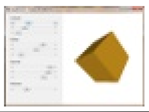
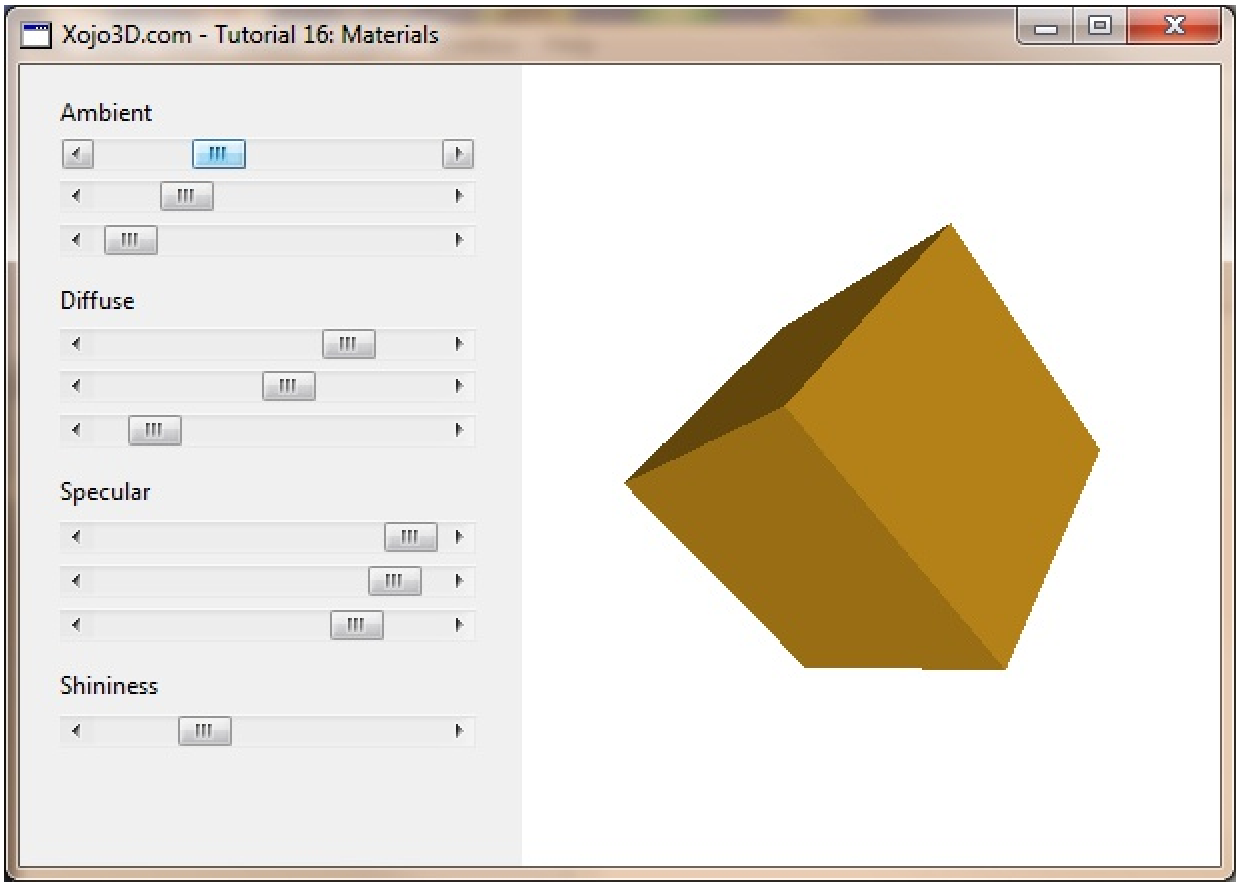


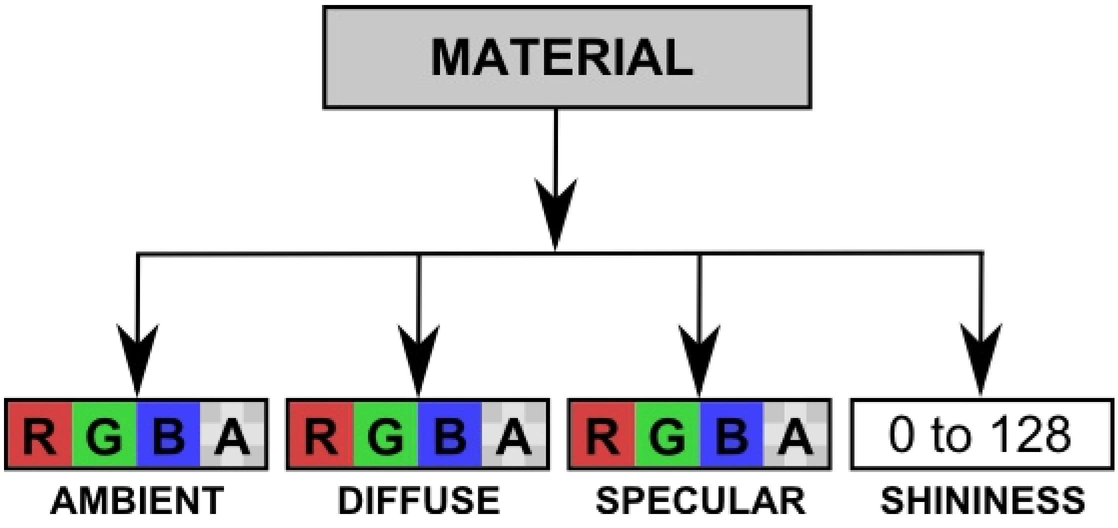
Tutorial 16: Materials

Different types of materials reflects differently in light, and by applying material properties to your models you can greatly enhance the aesthetics of your scenes. In this tutorial we explore materials, how materials reacts to light, and how to use this knowledge in the programming of your applications.



Theory

Similar to light sources, a material has **ambient**, **diffuse** and **specular** properties. There is also an additional shininess value that comes into play with materials, which determines the width of the specular peak of the material. This effectively determines the brightness and size of the reflection on a material. A higher shininess value results in a more focused (smaller and brighter) highlight.



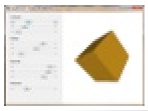
The shininess value is a floating point value between (and including) 0 and 128. The ambient, diffuse and specular components of the material are each broken down into red, green, blue and alpha (RGBA) values. These material properties determines how light affects the surface of a polygon during rendering.

The **ambient** colors define how the material appears when not in direct light. This value will typically be the same as the diffuse values.

The **diffuse** colors determine how the material reflects light. Essentially it defines what color the polygon will be when hit by light.

The **specular** colors is used in combination with the shininess value to determine how much a material shines when hit by light. A metal such as chrome for example, has a much higher shininess value than copper.

To help you get started with materials, the following table describes some common materials. All the alpha values are equal to 1 for these materials.



Material	Ambient			Diffuse			Specular			Shininess
	Red	Green	Blue	Red	Green	Blue	Red	Green	Blue	
Brass	0.329412	0.223529	0.027451	0.780392	0.568627	0.113725	0.992157	0.941176	0.807843	27.8974
Bronze	0.2125	0.1275	0.054	0.714	0.4284	0.18144	0.393548	0.271906	0.166721	25.6
Chrome	0.25	0.25	0.25	0.4	0.4	0.4	0.774597	0.774597	0.774597	76.8
Copper	0.19125	0.0735	0.0225	0.7038	0.27048	0.0828	0.256777	0.137622	0.086014	12.8
Emerald	0.0215	0.1745	0.0215	0.07568	0.61424	0.07568	0.633	0.727811	0.633	76.8
Gold	0.24725	0.1995	0.0745	0.75164	0.60648	0.22648	0.628281	0.555802	0.366065	51.2
Jade	0.135	0.2225	0.1575	0.54	0.89	0.63	0.316228	0.316228	0.316228	12.8
Obsidian	0.05375	0.05	0.06625	0.18275	0.17	0.22525	0.332741	0.328634	0.346435	38.4
Pearl	0.25	0.20725	0.20725	1	0.829	0.829	0.296648	0.296648	0.296648	11.264
Plastic (Black)	0	0	0	0.01	0.01	0.01	0.50	0.50	0.50	32
Plastic (Cyan)	0	0.1	0.06	0	0.50980392	0.50980392	0.50196078	0.50196078	0.50196078	32
Plastic (Green)	0	0	0	0.1	0.35	0.1	0.45	0.55	0.45	32
Plastic (Red)	0	0	0	0.5	0	0	0.7	0.6	0.6	32
Plastic (White)	0	0	0	0.55	0.55	0.55	0.7	0.7	0.7	32
Plastic (Yellow)	0	0	0	0.5	0.5	0	0.6	0.6	0.	32
Rubber (Black)	0.02	0.02	0.02	0.01	0.01	0.01	0.4	0.4	0.4	10
Rubber (Cyan)	0	0.05	0.05	0.4	0.5	0.5	0.04	0.7	0.7	10
Rubber (Green)	0	0.05	0	0.4	0.5	0.4	0.04	0.7	0.04	10
Rubber (Red)	0.05	0	0	0.5	0.4	0.4	0.7	0.04	0.04	10
Rubber (White)	0.05	0.05	0.05	0.5	0.5	0.5	0.7	0.7	0.7	10
Rubber (Yellow)	0.05	0.05	0	0.5	0.5	0.4	0.7	0.7	0.04	10
Ruby	0.1745	0.01175	0.01175	0.61424	0.04136	0.04136	0.727811	0.626959	0.626959	76.8
Silver	0.19225	0.19225	0.19225	0.50754	0.50754	0.50754	0.508273	0.508273	0.508273	51.2
Turquoise	0.1	0.18725	0.1745	0.396	0.74151	0.69102	0.297254	0.30829	0.306678	12.8

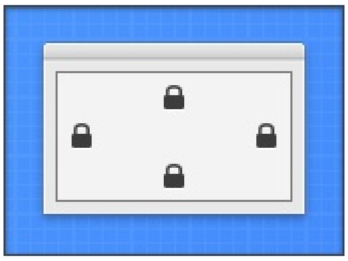
Tutorial Steps

1. Create a new Xojo desktop project.
2. Save your project.
3. Import the X3Core module.
   You can download the module from http://www.xojo3d.com/tutorials/tut016/x3core.zip.
4. Import the X3Test module.
   You can download the module from http://www.xojo3d.com/tutorials/tut016/x3test.zip.
5. Configure the following controls:

Control	Name	Text	Left	Top	Width	Period
Window	SurfaceWindow	-	-	-	-	-
OpenGLSurface	Surface	-	251	0	-	-
Label	lblAmbient	Ambient	20	14	-	-
ScrollBar	scrollAmbientRed	-	20	31	208	-
ScrollBar	scrollAmbientGreen	-	20	52	208	-
ScrollBar	scrollAmbientBlue	-	20	74	208	-
ScrollBar	scrollAmbientAlpha	-	20	95	208	-

Control	Name	Text	Left	Top	Width	Period
ScrollBar	scrollDiffuseRed	-	20	136	208	-
ScrollBar	scrollDiffuseGreen	-	20	158	208	-
ScrollBar	scrollDiffuseBlue	-	20	180	208	-
ScrollBar	scrollDiffuseAlpha	-	20	202	208	-
Label	lblSpecular	Specular	20	223	-	-
ScrollBar	scrollSpecularRed	-	20	247	208	-
ScrollBar	scrollSpecularGreen	-	20	269	208	-
ScrollBar	scrollSpecularBlue	-	20	291	208	-
ScrollBar	scrollSpecularAlpha	-	20	313	208	-
Label	lblSpecular	Shininess	20	339	-	-
ScrollBar	scrollShininess	-	20	363	208	-
Timer	tmrRotate	-	-	-	-	100

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



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

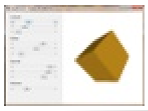
```
Surface.Render
```

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

```
Model = X3Test_MaterialCube()

GetMaterialSettings()
```

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

```
X3_Initialize

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



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

```
X3_SetPerspective Surface
```

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

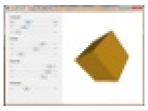
```
Dim i As Integer

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

OpenGL.glPushMatrix

OpenGL.glTranslatef 0, 0, -5

if Model <> nil then
  X3_RenderModel Model
end if

OpenGL.glPopMatrix
```

12. Add the following properties to *SurfaceWindow*:

Name	Type
Loading	Boolean
Model	X3Core.X3Model

13. Add a new class named "X3Material" to module *X3Core*.
14. Add the following properties to *X3Material*:

Name	Type
Ambient	X3Core.X3Color
Diffuse	X3Core.X3Color
Specular	X3Core.X3Color
Shininess	Double



15. Add the following method to X3Material:

```
Sub Constructor()
  Ambient = new X3Color(0.2, 0.2, 0.2)
  Diffuse = new X3Color(0.8, 0.8, 0.8)
  Specular = new X3Color(0, 0, 0)
  Shininess = 0
End Sub
```

16. Add the following method to *X3Material*:

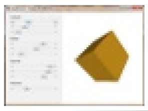
```
Function Clone() As X3Core.X3Material
  Dim mat As new X3Core.X3Material

  mat.Ambient = Ambient.Clone
  mat.Diffuse = Diffuse.Clone
  mat.Shininess = Shininess
  mat.Specular = Specular.Clone

  return mat
End Function
```

17. Add the following method to *X3Core*:

```
Sub X3_SetMaterial(material As X3Core.X3Material)
  Dim matMB As MemoryBlock

  matMB = material.Ambient.GetMemoryBlock()
  OpenGL.glMaterialfv(OpenGL.GL_FRONT, OpenGL.GL_AMBIENT, matMB)

  matMB = material.Diffuse.GetMemoryBlock()
  OpenGL.glMaterialfv(OpenGL.GL_FRONT, OpenGL.GL_DIFFUSE, matMB)

  matMB = material.Specular.GetMemoryBlock()
  OpenGL.glMaterialfv(OpenGL.GL_FRONT, OpenGL.GL_SPECULAR, matMB)

  OpenGL.glMaterialf(OpenGL.GL_FRONT_AND_BACK, OpenGL.GL_SHININESS,
              material.Shininess)
End Sub
```

18. Add the following method to *SurfaceWindow*:

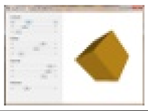
```
Sub GetMaterialSettings()
  Dim mat As X3Core.X3Material

  if Model <> nil then

// continue on next page
```



```
// continued from previous page

    Loading = true

    mat = Model.Material(0)

    scrollAmbientRed.Value = Round(mat.Ambient.Red * 100)
    scrollAmbientGreen.Value = Round(mat.Ambient.Green * 100)
    scrollAmbientBlue.Value = Round(mat.Ambient.Blue * 100)
    scrollAmbientAlpha.Value = Round(mat.Ambient.Alpha * 100)

    scrollDiffuseRed.Value = Round(mat.Diffuse.Red * 100)
    scrollDiffuseGreen.Value = Round(mat.Diffuse.Green * 100)
    scrollDiffuseBlue.Value = Round(mat.Diffuse.Blue * 100)
    scrollDiffuseAlpha.Value = Round(mat.Diffuse.Alpha * 100)

    scrollSpecularRed.Value = Round(mat.Specular.Red * 100)
    scrollSpecularGreen.Value = Round(mat.Specular.Green * 100)
    scrollSpecularBlue.Value = Round(mat.Specular.Blue * 100)
    scrollSpecularAlpha.Value = Round(mat.Specular.Alpha * 100)

    scrollShininess.Value = Round(mat.Shininess)

    Loading = false

  end if
End Sub
```

19. Add the following method to *SurfaceWindow*:

```
Sub ApplyMaterialSettings()
  Dim mat As X3Core.X3Material

  if (Model <> nil) and not Loading then

    mat = Model.Material(0)

    mat.Ambient.Red = scrollAmbientRed.Value / 100
    mat.Ambient.Green = scrollAmbientGreen.Value / 100
    mat.Ambient.Blue = scrollAmbientBlue.Value / 100
    mat.Ambient.Alpha = scrollAmbientAlpha.Value / 100

// continue on next page
```



```
// continued from previous page

    mat.Diffuse.Red = scrollDiffuseRed.Value / 100
    mat.Diffuse.Green = scrollDiffuseGreen.Value / 100
    mat.Diffuse.Blue = scrollDiffuseBlue.Value / 100
    mat.Diffuse.Alpha = scrollDiffuseAlpha.Value / 100

    mat.Specular.Red = scrollSpecularRed.Value / 100
    mat.Specular.Green = scrollSpecularGreen.Value / 100
    mat.Specular.Blue = scrollSpecularBlue.Value / 100
    mat.Specular.Alpha = scrollSpecularAlpha.Value / 100

    mat.Shininess = scrollShininess.Value

    Model.Invalidate = true

  end if
End Sub
```

20. Add the following line of code to ALL the ValueChanged events of ALL the scrollbars:

```
ApplyMaterialSettings
```

21. Add the following code to the *tmrRotate.Action* event handler:

```
if Model <> nil then
  Model.Rotation.Pitch(7)
  Model.Rotation.Yaw(14)
  Model.Rotation.Roll(7)
  Surface.Render
end if
```

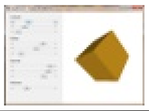
22. Save and run your project.

Analysis

The new X3Material class represents a material that can be used for polygons.

X3Sprite.Constructor:

```
Sub Constructor()
  Ambient = new X3Color(0.2, 0.2, 0.2)
  Diffuse = new X3Color(0.8, 0.8, 0.8)
  Specular = new X3Color(0, 0, 0)
  Shininess = 0
End Sub
```



The constructor of the X3Material class instantiates the components of the material with default values. These values can be updated afterward with the correct values for the material.

The different materials used by a model is stored in the model's Material() array. Polygons are then linked to these materials by storing the index of a material in the Polygon.MIndex property. During the rendering of the polygon the material pointed to by MIndex is used to render the polygon.

X3Core.X3_SetMaterial:

```
Sub X3_SetMaterial(material As X3Core.X3Material)
  Dim matMB As MemoryBlock

  matMB = material.Ambient.GetMemoryBlock()
  OpenGL.glMaterialfv(OpenGL.GL_FRONT, OpenGL.GL_AMBIENT, matMB)

  matMB = material.Diffuse.GetMemoryBlock()
  OpenGL.glMaterialfv(OpenGL.GL_FRONT, OpenGL.GL_DIFFUSE, matMB)

  matMB = material.Specular.GetMemoryBlock()
  OpenGL.glMaterialfv(OpenGL.GL_FRONT, OpenGL.GL_SPECULAR, matMB)

  OpenGL.glMaterialf(OpenGL.GL_FRONT_AND_BACK, OpenGL.GL_SHININESS,
               material.Shininess)
End Sub
```

X3_SetMaterial is a helper method used to configure OpenGL with a selected material. The material object to configure is simply passed as a parameter to X3_SetMaterial.