Lighting

Francesc Sebé

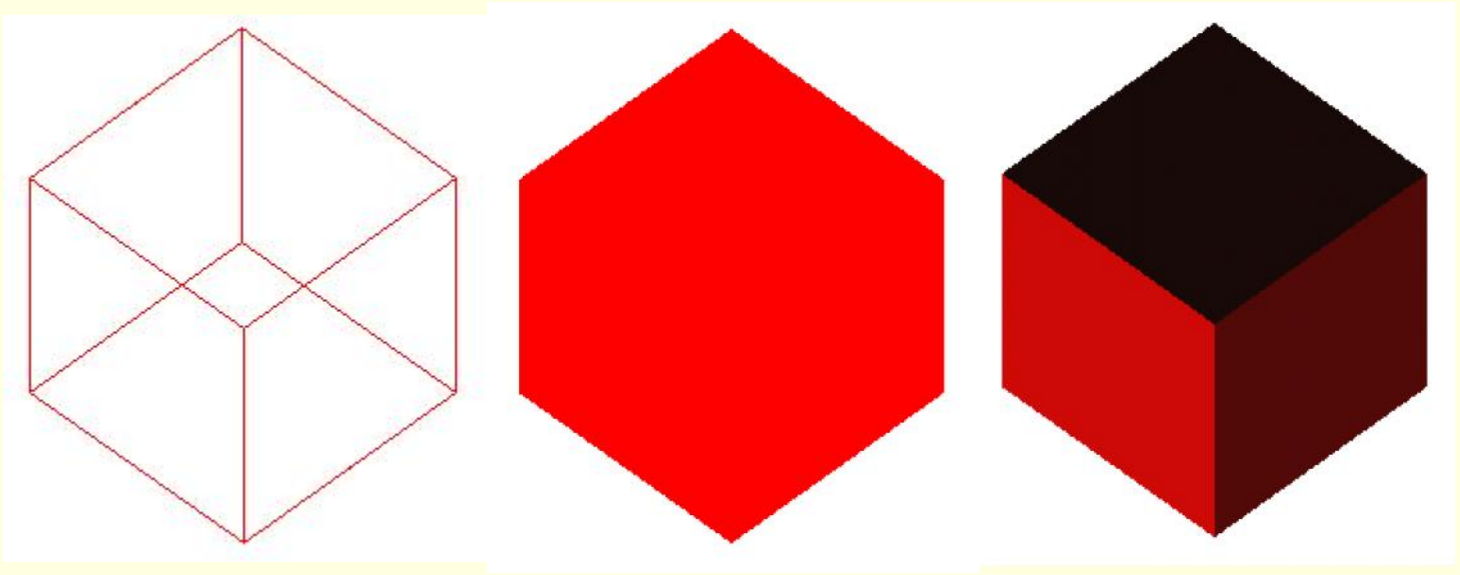
‘Computació gràfica i multimèdia’

Escola Politècnica Superior

Universitat de Lleida

Introduction

- A 3D cube:

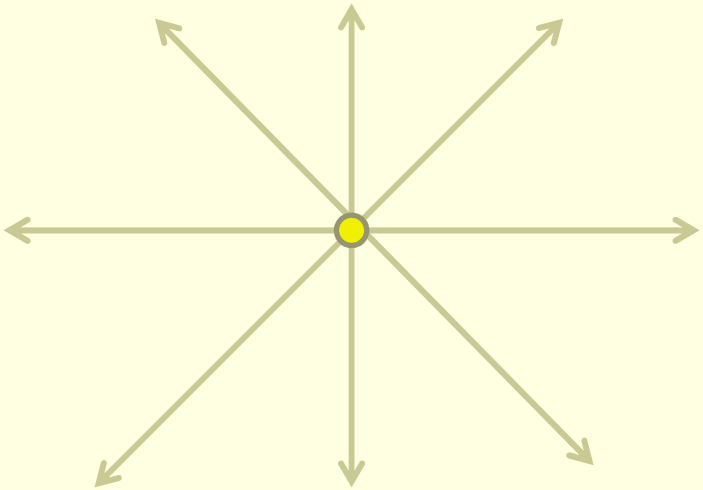


OpenGL lighting

- First of all:
  - glEnable(GL_LIGHTING);
- After that:
  - Color given by **glColor3f** is not considered.
  - Color **at vertices** computed from
    - Its material
    - The amount of received light

Point light sources

- A single point emitting radiant energy with a single color.
- It is given by its position and color of the emitted light.

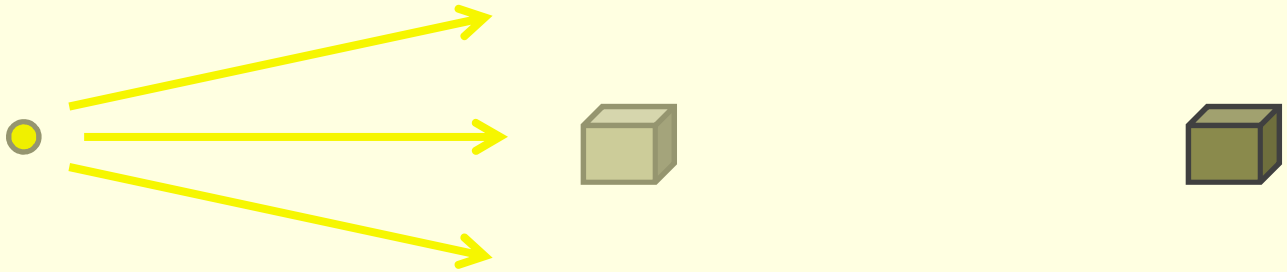


Infinitely distant light source

- Light source located very far from the scene
  - It illuminates from only one direction
  - Rays are parallel

Radial intensity attenuation

- Objects located near a point light source receive a higher intensity



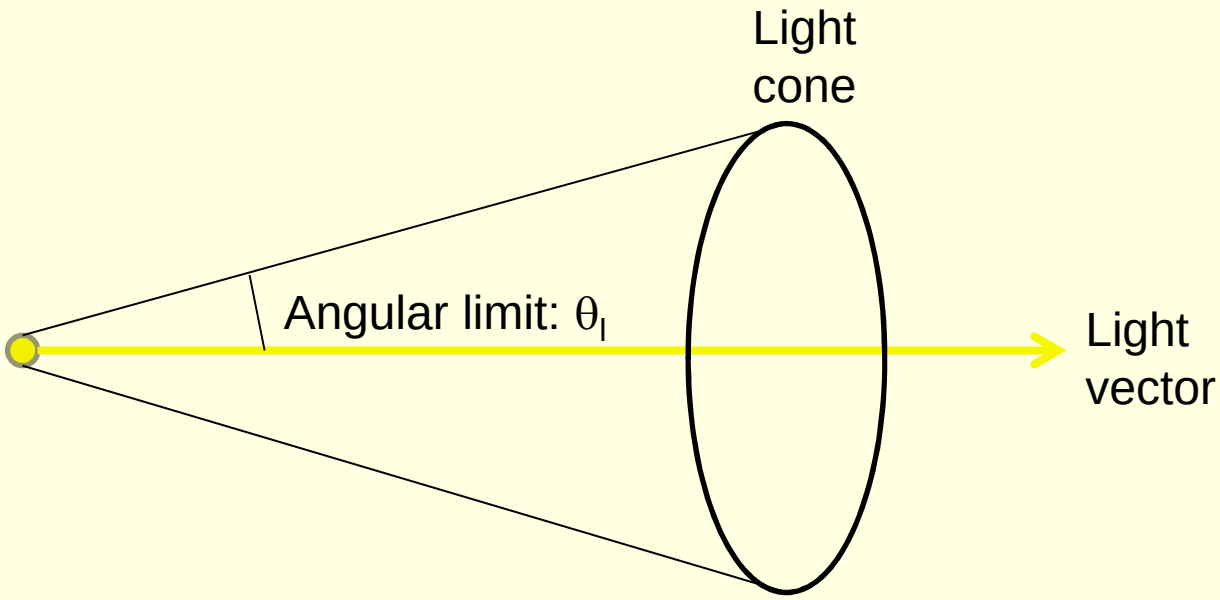
Radial intensity attenuation

- Attenuation is modelled with an inverse quadratic function:

$$f_{rad}(d_l) = \frac{1}{a_0 + a_1 d_l + a_2 d_l^2}$$

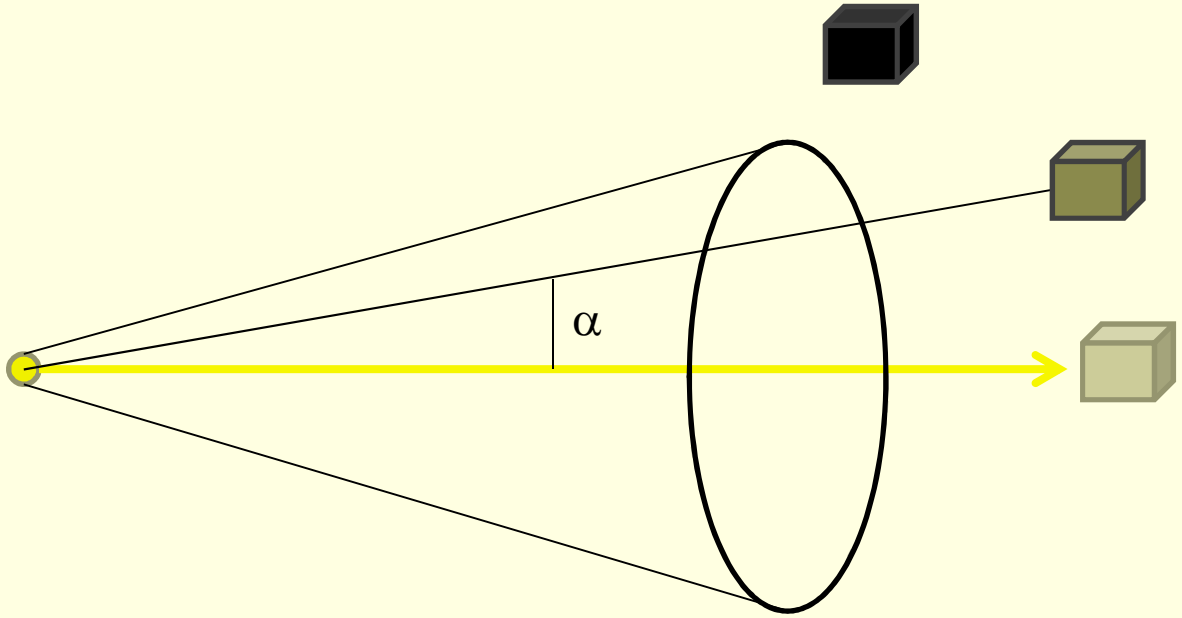
- Value a_0 prevents a very large intensity at points near the light source.
- Values a_0, a_1, a_2 have to be tuned.
- Infinitelly distant light sources are not attenuated.

Directional light sources



Angular intensity attenuation

- Only objects in the light cone receive illumination
- Light intensity decreases with α.



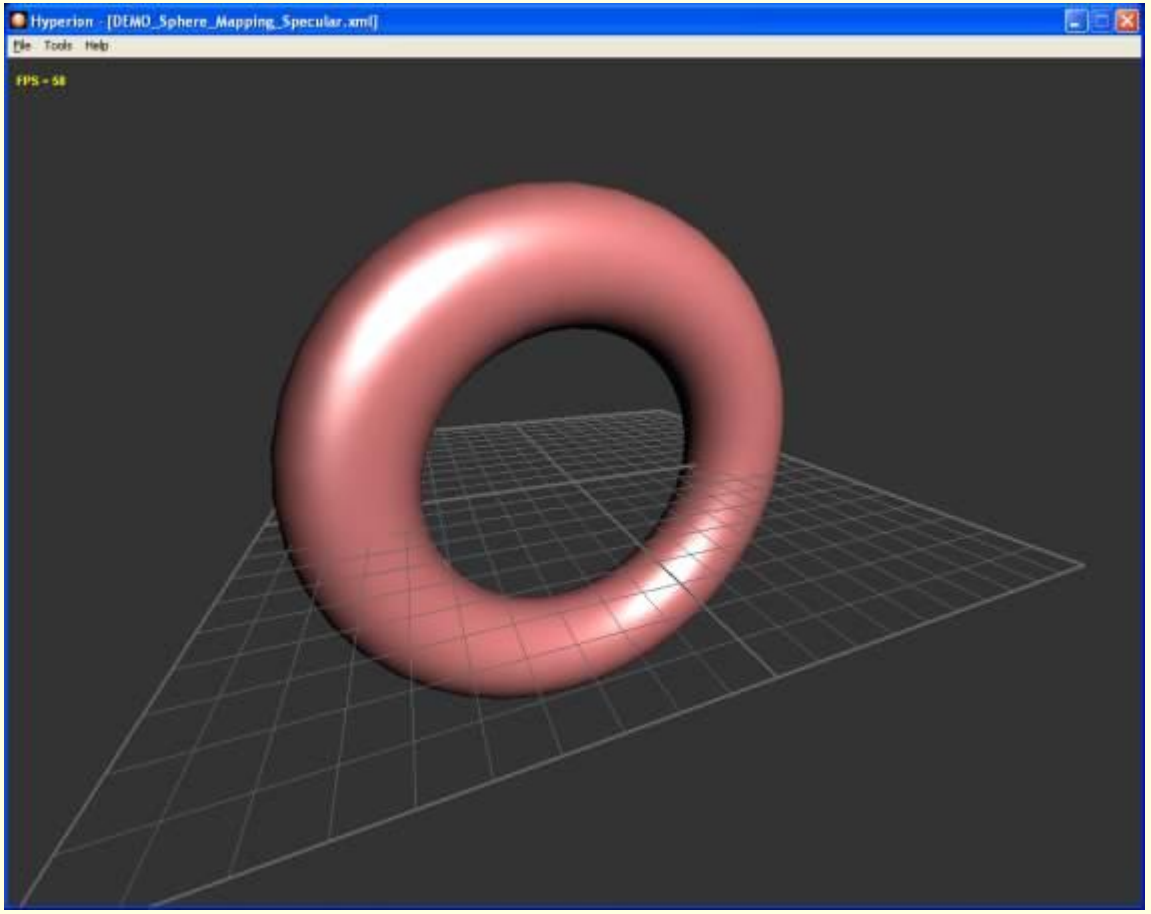
Angular intensity attenuation

$$f_{ang}(\sim) = \begin{cases} \cos^{a_l} \sim & ,\sim \leq \bar{\ }_l \\ 0 & ,otherwise \end{cases}$$

Background light

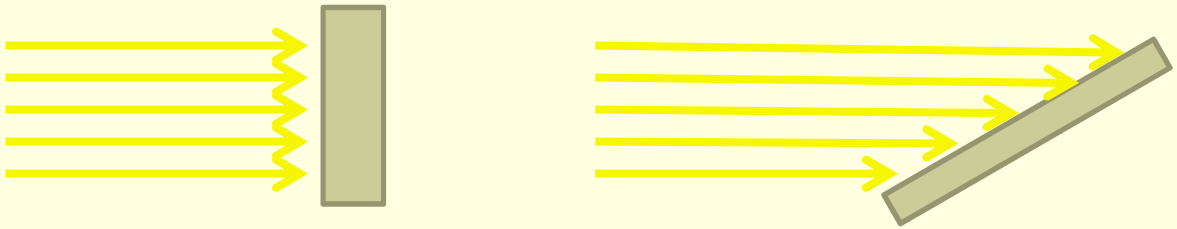
- A surface not directly exposed to a light source may still be visible due to background light.
  - This light reflects diffusely
- It is modeled as a light reaching all the vertices

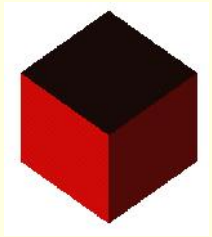
Specular and diffuse reflection



Diffuse reflection

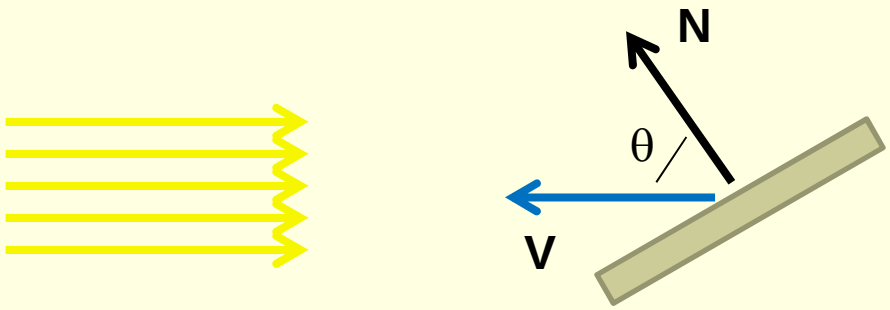
- The amount of light received by a surface depends on its orientation relative to the light source





Diffuse reflection

- Considering:
  - N: unit normal vector at a surface position
  - V: unit vector pointing to the light source
- Diffuse light intensity is proportional to $\cos\theta$ (if $|\theta|<90^{o}$).
- If $|\theta|>90^{o} \rightarrow$ No diffuse light



Location and color

- Glfloat vec[4];
- glLightfv (GL_LIGHT1, **GL_POSITION**, vec);
  - If vec[3]==0.0 → Very distant light
- glLightfv (GL_LIGHT1, **GL_AMBIENT**, vec);
- glLightfv (GL_LIGHT1, **GL_DIFFUSE**, vec);

Radial-intensity attenuation

- glLightfv (GL_LIGHT1, **GL_CONSTANT_ATTENUATION**, value);
- glLightfv (GL_LIGHT1, **GL_LINEAR_ATTENUATION**, value);
- glLightf (GL_LIGHT1, **GL_QUADRATIC_ATTENUATION**, value);

Directional light sources

- Glfloat dir[3];
- glLightfv (GL_LIGHT1, **GL_SPOT_DIRECTION**, dir);
- glLightf (GL_LIGHT1, **GL_SPOT_CUTOFF**, degrees);
- glLightf (GL_LIGHT1, **GL_SPOT_EXPONENT**, value); //1 … 128

Light enabling

- After all, glEnable (**GL_LIGHT1**);
- You can create multiple light sources:
  - GL_LIGHT2
  - GL_LIGHT3
  - (…)

Material definition

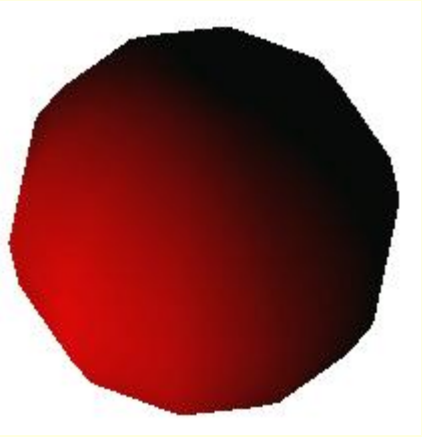
- When lighting is enabled, polygons are coloured according to their behaviour with respect to light reflection.
- You have to define their material properties.

Diffuse reflection coefficients

- Glfloat vec[4];
- glMaterialfv(GL_FRONT_AND_BACK, **GL_AMBIENT_AND_DIFFUSE**, vec);
- Diffuse reflection coefficients for each RGB color component
- A green object would receive
  - vec={**0.0, 1.0, 0.0**, 1.0}; //RGB

Surface approximation

- Surfaces are approximated by a polygon mesh
  - Different options for rendering

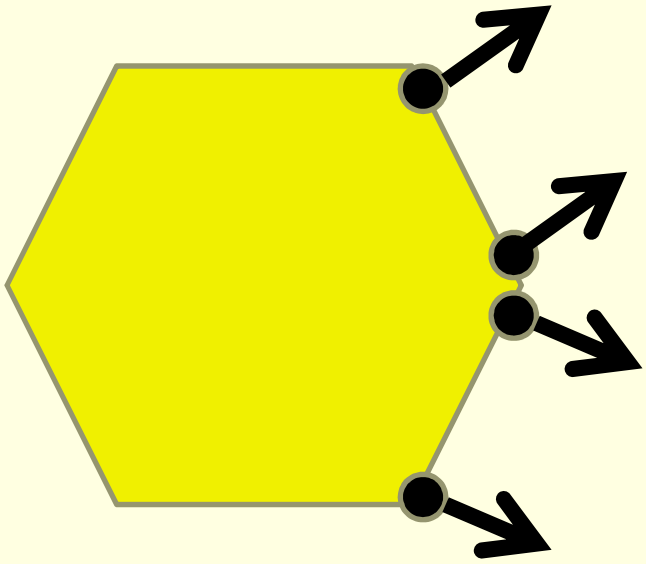


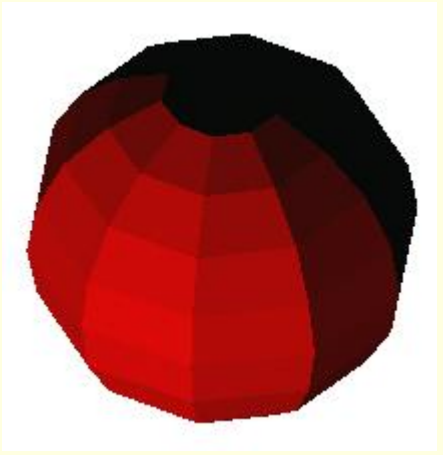
Surfaces with OpenGL

- OpenGL computes polygon color at polygon vertices
  - The color at the remaining points is interpolated

Surfaces with OpenGL

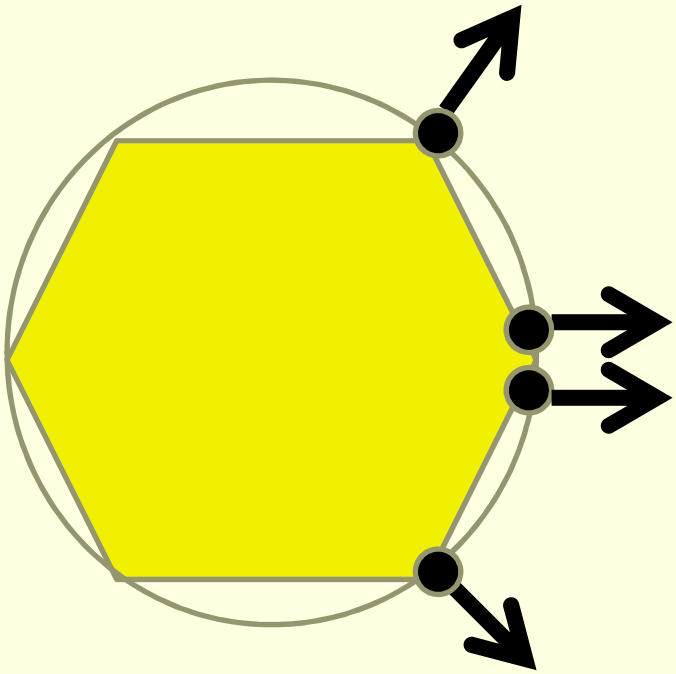
- If you are employing lighting, you should specify the normal vector at polygon vertices.
  - The same normal vector at each vertex produces the following result:

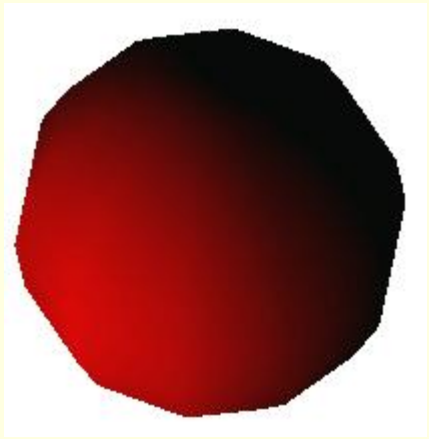




Surfaces with OpenGL

- If each vertex is assigned the normal vector it would have if it was the point of the surface we are approximating, the result is more realistic:





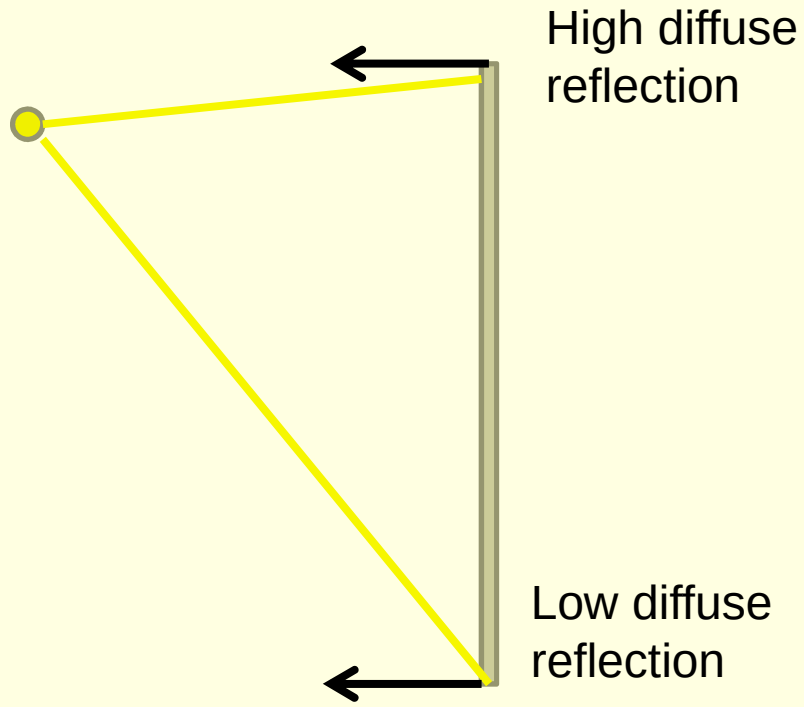
Surfaces with OpenGL

- **glShadeModel(GL_SMOOTH)**
- glBegin(GL_POLYGON);
  - glNormal3f(n_{x1},n_{y1},n_{z1}); // Unitary
  - glVertex3i(x_1,y_1,z_1);
  - (…)
  - glNormal3f(n_{x2},n_{y2},n_{z2});
  - glVertex3i(x_2,y_2,z_2);
- glEnd();

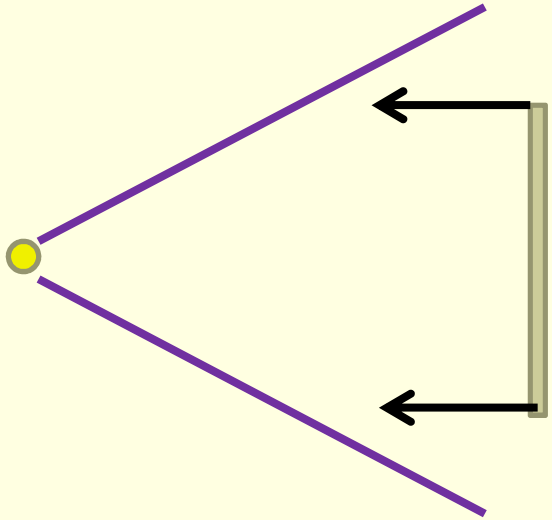
Surfaces with OpenGL

- With **glShadeModel(GL_FLAT)**,
  - OpenGL computes the color at just one vertex.
  - All points of the polygon receive the same color.

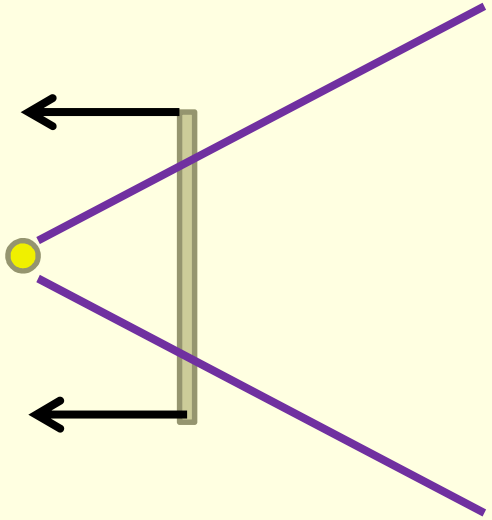
Color is computed at vertices



Be careful with directional light sources



Polygon vertices inside the light cone



Polygon vertices OUTSIDE the light cone

The polygon receives NO lighting