Computer Graphics

(Shading Algorithms)

Lecture 007

Shading (Gouraud and Phong)

- The first quality shading in CG was developed by H. Gouraud in 1971. In 1975 Phong improved on Gouraud's model and became the standard in mainstream 3D graphics.

- Phong shading has remained ubiquitous because it enables reality to be mimicked to an acceptable level at a reasonable cost.

- There are two separate considerations to shading the pixels onto which a polygon projects,

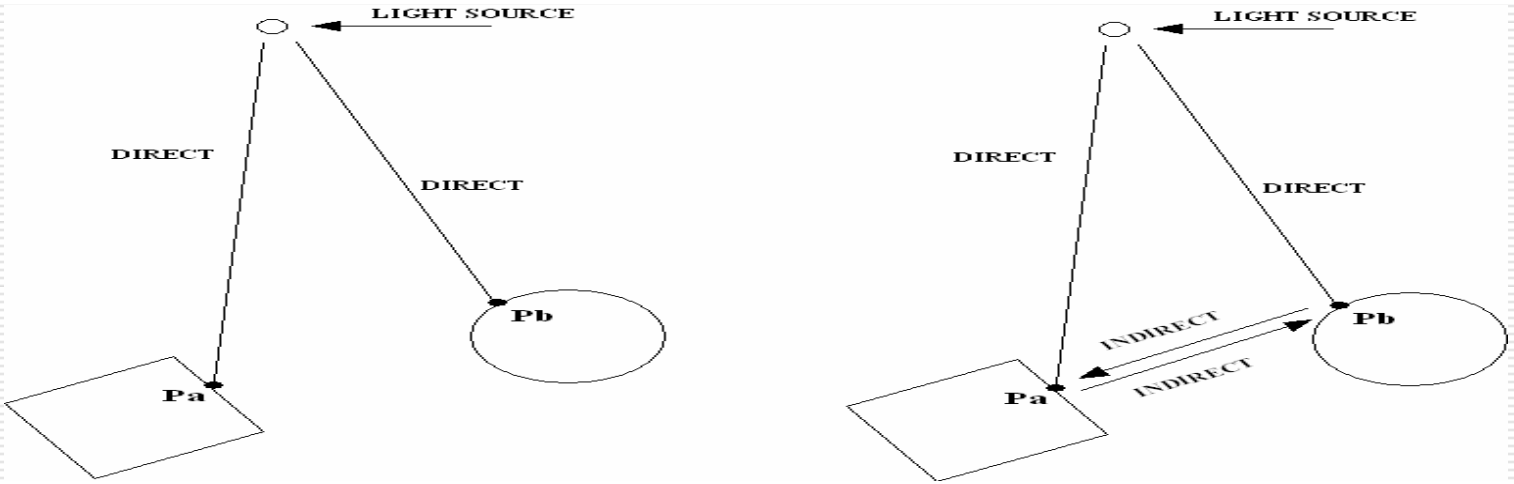
  - Local reflection models, i.e. a theoretical framework that gives us a way of calculating the light reflected at any point on the surface of an object,

  - Shading algorithms, i.e. how using this framework we can calculate the light intensity at the pixels onto which the polygon projects.

Local Reflection Model

- A local reflection model enables the calculation of the reflected light intensity from a point on the surface of an object.

- One common model evaluates the intensity of the reflected light as a function of the orientation of the surface at the point of interest with respect to the position of a light source and surface properties.



- Note that we refer to such a model as a local reflection model because it only considers direct illumination. Interaction with other objects that results in shadows and inter-reflection are not taken into account by local reflection models.

Local Reflection Model (cont.)

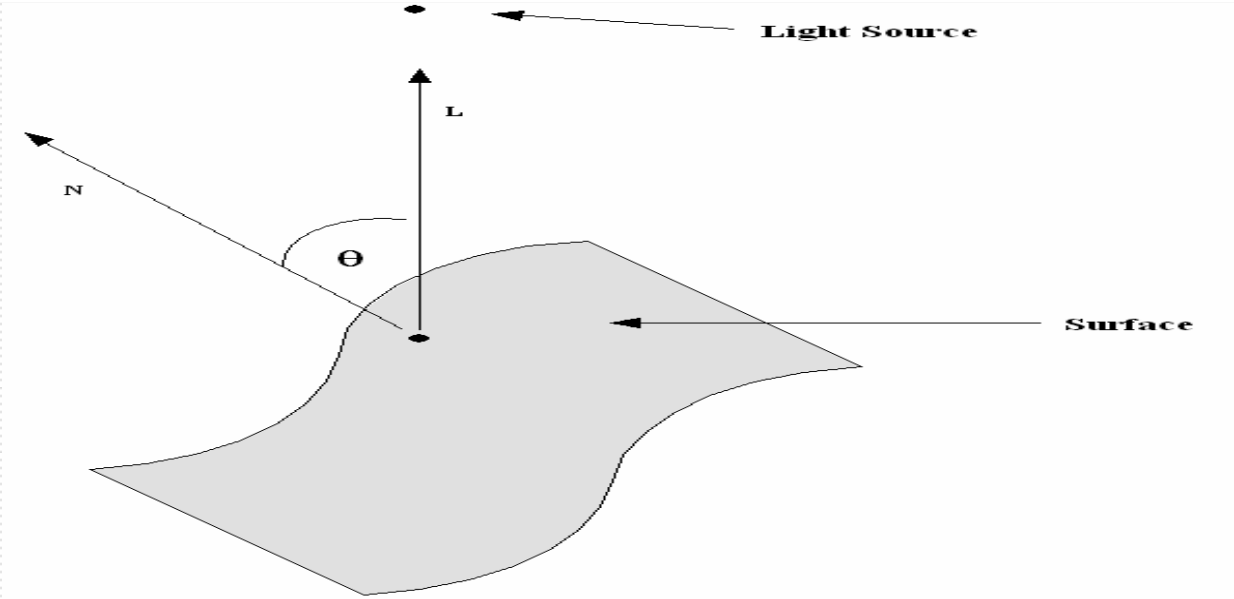
- The physical reflection phenomena that the local reflection model simulates are:
  - Perfect specular reflection
    - This occurs when incident light is reflected, without diverging, in the ‘mirror’ direction.
  - Imperfect specular reflection
    - This occurs when a thin beam of light strikes an imperfect mirror. In this case the surface would be physically rough.
  - Perfect diffuse reflection
    - This does not occur in practice but is used in ray tracing models simply because calculating interaction due to imperfect specular reflection is too expensive. A perfect diffuse surface reflects the light equally in all directions. Such a surface is usually called matte.

Phong Reflection Model

- The Phong reflection model considers the reflection from a surface to consist of three components linearly combined:
- Reflected light = ambient light + diffuse component + specular component
- Ambient light
  - This light source is a constant and simulates global or indirect illumination. This term is necessary because parts of a surface that cannot 'see' the light source, but which can be seen by the viewer (camera), need to be lit. Otherwise they would be rendered as black. Simply adding a constant side-steps the complexity of indirect or global illumination calculations.
- The combination of the diffuse and specular components determine the properties of the surface of the polygon to be rendered. For example, varnished wood has both a diffuse and specular components. Thus the physical nature of a surface is simulated by controlling the proportion of the diffuse a\to specular reflection and we have the reflected light:
- $I = k_a l_a + k_d l_d + k_s l_s$, where $k_a + k_d + k_s = 1$

Phong Reflection Model (Diffuse Component)

- The diffuse intensity is calculated as follows:
  - $I_d = I_i \cos \theta$
  - Where,
    - I_i is the intensity of the incident light
    - θ is the angle between the surface normal at the point of interest and the direction of the light source
    - In vector notation we get $I_d = I_i (L.N)$



Phong Reflection Model (Specular Component)

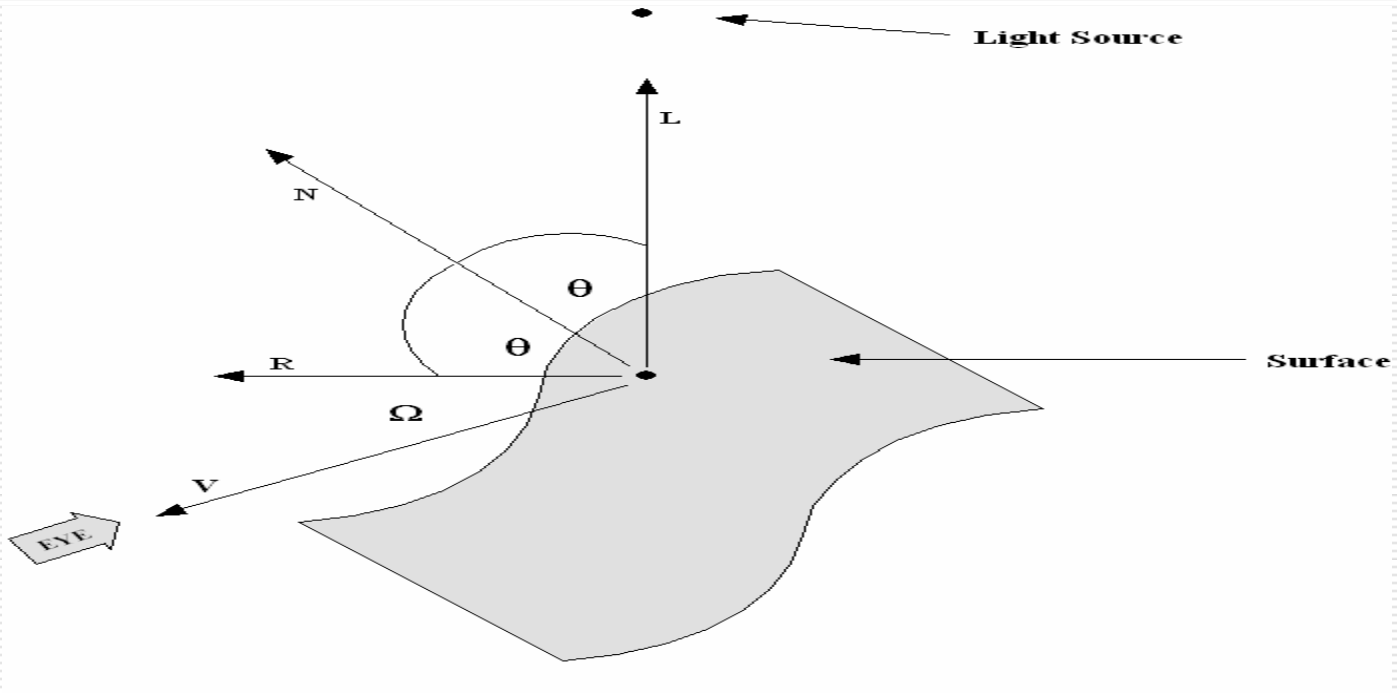
- The specular reflection consists of an image of the light source 'smeared' across an area of the surface resulting in what is commonly known as a highlight. A highlight is only seen by a viewer if the viewing direction is near to a mirror direction.

- Specular reflection is simulated using the following equation:
  - $I_s = I_i \cos^n \Omega$, where
    - Ω is the angle between the viewing direction and the mirror direction R
    - n is an index that simulates the degree of imperfection of a surface

- When n is set to infinity the surface is a perfect mirror – i.e. all reflected light emerges along the mirror direction. For other values of n an imperfect specular reflector is simulated.

- In vector notation we have: $I_s = I_i (R.V)^n$

Phong Reflection Model (Specular Component Geometry)



- Brining all components together we get the following:
  - $I = k_a I_a + I_i \ (k_d(L.N) + k_s(R.V)^n)$

Interpolative Shading Techniques

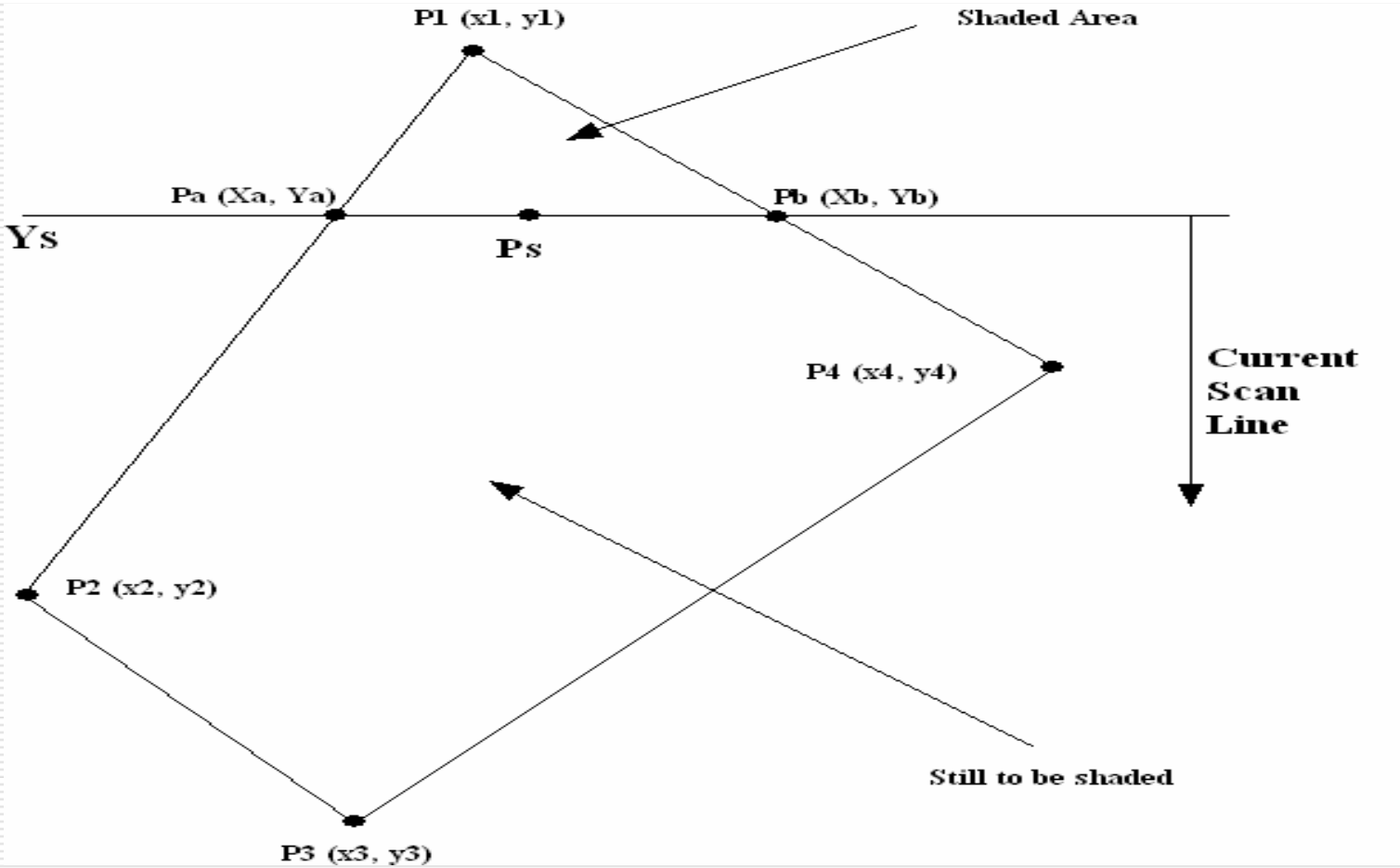
- Now that we have a way of calculating light intensity at a point, we can consider how to apply such a model to a polygon and calculate the light intensity over its surface.

- We will consider two classic techniques – Gouraud and Phong shading.

- Both techniques have been developed to interpolate information efficiently across the face of a polygon and to diminish the visibility of the polygon edges in the final shaded image.

- Information is interpolated from values at the vertices of a polygon. Phong interpolation gives the more accurate highlights. However Gouraud shading is considerably cheaper (in terms of computational requirements).

Bilinear Interpolation



Bilinear Interpolation (cont.)

- Bilinear interpolation is the foundation of the efficiency of this kind of shading.

- Interpolation proceeds by moving a scan line down through the pixel set representing the polygon and obtaining start and end values for a scan line by interpolating between the appropriate pair of vertex properties. Interpolation along a scan line then yields a value for the property at each pixel. The equations are:
  - $Pa = 1/(y1-y2) * [p1(ys-y2) + p2(y1-ys)]$
  - $Pb = 1/(y1-y4) * [p1(ys-y4) + p4(y1-ys)]$
  - $Ps = 1/(xb-xa) * [pa(xb-xs) + pb(xs-xa)]$

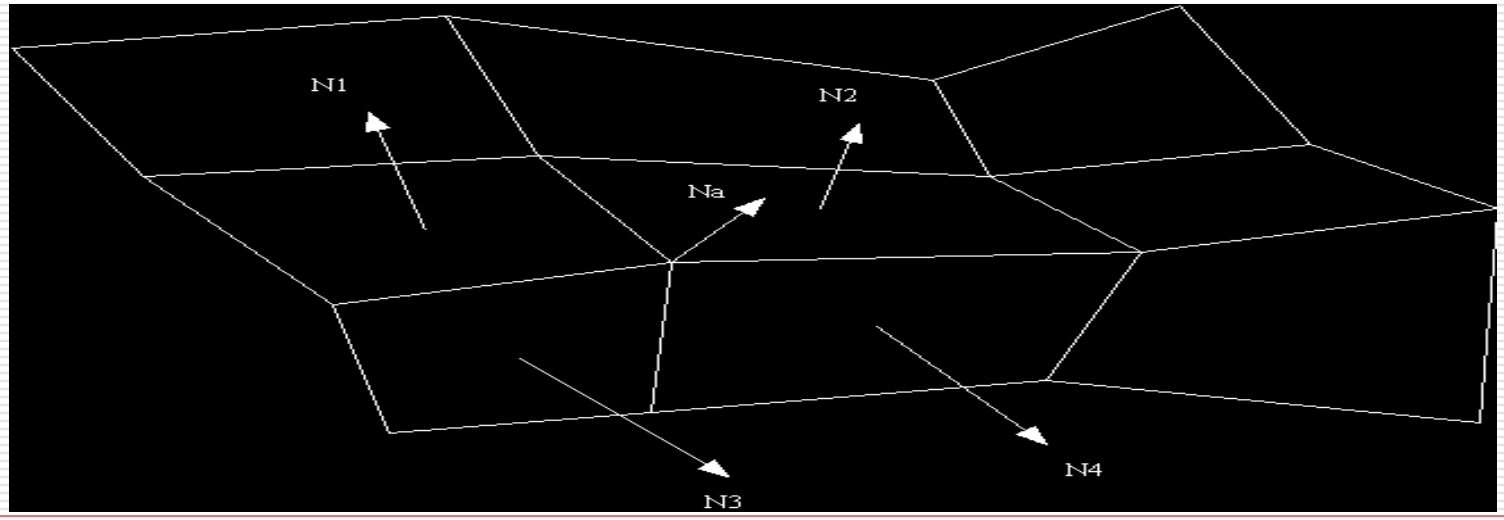
- Note that these equations would normally be implemented using an incremental form. The final equation, for example, would be something like Ps := Ps + Δp with the value Δp calculated once per scan line.

Gouraud Shading

- In Gouraud shading light intensity is calculated (using the model just discussed) at the vertices of the polygon and then interpolates between these intensities to find values at projected pixels.

- The property P (for bilinear interpolation) is set to the vertex intensity.

- Note that vertex normals are different from surface normals. If we consider a polygon in isolation then, of course, the vertex normals are all parallel. However in Gouraud shading we use vertex normals, which help in reducing the visibility of polygon edges.



Gouraud Shading (cont.)

- N_a is used to calculate an intensity at vertex A that is common to all the polygons that share vertex A

- The interpolation equations are implemented as incremental calculations. Suppose we define Δx to be the incremental distance along a scan line then ΔI, the change in intensity from one pixel to the next, is:

  - $\Delta I_s = \Delta x / x_b - x_a \quad * \quad (I_b - I_a)$

  - Hence:

    - $I_{s,n} = I_{s,n-1} + \Delta I_s$

- Because the intensity is calculated at vertices only the method cannot adequately deal with highlights and this is its major disadvantage. We deal with this problem in Phong shading.

Phong Shading

- Instead of interpolating only intensity values, this method interpolates also vertex normals.

- In Phong shading, we interpolate vertex normals across the polygon interior and calculate for each polygon pixel projection an interpolated normal. This interpolated normal is then used in the shading equation which is a applied for every pixel projection.

- The price that we pay for this improved model is efficiency. Vector interpolation consumes in general three times the cost of intensity interpolation.

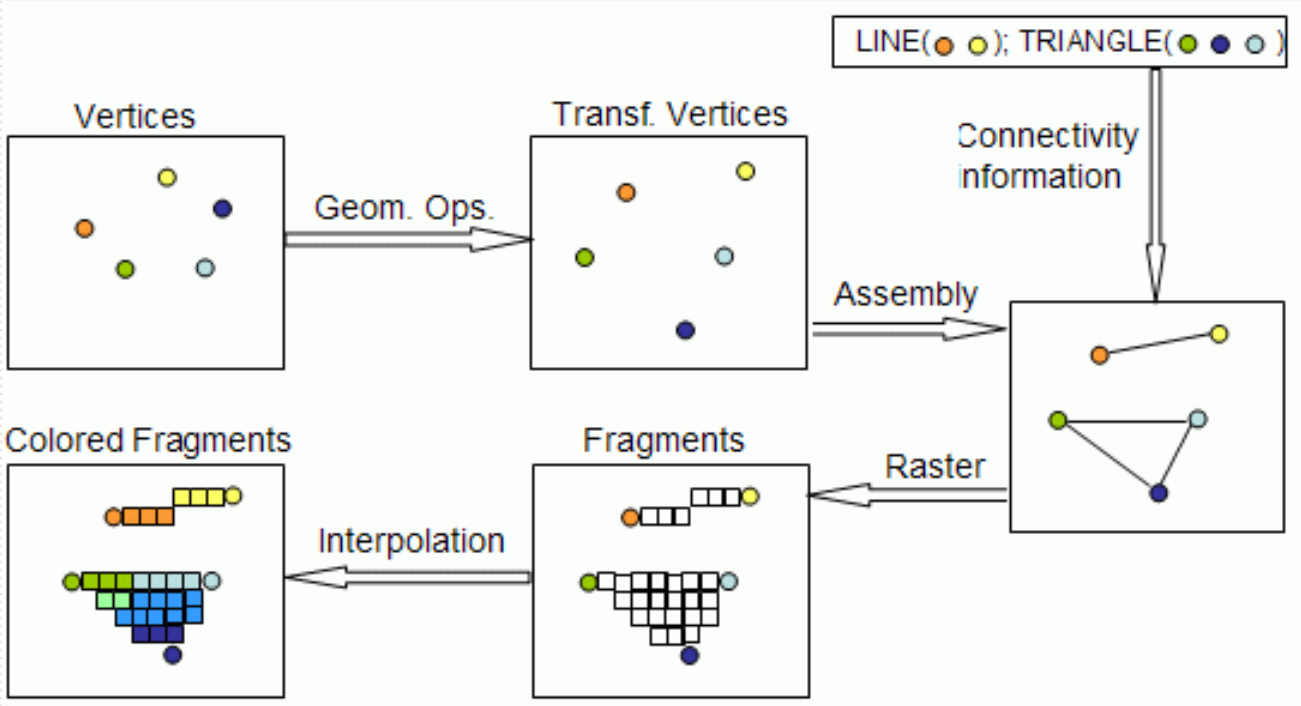
- An incremental approach can also be used to interpolate vertex normals.

Renderer Shading Options

- Wireframe
- Flat shaded polygons (only the polygon normal is used)
- Gouraud shading
- Phong shading
- Mixing Phong and Gouraud shading
  - In a scene consisting of specular and diffuse objects we can use Gouraud shading for the diffuse objects and only use Phong shading for the specular ones.

Programmable Shaders

- Both OpenGL and DirectX have their own programmable shader language.



Programmable Shaders (example)

- Programmers can write their own shading algorithms.
- The following is an example of a ‘toon’ shader created by using the OpenGL shading language