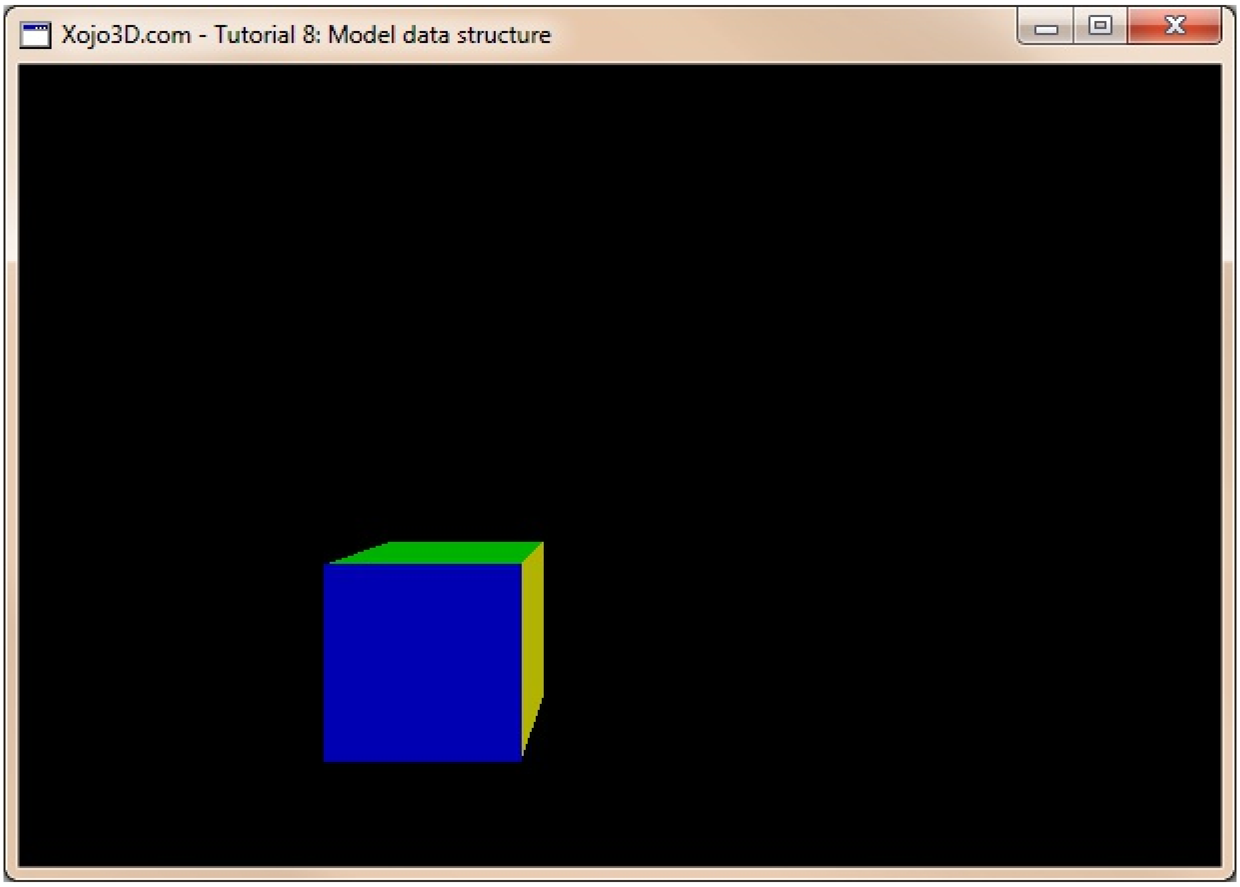
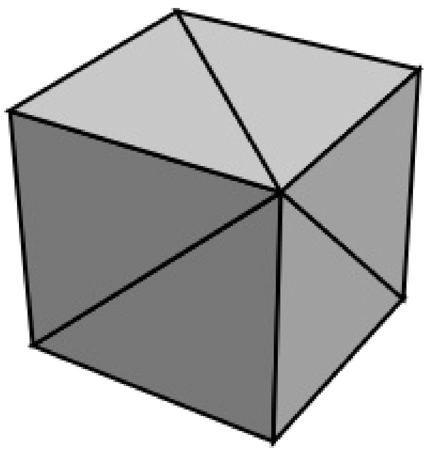
Tutorial 8: Model data structure

In this tutorial we design and implement a model data structure, that groups polygons into a collection to form a 3D object.



Theory

A model is a collection of triangular polygons, combined in such a way to form a 3D object. The image below illustrates a basic cube model built from 12 triangular polygons.



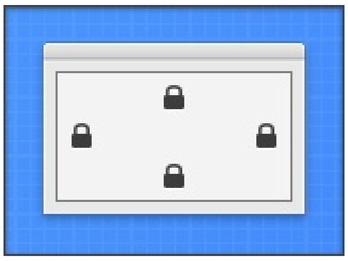
Our model data structure is a class that has a polygon array to store all the polygons of the model.

Tutorial Steps

1. Open Xojo.
2. In the Project Chooser select Desktop.
3. Enter "Tutorial008" 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/tut008/x3core.zip.

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

```
X3_Initialize
```

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

```
X3_SetPerspective Surface
```

12. Add a new class named "X3Model" to module *X3Core*.
13. Add the following property to *X3Model*:

Name	Type
Polygon()	X3Polygon

14. Add the following method to module X3Core:

```
Sub X3_RenderModel(model As X3Core.X3Model)
  Dim i, j As Integer
  Dim poly As X3Core.X3Polygon

  OpenGL.glBegin OpenGL.GL_TRIANGLES

  for i = 0 to model.Polygon.Ubound

    poly = model.Polygon(i)

    if poly.FillColor <> nil then
      OpenGL.glColor3d(poly.FillColor.Red, poly.FillColor.Green,
poly.FillColor.Blue)
    else
      OpenGL.glColor3d(1, 1, 1) ' set the color of the polygon
    end if

    for j = 0 to poly.Vertex.Ubound
      OpenGL.glVertex3d poly.Vertex(j).X, poly.Vertex(j).Y,
poly.Vertex(j).Z
    next j

  next i

  OpenGL.glEnd
End Sub
```

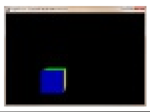
15. Import the X3Test module into your project.

You can download the module from http://www.xojo3d.com/tutorials/tut008/x3test.zip.

16. Add the following code to the Surface.Render event handler:

```
OpenGL.glPushMatrix

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

OpenGL.glTranslatef -2, -2, -8.0

X3_RenderModel X3Test_Cube1

OpenGL.glPopMatrix
```

17. Save and run your project.



Analysis

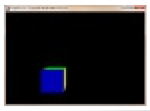
The X3Model class has an array of type X3Polygon that stores all the polygons of the model. Rendering the model is the simple task of looping through the polygons in this array, and rendering each one.

X3Core.X3_RenderModel:

```
Sub X3_RenderModel(model As X3Core.X3Model)

  Dim i, j As Integer
  Dim poly As X3Core.X3Polygon

  OpenGL.glBegin OpenGL.GL_TRIANGLES

  for i = 0 to model.Polygon.Ubound

    poly = model.Polygon(i)

    if poly.FillColor <> nil then
      OpenGL.glColor3d(poly.FillColor.Red, poly.FillColor.Green, 
poly.FillColor.Blue)
    else
      OpenGL.glColor3d(1, 1, 1) ' set the color of the polygon
    end if

    for j = 0 to poly.Vertex.Ubound
      OpenGL.glVertex3d poly.Vertex(j).X, poly.Vertex(j).Y, 
poly.Vertex(j).Z
    next j

  next i

  OpenGL.glEnd

End Sub
```

Large scenes have thousands of models. It therefore makes sense to have a dedicated method to render a model with. The X3_RenderModel method is just such a method, and takes as a parameter the model to render.

X3_RenderModel starts the drawing of triangular polygons with a call to glBegin. It then loops through all the polygons of the model. For each polygon the color of the polygon is set with a call to glColor3d, and then all the vertices of the polygon is added using a nested loop and the glVertex3d function.



Surface.Render:

```
OpenGL.glPushMatrix

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

OpenGL.glTranslatef -2, -2, -8.0

X3_RenderModel X3Test_Cube1

OpenGL.glPopMatrix
```

In the Surface.Render event handler, we use the X3_RenderModel method to render a test cube. The X3Test module provides convenient helper methods that can be used for testing during development. One such method is the X3Test_Cube1 method that generates a colored test cube.