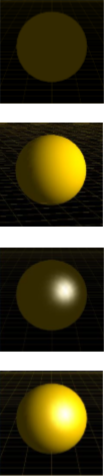
**Illumination**

**Light**

****In **Initialize**()

* **glEnable**(GL\_LIGHTING); **glEnable**(GL\_LIGHT0)

In **display**()

* float light\_pos[] = {20, 20, 0, 1}
* **glLightfv**(GL\_LIGHT0, GL\_POSITION, light\_pos);

lights position = imaginary point (can be transformed, no 3D primitive created)

**Three Types of Reflections**

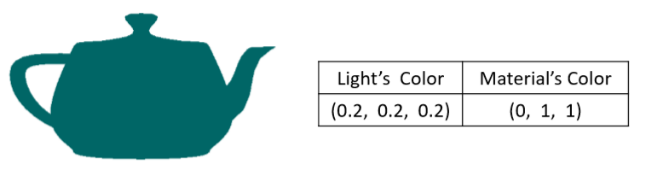
**Ambient reflection** – base level brightness of scene

**Diffuse reflection** – most common, intensity changes when angle between the light’s direction and the surface normal vector

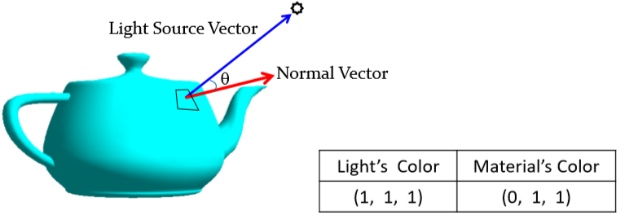
**Specular**  - view dependant, mirror like

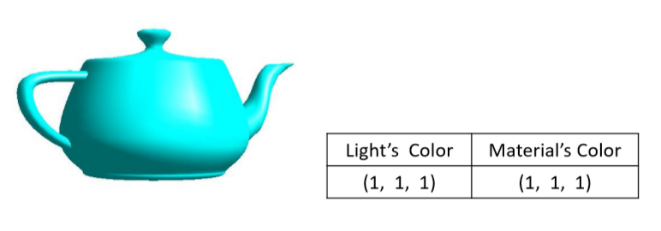
**Ambient+Diffuse+Specular** = net reflection of all of the above

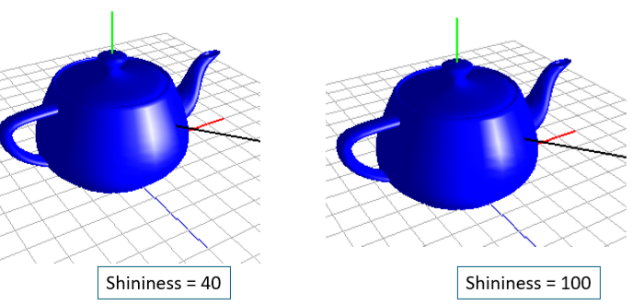
**Ambient Reflection**

* Constant in the scene.
* Typically, a low intensity grey e.g. (0.2, 0.2, 0.2)
* Ambient light interacts with the material colour to provide a uniform dark shade of the material colour

**Diffuse Reflection**

* ****Intensity = orientation of the surface relative to light’s direction.
* Intensity reduces = when the angle between the light’s direction and the surface normal vector Increases
* angle = 0; diffuse reflection = max
* angle >= 90o; diffuse reflection = min

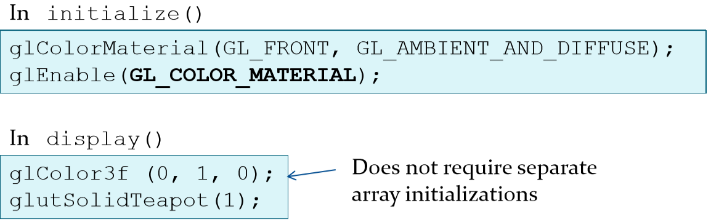
**Specular Reflection**

* Polished surfaces
* Doesn’t work with cubes us diffuse instead
* Reflection is directional and view dependent
* ****Material = white for bright highlight

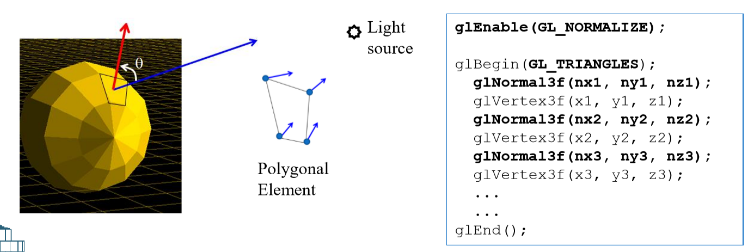
**“Shininess” of Specular Reflection**

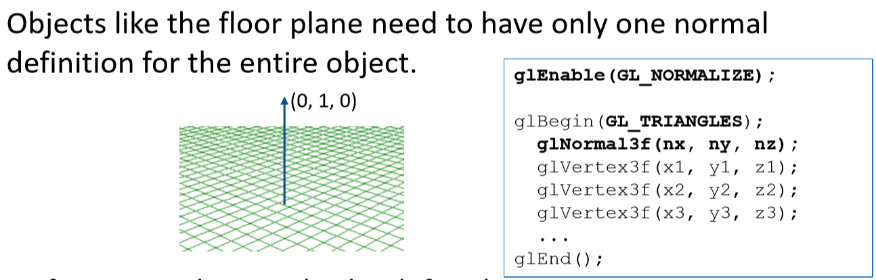
* Width of specular highlight = GL\_SHININESS
* Shininess Increase = less concentrated
* Shininess Decrease = more concentrated

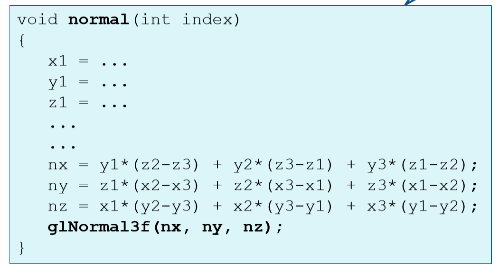
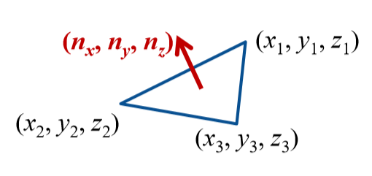
**Material Colour**

* When lighting enabled = **glColorX(…)** is ignored and **glMaterialfv(…)** used instead
* For lighting use **glMaterialfv(…)**
* This leads to cumbersome code
* If we want to use **glColor3f(...)** we do ->->->

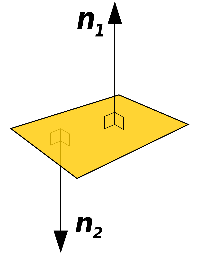
**Per-Vertex Lighting**

* Lighting occurs for only vertices of polygonal objects
* Interior filled = done by Interpolation
* Lighting calculations = require normal vector, **glNormal3f**(x1, y1, z1);

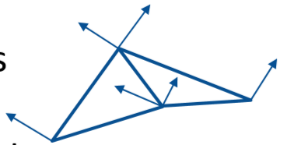
**Surface Normal Vectors**

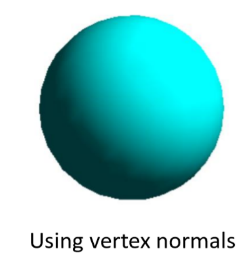
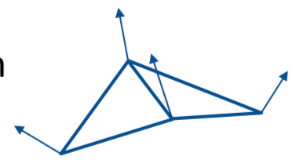
* Each vertex needs a normal apart from the floor(because it is flat/has one face/surface)

Example of a Surface Normal Vector



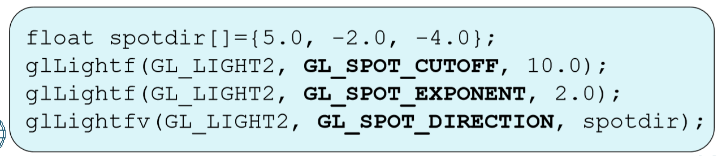
Normal vector= A **perpendicular** **vector** to another object, such as a surface or plane. Often we refer to a unit **normal vector** n, which is a **normal vector** of length one.

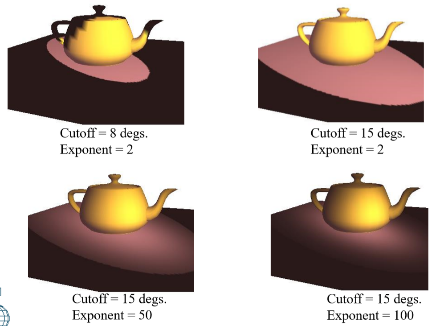
**Face normal vs Vertex normal**

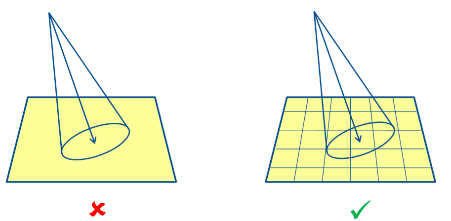
* Face-based normal = Each vertex of a triangle has the same normal vector. This gives a nearly constant colour for each face and reveals the polygonal structure of the object.
* Per-vertex normal = Each vertex of a triangle has their own distinct normal vector. This gives a smooth variation of shades across the surface.

**Spot Lights**

* Default = Omni-directional light (emit light in all directions)

Convert to spot light:

* A cutoff angle = half cone angle of the spotlight.
* A spot direction. This vector specifies the cone’s axis.
* A spot exponent. This specifies how fast the intensity drops off as a vertex is moved from the centre of the spotlight towards its edge.

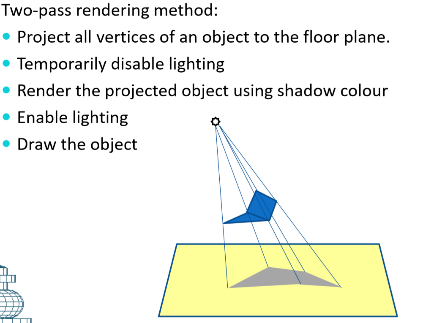
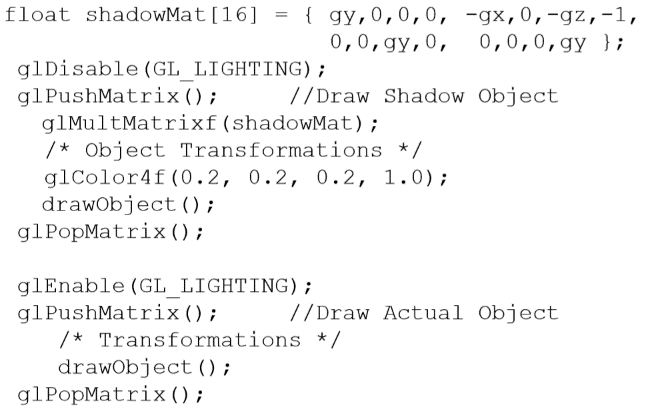
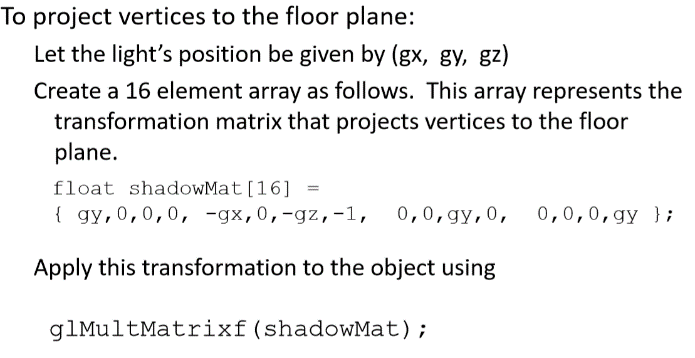
**OpenGL Lighting**

Plane has to be subdivided into small quads = lighting calculations are only done at the vertices of every polygon.

**Shadows**

Two Types:

* Backface shadows: A shadow on an object’s surface that is oriented away from light.
* Projected shadows/cast shadows: Shadows cast by a part of an object’s surface on either the same or a different object.

**Planar Shadows on Floor Plane (y=0)**