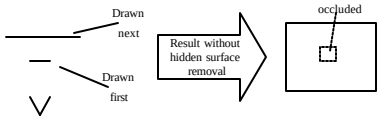
Computer Graphics Course 2001

OpenGL - Lighting, Shading and Material Properties

Hidden Surface Removal

- First we begin with hidden surface removal. When drawing objects in order which does not match the order of their appearance (distance from the camera) we get wrong occlusions.



- Note: the order is view dependent, therefore for each viewpoint a different drawing order should be found.

Hidden Surface Removal

- OpenGL solves this problem by holding a depth-map called "Z-Buffer". This buffer holds the depths (distances on the Z direction) of each pixel drawn on the frame buffer. Then, when a new object is painted, a depth test determines for each pixel if it should be updated or not.
- To turn this mechanism on, the following steps should be taken:
  - gluInitDisplayMode(GLUT_DEPTH | ...) ;
  - glEnable(GL_DEPTH_TEST) ;
  - glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT) ;

OpenGL's Shading models

- **glShadeModel**(GLenum mode) - Sets the shading model which can be either GL_SMOOTH or GL_FLAT.
- FLAT shading model:
  - The color of one particular vertex is used to render the entire primitive.

- SMOOTH shading model:
  - The color used to render the primitive is a result of color interpolation between the primitive's vertices' colors.

OpenGL's Lighting model

- OpenGL supports a lighting model which includes:
  - Multiple light sources.
  - Spots, Points and directional light sources.
  - Different material properties.
  - Surface normals.
- The lighting model uses above factor to estimate vertices colors.
- **glEnable**(GL_LIGHTING) - will enable OpenGL's lighting model. Once this is enables the **glColor** doesn't effects the vertices colors (material color attributes are now responsible for the objects own color).

OpenGL's Lighting model

- The Lighting Model:

$$\text{vertex color} = emission_{material} + ambient_{light\_model} * ambient_{material} + \sum_{\substack{lights \\ sources}} \text{attenuation\_factor} * \text{spotlight\_effect} * (ambient + diffuse + specular)_i$$

- **glEnable**(GL_LIGHTX) - X = 0...7 - enable light source.
- **glLight**{if}(GLenum lightnum, GLenum pname, TYPE param)
- **glLight**{if}v(GLenum lightnum, GLenum pname, TYPE *param) - sets the property *pname* of light *lightnum* to be *param*:
  - GL_AMBIENT (R,G,B,A) RGBA values.
  - GL_DIFFUSE (R,G,B,A) RGBA values.
  - GL_SPECULAR (R,G,B,A) RGBA values.
  - GL_POSITION (X,Y,Z,W) Position in space.

OpenGL's Lighting model

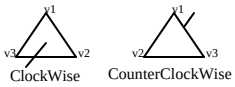
GL_SPOT_DIRECTION	(x,y,z)		Direction vector
GL_SPOT_EXPONENT	e		spotlight exponent
GL_SPOT_CUTOFF	ang		spotlight cutoff angle
GL_CONSTANT_ATTENUATION		kc	Const. Attn. Factor
GL_LINEAR_ATTENUATION		kl	Linear Attn. Factor
GL_CONSTANT_ATTENUATION		kq	Quadric Attn. Factor

- **glMaterial**{if}(GLenum face, GLenum pname, TYPE param) **glMaterial**{if}**v(**GLenum face, GLenum pname, TYPE *param) - sets the vertex material property *pname* to be *param* on *face*.

GL_AMBIENT	(R,G,B,A)	RGBA color
GL_DIFFUSE	(R,G,B,A)	RGBA color
GL_SPECULAR	(R,G,B,A)	RGBA color
GL_SHININESS	s	specular exponent
GL_EMISSION	e	emissive color of vertex.

OpenGL's Lighting model

- Argument face is:
  - GL_FRONT
  - GL_BACK
  - GL_FRONT_AND_BACK
- This determines on which face of the primitive to apply the **glMaterial**.
- Primitives' sides are determined by the order of their vertices:



- **glFrontFace**(GLenum mode) - determines which order will define the front face. Arguments: GL_CCW, GL_CW (order on the projected windows)

OpenGL's Lighting model

$$\text{vertex color} = emission_{material} + ambient_{light\_model} * ambient_{material} + \sum_{\substack{lights \\ sources}} \text{attenuation\_factor} * \text{spotlight\_effect} * (ambient + diffuse + specular)_i$$

- Emission term: The material emissive light value (GL_EMISSION)
- Attenuation Factor:

d – distance between light source and vertex
k_c – GL_CONSTANT_ATTENUATION
k_l – GL_LINEAR_ATTENUATION
k_q – GL_QUADRIC_ATTENUATION

$$\frac{1}{k_c + k_l d + k_q d^2}$$

- Spotlight Effect:

$\mathbf{v} = (v_x, v_y, v_z)$ is the unit vector that points from the spotlight to the vertex.
$\mathbf{d} = (d_x, d_y, d_z)$ is the spots direction (GL_SPOT_DIRECTION)

$$= \begin{cases} 1 & \text{If the light is not a spotlight} \\ 0 & \text{If the vertex is out of the light cone} \\ \max\{\mathbf{v} \cdot \mathbf{d}, 0\}^{GL\_SPOT\_EXPONENT} & otherwise \end{cases}$$

OpenGL's Lighting model

- Ambient Term: light's ambient color multiplied by the materials (GL_AMBIENT): $ambient_{material} * ambient_{light}$
- Note: The multiplication above is individually for the R,G,B and A color components.
- Diffuse Term: The direct light that reaches the vertex. It is directional depended: $(\max\{\mathbf{L} \cdot \mathbf{n}, 0\}) \cdot diffuse_{material} * diffuse_{light}$
  $\mathbf{L} = (L_x, L_y, L_z)$ Is the unit vector that points from the vertex to the light position (GL_POSITION)
  $\mathbf{n} = (n_x, n_y, n_z)$ Is the unit normal vector at the vertex.
- **glNormal3**{bsidf}(TYPE nx, TYPE ny, TYPE nz) **glNormal3**{bsidf}**v(**TYPE *v) - Defines the current normal vector. Next time **glVertex** will be called, the current normal will be assigned to the vertex.
- Note: OpenGL - can receive non-unit normals and normalize them if **glEnable**(GL_NORMALIZE) is called.

OpenGL's Lighting model

- Specular Term: Is calculated as follows:

$\mathbf{n}$ – vertex normal vactor
$\mathbf{s}$ = normalized sum of the two unit vectors pointing from vertex to light position and from vertex to viewer position

$$(\max\{\mathbf{s} \cdot \mathbf{n}, 0\})^{shinines}$$

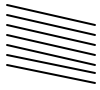
- All together gives:

$$\text{vertex color} = emission_{material} + \sum_{vertices} \left(\frac{1}{k_c + k_l d + k_q d^2}\right)_i * spotlight effect_i * ambient_{material} * (ambient_{light})_i + (\max\{\mathbf{L}_i \cdot \mathbf{n}, 0\}) * diffuse_{material} * (diffuse_{light})_i + (\max\{\mathbf{s}_i \cdot \mathbf{n}, 0\})^{shininess_i} * specular_{material} * (specular_{light})_i +$$

OpenGL's Lighting model

- Point:

- Distant:



- Spot:

OpenGL's Lighting model

- Light sources types:
  - Point - Light coming from a single point in 3D space. This is the default light source.
  - Distant/directional - Light coming from a direction. Light Rays a parallel. This is specified by putting 0 zero in the fourth coordinate of the GL_POSITION attribute of the light source. (Remember what vectors of the type (x,y,z,0) mean in projective spaces).
  - Spot - Light coming from a point same as in Point lights, but has a direction around which light intensity drops. Specified by setting GL_SPOT_CUTOFF to be less than 180.

OpenGL's Lighting model

- Lights Position: we can think of three relations between the light position and the objects position:
  - A light position that remains fixed.
  - A light that moves around a stationary object.
  - A light that moves along with the viewpoint.
- A important fact needed in order to achieve the above cases is:
- When **glLight** is called to specify the position or the direction of a light source, the position or direction is transformed by the current modelview matrix.

OpenGL's Lighting model

- A light position that remains fixed:
- In this case lights (pos/dirs) should go only through the viewing transformations.
- This means that we will specify the light pos/dirs After view trans. Has been defined but before model trans has been defined:
  - glMatrixMode(GL_MODELVIEW);
  - glLookAt / any view transform
  - **glLight**(GL_POSITION/DIRECTION)
  - glRotate/glTranslate/glScale - on the objects.

OpenGL's Lighting model

- A light that moves around a stationary object:
- In this case lights (pos/dirs) should go only through the viewing and model transformations, while the objects will go only through view transformations.
- This means that we will specify the light pos/dirs After view and model trans. But the objects won't go through the model trans:
  - glMatrixMode(GL_MODELVIEW);
  - glLookAt / any view transform
  - glPushMatrix();
  - glTranslate() / glRotate() (for light pos and dir)
  - **glLight**(GL_POSITION/DIRECTION)
  - glPopMatrix();
  - glBegin() ... glEnd() // Draw the objects.

OpenGL's Lighting model

- A light that moves along with the viewpoint
- In this case lights (pos/dirs) should not go through any transformations. They stay in the camera (eyes) coordinate system.
- First we call **glLight** when modelview is identity*. Then we specify view and model transformations for the objects:
  - glMatrixMode(GL_MODELVIEW);
  - glLoadIdentity(); (*or any location / rotation in eye coord. sys.)
  - **glLight**(GL_POSITION/DIRECTION)
  - glLookAt / any view transform
  - glTranslate() / glRotate() (for objects)
  - glBegin() ... glEnd() // Draw the objects.