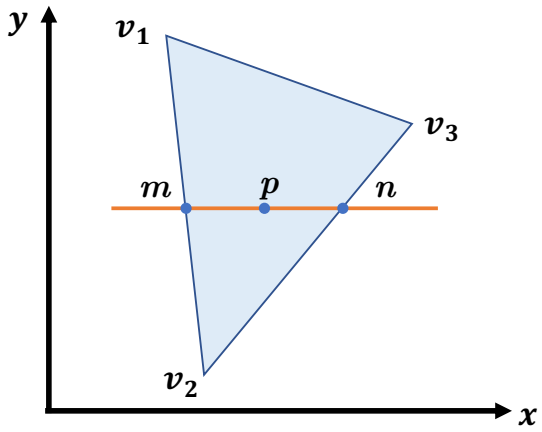
Homework 2: Solutions
CSE 490V: Virtual Reality Systems

Student Name
student@uw.edu

1 Theoretical Part

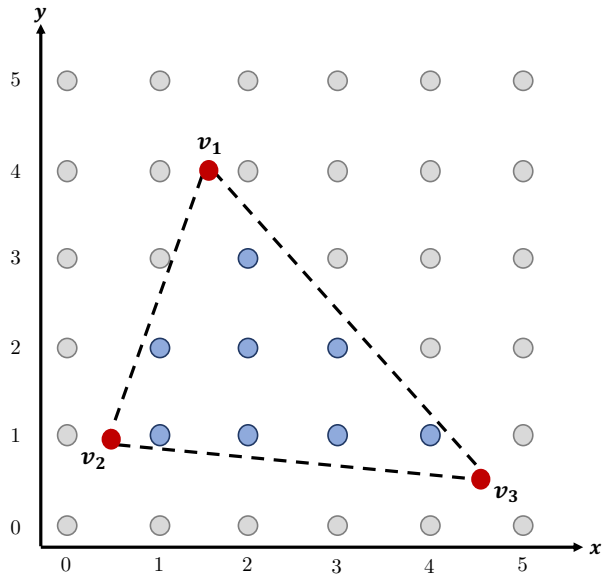
1.1 Simple Scanline Interpolation (*5pts*)

Given the polygon, shown in the first figure below, with vertices $\mathbf{v_1}$, $\mathbf{v_2}$, $\mathbf{v_3}$ with coordinates $(x_i,\ y_i)$ for $i = 1, 2, 3$ and intensities at each vertex I_1, I_2, I_3, calculate the intensity at point $\mathbf{p}$ at $(x,\ y)$. Assume the coordinates of $\mathbf{m}$ and $\mathbf{n}$ are (x_m, y_m) and (x_n, y_n), respectively.



Answer:

Write your answer to this question here.



1.2 Phong Lighting (*30pts*)

In this problem, we are going to "reverse transform" vertices from window space to view space where lighting calculations actually happen, and assign colors to each vertex based on the Phong lighting model. As illustrated in the figure below, suppose we have a screen window with a resolution of 6×6 pixels and with both x and y components of the pixel coordinates ranging from 0 to 5. The red points $\mathbf{v_1}$, $\mathbf{v_2}$ and $\mathbf{v_3}$ are vertices in window space, while the blue points represent the set of fragments that were determined by the rasterizer to lie within the primitive (i.e., the triangle spanned by the three vertices). You can think of fragments as a regular grid of pixels, each associated with a number of attributes such as 2D position, RGB color, normal, depth, alpha value, etc. The rasterizer determines which fragments are inside a primitive and it interpolates vertex attributes such that each fragment inside the primitive receives a set of interpolated attributes. Refer to the lecture slides and Marschner's textbook for additional details.

Given parameters of the three vertices:

- **vertex coordinates in window space:** $\mathbf{v}_1 = (1.5, 4)$, $\mathbf{v}_2 = (0.5, 1)$, $\mathbf{v}_3 = (4.5, 0.5)$
- **depth values in window space with range** $[0,\ 1]$**:** $z_1 = 0.7$, $z_2 = 0.5$, $z_3 = 0.3$
- **normals in view space:** $\mathbf{n}_1 = (-\frac{2}{3},\ \frac{2}{3},\ \frac{1}{3})^T$, $\mathbf{n}_2 = (-\frac{2}{3},\ -\frac{2}{3},\ \frac{1}{3})^T$, $\mathbf{n}_3 = (\frac{2}{3},\ -\frac{2}{3},\ \frac{1}{3})^T$

(i) Compute 3D coordinates of all three vertices in view space, given the following parameters: (*15pts*)

- $aspect = \frac{\text{width}}{\text{height}} = 1$
- $fovy = 90^\circ$
- $zNear = 2$, $zFar = 22$

Note that projection matrix can be constructed as follows.

$$M_{proj} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{6}{5} & -\frac{22}{5} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

and you may use this matrix in your calculation.

Hint: If you have a vector in normalized device coordinates (NDC), you can transform it into clip space by multiplying $\mathbf{v}_{ndc}$ by w_{clip} and plugging in w_{clip} as the fourth element. The problem is that we do not know w_{clip} directly. However, we can compute it from z_{ndc} and certain portions of the projection matrix. Assume that the projection matrix has the following structure

$$M_{proj} = \begin{pmatrix} \star & \star & \star & \star \\ \star & \star & \star & \star \\ 0 & 0 & T_1 & T_2 \\ 0 & 0 & E_1 & 0 \end{pmatrix},$$

where $\star$ denotes elements that are not being used. Since we know

$$z_{clip} = T_1 \times z_{view} + T_2$$

$$w_{clip} = E_1 \times z_{view}$$

$$z_{ndc} = z_{clip}/w_{clip}$$

Then we can find w_{clip} as follows.

$$w_{clip} = \frac{T_2}{z_{ndc} - \frac{T_1}{E_1}}$$

The derivation is not covered here, but you are encouraged to derive it by yourself; this will not be graded.

(ii) Compute the color of each vertex using the Phong lighting model given the following conditions: (*10pts*)

- point light source located at $\mathbf{l} = (5, 5, 5)$ in view space with RGB color $\mathbf{I} = (10,\ 10,\ 10)^T$
- diffuse material properties: $\mathbf{m_1} = (1,\ 0,\ 0)^T$, $\mathbf{m_2} = (0,\ 1,\ 0)^T$, $\mathbf{m_3} = (0,\ 0,\ 1)^T$
- neglect specular and ambient components of lighting
- consider square distance falloff ($k_c = k_l = 0,\ k_q = 1$)

Note: In the Gouraud shading model (i.e., per-vertex lighting), the lighting calculations are done for each vertex similar to what you just calculated. Afterward, the rasterizer will interpolate the color values of the three vertices over the entire triangle (similar in spirit to what you just did). Remember that the Phong shading model (i.e., per-fragment lighting) uses the rasterizer to interpolate the vertex positions and also the normals over the triangle and then performs the lighting calculations per fragment.

In this particular case, Gouraud shading includes lighting calculations for the three vertices whereas Phong shading would require lighting calculations for all eight fragments that are inside this triangle. This would make per-fragment lighting significantly slower.

(iii) Oftentimes, per-fragment lighting involves more calculations than per-vertex lighting. Are there cases when this is not true? When specifically would Phong shading (i.e., per-fragment lighting) be faster than Gouraud shading (i.e., per-vertex lighting)? (*5pts*)

Answer:

(i) Write your answer to this question here.

(ii) Write your answer to this question here.

(iii) Write your answer to this question here.

---

2 Programming Part

2.1.1.3 Attenuation Factor Observations Does enabling attenuation have a major impact on the appearance of the scene? If so, how so?

Answer:

Write your answer to this question here.

2.2.2.1 Gouraud vs. Phong Shading Comparison Compare Gouraud shading to Phong shading. How are they different? What are the benefits and downsides of each shading method? Specifically, comment on both quality of shading and computational load. You don't need to actually measure any runtimes here, just briefly discuss them theoretically.

Answer:

Write your answer to this question here.
