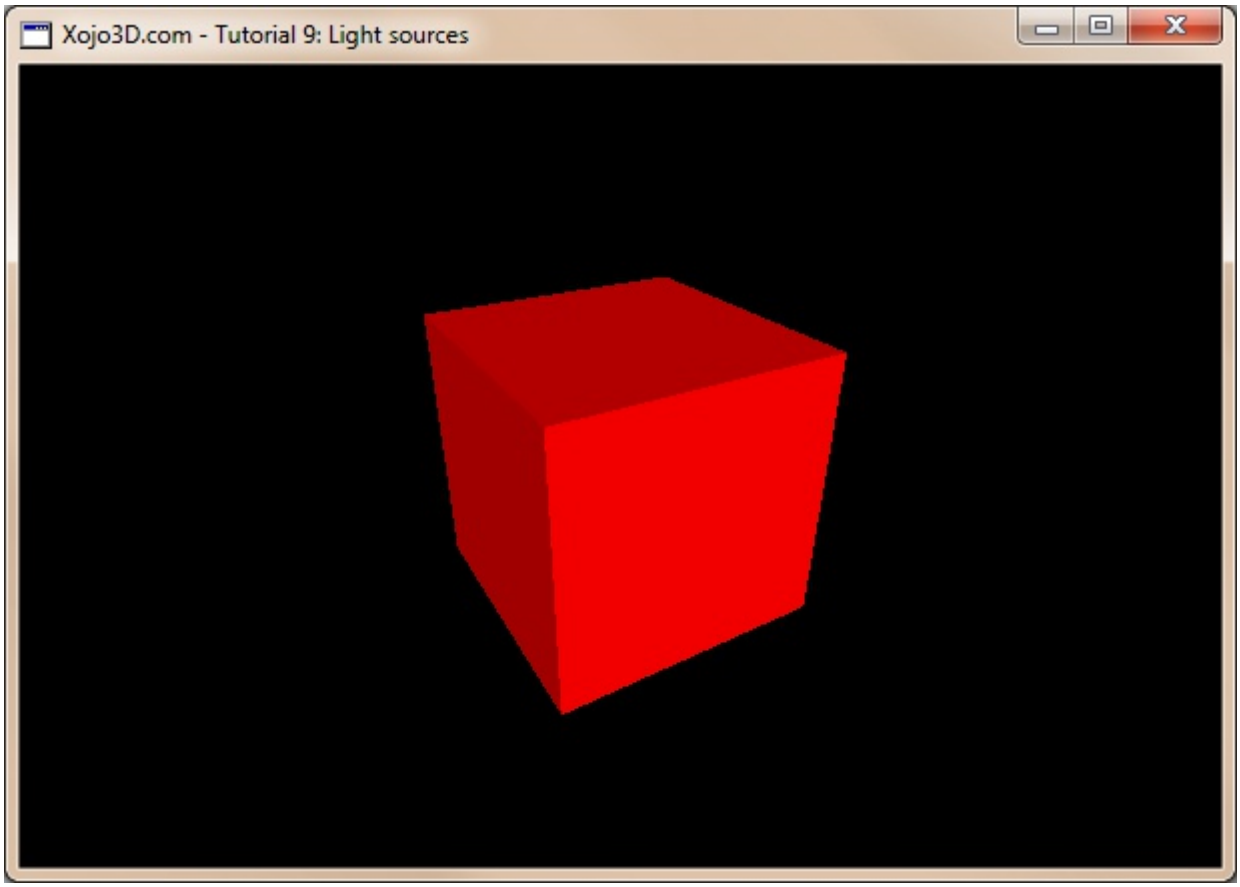
Tutorial 9: Light sources

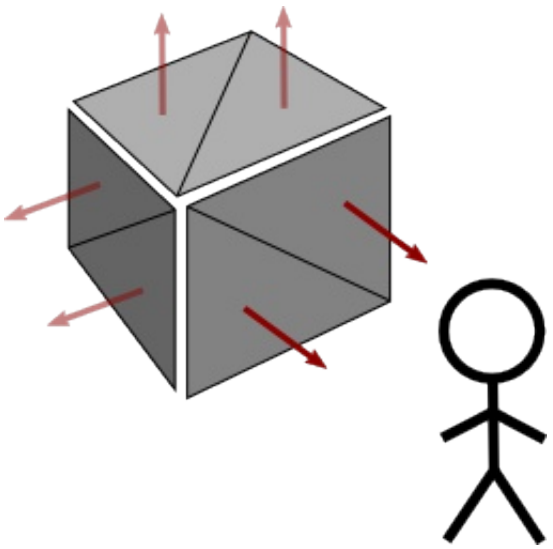
Similar to photography, lighting is essential when creating exceptional 3D scenes. In this tutorial you will learn how to add light sources to your scenes.



Theory

Lighting effects are achieved by making use of vectors. A vector is a quantity that has both magnitude (size) and direction.

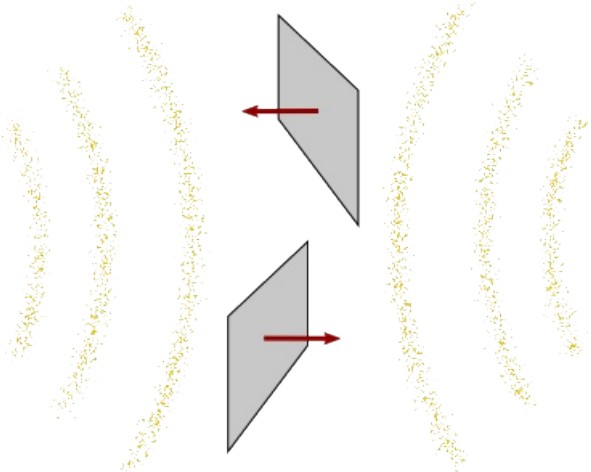
Each polygon has a **normal vector**, which is the direction perpendicular to the surface of the polygon (the direction that the polygon is facing). For example, if you look towards a cube, the direction of the two normals of the front polygons will be pointing towards you.

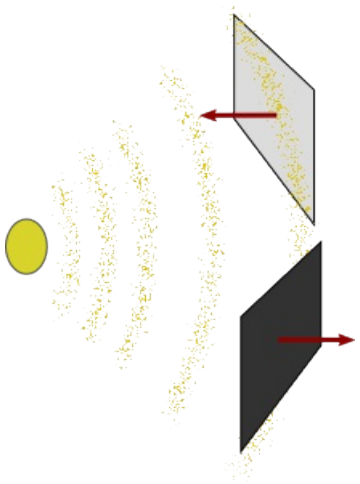


The red arrows in the image above indicate the normal vectors of the polygons.

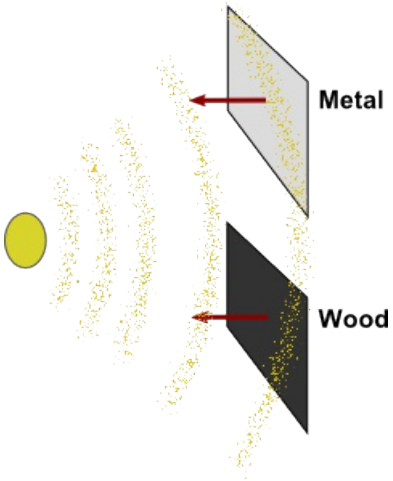
To make use of OpenGL's lighting features, you need to add light sources to your scene. An OpenGL light source has a **position** and **ambient**, **diffuse** and **specular** properties.

Ambient light, is the light that is present everywhere, and is distributed equally to all polygons. The direction of your polygon normal does not affect ambient light. The brighter the ambient light, the brighter all polygons will appear to be in your scene.





Diffuse light originates from a specific point in space (e.g. the sun). It determines how bright your polygon will be, based on the direction of your polygon's normal vector. If your polygon is facing towards a diffuse light source it will be brighter than when it is facing away from the light source.



Specular light also originates from a specific point in space, but reflects differently then diffuse light. Specular light uses the properties of a material to determine brightness. For example, a smooth surface such as a metal will reflect specular light better than a dull surface such as plaster on a brick wall.

Light and color, when applied correctly, can give your scenes exceptional beauty. As a habit, learn to experiment a lot with light and colors.

Tutorial Steps

1. Open Xojo.
2. In the Project Chooser select Desktop.
3. Enter "Tutorial009" 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/tut009/x3core.zip.
10. Add the following code to the *Surface.Resized* event handler:

```
X3_SetPerspective Surface
```

11. Add the following property to *X3Color*:

Name	Type
Alpha	Double

12. Add the following method to *X3Color*:

```
Function GetMemoryBlock() As MemoryBlock
  Dim mblock As new MemoryBlock(16)

  mblock.SingleValue(0) = Red
  mblock.SingleValue(4) = Green
  mblock.SingleValue(8) = Blue
  mblock.SingleValue(12) = Alpha

  return mblock
End Function
```

13. Remove the existing Contructor method of *X3Color*.
14. Add the following Constructor to *X3Color*.

```
Sub Constructor(initRed As Double,
                initGreen As Double,
                initBlue As Double,
                initAlpha As Double = 1)
  Red = initRed
  Green = initGreen
  Blue = initBlue
  Alpha = initAlpha
End Sub
```

15. Add the following property to *X3Polygon*:

Name	Type
Normal	X3Vector

16. Add the following Constructor to *X3Polygon*:

```
Sub Constructor()
  Normal = new X3Vector(0, 0, 0)
End Sub
```

17. Add the following second Constructor to *X3Polygon*:

```
Sub Constructor(normalX As Double,
                normalY As Double,
                normalZ As Double)
  Normal = new X3Vector(normalX, normalY, normalZ)
End Sub
```

18. Add a new class named "X3Light" to module *X3Core*.

19. Add the following properties to *X3Light*:

Name	Type
Ambient	X3Color
Diffuse	X3Color
Position	X3Vector
Specular	X3Color

20. Add the following method to *X3Light*.

```
Sub Constructor(xPos As Double, yPos As Double, zPos As Double)
  Position = new X3Vector(xPos, yPos, zPos)
  Ambient = new X3Color(0, 0, 0, 1)
  Diffuse = new X3Color(1, 1, 1, 1)
  Specular = new X3Color(1, 1, 1, 1)
End Sub
```

21. Add the following method to *X3Vector*:

```
Function GetMemoryBlock() As MemoryBlock
  Dim mblock As new MemoryBlock(16)

  mblock.SingleValue(0) = X
  mblock.SingleValue(4) = Y
  mblock.SingleValue(8) = Z
  mblock.SingleValue(12) = 0

  return mblock
End Function
```

22. 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
```

23. Add the following method to module *X3Core*:

```
Sub X3_EnableLight(lightIndex As Integer, light As X3Core.X3Light)
  OpenGL.glLightfv lightIndex, OpenGL.GL_POSITION,
light.Position.GetMemoryBlock
  OpenGL.glLightfv lightIndex, OpenGL.GL_AMBIENT,
light.Ambient.GetMemoryBlock
  OpenGL.glLightfv lightIndex, OpenGL.GL_DIFFUSE,
light.Diffuse.GetMemoryBlock
  OpenGL.glLightfv lightIndex, OpenGL.GL_SPECULAR,
light.Specular.GetMemoryBlock

  OpenGL.glEnable lightIndex
End Sub
```

24. Replace the code in the *X3Core.X3_RenderModel* method with the following:

```
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)
  end if

  OpenGL.glNormal3d poly.Normal.X, poly.Normal.Y, poly.Normal.Z

  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
```

25. Import the X3Test module into your project.

You can download the module from http://www.xojo3d.com/tutorials/tut009/x3test.zip.

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

```
X3_Initialize

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

27. 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 0, 0, -5.0

OpenGL.glRotated(30, 1, 0, 0)
OpenGL.glRotated(30, 0, 1, 0)

X3_RenderModel X3Test_Cube2

OpenGL.glPopMatrix
```

28. Save and run your project.

Analysis

The GetMemoryBlock methods added to X3Color and X3Vector return MemoryBlock objects that contain color and position information respectively. These MemoryBlock objects are in a format that is compatible with OpenGL, and is used when passing color and position information to OpenGL function calls.

X3Color.Constructor:

```
Sub Constructor(initRed As Double, initGreen As Double,
                    initBlue As Double, initAlpha As Double = 1)
  Red = initRed
  Green = initGreen
  Blue = initBlue
  Alpha = initAlpha
End Sub
```

The new constructor of X3Color now has an optional initAlpha parameter with a default value of 1. This parameter is used to instantiate the new Alpha property added to X3Color. The alpha value determines the transparency of a color, with 1 being completely opaque (not see through) and 0 being completely transparent (see through).

Two new constructors were added to X3Polygon:

```
Sub Constructor()
  Normal = new X3Vector(0, 0, 0)
End Sub
```

The first constructor simply initializes the normal of the polygon to a NULL vector if default values aren't provided for the normal.

```
Sub Constructor(normalX As Double,
                normalY As Double,
                normalZ As Double)
  Normal = new X3Vector(normalX, normalY, normalZ)
End Sub
```

The second constructor initializes the normal of the polygon using the values provided by the parameters of the constructor.

X3Light.Constructor:

```
Sub Constructor(xPos As Double, yPos As Double, zPos As Double)
  Position = new X3Vector(xPos, yPos, zPos)
  Ambient = new X3Color(0, 0, 0, 1)
  Diffuse = new X3Color(1, 1, 1, 1)
  Specular = new X3Color(1, 1, 1, 1)
End Sub
```

An OpenGL light source has many adjustable properties. Our constructor simply sets up the position of the light source, and default values for the ambient, diffuse and specular properties that should be sufficient for most purposes.

X3Core.X3_Initialize:

```
OpenGL.glEnable OpenGL.GL_LIGHTING

OpenGL.glEnable OpenGL.GL_COLOR_MATERIAL
```

Two new instructions were added to our X3_Initialize method. The first instruction enables OpenGL's lighting. Without this instruction light sources will be ignored during rendering.

Enabling OpenGL lighting disables the glColor command. By enabling GL_COLOR_MATERIAL with our second instruction, the glColor instruction works again when lighting is enabled.

X3Core.X3_EnableLight:

```
Sub X3_EnableLight(lightIndex As Integer, light As X3Core.X3Light)
  OpenGL.glLightfv lightIndex, OpenGL.GL_POSITION,
light.Position.GetMemoryBlock
  OpenGL.glLightfv lightIndex, OpenGL.GL_AMBIENT,
light.Ambient.GetMemoryBlock
  OpenGL.glLightfv lightIndex, OpenGL.GL_DIFFUSE,
light.Diffuse.GetMemoryBlock
  OpenGL.glLightfv lightIndex, OpenGL.GL_SPECULAR,
light.Specular.GetMemoryBlock

  OpenGL.glEnable lightIndex
End Sub
```

The X3_EnableLight method is simply a helper method that makes it easier for us to configure OpenGL with our X3Light objects.

The OpenGL environment can be configured with up to GL_MAX_LIGHTS light sources. The first parameter, lightIndex, is the identifier of the light source of to be configured. This value can be anything between 0 and GL_MAX_LIGHTS).

The second X3Light parameter is the light source to configure.

X3Core.X3_RenderModel:

```
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)
  end if

  OpenGL.glNormal3d poly.Normal.X, poly.Normal.Y, poly.Normal.Z

// continue on next page
```

```
// continued from previous page

  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
```

X3Core.X3_RenderModel has one new instruction added to it... glNormal3d.

This instruction simply sets up the normal of the next polygon to be drawn, to ensure that OpenGL can apply the correct lighting effects to the polygon, based on the direction that the polygon is facing.