The Virtual Reality Modeling Language

4 Concepts

4.14 Lighting model

4.14.1 Introduction

The VRML lighting model provides detailed equations which define the colours to apply to each geometric object. For each object, the values of the Material node, Color node and texture currently being applied to the object are combined with the lights illuminating the object and the currently bound Fog node. These equations are designed to simulate the physical properties of light striking a surface.

4.14.2 Lighting 'off'

A Shape node is unlit if either of the following is true:

a. The shape's *appearance* field is NULL (default).
b. The *material* field in the Appearance node is NULL (default).

Note the special cases of geometry nodes that do not support lighting (see 6.24, IndexedLineSet, and 6.36, PointSet, for details).

If the shape is unlit, the colour (I_{rgb}) and alpha (A, 1-transparency) of the shape at each point on the shape's geometry is given in Table 4.5.

Table 4.5 -- Unlit colour and alpha mapping

Texture type	Colour per-vertex or per-face	Colour NULL
No texture	$I_{rgb} = I_{Crgb}$ $A = 1$	$I_{rgb} = (1, 1, 1)$ $A = 1$
Intensity (one-component)	$I_{rgb} = I_T \times I_{Crgb}$ $A = 1$	$I_{rgb} = (I_T, I_T, I_T)$ $A = 1$
Intensity+Alpha (two-component)	$I_{rgb} = I_T \times I_{Crgb}$ $A = A_T$	$I_{rgb} = (I_T, I_T, I_T)$ $A = A_T$
RGB (three-component)	$I_{rgb} = I_{Trgb}$ $A = 1$	$I_{rgb} = I_{Trgb}$ $A = 1$
RGBA	$I_{rgb} = I_{Trgb}$	$I_{rgb} = I_{Trgb}$

(four-component)	$A = A_T$	$A = A_T$

where:

A_T = normalized [0, 1] alpha value from 2 or 4 component texture image
I_{Crgb} = interpolated per-vertex colour, or per-face colour, from Color node
I_T = normalized [0, 1] intensity from 1 or 2 component texture image
I_{Trgb}= colour from 3-4 component texture image

4.14.3 Lighting 'on'

If the shape is lit (i.e., a Material and an Appearance node are specified for the Shape), the Color and Texture nodes determine the diffuse colour for the lighting equation as specified in Table 4.6.

Table 4.6 -- Lit colour and alpha mapping

Texture type	Colour per-vertex or per-face	Color node NULL
No texture	$O_{Drgb} = I_{Crgb}$ $A = 1\text{-}T_M$	$O_{Drgb} = I_{Drgb}$ $A = 1\text{-}T_M$
Intensity texture (one-component)	$O_{Drgb} = I_T \times I_{Crgb}$ $A = 1\text{-}T_M$	$O_{Drgb} = I_T \times I_{Drgb}$ $A = 1\text{-}T_M$
Intensity+Alpha texture (two-component)	$O_{Drgb} = I_T \times I_{Crgb}$ $A = A_T$	$O_{Drgb} = I_T \times I_{Drgb}$ $A = A_T$
RGB texture (three-component)	$O_{Drgb} = I_{Trgb}$ $A = 1\text{-}T_M$	$O_{Drgb} = I_{Trgb}$ $A = 1\text{-}T_M$
RGBA texture (four-component)	$O_{Drgb} = I_{Trgb}$ $A = A_T$	$O_{Drgb} = I_{Trgb}$ $A = A_T$

where:

I_{Drgb} = material *diffuseColor*
O_{Drgb} = diffuse factor, used in lighting equations below
T_M = material *transparency*

All other terms are as defined in 4.14.2, Lighting `off'.

4.14.4 Lighting equations

An ideal VRML implementation will evaluate the following lighting equation at each point on a lit surface. RGB intensities at each point on a geometry (I_{rgb}) are given by:

$$I_{rgb} = I_{Frgb} \times (1 - f_0) + f_0 \times (O_{Ergb} + \text{SUM}(\, on_i \times attenuation_i \times spot_i \times I_{Lrgb} \times (ambient_i + diffuse_i + specular_i)))$$

where:

$attenuation_i = 1 / \max(c_1 + c_2 \times d_L + c_3 \times d_L^2 , 1)$
$ambient_i = I_{ia} \times O_{Drgb} \times O_a$

$diffuse_i = I_i \times O_{Drgb} \times (\mathbf{N} \cdot \mathbf{L})$
$specular_i = I_i \times O_{Srgb} \times (\mathbf{N} \cdot ((\mathbf{L} + \mathbf{v}) / |\mathbf{L} + \mathbf{v}|))^{shininess \times 128}$

and:

· = modified vector dot product: if dot product < 0, then 0.0, otherwise, dot product
c_1 , c_2, c_3 = light i *attenuation*
d_V = distance from point on geometry to viewer's position, in coordinate system of current fog node
d_L = distance from light to point on geometry, in light's coordinate system
f_0 = Fog interpolant, see Table 4.8 for calculation
I_{Frgb} = currently bound fog's *color*
I_{Lrgb} = light i *color*
I_i = light i *intensity*
I_{ia} = light i *ambientIntensity*
L = (Point/SpotLight) normalized vector from point on geometry to light source i position
L = (DirectionalLight) -direction of light source i
N = normalized normal vector at this point on geometry (interpolated from vertex normals specified in Normal node or calculated by browser)
O_a = Material *ambientIntensity*
O_{Drgb} = diffuse colour, from Material node, Color node, and/or texture node
O_{Ergb} = Material *emissiveColor*
O_{Srgb} = Material *specularColor*
on_i = 1, if light source i affects this point on the geometry,
0, if light source i does not affect this geometry (if farther away than *radius* for PointLight or SpotLight, outside of enclosing Group/Transform for DirectionalLights, or *on* field is FALSE)
shininess = Material *shininess*
spotAngle = acos($-\mathbf{L} \cdot \mathbf{spotDir}_i$)
$spot_{BW}$ = SpotLight i beamWidth
$spot_{CO}$ = SpotLight i *cutOffAngle*
$spot_i$ = spotlight factor, see Table 4.7 for calculation
$\mathbf{spotDir}_i$ = normalized SpotLight i *direction*
SUM: sum over all light sources i
v = normalized vector from point on geometry to viewer's position

Table 4.7 -- Calculation of the spotlight factor

Condition (in order)	$spot_i$ =
$light_i$ is PointLight or DirectionalLight	1
spotAngle >= $spot_{CO}$	0

spotAngle <= $spot_{BW}$	1
$spot_{BW}$ < spotAngle < $spot_{CO}$	(spotAngle - $spot_{CO}$) / ($spot_{BW}$-$spot_{CO}$)

Table 4.8 -- Calculation of the fog interpolant

Condition	**f_0 =**
no fog	1
fogType "LINEAR", d_v < fogVisibility	(fogVisibility-d_v) / fogVisibility
fogType "LINEAR", $d_v \geq$ fogVisibility	0
fogType "EXPONENTIAL", d_v < fogVisibility	exp(-d_v / (fogVisibility-d_v))
fogType "EXPONENTIAL", $d_v \geq$ fogVisibility	0

4.14.5 References

The VRML lighting equations are based on the simple illumination equations given in

Foley, van Dam, Feiner and Hughes, Computer Graphics Principles and Practice, 2nd Edition, Addison Wesley, Reading, MA, 1990.
http://www.awl.com

"The OpenGL Graphics System: A Specification (Version 1.1)," Silicon Graphics, Inc., 1995.
http://www.sgi.com/Technology/openGL/glspec1.1/glspec.html