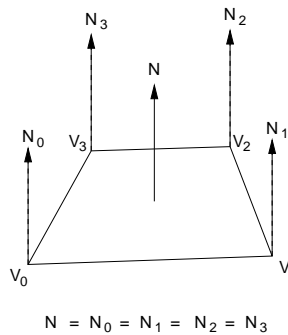


Flat, Gouraud, and Phong Shading

CptS 442/542

1. Flat Shading

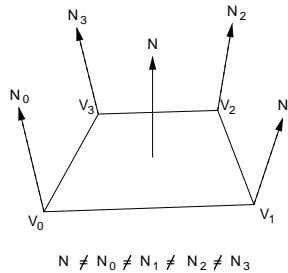
- Face is considered planar, i.e., one normal
- All vertices share the same normal



- Illumination model applied to one vertex to determine its color. This color is replicated for every resulting pixel.
- `glShadeModel(GL_FLAT);`

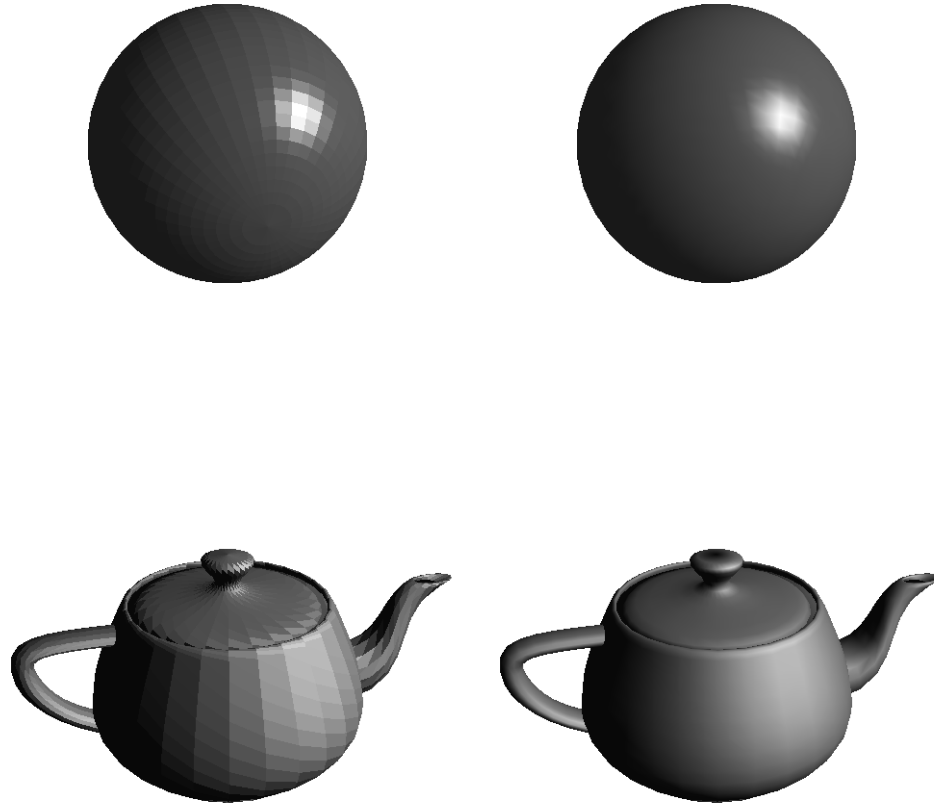
2. Smooth Shading

- Faces are (probably) still planar, but shading can make a mesh that is the skin of a smooth surface (e.g., teapot) appear smooth.
- Each vertex normal may be different.



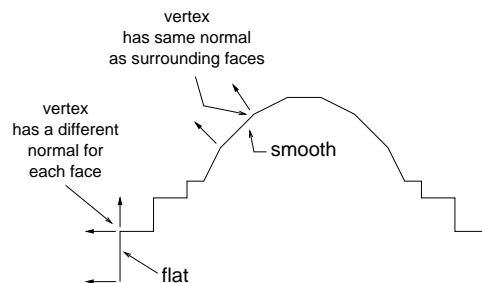
- Two primary algorithms for computing colors
 - *Gouraud shading*: Illumination model applied at each vertex to determine color associated with each vertex. Color is interpolated across the surface for each pixel.
 - *Phong Shading*: Normals are interpolated across the surface, and the illumination model is applied to each pixel.
- `glShadeModel(GL_SMOOTH);`
OpenGL uses Gouraud shading when smooth shading is specified.

3. Flat vs Smooth



The object's silhouette will expose flatness of surface even when smooth shading is used. Why?

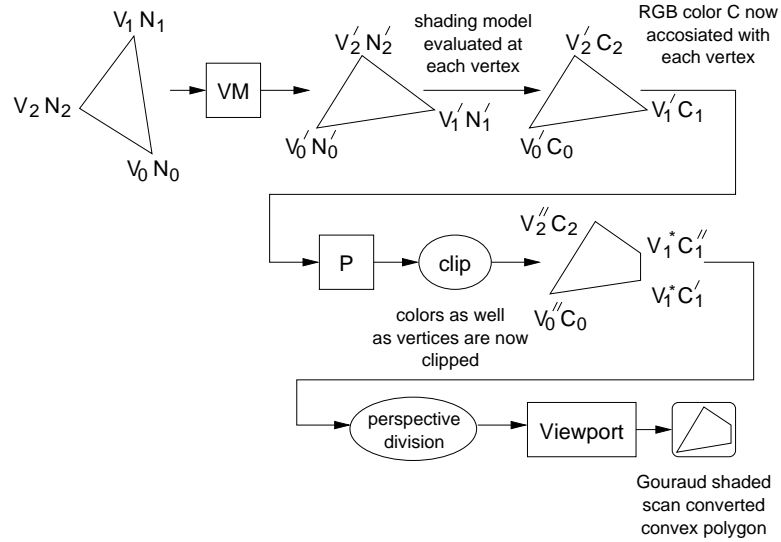
4. Flat and Smooth



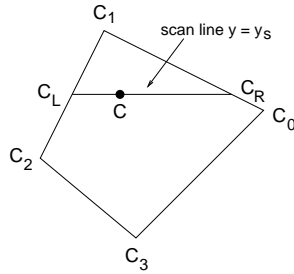
How we specify vertex normals depends on if we are approximating a smooth surface or a flat surface with sharp corners.

- For sharp corners just use face normal for each vertex when rendering face.
- For smooth corners, average normals for adjacent faces and use the same normal for each face.
- Use `glShadeModel(GL_SMOOTH)` to mix and match.

5. Gouraud shading



- Scan converting convex polygon.
 - We use linear interpolation of vertex colors to compute the colors of the interior pixels:



$$C_L = (1 - t)C_1 + tC_2, \quad t = \frac{y_s - y_1}{y_2 - y_1}$$

$$C_R = (1 - t)C_1 + tC_0, \quad t = \frac{y_s - y_1}{y_0 - y_1}$$

$$C = (1 - t)C_L + tC_R, \quad t = \frac{x - x_L}{x_0 - x_L}$$

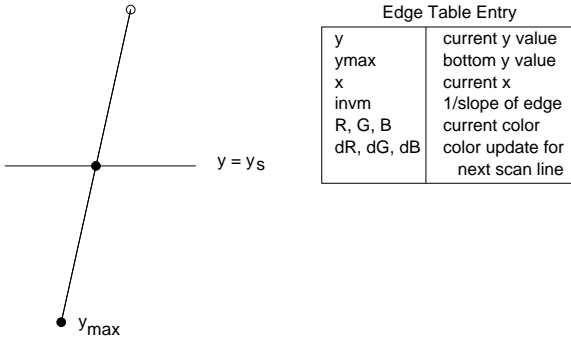
- To avoid the multiplications and divisions at each scan line we use *forward differencing*. While updating C_L , we have

$$t = \frac{y_s - y_1}{y_2 - y_1}, \quad t' = \frac{(y_s + 1) - y_1}{y_2 - y_1}$$

at scan line $y = y_s$ and $y = y_s + 1$ respectively. Now we determine how C_L changes as we move to the next scan line:

$$\begin{aligned} C_L &= (1 - t)C_1 + tC_2 \quad (\text{at } y = y_s) \\ C'_L &= (1 - t')C_1 + t'C_2 \quad (\text{at } y = y_s + 1) \\ \Delta C_L &= C'_L - C_L \\ &= (t' - t)(C_2 - C_1) \\ &= \frac{((y_s + 1) - y_1) - (y_s - y_1)}{y_2 - y_1} (C_2 - C_1) \\ &= \frac{C_2 - C_1}{y_2 - y_1}. \end{aligned}$$

– Entry in Edge Table



– Similarly, the update constant ΔC for C is

$$\Delta C = \frac{C_R - C_L}{x_r - x_l}.$$

– There is exactly two edges in the active edge table (AET) while filling a convex polygon. To fill a span we need to find the left and right edges by comparing the edges' x -values.

```

find left and right edges;
dR = (Rright - Rleft)/(xright - xleft);
dG = (Gright - Gleft)/(xright - xleft);
dB = (Bright - Bleft)/(xright - xleft);
for (x = xleft, R = Rleft, G = Gleft, B = Bleft;
     x <= xright;
     x++, R += dR, G += dG, B += dB)
    fill pixel (x,y) with color R,G,B;
    
```

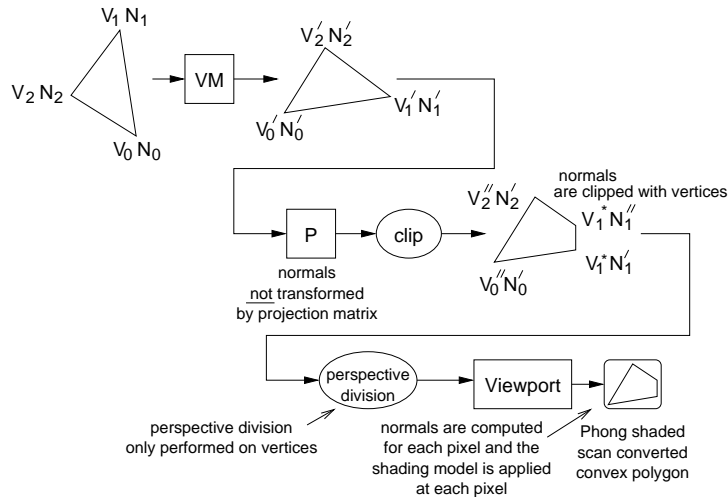
- Color clipping

In our 4-D clipping algorithm, we found the t value for the intersection of an edge with the canonical clipping volume (CCV). We use this t -value to determine the interpolated color C at the intersection point

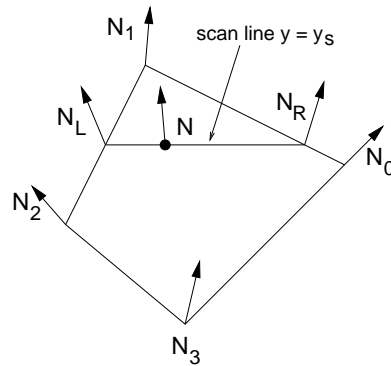
$$C = (1 - t)C_0 + tC_1.$$

6. Phong Shading

- Normals are transformed by inverse transpose of model-view matrix $((VM)^{-1})^T$ and passed are down the pipeline all the way to the rasterization step. The illumination model is applied at *every* pixel in the rendered polygon.
- Graphics pipeline if (when) OpenGL supported Phong shading



- Scan converting convex polygon



Normals are linearly interpolated at follows.

$$\begin{aligned} \vec{N}_L &= (1-t)\vec{N}_1 + t\vec{N}_2, & t &= \frac{y_s - y_1}{y_2 - y_1} \\ \vec{N}_R &= (1-t)\vec{N}_1 + t\vec{N}_0, & t &= \frac{y_s - y_1}{y_0 - y_1} \\ \vec{N} &= (1-t)\vec{N}_L + t\vec{N}_R, & t &= \frac{x - x_L}{x_0 - x_L} \end{aligned}$$

- Phong shading is more accurate, but more expensive. Specular highlights that may be missed with Gouraud shading are captured by a Phong shader.