

# CPSC 314

## Assignment 3

Due 4PM, Nov 25, 2005 (You can submit the programming part till Monday Nov 28 with no penalty or grace day use)

Answer the questions in the spaces provided on the question sheets. If you run out of space for an answer, use separate pages and staple them to your assignment.

**Note:** Some of the questions in the assignment are based on material to be covered between now and Nov 25. Answer those after the topics are covered in class.

Name: \_\_\_\_\_

Student Number: \_\_\_\_\_

Question 1	/ 10
Question 2	/ 5
Question 3	/ 5
Question 4	/ 90
TOTAL	/ 100

## 1. Light and shading

- (a) Given a scene with two non specular objects, one yellow ( $k_a = k_d = (1, 1, 0)$ ) and one red ( $k_a = k_d = (1, 0, 0)$ ), classify the following statement as true or false. Explain.
- i. (1 point) Given a single point light source with intensity  $I_p = (1, 0, 0)$  the objects will have the same shading.

No. Because it is a point source, we must use the  $k_d I_p \cos(\theta)$  term of the lighting calculations. The angle may be different between the surfaces of the two objects and the light source therefore the shading will be different.

- ii. (1 point) Given a single ambient light source with intensity  $I_a = (1, 0, 0)$  the objects will have the same shading.

Yes. The ambient calculation  $k_a I_a$  is the same value for both objects and remains the same regardless of their positions and orientations, therefore the shading will be identical.

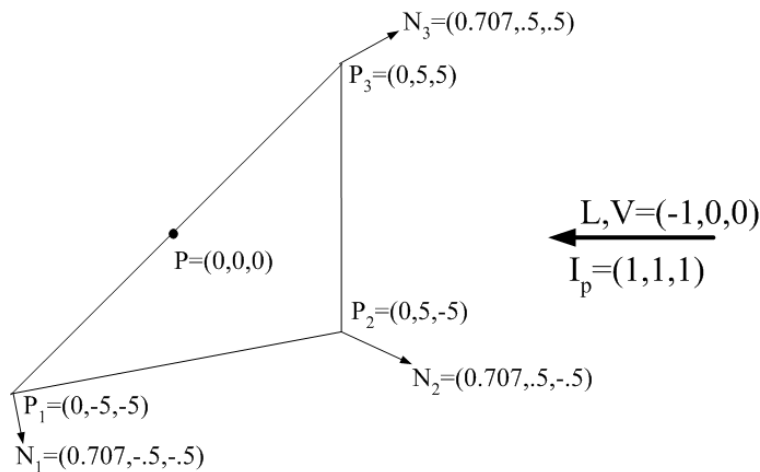
- (b) (1 point) Write the openGL code for defining the following lighting scenario with three light sources: ambient light source with intensity  $I_a = (0.3, 0, 0)$ ; directional light with direction  $(1, 0, 0)$  and intensity  $(0.6, 0.6, 0.6)$ ; point light at  $(10, 0, 0)$ .

```
float reset[]      = { 0.f, 0.f, 0.f, 1.f };
float point[]      = { 10.f, 0.f, 0.f, 1.f };
float directional[] = { -1.f, 0.f, 0.f, 0.f };
float ambient[]    = { 0.3f, 0.f, 0.f, 1.f };
float intensity[]  = { 0.6f, 0.6f, 0.6f, 1.f };
// enable lighting
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_LIGHT1);
glEnable(GL_LIGHT2);
// set up light 0 as ambient light
glLightfv( GL_LIGHT0, GL_AMBIENT,  ambient );
glLightfv( GL_LIGHT0, GL_DIFFUSE,  reset );
glLightfv( GL_LIGHT0, GL_SPECULAR,  reset );
// set up light 1 as directional light
glLightfv( GL_LIGHT1, GL_POSITION,  directional );
glLightfv( GL_LIGHT1, GL_AMBIENT,  reset );
glLightfv( GL_LIGHT1, GL_DIFFUSE,  intensity );
glLightfv( GL_LIGHT1, GL_SPECULAR,  intensity );
// set up light 2 as point light (other params unimportant)
glLightfv( GL_LIGHT2, GL_POSITION,  point );
glLightfv( GL_LIGHT2, GL_DIFFUSE,  intensity );
```

- (c) (1 point) In OpenGL define the material properties for a triangle with  $k_a = (1, .5, .5)$ ,  $k_d = (1, .5, .5)$ ,  $k_s = (.5, .5, .5)$  and specularity coefficient  $n = 16$ .

```
float ambient[]      = { 1.f, 0.5f, 0.5f, 1.f };
float diffuse[]     = { 1.f, 0.5f, 0.5f, 1.f };
float specular[]    = { 0.5f, 0.5f, 0.5f, 1.f };
float shininess     = 16.0f;
glMaterialfv(GL_FRONT, GL_AMBIENT, ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, specular);
glMaterialf(GL_FRONT, GL_SHININESS, shininess);
```

- (d) In the scene below there is one directional light source at infinity  $(\infty, 0, 0)$  with direction  $(-1, 0, 0)$ . The view direction is the same as light direction  $(-1, 0, 0)$ . The shading coefficients for the triangle are  $k_a = k_d = (1, 0, 0)$ ,  $k_s = (0, 1, 0)$  and the specularity coefficient is  $n = \infty$ .



Compute the color at point  $P$  on the triangle using the following shading algorithms (use per-face or per-vertex normals as necessary):

- i. (2 points) Flat shading,

We compute the normal to the triangle using the point coordinates.

$$N = \frac{(P_2 - P_1) \times (P_3 - P_1)}{\|(P_2 - P_1) \times (P_3 - P_1)\|} = (1, 0, 0)$$

We then perform lighting calculations with this normal.

$$color = k_a I_p + k_d I_p (-L \cdot N) + k_s ((2N(-L \cdot N) + L) \cdot -V)^\infty = (2, 1, 0)$$

This is the color of the entire triangle including point  $P$ .

- ii. (2 points) Gourard shading,

For this shading model we calculate the colors at each of the vertices using the following calculations

$$color_1 = k_a I_p + k_d I_p (-L \cdot N_1) + k_s ((2N_1(-L \cdot N_1) + L) \cdot -V)^\infty = (1.707, 0, 0)$$

$$color_2 = k_a I_p + k_d I_p (-L \cdot N_2) + k_s ((2N_2(-L \cdot N_2) + L) \cdot -V)^\infty = (1.707, 0, 0)$$

$$color_3 = k_a I_p + k_d I_p (-L \cdot N_3) + k_s ((2N_3(-L \cdot N_3) + L) \cdot -V)^\infty = (1.707, 0, 0)$$

We then interpolate the color at  $P$  using its barycentric coordinates  $(0.5, 0.0, 0.5)$ , yielding the color  $0.5 * color_1 + 0.0 * color_2 + 0.5 * color_3 = (1.707, 0, 0)$ . Note that the specular term never contributes in these equations because  $1 > ((2N(-L \cdot N) + L) \cdot -V)$  for each normal and  $x^\infty = 0$  if  $0 \leq x < 1$ .

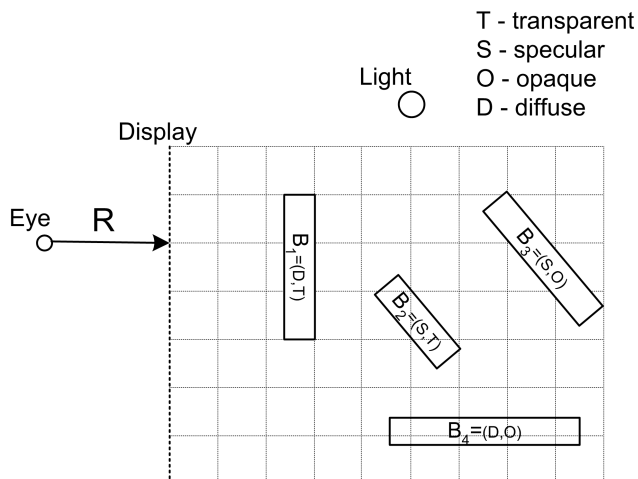
- iii. (2 points) Phong shading.

For this shading model we interpolate the normal at  $P$  using its barycentric coordinates  $(0.5, 0.0, 0.5)$ . This gives us the normal  $N = 0.5 * N_1 + 0.0 * N_2 + 0.5 * N_3 = (0.707, 0, 0)$  which we normalize to  $(1.0, 0.0, 0.0)$ . We then use the familiar lighting formula to calculate the color.

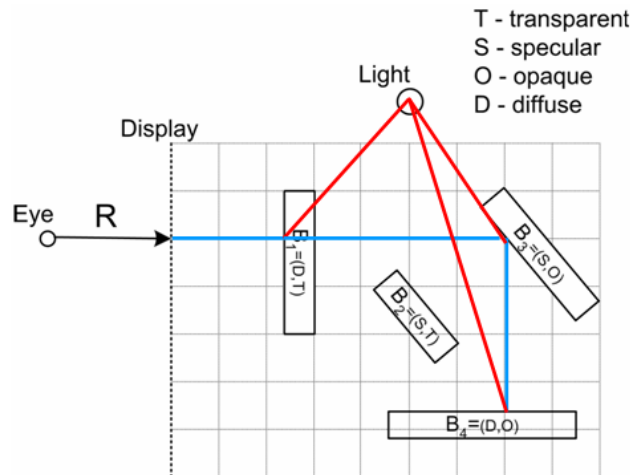
$$color_P = k_a I_p + k_d I_p (-L \cdot N) + k_s ((2N(-L \cdot N) + L) \cdot -V)^\infty = (2, 1, 0)$$

## 2. Ray-Tracing

- (a) (3 points) Draw the ray tree for the ray  $R$  shown below. Assume index of refraction  $c_1$  for air is 1 and index of refraction for all the transparent objects in the scene is  $c_2 = \frac{1}{\sqrt{2}}$ . Use Snell's law to obtain refraction angles.



Since the ray is orthogonal to the surface of  $B_1$  there is no change in the angle as it passes through it. It then bounces off the mirror surface of  $B_3$  and we follow the reflected ray to  $B_4$ .



answer:

- (b) (2 points) Assume the transparency coefficient  $\alpha$  for the transparent objects is .5, the light intensity is  $I_p = (1, 1, 1)$  (no other lights), and the diffuse/specular coefficients for the objects are  $k_d^1 = (1, 0, 0)$ ,  $k_s^1 = (0, 0, 0)$ ,  $k_d^2 = (0, 0, 0)$ ,  $k_s^2 = (1, 1, 1)$ ,  $k_d^3 = (0, 0, 0)$ ,  $k_s^3 = (1, 1, 1)$ ,  $k_d^4 = (0, 1, 0)$ ,  $k_s^4 = (0, 0, 0)$ . What is the color returned by the ray tracing algorithm for ray  $R$ ?

For each lighting calculation we sum the transparent, specular, and surface colors with the following respective coefficients  $\alpha$ ,  $\beta$ ,  $\gamma$  where  $\alpha + \beta + \gamma = 1$ .

Let's first determine these coefficients for each object we intersect in our raytree diagram. Because  $B_1$  is a transparent, diffuse object its coefficients are  $\alpha_1 = 0.5$  (given above),  $\beta_1 = 0$ ,  $\gamma_1 = (1 - \alpha) = 0.5$ .  $B_3$  is opaque and perfectly specular (a mirror surface) and so its coefficients are  $\alpha_3 = 0$ ,  $\beta_3 = 1$ ,  $\gamma_3 = 0$ .  $B_4$  is opaque and diffuse so its coefficients are  $\alpha_4 = 0$ ,  $\beta_4 = 0$ ,  $\gamma_4 = 1$ .

Next, we formulate the lighting calculations for each intersection. For  $B_4$

$$color_{B_4} = \gamma_4 k_d^4 I_p (L_4 \cdot N_4) = (0, L_4 \cdot N_4, 0)$$

where  $N_4$  is the normal to  $B_4$  and  $L_4$  is the normalized vector from the point designated in our ray-tree diagram and the light. For  $B_3$

$$color_{B_3} = \beta_3 k_s^3 * color_{B_4} = (0, L_4 \cdot N_4, 0)$$

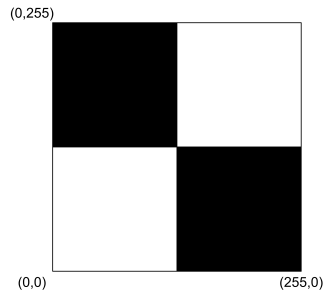
because we're reflecting the color from  $B_4$ . Finally, for  $B_1$

$$color_{B_1} = \alpha_1 color_{B_3} + \gamma_1 k_d^1 I_p (L_1 \cdot N_1) = (\gamma_1 L_1 \cdot N_1, \alpha_1 L_4 \cdot N_4, 0) = (.5 * L_1 \cdot N_1, .5 * L_4 \cdot N_4, 0)$$

because we're considering both the transparent color and the surface color. Our final color is  $color_{B_1}$ .

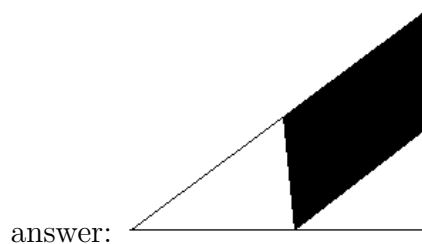
## 3. Texture Mapping.

- (a) (3 points) The following texture is stored in the array *image* of size  $imgx \times imgy$  ( $256 \times 256$ ).

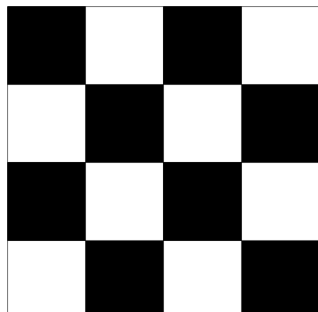


Draw the textured triangle produced by the following code:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, imgx, imgy, 0,
             GL_RGBA, GL_UNSIGNED_BYTE, image);
glEnable(GL_TEXTURE_2D);
glBegin(GL_POLYGON);
glTexCoord2f( 0, 0);
glVertex3d( 0, 0, 0 );
glTexCoord2f( 1, 1);
glVertex3d( 1, 0, 0 );
glTexCoord2f( 0, 1);
glVertex3d( 1, 1, 0 );
glEnd();
```

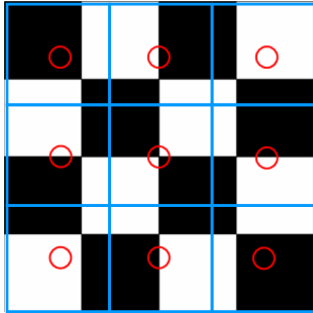


- (b) (2 points) The texture below is stored in a  $4 \times 4$  “texel” array.

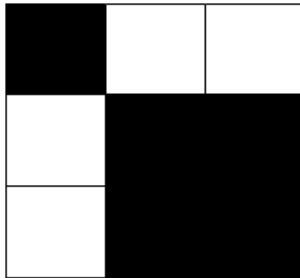


How will this texture look when mapped to a square of  $3 \times 3$  pixels? Draw and explain.

The key is to divide the texture into a uniform 3 by 3 grid and sample from the center of each cell in the grid like so:



You will note that several of these samples occur exactly between the two (and even four) color regions. So which color do we choose? The key is to develop a consistent rule about which side to choose. For the patch below, if our sample falls between two colors horizontally, then we choose the left color. Likewise, if our sample falls between two colors vertically then we choose the top color. The key is to keep your selections *consistent*.



answer: