COMP 4002 Assignment 3 - Ryan Seys - 100817604

<u>Question / Task 3.</u>

<u>9.1) Which model is better? Why?</u>

Each model has their strengths and weaknesses so it can be said that neither is "better" than the other and it only depends on your use case. The Gouraud lighting model only computes the lighting on each vertex, rather than each pixel, so typically Gouraud is a faster process than the Phong model which on contrary computes lighting on every pixel. The Phong model has the advantage that because each pixel is processed, typically results are more realistic, despite being more computationally expensive. If the machine you're running the shader code on cannot maintain a desired framerate using the Phong lighting model, it may be suggested that you use the Gouraud model instead in order to gain performance and hopefully hit the desired framerate.

<u>9.2) To your opinion is it worth to use the Phong model from performance point of view and from programming efforts point of view?</u>

Given the current state of CPU and GPU computational power, for most simple simulations such as those used in these assignments, the performance hit imposed by using the Phong model will be minimal and you shouldn't see any noticable performance degradation.

For large complex models with many vertices such as those used in realistic high resolution real time 3D games on next-generation consoles and PCs, a Phong model may be too computationally expensive and thus the performance hit to framerate may be noticable. Also, if a model has a high number of vertices, the noticable difference between the Phong model and Gouraud model may be negligible because the lighting differences will be subtle on high-density objects.

That being said, if the framerate drops below 60 frames per second, one may choose to move to a simpler lighting model such as Gouraud in hopes that it may improve performance.

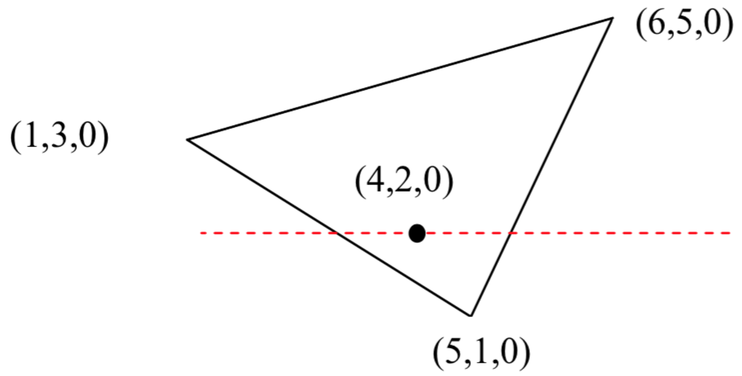
To clarify, I would have to say that in my opinion, for smaller models on a typical modern machine, the Phong model is definitely worth it from a performance point of view as the performance hit will be negligible. For some cases in larger models or slower machines, the performance hit and less-noticable visual difference between Gouraud and Phong may not be worth it if you see a noticable drop in framerate.

From a programming efforts point of view, there was no real difference between modeling lighting in the vertex shader versus modeling it in the fragment shader (as is used in the Phong model). From this perspective, I would have to say that there was minimal effort used in converting between the two lighting models and thus using Phong is definitely worth it from a programming point of view given the better results I acheived.

<u>Question / Task 4.</u>

Perform bilinear interpolation.

<u>What values of colour and normal be transferred to the fragment shader by the GPU?</u>



V1 – position(5,1,0), (colour (0.9, 0.7, 0.1), normal (1, 3, 3)
V2 – position(1,3,0), colour (1.0, 0.3, 0.9), normal (2, 2, 2)
V3 – position(6,5,0), colour (1.0, 1.0, 0.1), normal (0, 2, 2)

Intersection of scan line and $V_{12}$ (i.e. from (5, 1, 0) to (1, 3, 0)) **denoted by A**
Intersection of scan line and $V_{13}$ (i.e. from (5, 1, 0) to (6, 5, 0)) **denoted by B**
Point (4, 2, 0) **denoted by P**

**Solving for position of A:**

$A_y = P_y = 2$
$A_z = P_z = 0$

$A_y = V_{1y} + ((A_x - V_{1x)} / (V_{2x} - V_{1x}))*(V_{2y} - V_{1y})$
$2 = 1 + ((A_x - 5) / (1 - 5)) * (3 - 1)$
$A_x = 3$

Therefore, A = (3, 2, 0)

**Solving for color at A (denoted by $A_C$):**

$A_C = V_{1C} + (|V_{1A}| / |V_{12}| ) * (V_{2C} - V_{1C})$

$A_C$ = (0.9, 0.7, 0.1) + (| (3, 2, 0) - (5, 1, 0) | / | (1, 3, 0) - (5, 1, 0) |) * ((1.0, 0.3, 0.9) - (0.9, 0.7, 0.1))

$A_C$ = (0.95, 0.5, 0.5) [i.e. because A's position is exactly halfway between $V_1$ and $V_2$, the color of A is exactly the mean of both colors for $V_1$ and $V_2$]

**Solving for normal at A (denoted by $A_N$):**

$A_N = V_{1N} + (|V_{1A}| / |V_{12}|) * (V_{2N} - V_{1N})$

$A_N$ = (1, 3, 3) + (| (3, 2, 0) - (5, 1, 0) | / | (1, 3, 0) - (5, 1, 0) |) * (( 2, 2, 2) - (1, 3, 3))

$A_N$ = (1.5, 2.5, 2.5) [i.e. because A's position is exactly halfway between $V_1$ and $V_2$, the normal of A is exactly the mean of both normals for $V_1$ and $V_2$ much like color was.]

**Solving for position of B:**

$B_y = P_y = 2$
$B_z = P_z = 0$

$B_y = V_{1y} + ((B_x - V_{1x)} / (V_{3x} - V_{1x}))*(V_{3y} - V_{1y})$
2 = 1 + $(B_x$ - 5) / (6 - 5)) * (5 - 1)
$B_x$ = 21 / 4 = 5.25

Therefore, B = (5.25, 2, 0)

**Solving for color at B (denoted by $B_C$):**

$B_C = V_{1C} + (|V_{1B}| / |V_{13}|) * (V_{3C} - V_{1C})$

$B_C$ = (0.9, 0.7, 0.1) + (| (5.25, 2, 0) - (5, 1, 0) | / | (1, 3, 0) - (6, 5, 0) |) * ((1.0, 1.0, 0.1) - (0.9, 0.7, 0.1))

$B_C$ = (0.919141, 0.757423, 0.1)

**Solving for normal at B (denoted by $B_N$):**

$B_N = V_{1N} + (|V_{1B}| / |V_{13}|) * (V_{3N} - V_{1N})$

$B_N$ = (1, 3, 3) + (| (5.25, 2, 0) - (5, 1, 0) | / | (6, 5, 0) - (5, 1, 0) |) * ((0, 2, 2) - (1, 3, 3))

$B_N$ = (0.80859, 2.80859, 2.80859)

**Solving for color at P (denoted by $P_C$):**

$P_C = A_C + (|V_{AP}| / |V_{AB}|) * (B_C - A_C)$

$P_C$ = (0.95, 0.5, 0.5) + (| (4, 2, 0) - (3, 2, 0) | / | (5.25, 2, 0) - (3, 2, 0) |) * ((0.919141, 0.757423, 0.1) - (0.95, 0.5, 0.5))

$P_C$ = (0.95, 0.5, 0.5) + (1/2.25)(-0.030859, 0.257423, -0.4)

$P_C$ = (0.936285, 0.61441, 0.322222)

**Solving for normal at P (denoted by $P_N$):**

$P_N = A_N + (|V_{AP}| / |V_{AB}|) * (B_N - A_N)$

$P_N$ = (1.5, 2.5, 2.5) + (| (4, 2, 0) - (3, 2, 0) | / | (5.25, 2, 0) - (3, 2, 0) |) * ((0.80859, 2.80859, 2.80859) - (1.5, 2.5, 2.5))

$P_N$ = (1.5, 2.5, 2.5) + (1/2.25)(-0.69141, 0.30859, 0.30859)

$P_N$ = (1.19271, 2.63715, 2.63715)

Therefore, the values of colour and normal be transferred to the fragment shader are:

color = $P_C$ = (0.936285, 0.61441, 0.322222)
normal = $P_N$ = (1.19271, 2.63715, 2.63715)