NetID___wzm416_________Name______Wenjie Zhang_____________________________
CS351 Introduction to Computer Graphics **TestC: Better Lights & Materials**
Winter Quarter, 2016 **100 pts, 7% of grade**

(48 pts) True/False: **3pts each** (See Canvas →Pages→WebGL: the Full Specifications→GLSL-ES 1.0 Spec)
Copy-and-paste your choice of these highlighted answers: "True" or "False")

1)_____ True _____GLSL ES shader programs are fed to the GLSL ES compiler as array of one or more strings (concatenated when loaded). Each string may contain multiple lines separated by newlines.

2)______ True ____GLSL ES permits you to declare, define, and call your own functions for use in shader programs.

3)___ False _______Just like the C/C++ pre-processor, the GLSL ES preprocessor can accept macros with parameters. For example, you can write a GLSL macro for absolute-value that contains: `abs(x) ((x)<0? (-x): (x))`

4)___ False _______GLSL ES functions, ones that use the **`inout`** qualifier before a formal parameter, are granted the ability to make pass-by-reference accesses to its arguments, but (like C/C++) requires use of pointer arguments in the function call.

5)_ True _________A GLSL ES function declared with a vec3 return value could be re-written to supply that same vec3 result
value as one of its arguments. For example, the value returned by the function declared by
`vec3 myFunc(in float src);` could also be obtained by calling a re-written version declared by
`void yourFunc(in float src, out vec3 dest);`

6 True Parameters for functions in GLSL ES that were written with the '**`in`**' qualifier can also be written without any qualifier at all. For example, the function prototype **`vec3 myFunc(in float src);`** could be re-written as **`vec3 myFunc(float src);`** without causing error or changing its capabilities.

7)___ False _______Just like a C/C++ **`for() loop`**, the GLSL ES **`for()`** loop allows multiple indices and can change its counting increment as it runs. This enables a rich variety of complex iteration schemes for GLSL ES, such as:
`for(int i=0, int j=1; i<j+3; i=j+2, j=i+k){ k = myFunc(i,j); …}`

8)____ False ______GLSL ES offers multi-dimensional arrays of float, vec, and matrix members for use in your shaders.

9 True Just like the C/C++ pre-processor, the GLSL ES preprocessor supports conditional compiling. For example, it can selectively execute code between lines that begin with: **`#if`**, **`#ifdef`**, **`#ifndef`**, and **`#endif`**.

10)__ False ________To declare structures in GLSL ES you *may* use the '**`typedef`**' mechanism (just like C/C++, & not req'd).

11)__ False ________The GLSL ES shader language permits declaration and use of 'attribute' variables as either local variables or global variables, and these global 'attribute' variables are always available for use in your fragment shader programs.

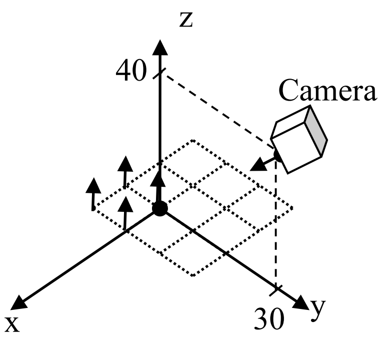
12)___ False _______If your GLSL ES Vertex Shader and Fragment Shader programs correctly compute the Blinn-Phong lighting and Phong shading, they also automatically compute shadows cast on the ground-plane from any object above it.

13)___ False _______In the Phong lighting model, if we change by 45 degrees the direction of the surface normal vector for one vertex illuminated by one light source, the on-screen result from the diffuse lighting term for that vertex will not change.

14)____ False ______The same matrix that correctly transforms vertex positions from model coordinates to world coordinates will correctly transform vertex normal vectors from model coordinates to world coordinates

15)____ True ______In the Phong lighting model, if we change by 45 degrees the direction of the surface normal vector for one vertex illuminated by one light source, the on-screen result from the ambient lighting term for that vertex will not change.

16)___ True _______The Phong lighting model can be computed correctly in either the 'world' or the 'eye' coordinate system.



Suppose we use your Project C interactive 3D rendering program to view this simple synthetic scene using Phong materials, lighting, and shading.
The scene includes ground plane; a 3x3 grid of axis-aligned 'tiles'—adjacent, flat, solid (not wireframe) squares, with all four vertices in the z=0 plane as shown, drawn by using two TRIANGLES primitives for each tile.
-- four of these tiles have a corner vertex at the origin (0,0,0),
--adjacent tiles share the same vertex locations for their corners
-- Your vertex buffer objects store triangle vertices in CCW order,
so that each tile's 'front' or 'outwards' direction is +z.
-- Each vertex of each tile includes a surface normal vector attribute. The figure above shows the surface normal vectors for just one tile; all normal vectors have the value N = (0, 0, +1), pointing in the +z direction.
--The camera field-of-view is large enough to see the entire 3x3 ground plane, and its location in world coordinates is

Camera Center of Projection:	(0,30,40) as shown above
Camera Aiming point:	(0,0,0)
Camera View Up Vector:	(0,0,1).

--The Phong materials properties used to render this 36-vertex, 18-triangle surface are:

Ambient Reflectance:	(r,g,b) = (0.4, 0.3, 0.2)	Specular Reflectance :	(r,g,b) = (1.0, 1.0, 1.0)
Diffuse Reflectance:	(r,g,b) = (0.0, 1.0, 0.0)	Specular Exponent, or 'Shinyness':	Se = 100
		Emissive Color:	(r,g,b) = (0.1, 0.05, 0.2)

--Lighting: our program defines one Phong light source in the 'world' coordinate system, with these settings:

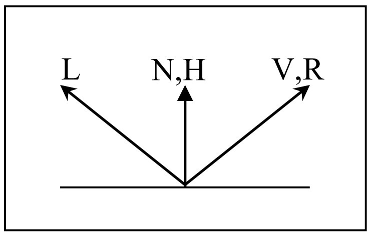
Ambient Intensity:	(r,g,b) = (0.1, 0.2, 0.3)	Specular Intensity :	(r,g,b) = (0.85, 0.85, 0.85)
Diffuse Intensity:	(r,g,b) = (0.4, 0.5, 0.4)		

Apply the basic principles of Phong lighting and materials to solve these problems:
(you will not need any fancy mathematics, vector diagrams, or complex calculations to find these answers!)

17) **(6 pts)** With the camera fixed at world-coordinate-system location (0, 30, 40) as shown we move the light source, and watch as the specular highlight moves on the ground-plane. We stop the light when the world-space origin point (0,0,0) on the ground plane looks brightest on-screen, with a round specular highlight centered at (0,0,0). Where is the light?
--At this light position (Lx, Ly, Lz), the specular highlight intensity at surface point (0,0,0) reaches its maximum.
--The distance from the light source to the origin is 100.
--The distance from the camera to the origin is 50. (because sqrt($30^2 + 40^2 + 0^2$) = sqrt(2500) = 50)
Where is the light source in x,y,z coordinates?

Helpful Reflection Figure:
(but not really necessary)

(Lx, Ly, Lz) = (0, -60, 80)



18) Suppose our interactive 3D rendering program can toggle between Phong lighting (which computes the reflectance vector **R**) and Blinn-Phong lighting (which uses the 'halfway' angle approximation) each time we press a button on the keyboard. If we press that button repeatedly,

a) **(3 pts)** Will the specular highlight appear to change position on the ground plane? Remember, the camera is at is at (0, 30, 40), and it sees a specular highlight centered at the origin point (0,0,0). Specifically, will the center of the specular highlight appear anywhere other than the origin for both Phong and Blinn-Phong lighting? **Answer 'Yes' or 'No' AND explain why** your answer is correct.

No. because the light and camera position are not change the specular Highlight is also not change. And
For the Phong Lighting Is*Ks*max $(0, (R \cdot V)^{Se})$*Att,
For the Blinn-Phong Lighting Ks*Is*Att*max $(0, (N \cdot \mathrm{H})^{Se})$,
In this case R, V, N, H is const value so the center of the specular highlight is not change.

b) **(3 pts)** In the camera image from (0, 30, 40), how does the specular highlight appear to change size and shape? **Specify which** specular highlight will appear to cover a larger area? Choose only one: Phong? Blinn-Phong? Neither? **AND explain why** your answer is correct. (Hint: Look around online – Google 'Blinn-Phong' etc.)

Blinn-Phong
The Blinn-Phong Lighting has a larger area since compared with Phong lighting $N \cdot \mathrm{H}$ is changing relatively slow (for Blinn-Phong it calculate halfway vector), So the specular highlight of Blinn-Phong can cover larger area.

19) Suppose that the camera image from (0, 30, 40) shows the specular highlight from one light centered at the origin on the ground plane (same as in questions 17, 18a, 18b). If we move the camera to location (10, 30, 40), AND we move the light-source by +10 in the x direction: Highlight your choice below:

a)**(3 pts)** how far does the center of the specular highlight appear to move if measured on the ground-plane?

a) the specular highlight stays fixed at the origin, at (0,0,0)
b) the specular highlight appears to move away from the origin, by a distance of less than 10
[✓ c) the specular highlight appears to move away from the origin, by a distance exactly 10]
d) the specular highlight appears to move away from the origin, by a distance of more than 10

b)**(3 pts)** What is the specular highlight's new location on the ground plane?

a) (a,0,0) where a=+10.0 (give a signed numerical value: e.g. +5.3, -2.4)
b) (0,a,0)
c) (0,0,a)
d) (a,b,c) where a=_____________, b=_____________, c= _____________ (signed numerical values)

REMEMBER:
at the center of the specular highlight, the angle between N, L vectors exactly equals the angle between the N,V vectors

20) Suppose that the camera image from (0, 30, 40) shows the specular highlight from one light centered at the origin on the ground plane, as before. Recall that the camera is distance 50 from the origin, and the light is distance 100 from the origin. If we lower the camera by 10 units in z to location (0, 30, 30), and we ALSO lower the light-source 10 units in z:

a)**(3 pts)** how far does the center of the specular highlight appear to move if measured on the ground-plane?

a) the specular highlight stays fixed at the origin, at (0,0,0)
[✓ b) the specular highlight appears to move away from the origin, by a distance less than 10]
c) the specular highlight appears to move away from the origin, by a distance of exactly 10
d) the specular highlight appears to move away from the origin, by a distance of greater than 10.

b)**(3 pts)** What is the specular highlight's new (x,y,z) location on the ground plane? Highlight your answer:

-- for the new x coordinate (choose only one of these 3): x<0; [✓ x=0;] x>0;

-- for the new y coordinate (choose only one of these 3): y<0; y=0; y>0;

-- for the new z coordinate (choose only one of these 3): z<0; z=0; z>0;

21)**(6 pts)**Next, we interactively adjust the ground-plane Phong material 'shinyness', its 'Specular Exponent' 'Se' to make the specular highlight appear ***smaller*** on-screen, Did the value of 'Se' get larger or smaller? Highlight your answer:

(Larger Se / Smaller Se) (3pt) Why? (3pts):

The specular light* specular reflectance = Is*Ks*max $(0, (R \cdot V)^{Se})$*Att
And R$\cdot V$ is less than 1. So if Se becomes bigger, the specular light* specular reflectance become smaller and the highlight becomes smaller.

22) Now suppose we re-position the light source directly above the surface, at (0, 0, 40) and in the image from our camera located at (0, 30, 40) we find the center of the specular highlight has moved away from the origin (0,0,0) on the ground plane. What is the new ground-plane location of the specular highlight?

a) **(2 pts)** the x coordinate: 0.0

b) **(2 pts)** the y coordinate: ______15.0__________ (z=0 on ground plane)

23) Find the on-screen color for the ground-plane point at the origin (0,0,0).
With no distance attenuation (Att = 1.0), compute colors ($0 \leq r,g,b < 1$) for each of the Blinn-Phong lighting terms:

a) **(3 pts)** Emissive: (r,g,b) = (0.1, 0.05, 0.20)

b) **(3 pts)** Ambient: (r,g,b) = (0.04,0.06,0.06)

c) **(3 pts)** Diffuse: (r,g,b) = (0.0, 0.5, 0.0)

d) **(3 pts)** Blinn-Phong Specular: (r,g,b) = (0.85, 0.85, 0.85)

24) **(6pts)** Suppose we keep the light-source stationary and move our camera to a different location, but continue to aim it's 'lookAt' the surface point at the origin. Will any of the color components of any ground-plane point change on-screen as we move the camera?

(Highlight your answers)

Ambient: (yes / [✓ no])(1pt) Why? Explain (1pt)

Ambient=Ia*Ka, so it doesn't depend on the camera's position and direction.

Diffuse: (yes / [✓ no])(1pt) Why? Explain(1pt)

Diffuse=Kd*Id*Att*max(0, (N$\cdot L$)), so it doesn't depend on the camera's position and direction.

Specular: ([✓ yes] / no)(1pt) Why? Explain(1pt)

Specular=Ks*Is*Att*max $(0, (N \cdot \mathrm{H})^{Se})$,where H=(V+L)/2, that means it depends on view point(eyedirection).

Thank you for taking Introduction to Computer Graphics: I hope you will find it contents interesting and useful!
If you'd like to learn about graphics by writing a ray-tracer and a particle-system, try Intermediate Graphics, Spring qtr.