Dithering and Rendering

CS116B
Chris Pollett
Apr 18, 2004.

Outline

- Dithering Techniques
- Constant-Intensity Surface Rendering
- Gouraud Surface Rendering
- Phong Surface Rendering
- Fast Phong Surface Rendering

Dithering Techniques

- Last day, we talked about using dithered noise with intensity values to get halftones without reducing resolution.
- Another method to do dithering is called **ordered dithering**.
- In this method, a **dither matrix** D_n is used.

$$D_2 = \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix} \quad D_3 = \begin{bmatrix} 7 & 2 & 6 \\ 4 & 0 & 1 \\ 3 & 8 & 5 \end{bmatrix} \quad D_n = \begin{bmatrix} 4D_{n/2} + D_2(1,1)U_{n/2} & 4D_{n/2} + D_2(1,2)U_{n/2} \\ 4D_{n/2} + D_2(2,1)U_{n/2} & 4D_{n/2} + D_2(2,2)U_{n/2} \end{bmatrix}$$

- Here U_n is the n x n all 1's matrix.
- Given a pixel (x,y) and a intensity $0 <= I <= n^2$. We calculate
  $j = (x \bmod n) +1$, $k = (y \bmod n) +1$ and check if

  $I > D_n(j,k)$. If yes, turn the pixel on, else off.

Still More Dithering

- Another dithering technique is called **error diffusion**.
- In this technique, the error between an input intensity and the selected intensity of a pixel position is spread out to the pixels to the right and below the current pixel position.
- Suppose I is intensity value in image at (i,j) and I' is the intensity we can display at this location. Then we compute E=I-I' and add to intensities of neighbors: (i+1,j), (i-1, j+1), (i, j+1), (i+1, j+1); the values a*E, b*E, c*E, d*E. Here we want a+b+c+d <=1. For example, could use a= 7/16, b= 3/16, c= 5/15, d= 1/16.
- One problem with this technique is that it can cause 'ghosts' on parts of the image to show up.

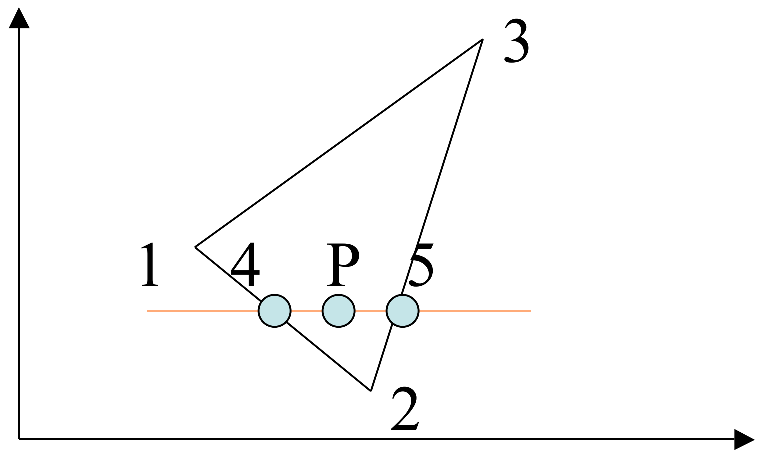
Constant-Intensity Surface Rendering

- We want to use our lighting model to compute intensity values for points on our surfaces.
- **Constant surface-rendering** or **flat surface** rendering is probably the fastest way to do this:
  - We assign the same color to all points in a projected surface.
- It is a reasonable approximation when:
  - The polygon is one face of a polyhedron and is not a section of a curved-surface approximation mesh.
  - All light sources are far enough away that **N.L** is approximately constant across the surface.
  - The viewing position is far enough so that **V.R** is approximately constant across the surface. **R** being the reflected light vector; **V** being the viewing vector.

Gouraud Surface Rendering

- We process each polygon in the scene in the following way:
    1. Determine the average unit normal vector at each vertex of the polygon (to do this we average the polygon normals of polygons containing this vertex).
    2. Apply the lighting model to get an intensity of this vertex.
    3. Linearly interpolate the vertex intensities over the projected area of the polygon.

Example of Step 3



- $I_4 = (y_4-y_2)/(y_1-y_2)I_1 + (y_1-y_4)/(y_1-y_2)I_2$
- Could calculate I_5 similarly.
- From this, $I_p=(x_5-x_p)/(x_5-x_4)I_4 + (x_p-x_4)/(x_5-x_4)I_5$
- These numbers can be computed incrementally along the scan-line. And similarly, could be incrementally updated between scan-lines.

Phong Surface Rendering

- A more accurate but slower way to do interpolation is to interpolate the normals across the surface.
- This is called **Phong-surface rendering**.
- In this procedure, we:
  - determine the average unit normal vector at each vertex of the polygon.
  - linearly interpolate the vertex normal over the projected area of the polygon.
  - apply an illumination model to positions along scan lines so as to calculate pixel intensities using the interpolated normals.

Fast Phong Surface Rendering

- Recalculating intensities for each pixel in the Phong set-up is slow.
- So instead we try to approximate these calculations using a truncated Taylor-series expansion and by limiting the polygons to be triangles.
- Let $\mathbf{N} = \mathbf{A}x+\mathbf{B}y +\mathbf{C}$ be the normal calculated for position (x,y). Where **A,B,C** is determined from the 3 vertex normals $\mathbf{N_k}$, k=1,2,3.
- $I_{diff}(x,y) = \mathbf{L}.\mathbf{N}/(|\mathbf{L}||\mathbf{N}|)$

$= [(\mathbf{L}.\mathbf{A})x+ (\mathbf{L}. \mathbf{B})y + (\mathbf{L}. \mathbf{C})]/(|\mathbf{L}|| \mathbf{A}x+\mathbf{B}y +\mathbf{C}|)$

More Fast Phong

- This last expression can be written as:

  $I_{diff}(x,y) = [ax+by+c]/[dx^2+exy + fy^2 + gx+hx+i]^{1/2}$

  for example a = **L.A**/|**L**|

- Taylor expanding, this we can write this as:

  $I_{diff}(x,y) = T_5x^2 + T_4xy + T_3y^2 + T_2x + T_1y + T_0$ where the constant T_i can be found in the book.

- Can do similar tricks to approximate the specular intensity.