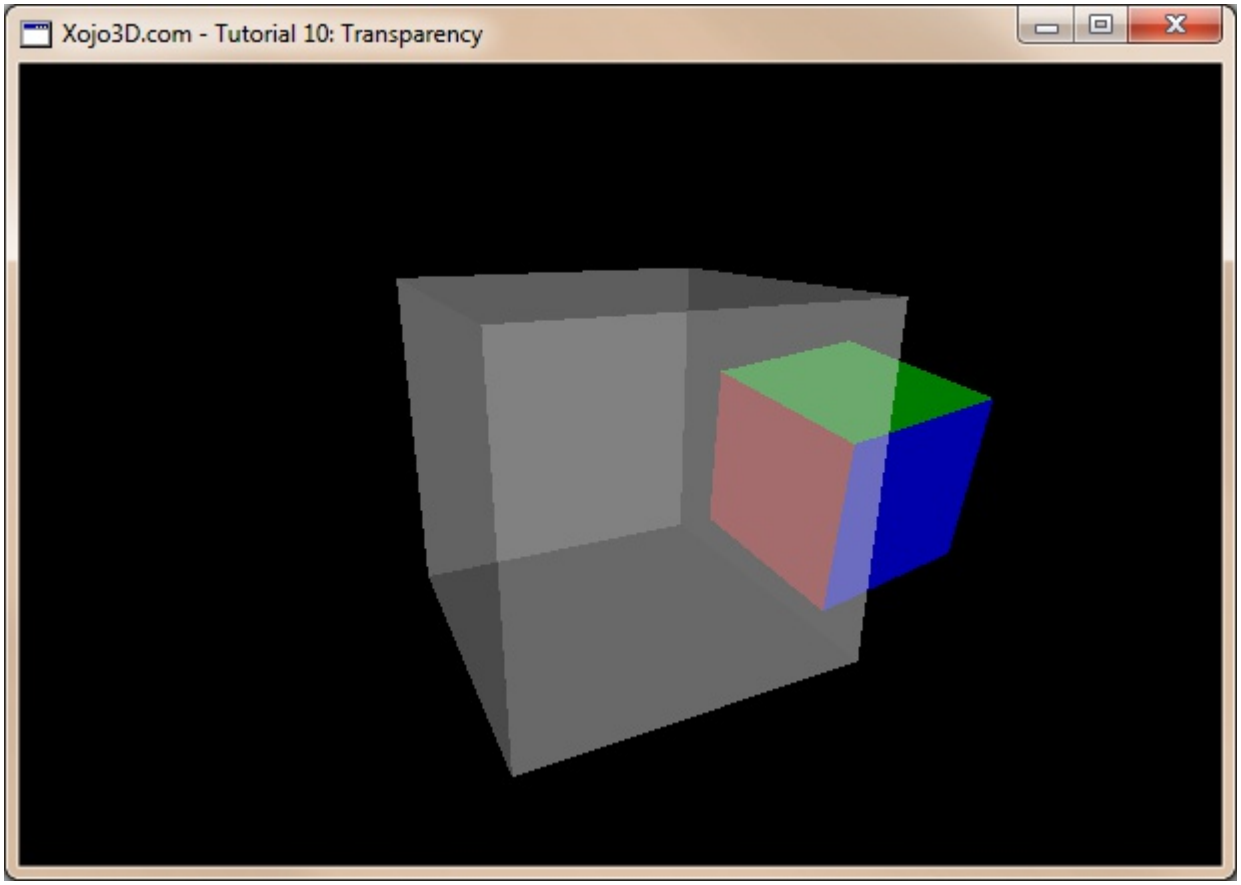
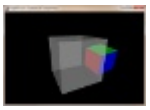
Tutorial 10: Transparency

Some polygons are transparent (see through), and we need to render them as such. The window of a car and a glass cup are examples of transparent objects. This tutorial shows you how to add transparency to your polygons and models.



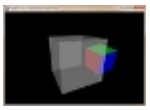


Theory

Consider the two polygons below. These are two opaque polygons, with the green polygon behind the blue polygon.

Now consider the same two polygons again, but this time the blue polygon is transparent. Notice how we can see through the blue polygon.

The value used to specify the level of transparency is better known as the **alpha** value. The alpha value is a real value that ranges from 0 (completely transparent) to 1 (opaque). Below are some examples of different alpha values.



Tutorial Steps

1. Open Xojo.
2. In the Project Chooser select Desktop.
3. Enter "Tutorial010" as the Application Name, and click OK.
4. Save your project.
5. Configure the following controls:

Control	Name	DoubleBuffer	Left	Top	Maximize Button
Window	SurfaceWindow	-	-	-	ON
OpenGLSurface	Surface	ON	0	0	-

6. Position and size *Surface* to fill the whole window, and set its locking to left, top, bottom and right.

7. Add the following code to the *SurfaceWindow.Open* event handler:

```
Self.MouseCursor = System.Cursors.StandardPointer
```

8. Add the following code to the *SurfaceWindow.Paint* event handler:

```
Surface.Render
```

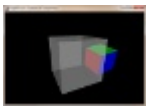
9. Import the X3Core module, created in the previous tutorial.
   You can download the module from http://www.xojo3d.com/tutorials/tut010/x3core.zip.

10. Add the following code to the *Surface.Open* event handler:

```
X3_Initialize

X3_EnableLight OpenGL.GL_LIGHT0, new X3Core.X3Light(0, 0, 1)
```

11. Add the following code to the *Surface.Resized* event handler:

```
X3_SetPerspective Surface
```



12. Replace the code in the *X3Core.X3_Initialize* method with the following:

```
OpenGL.glEnable OpenGL.GL_DEPTH_TEST
OpenGL.glDepthMask OpenGL.GL_TRUE

OpenGL.glCullFace OpenGL.GL_BACK
OpenGL.glEnable OpenGL.GL_CULL_FACE

OpenGL.glEnable OpenGL.GL_LIGHTING

OpenGL.glEnable OpenGL.GL_COLOR_MATERIAL

OpenGL.glEnable OpenGL.GL_BLEND
OpenGL.glBlendFunc OpenGL.GL_SRC_ALPHA, OpenGL.GL_ONE_MINUS_SRC_ALPHA
```

13. Make the following changes in the *X3Core.X3_RenderModel* method:

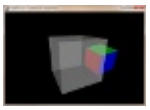
```
' Replace...
' OpenGL.glColor3d(poly.FillColor.Red, poly.FillColor.Green,
poly.FillColor.Blue)
' with

OpenGL.glColor4d(poly.FillColor.Red, poly.FillColor.Green,
poly.FillColor.Blue, poly.FillColor.Alpha)

' and replace...
' OpenGL.glColor3d(1, 1, 1)
' with

OpenGL.glColor4d(1, 1, 1, 1)
```

14. Import the X3Test module into your project.

You can download the module from http://www.xojo3d.com/tutorials/tut010/x3test.zip.

15. Add the following code to the *Surface.Render* event handler:

```
OpenGL.glClearColor(0, 0, 0, 1)
OpenGL.glClear(OpenGL.GL_COLOR_BUFFER_BIT + OpenGL.GL_DEPTH_BUFFER_BIT)

// continue on next page
```



```
// continued from previous page

' render colored cube in background

OpenGL.glPushMatrix
OpenGL.glTranslatef 2.5, 0, -8.0
OpenGL.glRotated(30, 1, 0, 0)
OpenGL.glRotated(30, 0, 1, 0)
X3_RenderModel X3Test_Cube1
OpenGL.glPopMatrix

' render transparent cube

OpenGL.glPushMatrix
OpenGL.glTranslatef 0, 0, -4.0
OpenGL.glRotated(20, 1, 0, 0)
OpenGL.glRotated(25, 0, 1, 0)
X3_RenderModel X3Test_Cube3
OpenGL.glPopMatrix
```

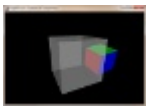
16. Save and run your project.

Analysis

X3Core.X3_Initialize:

```
OpenGL.glEnable OpenGL.GL_DEPTH_TEST
OpenGL.glDepthMask OpenGL.GL_TRUE

OpenGL.glCullFace OpenGL.GL_BACK
OpenGL.glEnable OpenGL.GL_CULL_FACE

OpenGL.glEnable OpenGL.GL_LIGHTING

OpenGL.glEnable OpenGL.GL_COLOR_MATERIAL

OpenGL.glEnable OpenGL.GL_BLEND
OpenGL.glBlendFunc OpenGL.GL_SRC_ALPHA, OpenGL.GL_ONE_MINUS_SRC_ALPHA
```

We added two new instructions to our initialization routine. For transparent rendering we first enabled GL_BLEND with the glEnable instruction.



Next we used glBlendFunc to indicate to OpenGL how the source color (the color of the polygon we're drawing) should be combined with the destination color (the color already in the color buffer). The final color is calculated as follow:

Final color = (Source color * Source blending factor) + (Destination color * Destination blending factor)

Using GL_SRC_ALPHA and GL_ONE_MINUS_SRC_ALPHA as source and destination factors respectively, is a common method to render semi-transparent polygons.

X3Core.X3_RenderModel:

```
OpenGL.glColor4d(poly.FillColor.Red, poly.FillColor.Green,
                 poly.FillColor.Blue, poly.FillColor.Alpha)
```

Notice that we now use the glColor4d function, instead of the glColor3d function, to set our drawing color. The additional alpha value added to the parameter list, gives us the ability to specify the transparency of the color, in addition to its red, green and blue components.