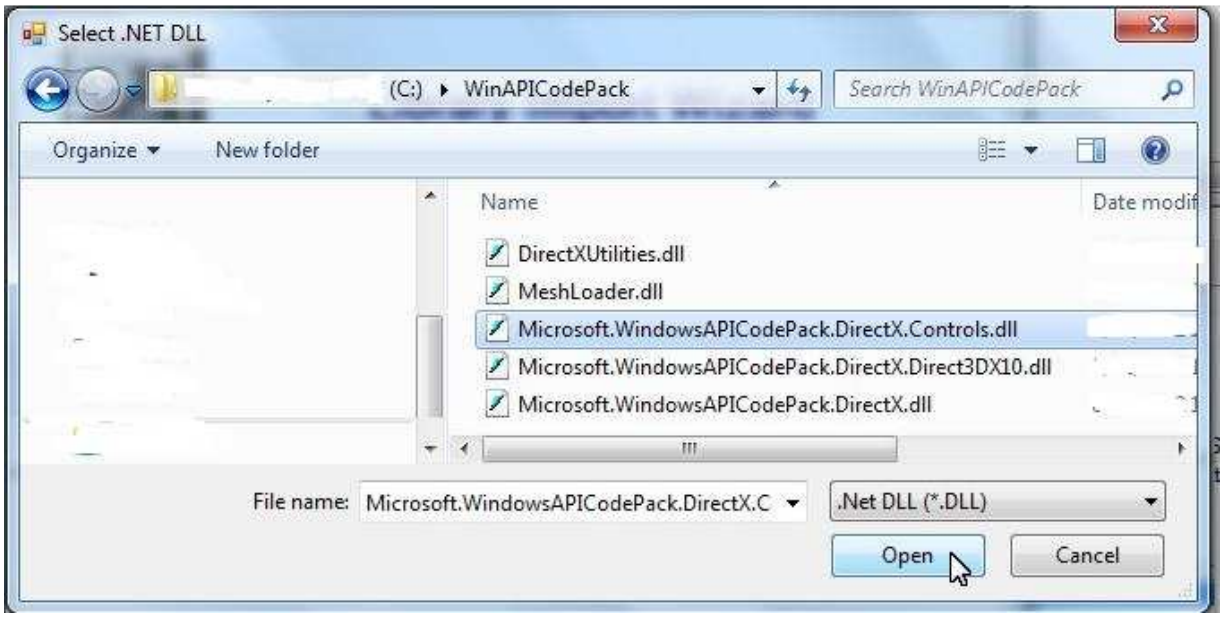
Add DirectControl to the toolbox:

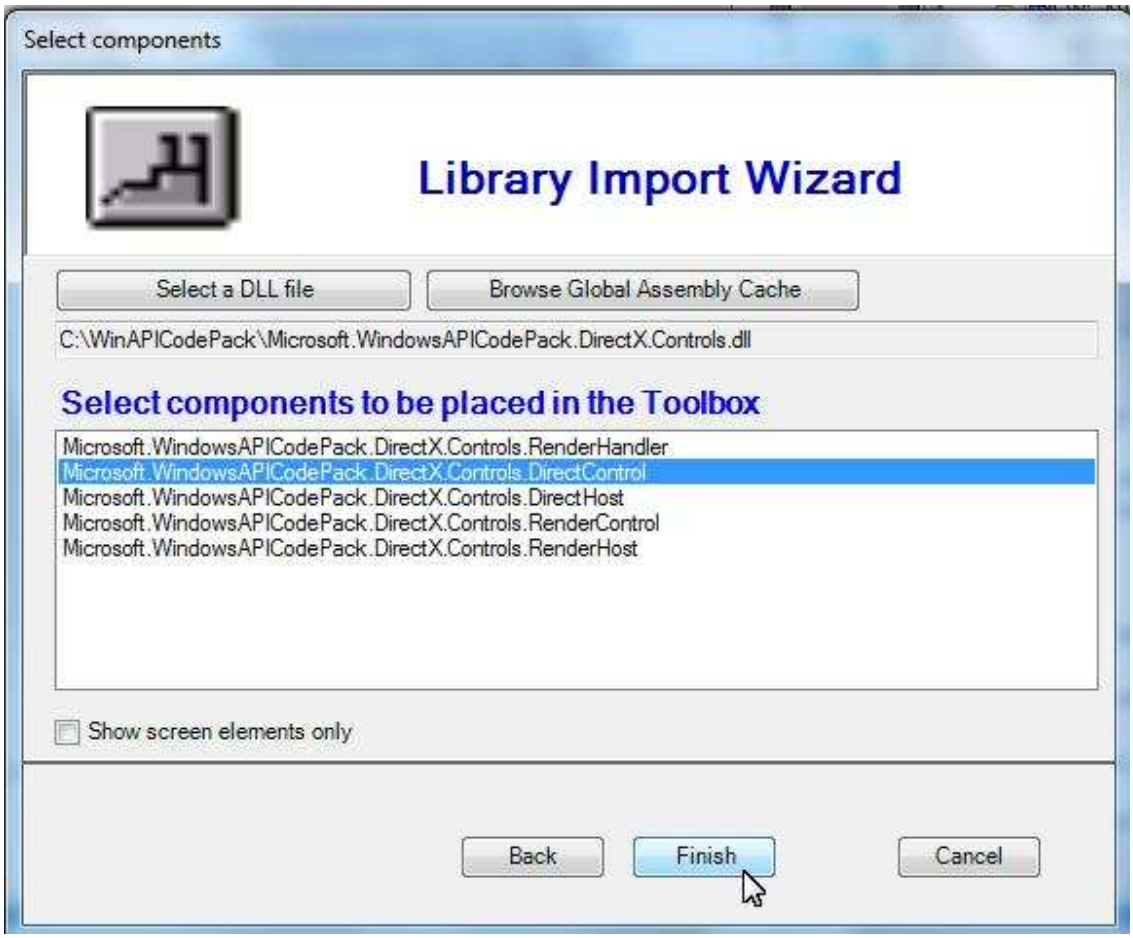



Library Import
Library Importer
This tool imports ActiveX components or Microsoft .NET classes into your projects. The imported components will appear in the Toolbox under the tabs named after your projects.
To import an ActiveX component some new DLL files will be created in your project folder to wrap the ActiveX component to be imported in .Net classes.
Start
Finish

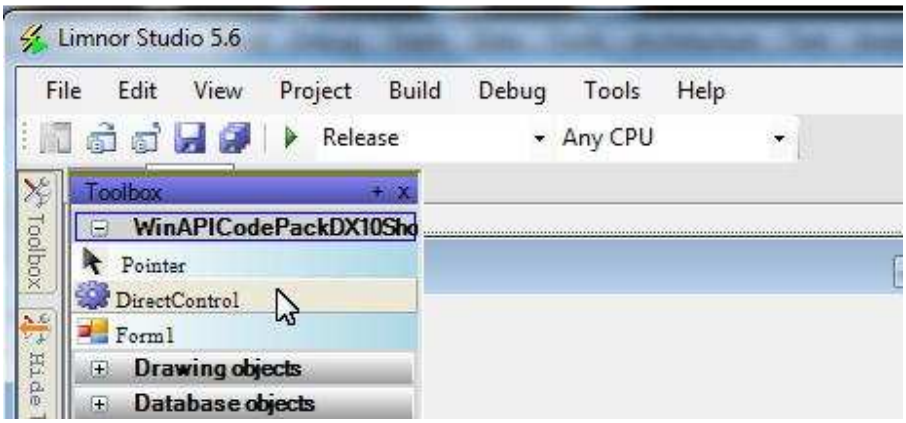

Select library type
Library Import Wizard
Import .NET classes
Import ActiveX components
Back
Next
Cancel


Select components
Library Import Wizard
Select a DLL file
Browse Global Assembly Cache

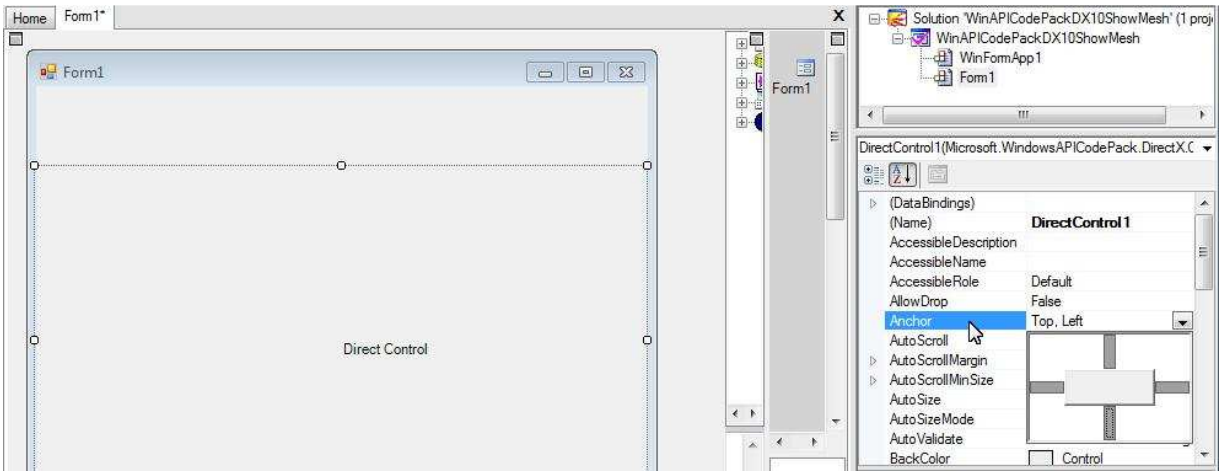




DirectControl appears in the Toolbox. Drop it to the form:

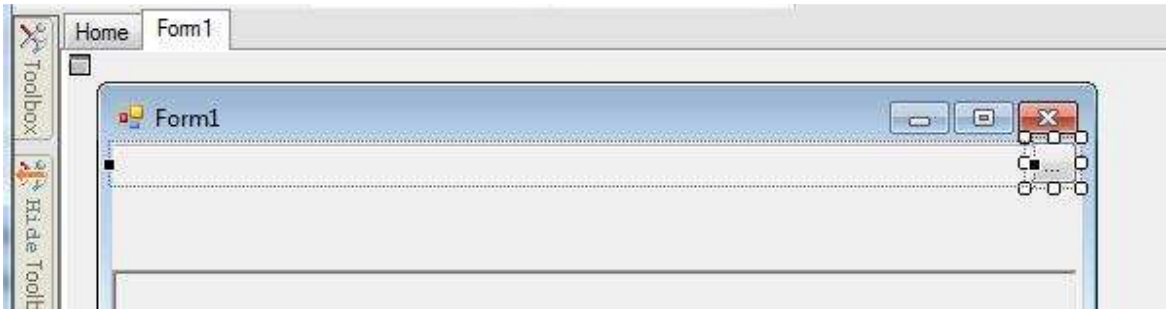


Set properties for the DirectControl to your preferences:

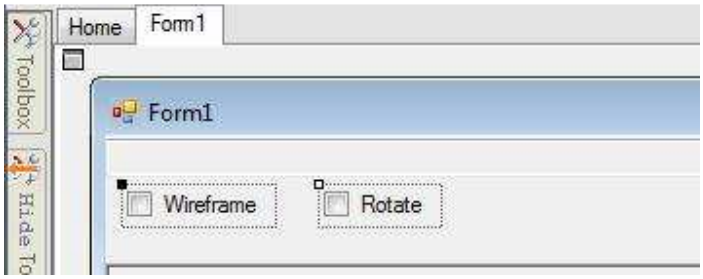


Sample User Interface

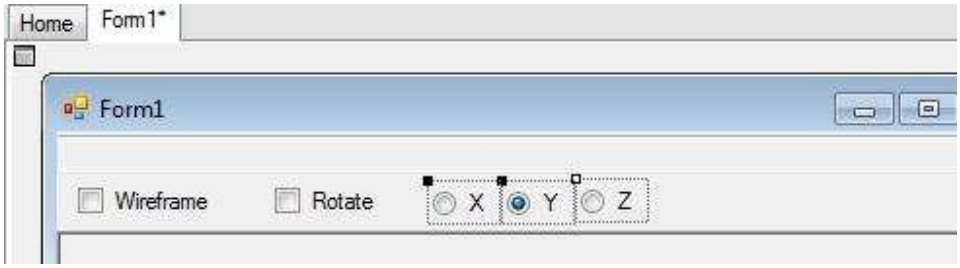
Use a text box to show sample 3D file to be displayed. Use a button to browse and load a 3D file:



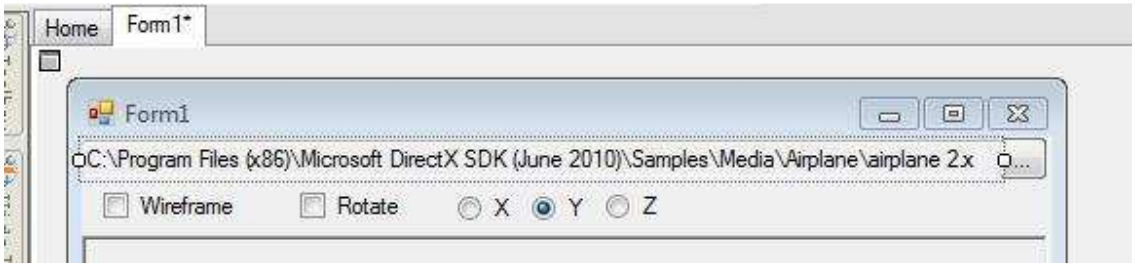
Use a check box to indicate whether wireframe is displayed. Use a check box to indicate whether to rotate the 3D object.



Use a set of radio buttons to indicate rotation axis:



Use the airplane file provided by the DirectX SDK as our default media file:

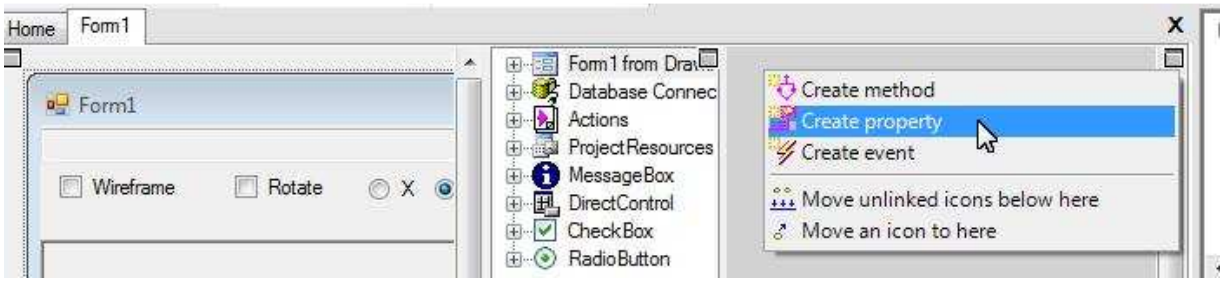


Objects for Making 3D Display

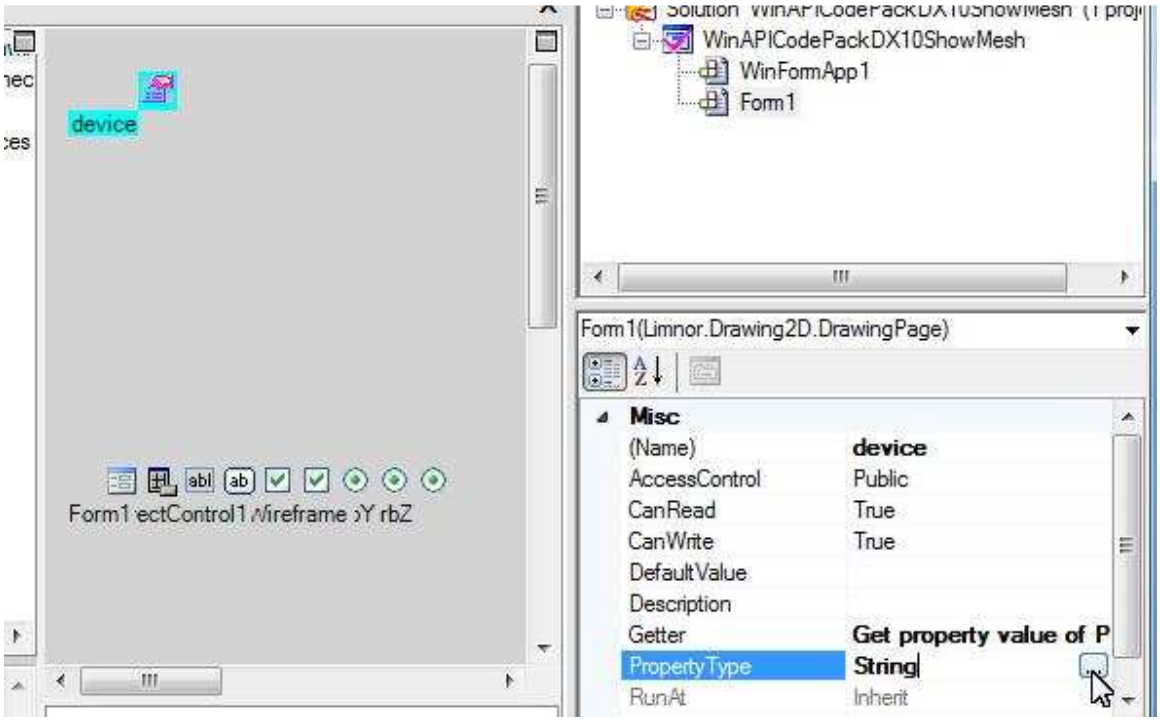
This sample needs following objects from Windows API Code Pack for making 3D display.

D3DDevice

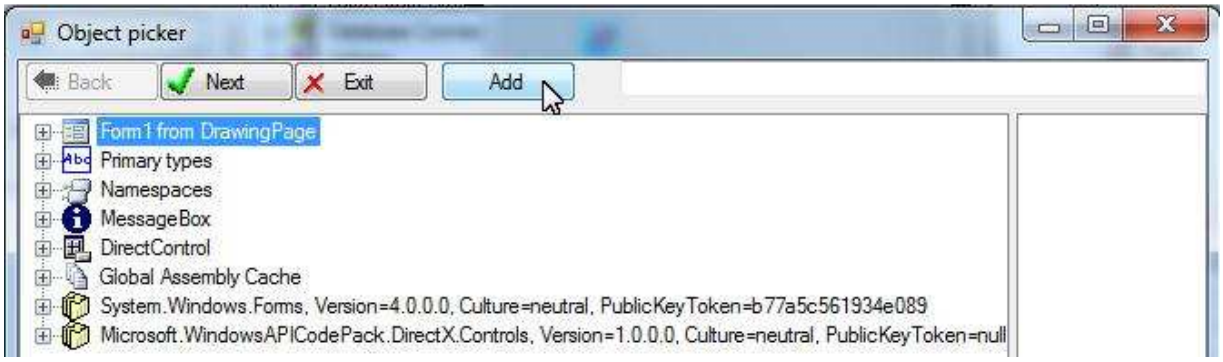
The most basic object for 3D is D3DDevice. Add a D3DDevice property to the form.

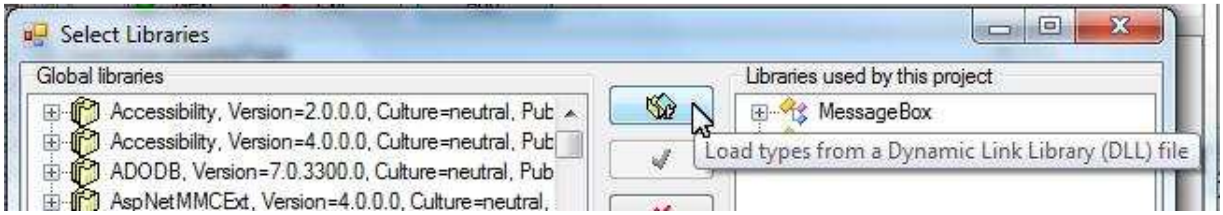


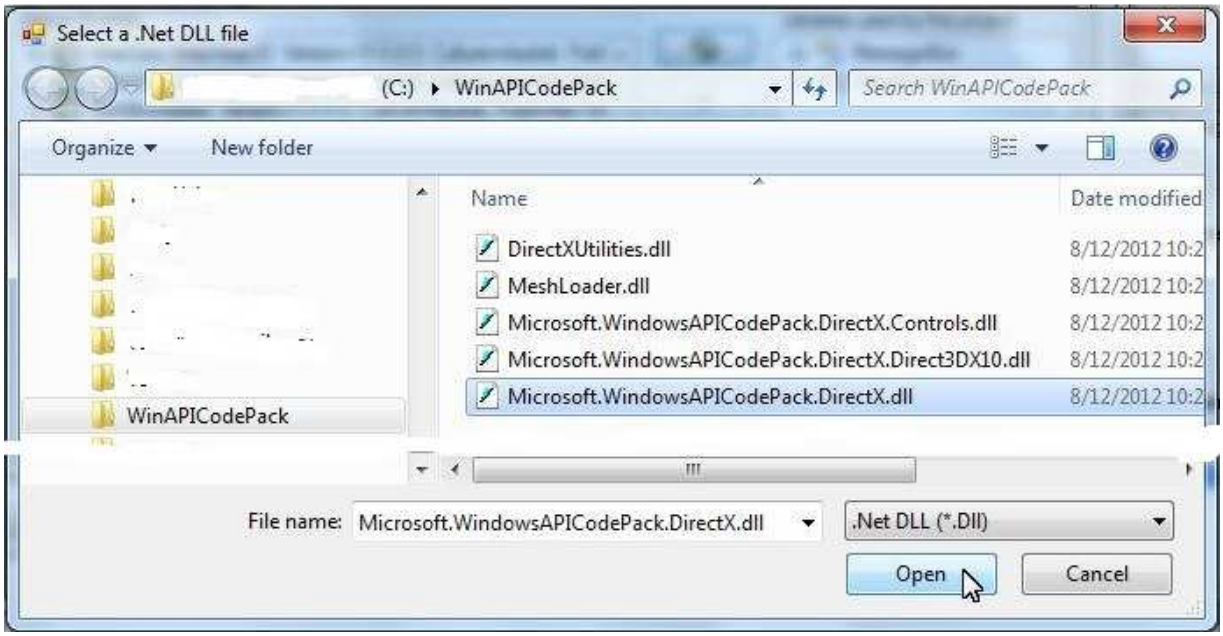
Rename the property “device” and set its property type to D3DDevice:



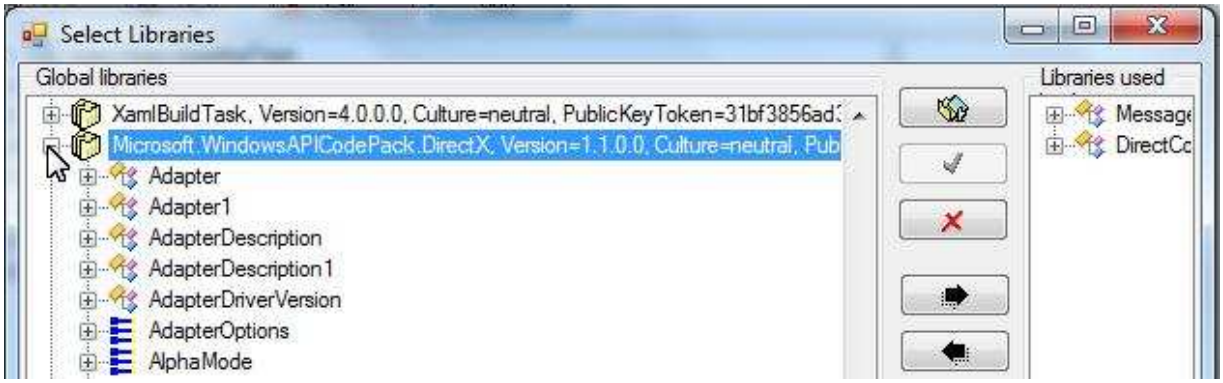
D3DDevice is in Microsoft.WindowsAPICodePack.DirectX.dll. Click Add to add D3DDevice to the class list:



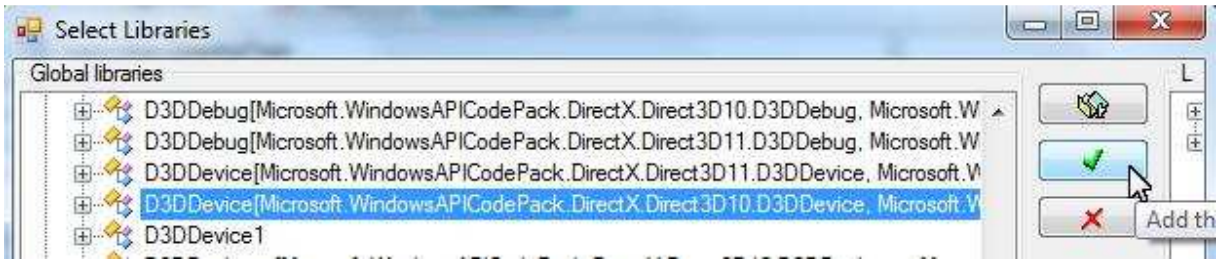


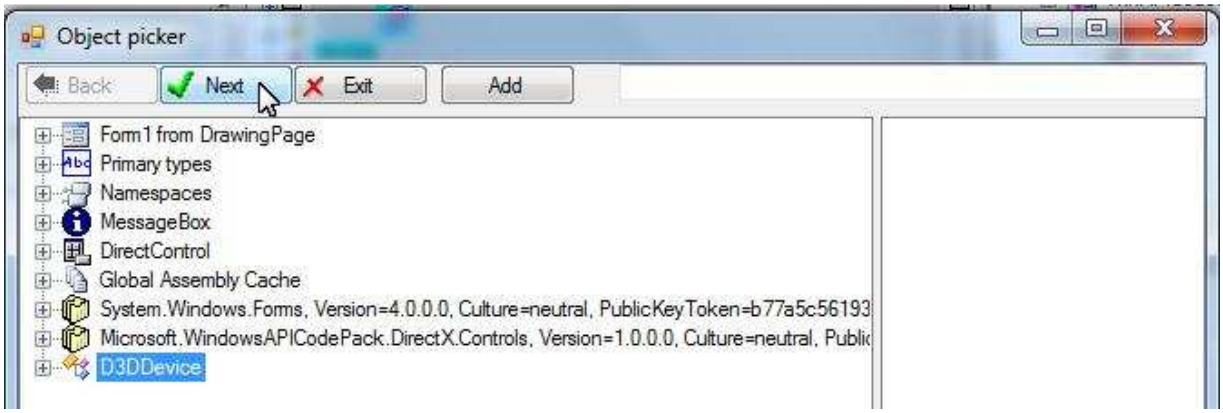


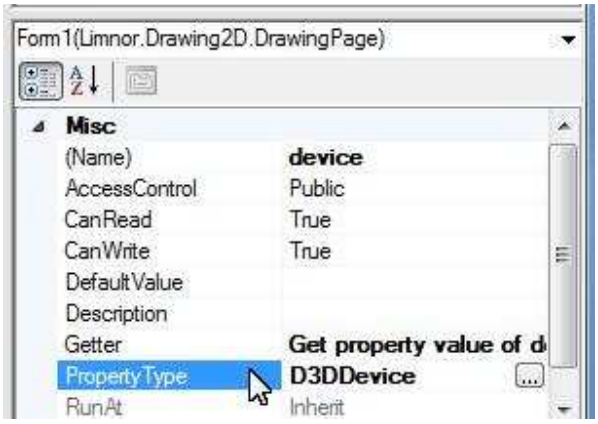
Find D3DDevice in the file:



This sample only uses DirectX10. Choose D3DDevice from DirectX3D10:

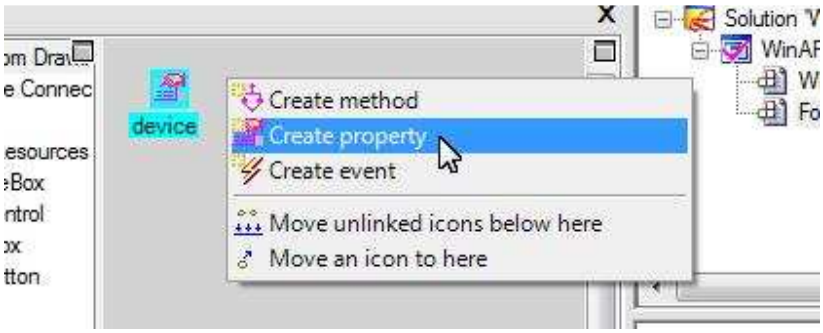




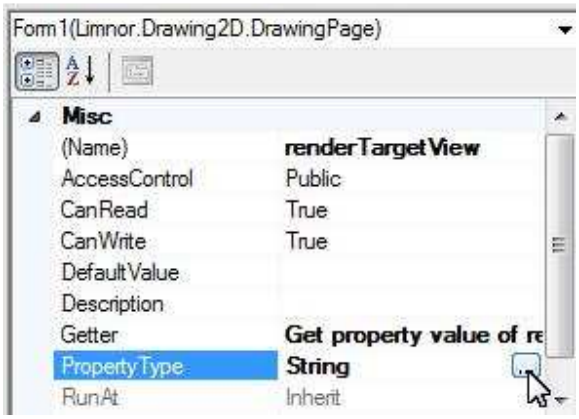


RenderTargetView

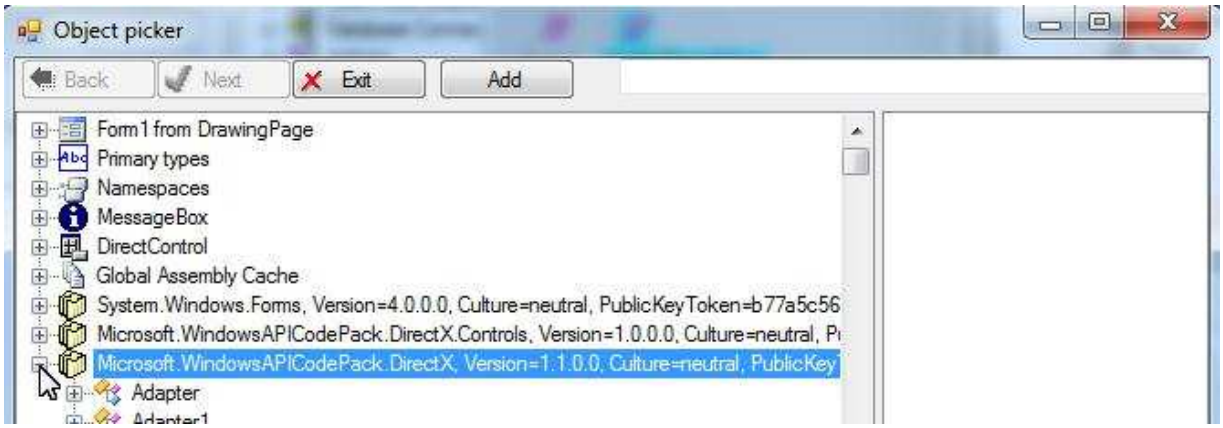
Class RenderTargetView represents render-target-view. Add a property of RenderTargetView to the form:



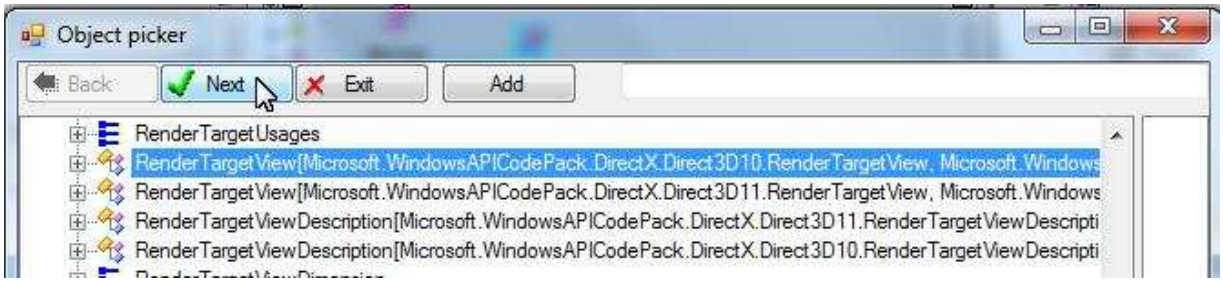
Name the property "`renderTargetView`". Set its property type to RenderTargetView.



Find class RenderTargetView in Microsoft.WindowsAPICodePack.DirectX.dll:



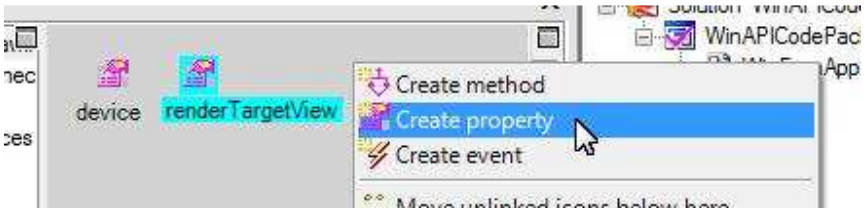
Choose RenderTargetView from Direct3D10:



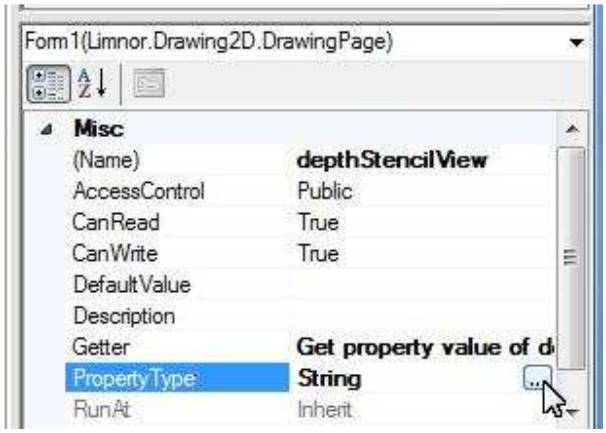
DepthStencilView

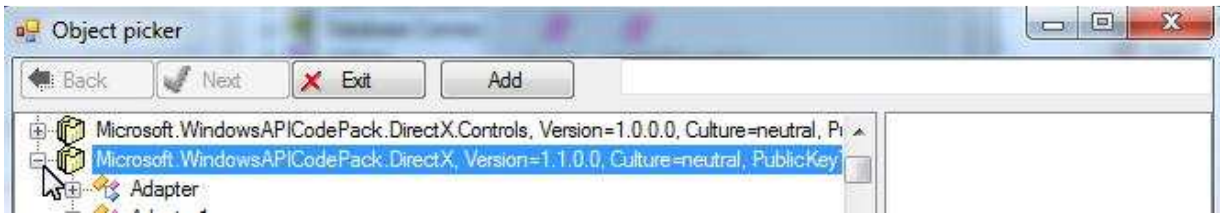
Class DepthStencilView allows the device to use depth and stencil resources which contains state that controls how depth and stencil data impacts rendering.

Add a DepthStencilView property to the form:

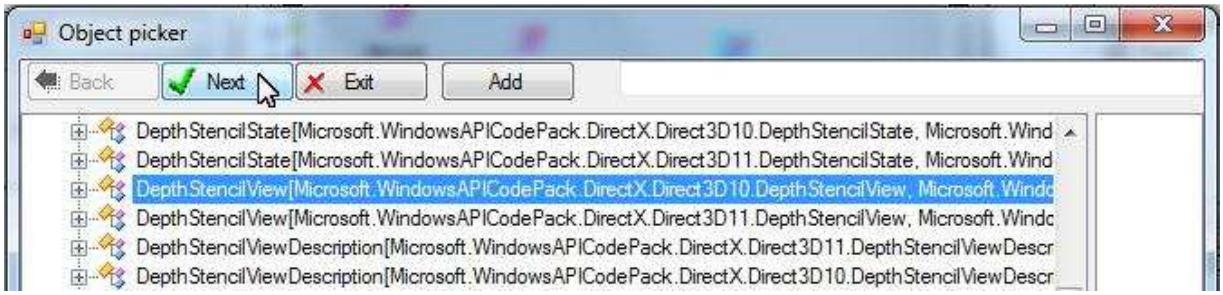


Name the property "`depthStencilView`". Find class DepthStencilView in Microsoft.WindowsAPICodePack.DirectX.dll:





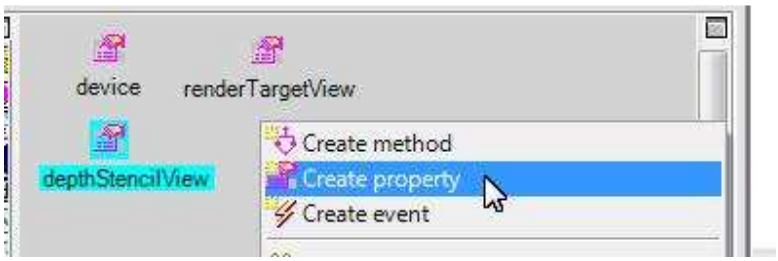
Choose DepthStencilView from Direct3D10:



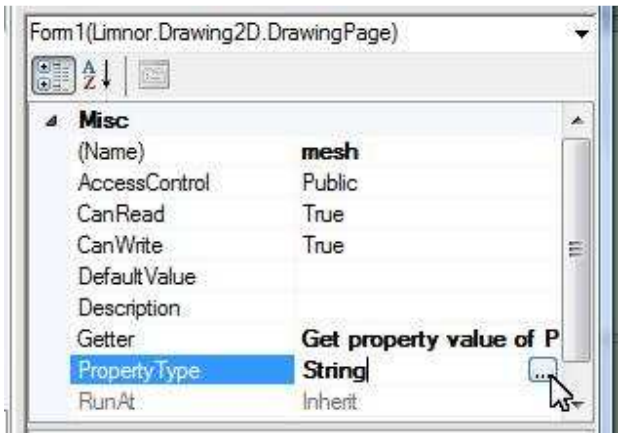
XMesh

A mesh is a single vertex buffer and index buffer, containing all the vertices for a specific renderable instance. A mesh file can be loaded into an XMesh object.

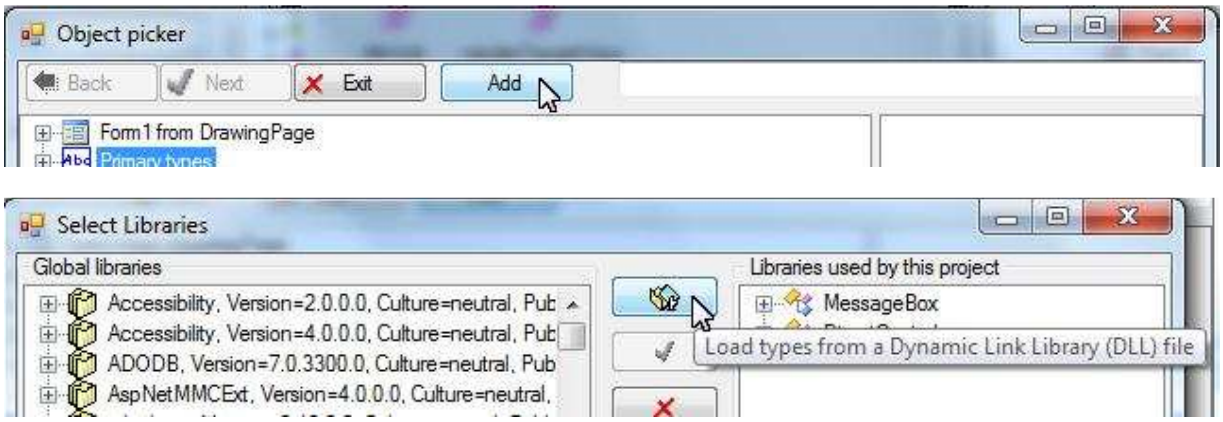
Add an XMesh property to the form to represent what we are going to display.

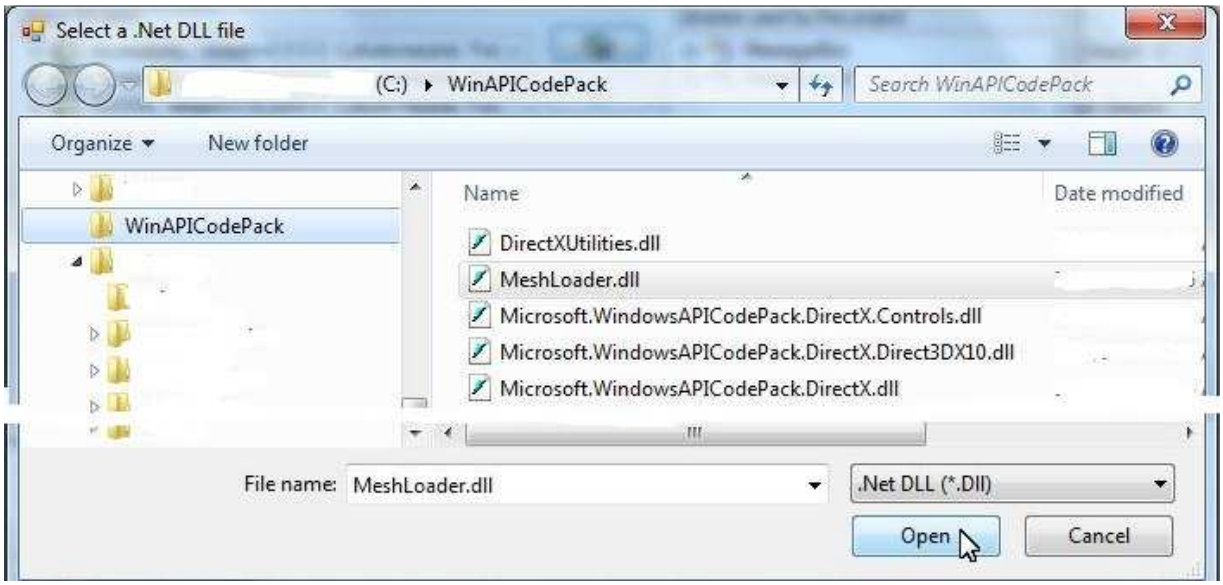


Name it “mesh” and change its property type to XMesh:

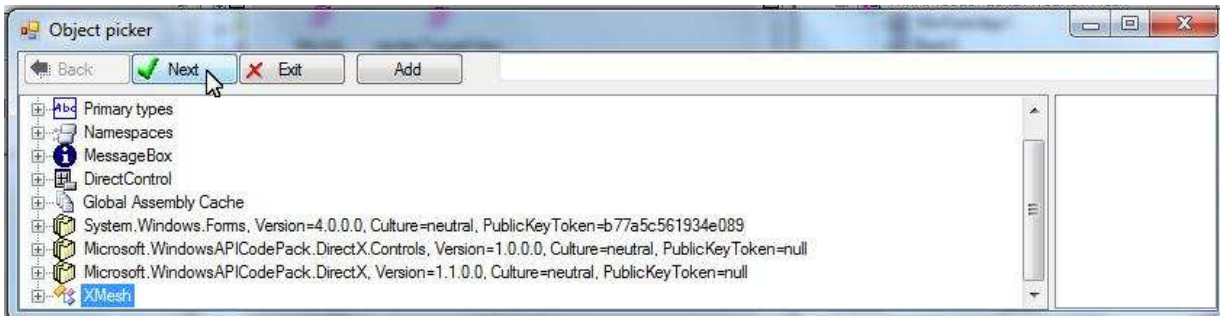


Class XMesh is in MeshLoader.DLL. Click “Add” to load the DLL:



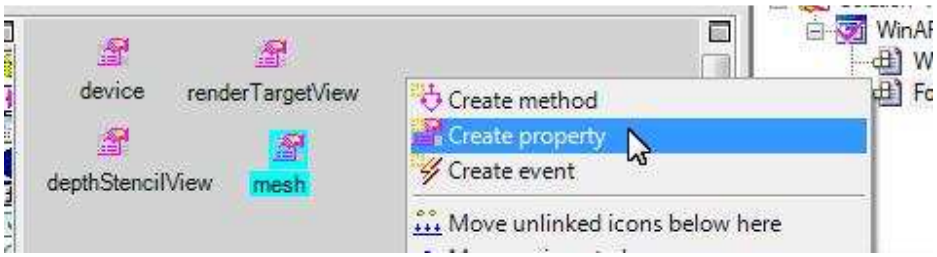


Select XMesh:

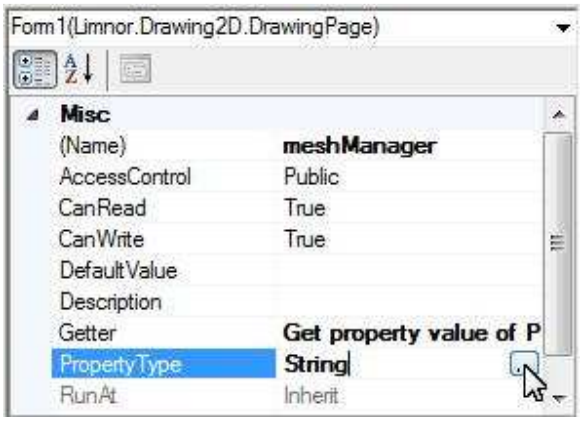


XMeshManager

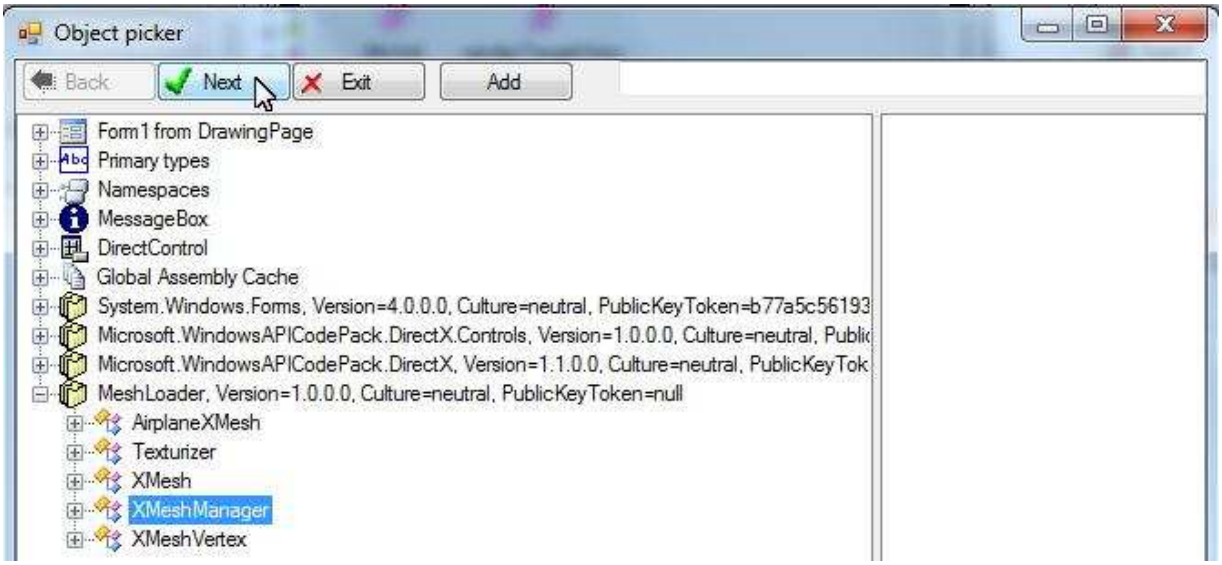
Class XMeshManager is for managing meshes. Add an XMeshManager property to the form.



Name it "meshManager". Set its property type to XMeshManager:



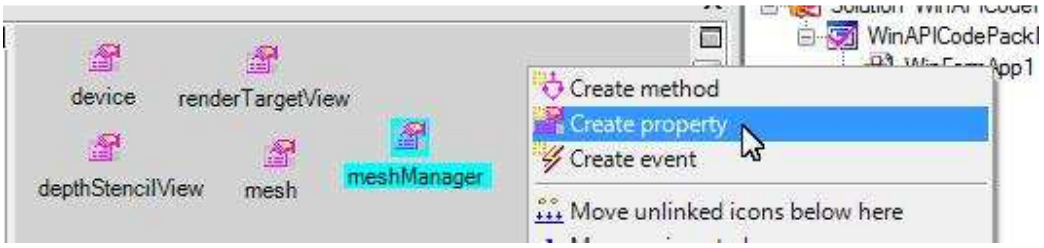
Select XMeshManager:



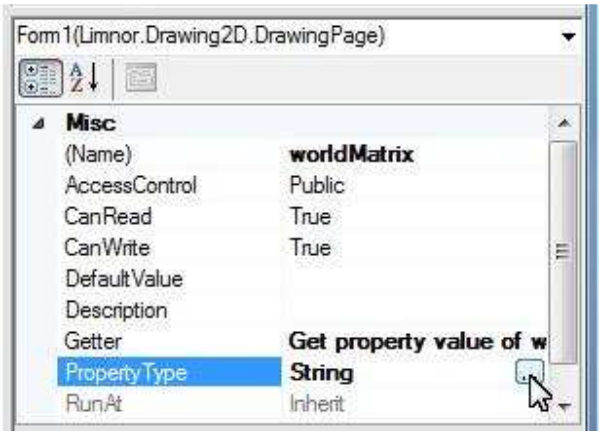
World Viewing Angle

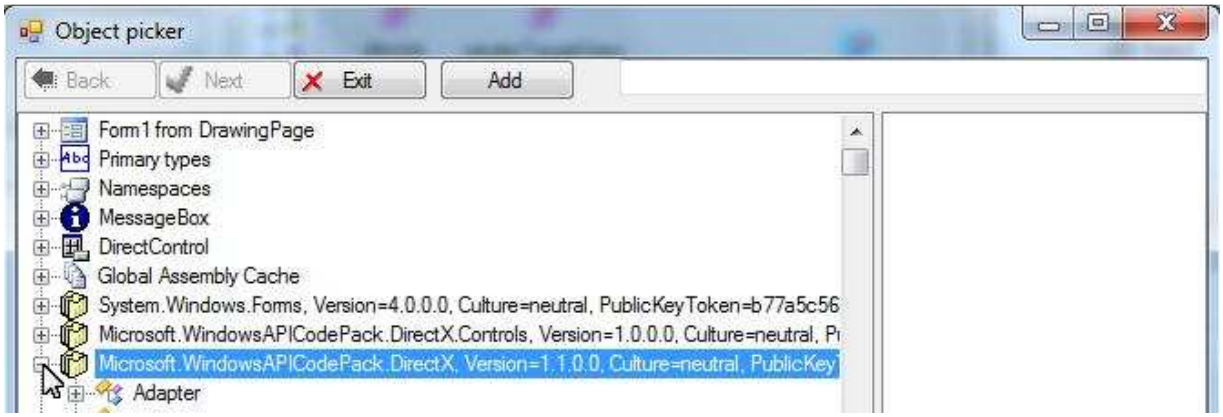
World viewing angle is represented by a 4 by 4 matrix, represented by a Matrix4x4F object.

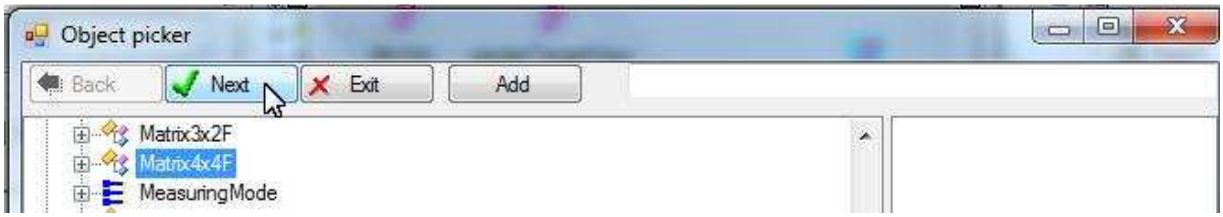
Add a Matrix4x4F property to the form.



Name it "worldMatrix" and change its property type to Matrix4x4F:

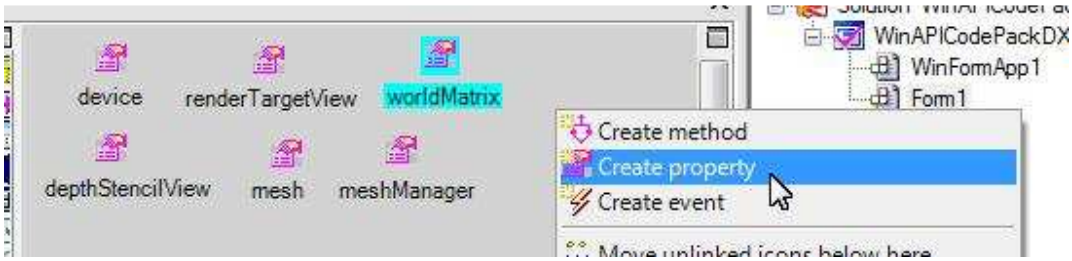




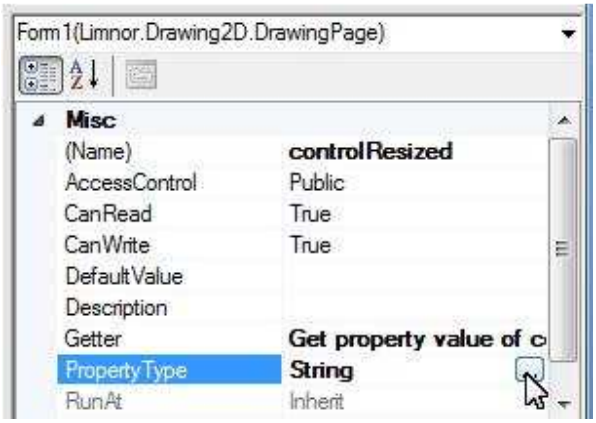


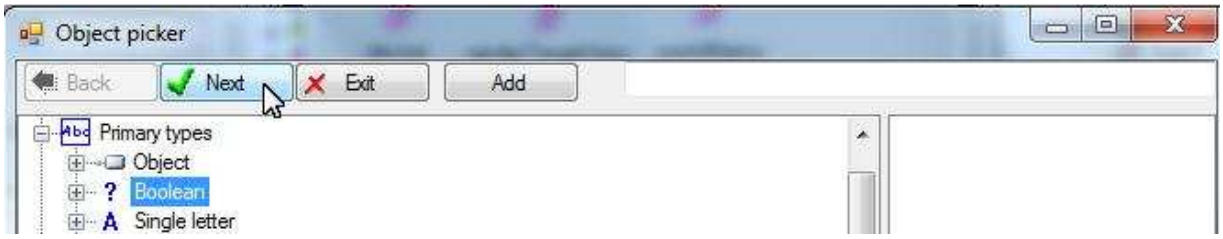
Monitor Control Size Change

We allow form resizing by the user and automatically resizing the DirectControl by setting its Anchor property. On changing the size of the DirectControl we need to reset the 3D device. We add a Boolean property to monitor the control size change:

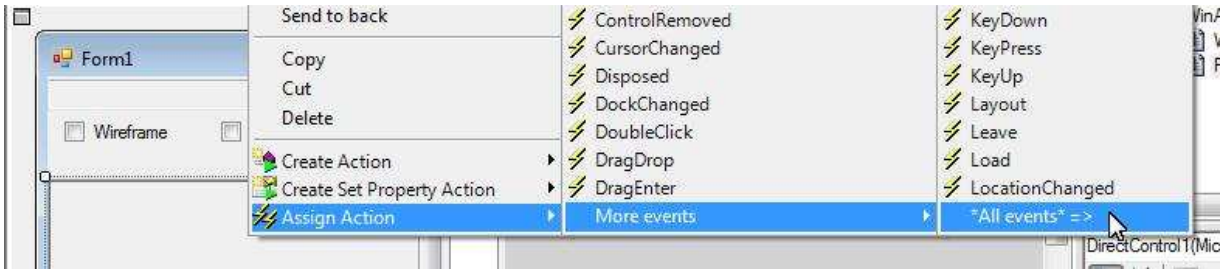


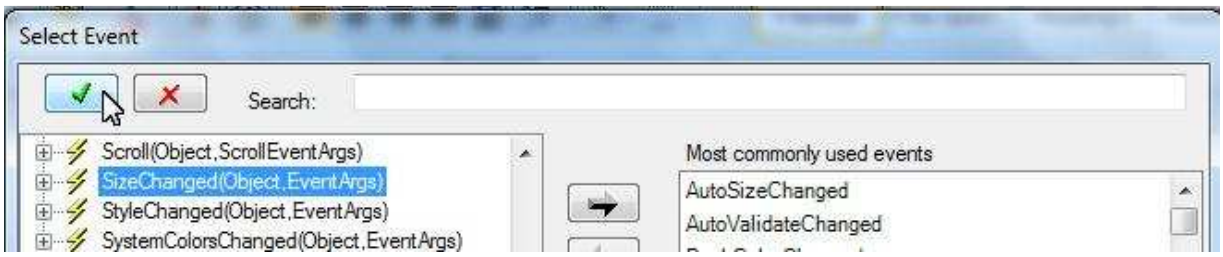
Name it “controlResized”. Set its property type to Boolean:

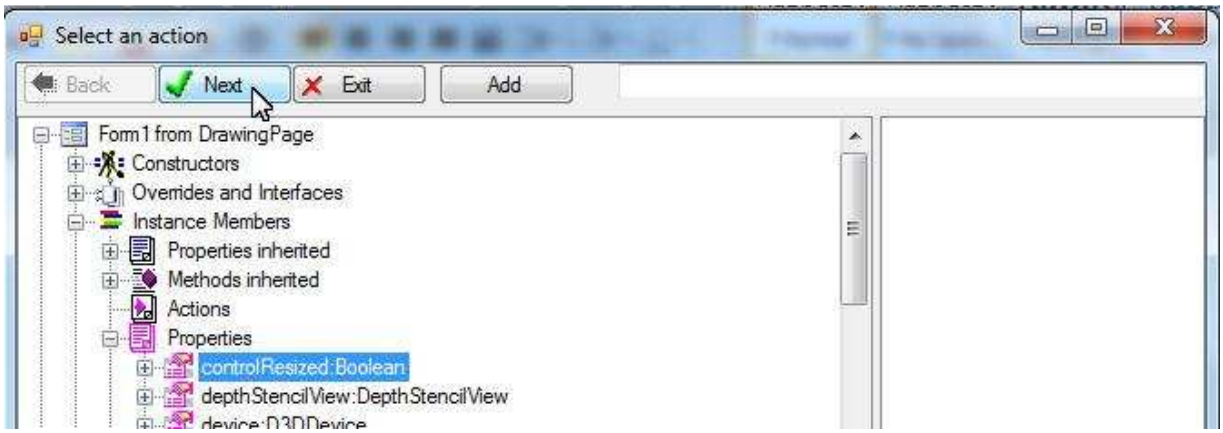


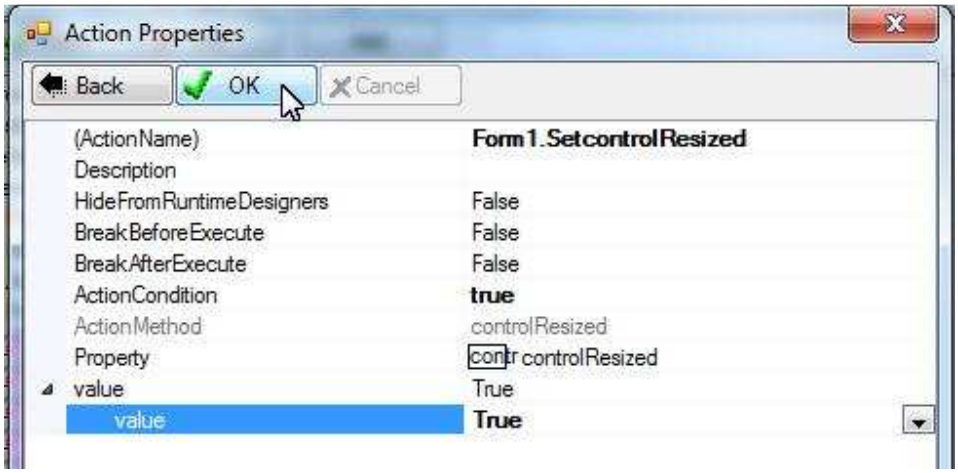


Handle SizeChanged event of the DirectControl to set controlResized to True:



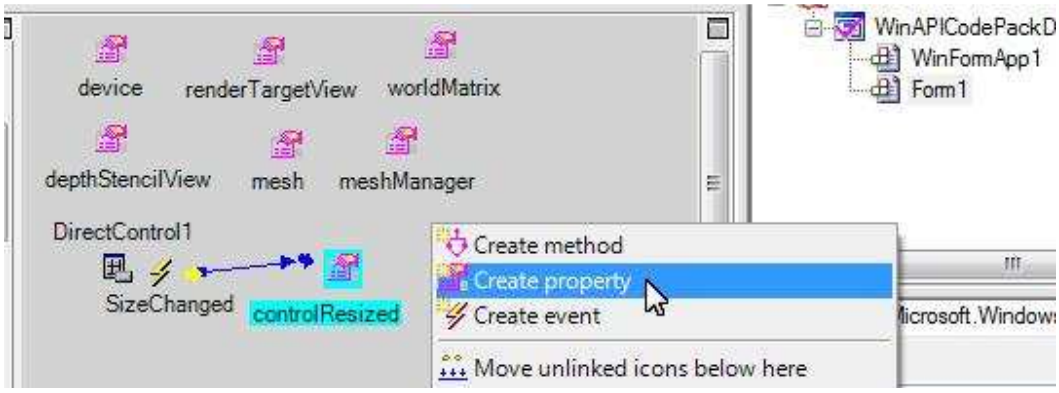




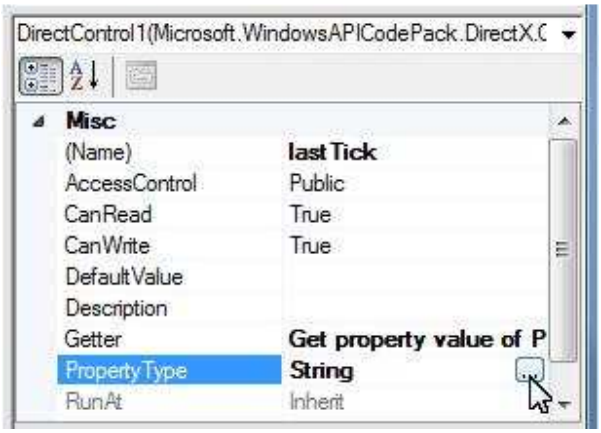


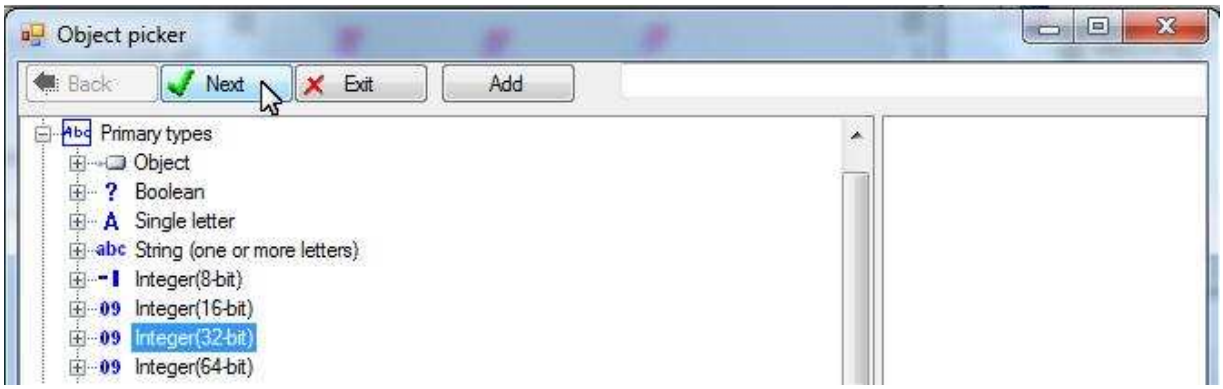
Calculate Rotation Angle by Time Interval

DirectControl will paint again and again. On every time of painting we increase viewing angle, the result will be a rotating scene. We may use the time interval between consecutive paintings to calculate the viewing angle. To get the time interval between consecutive paintings we need to remember the last painting time. We use an integer property to remember the last painting time.



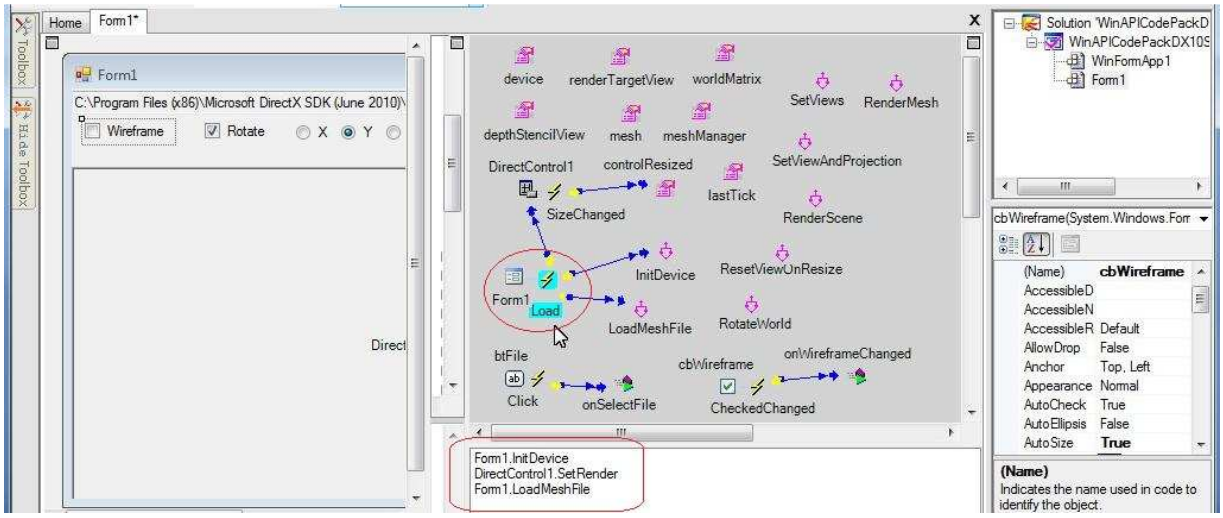
Name it “lastTick” and set its property type to integer:





Initializations

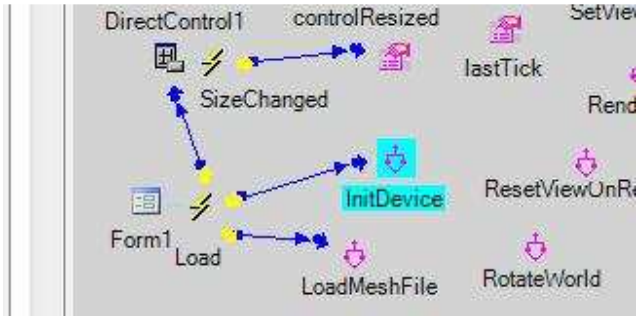
For this sample, we do initializations by three actions in Load event of the form.

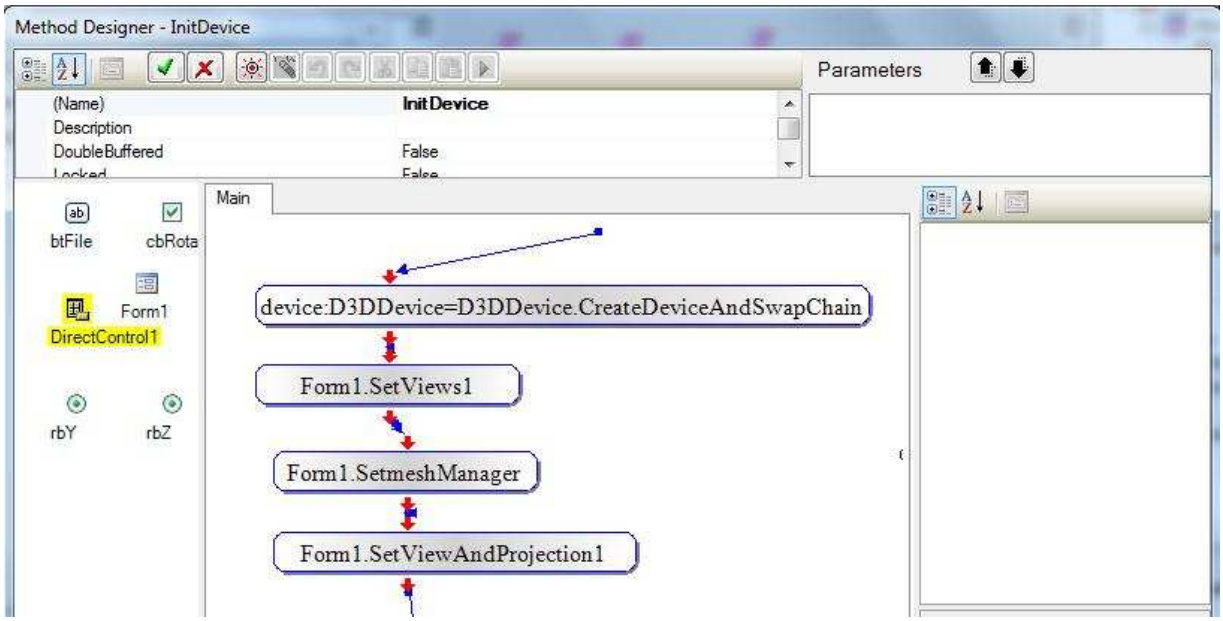


The first action initializes 3D device. The second action tells DirectControl1 how to draw 3D scene. The last action load mesh file.

InitDevice

InitDevice is a method created for initialize the device:





The first action creates 3D device object using the handle of DirectControl1.

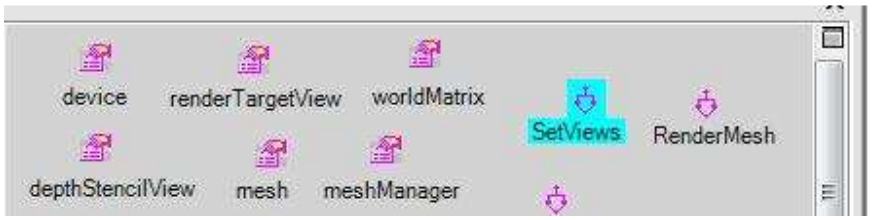
The second action creates views.

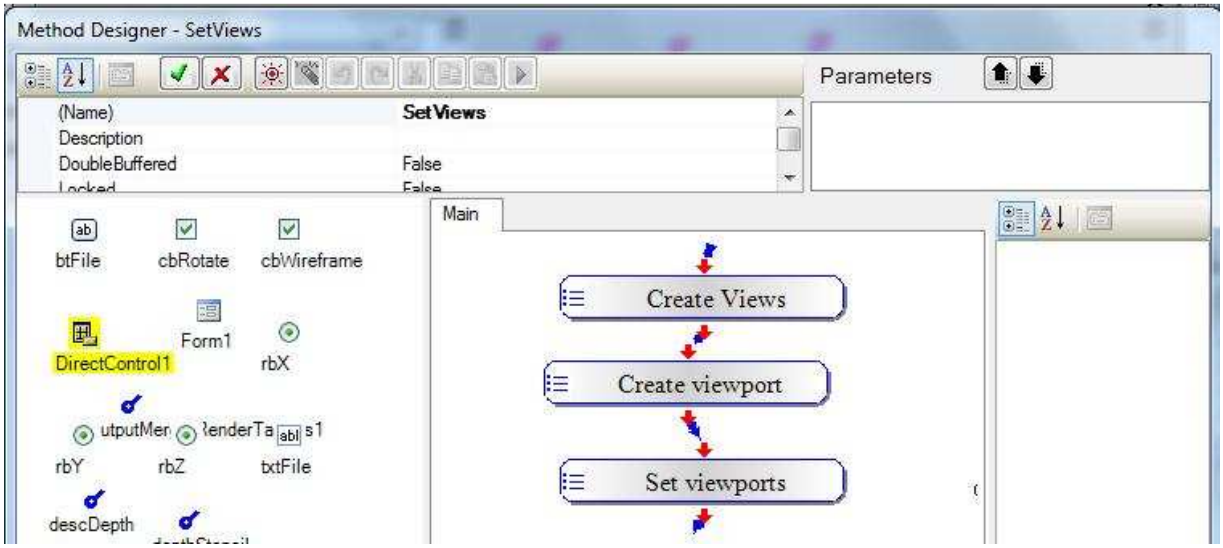
The third action creates mesh manager object.

The last action prepares view and projection of the 3D world.

Set Views

SetViews is a method for creating views for the device.

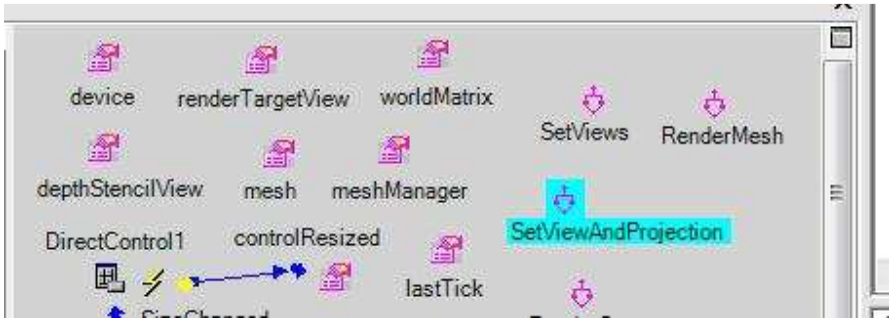


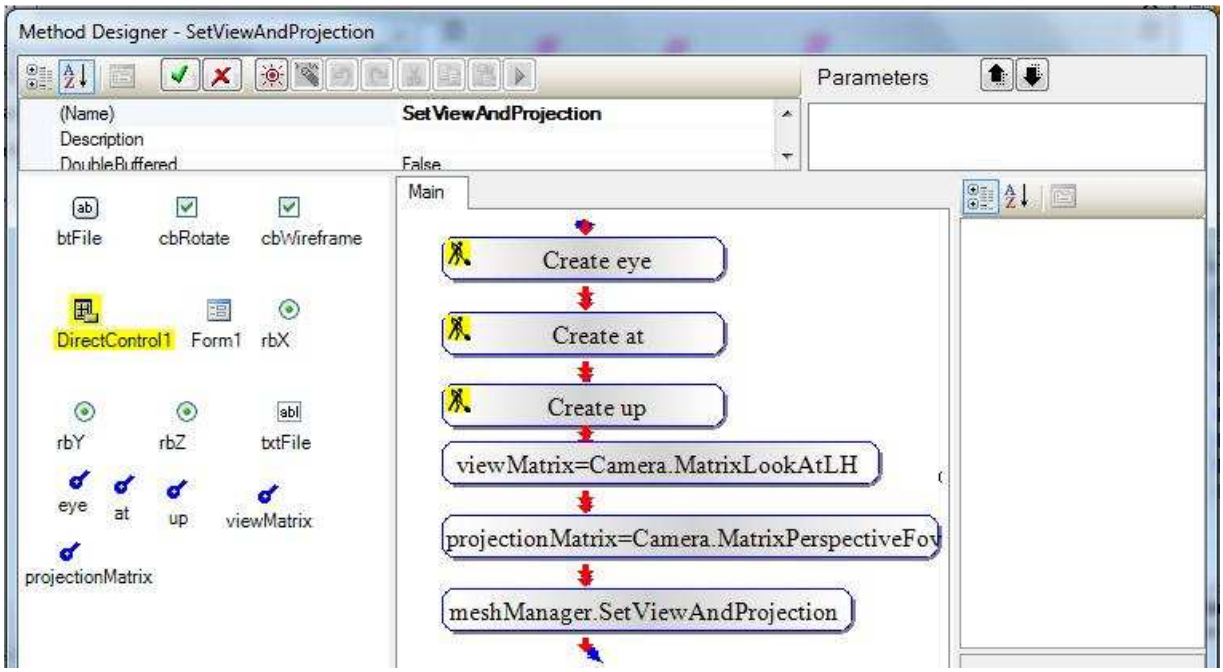


The first action-list creates a render target view and a depth-stencil view. The second action-list creates a view port using the views. The last action-list sets the view port to the 3D device.

Set View and Projection

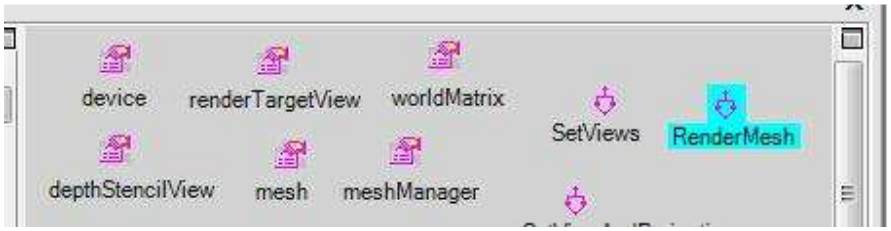
SetViewAndProject is a method for setting view and projection of the 3D world.

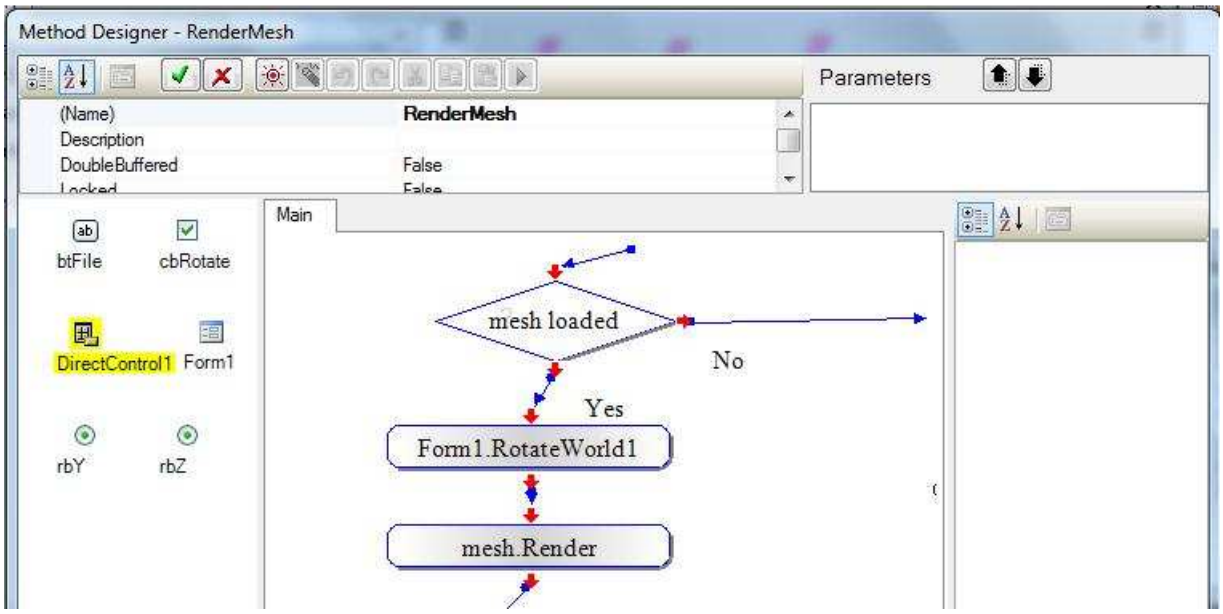




Render Mesh

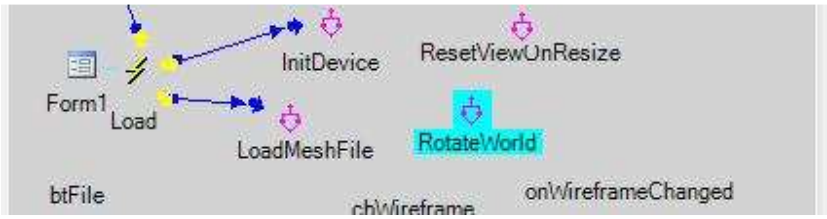
RenderMesh is a method for rendering the mesh file on the 3D scene:

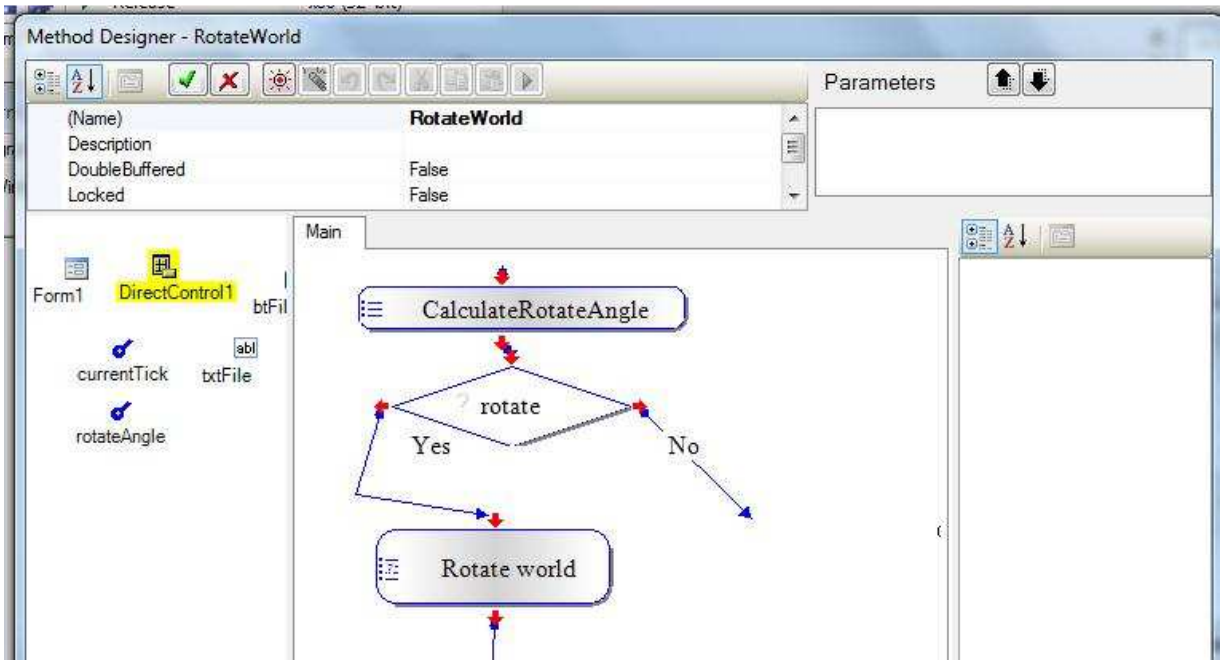




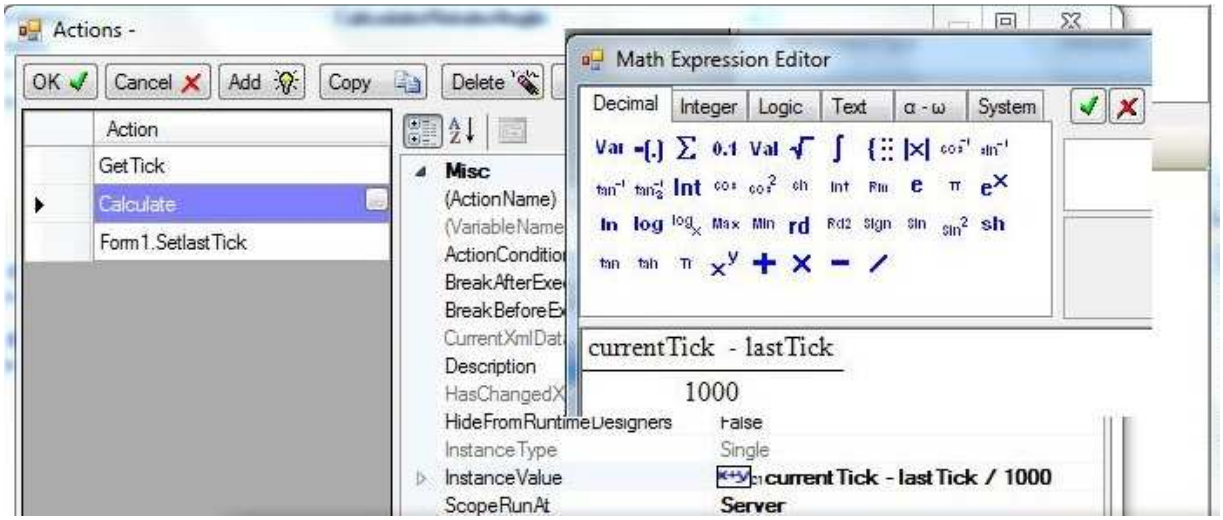
Rotate the world

RotateWorld is a method for rotating the world:





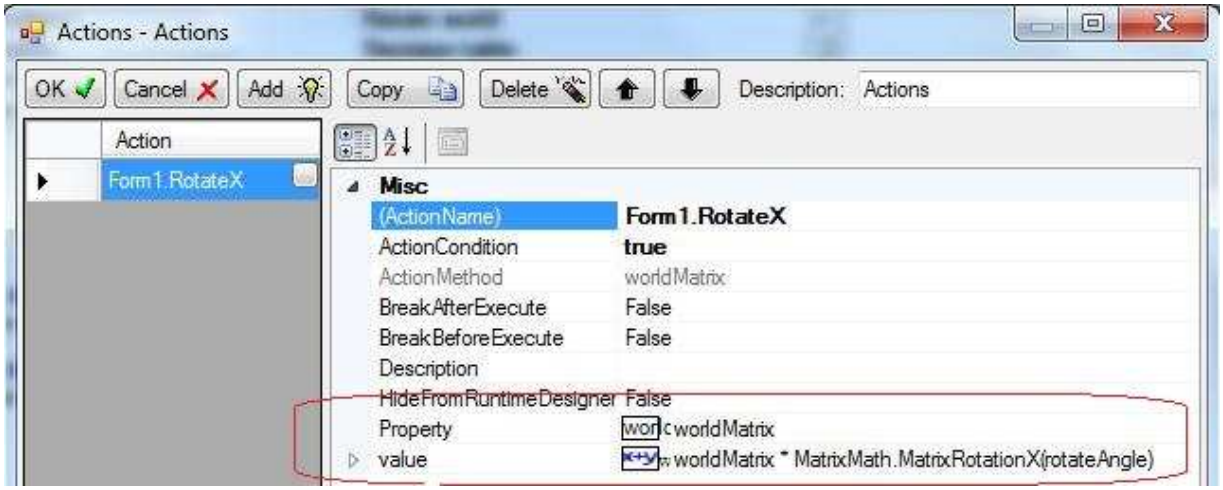
The first action-list calculates the rotation angle by time interval between the consecutive paintings:



“Rotate world” is a decision table which rotates the world on an axis based on the states of radio buttons:

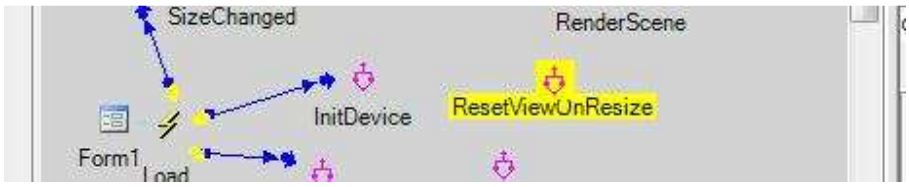
Each “Actions” contains one action to rotate on one axis.

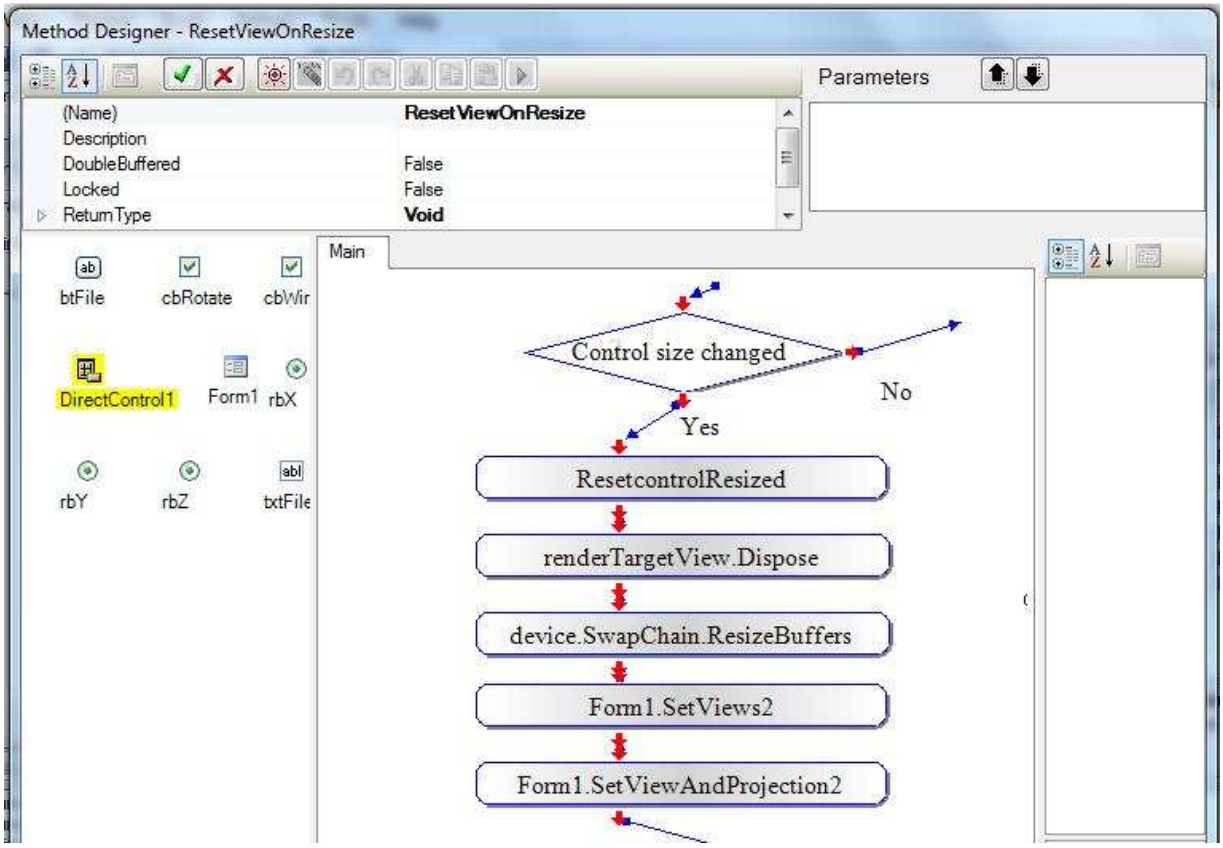
For example, “Actions” in the first row contains one action, Form1.RotateX, which rotates the world on x-axis:



Reset Views

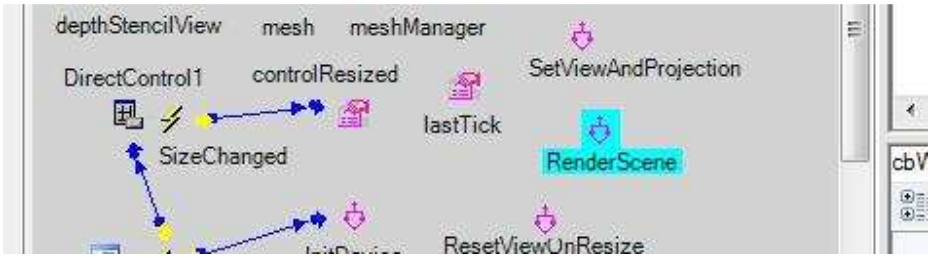
ResetViewOnResize is a method for resetting views if the size of DirectControl1 is changed:

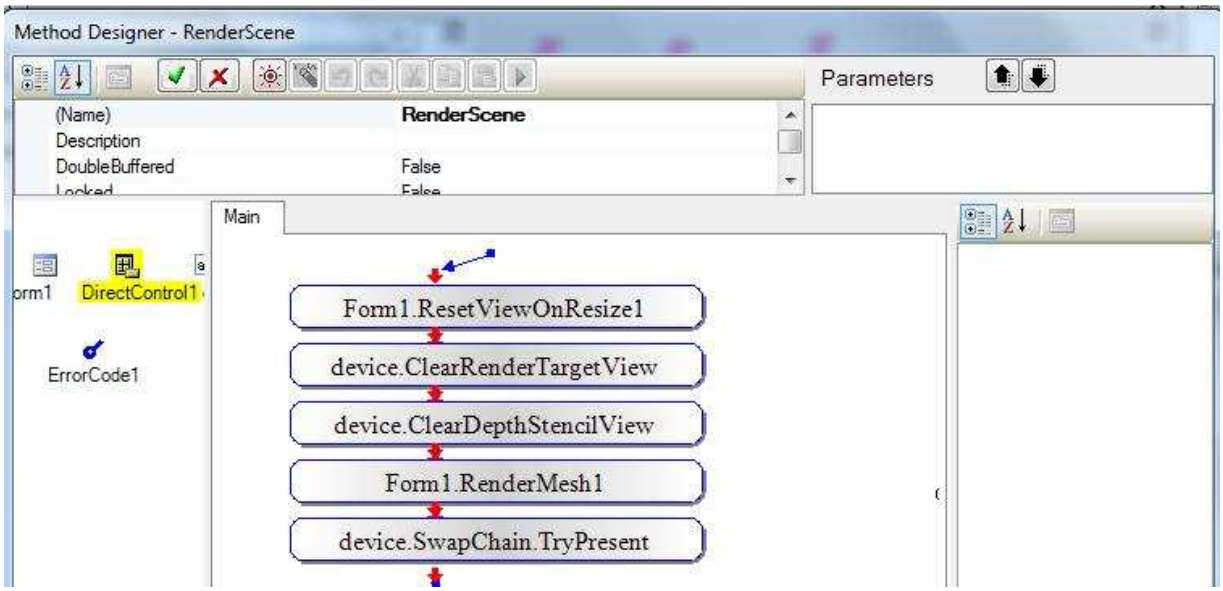




Render Scene

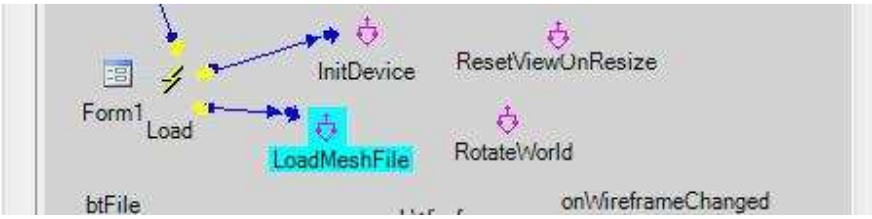
RenderScene is a method for rendering 3D scene. This method is assigned to DirectControl1 at Load event. DirectControl1 uses this method to draw 3D scene.

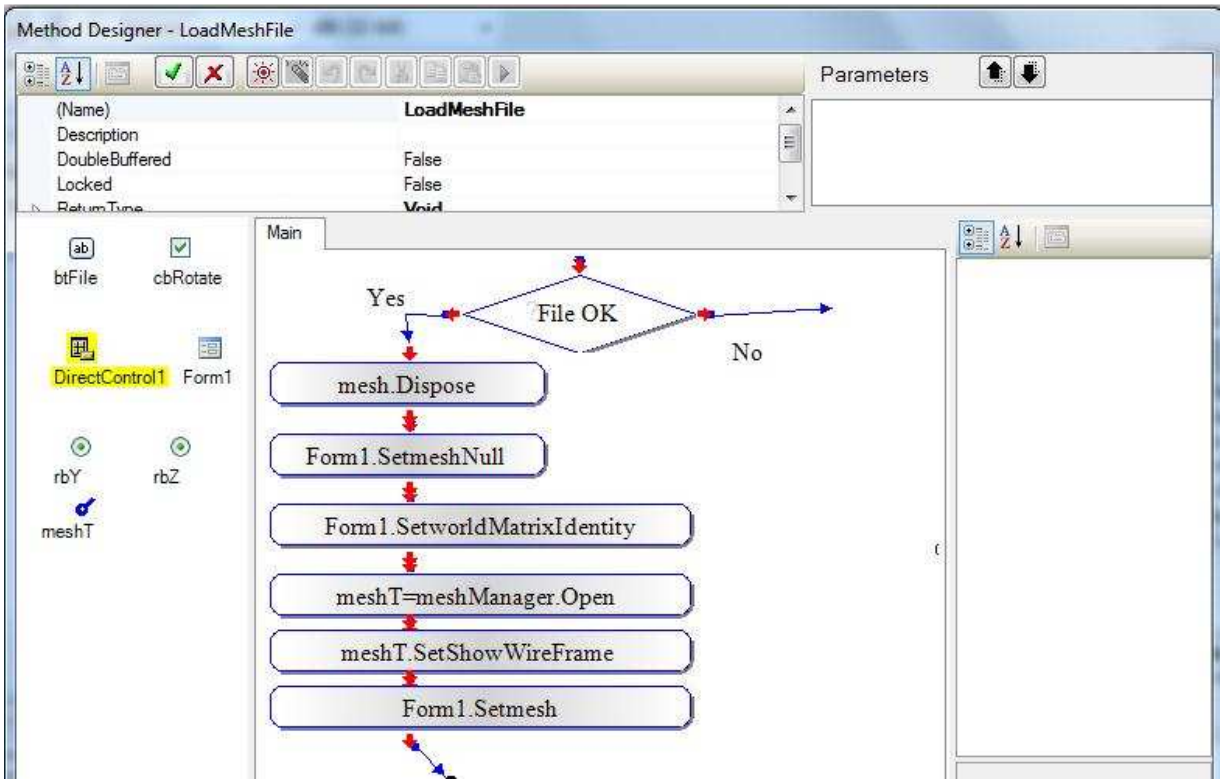




Load Mesh File

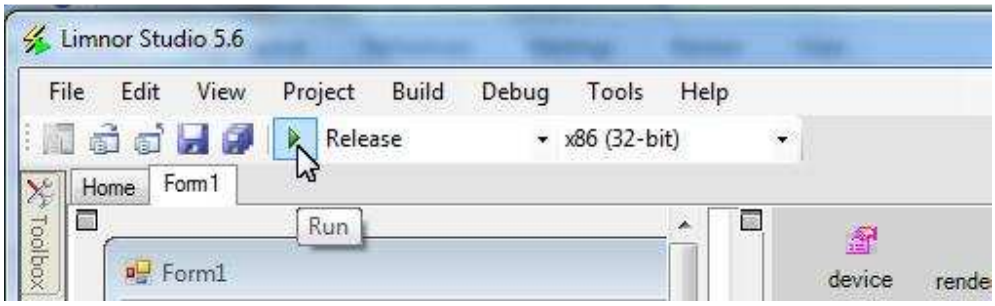
LoadMeshFile is a method for loading a mesh file.



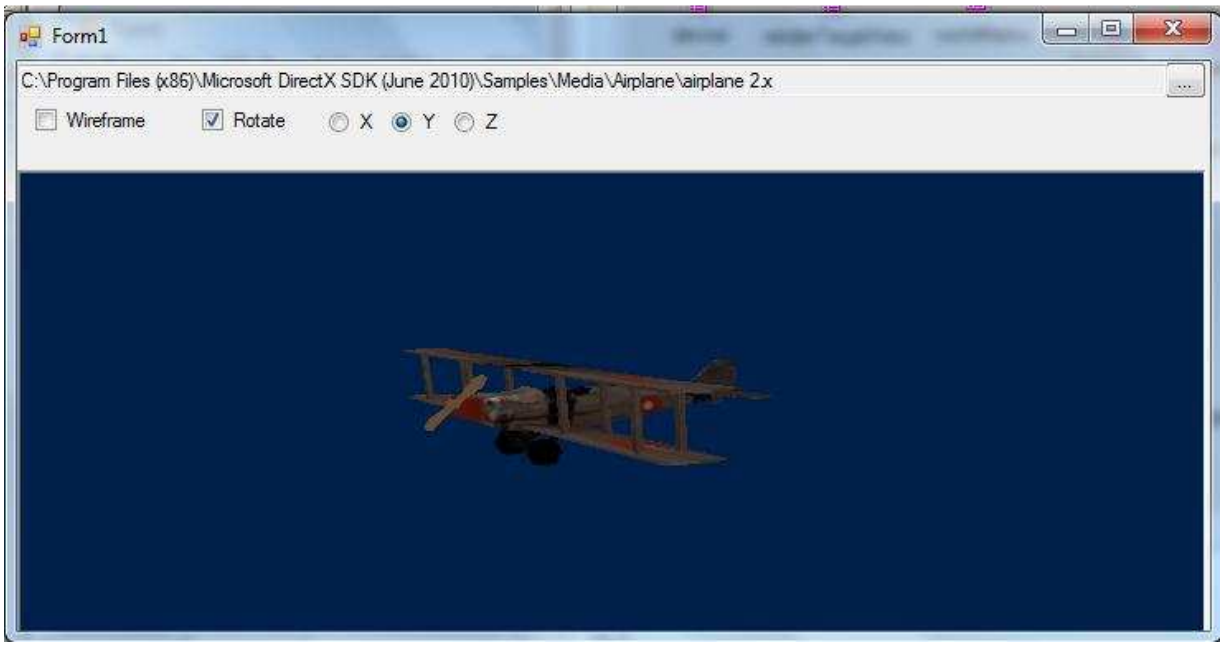


Test

We may click Run button to compile and run the application.



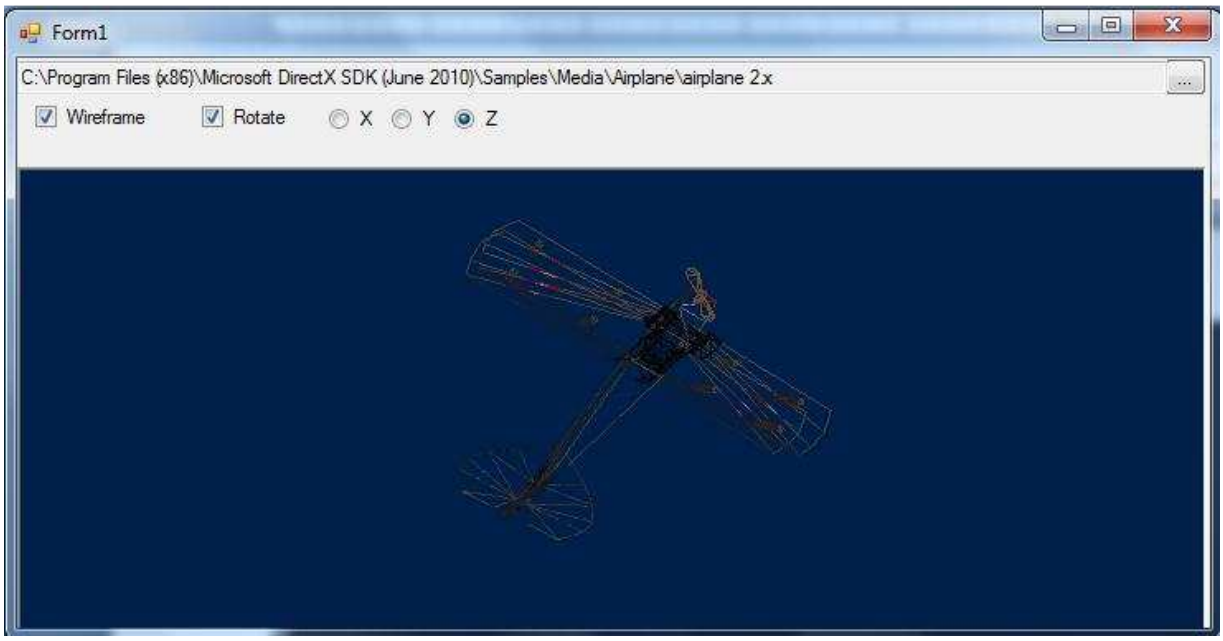
The form appears. File “airplane 2.x” is loaded and a 3D airplane is displayed on DirectControl. By the default settings the airplane is rotating on Y-axis:



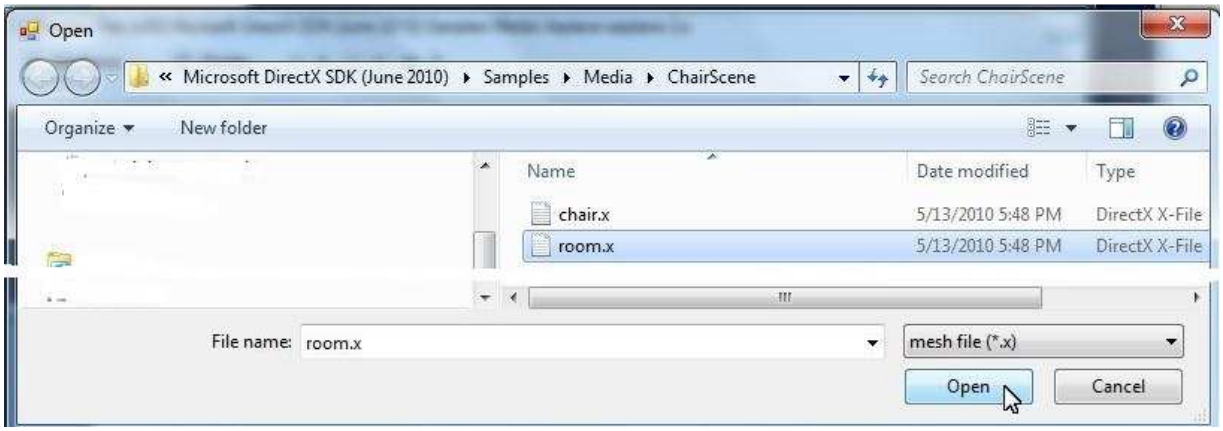
Click radio button X, the airplane starts rotating on X-axis:

Click radio button Z, the airplane starts rotating on Z-axis:

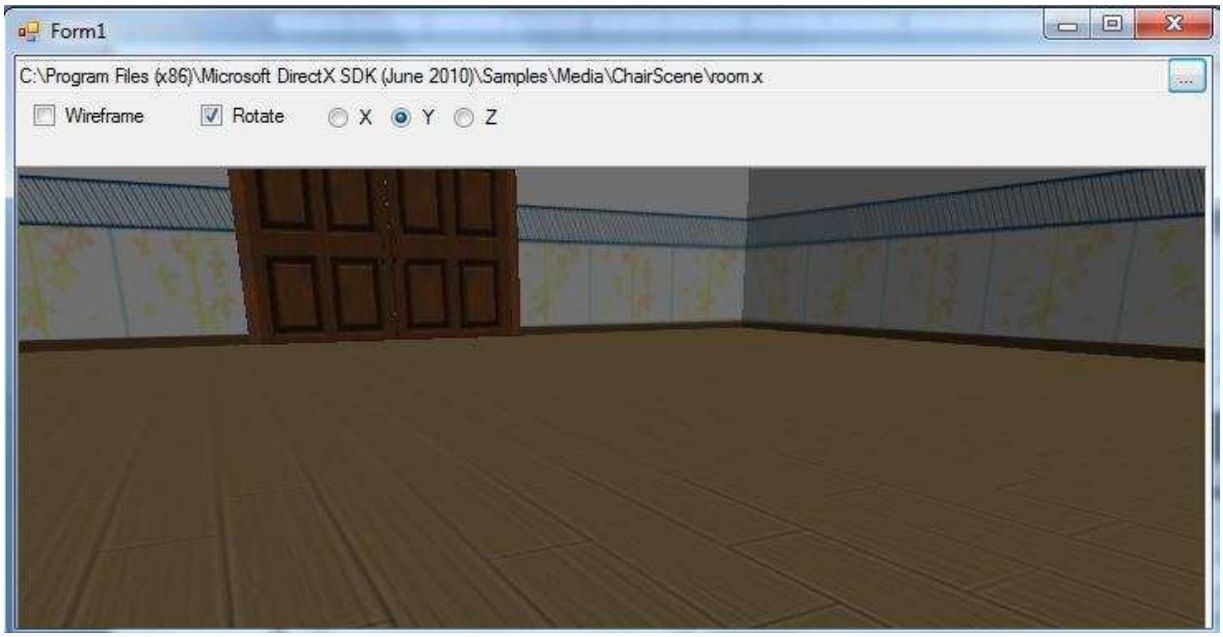
Check “Wireframe” checkbox. The airplane shows wireframe:

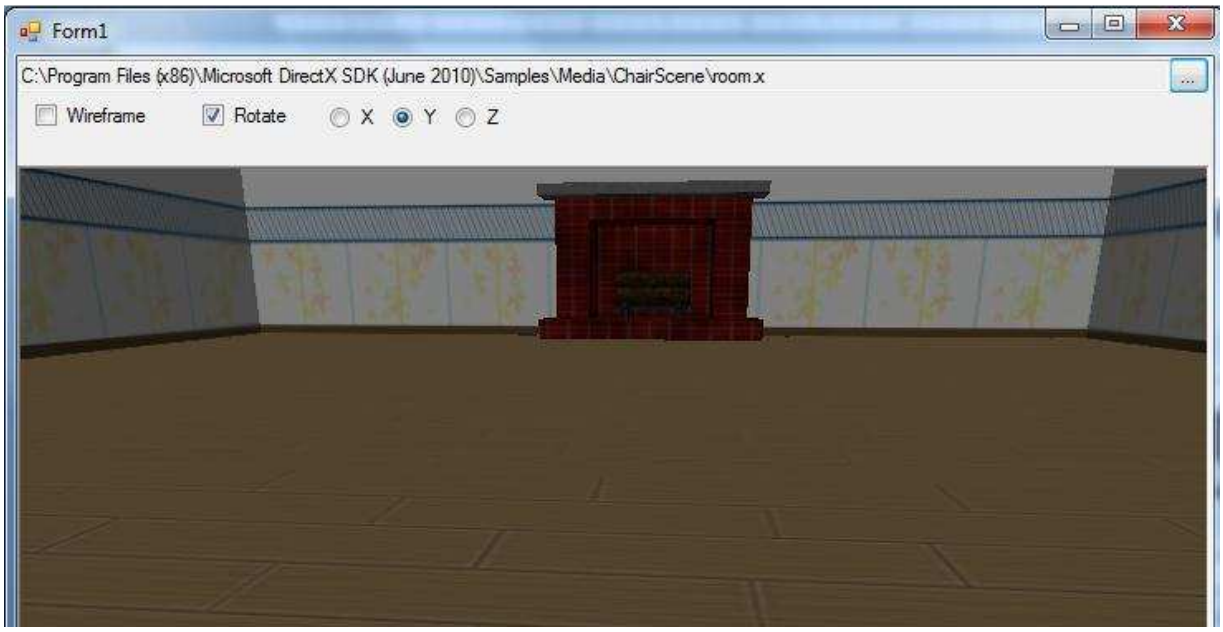


Click button “…” to load another mesh file:



The file is loaded and shows a rotating room:





This sample project can be downloaded from

http://www.limnor.com/studio/WinAPICodePackDX10ShowMesh.zip

DLL files for Windows API Code Pack can be downloaded from:

http://www.limnor.com/studio/WinAPICodePackBin.zip