NetID !!!SOLUTION!!

CS351-1 **Test C: Better Lights & Materials**

Winter Quarter, 2018 **3 pages, 100 pts, 7% of grade**

Definitions:
L is the 3D position of a point-light source,
P is the 3D position of a point located on a smooth, very-shiny, mirror-like surface of a 3D part, and
N is the unit-length 3D vector perpendicular to the surface at point **P**, aimed outwards from the 3D part, and
C is the 3D position of a perspective camera's center-of-projection (COP).
The light at **L** illuminates the surface at point **P**, and the camera at **C** views the point **P** on the surface.
For each of the points **L**, **P** and **C** we know their Cartesian world-space coordinate values (x,y,z).
For the unit-length vector **N** we know the Cartesian world-space coordinate values (x,y,z).

The questions below each have one or more correct answers, and zero or more incorrect answers.
To earn all points, HIGHLIGHT every correct answer, and leave unmarked every incorrect answer.
We penalize each HIGHLIGHTed incorrect answer, and also penalize each unmarked correct answer.

1) (**3 pts*6**) The dot-product of the vector (**L - P**) and the vector **N** yields a signed, scalar value. This value is:

a) cos(θ), where θ is the angle measured from the (L-P) vector to the N vector
b) cos(-θ) if the magnitude of vector (**L-P**) is 1.0
c) sin(θ), !FALSE! A·B = ||A|| ||B|| cos(θ)
d) A*c !FALSE! N is normalized; no effect. Missing P e depends on only L and N values.
e) B*cos(θ), where B is a positive scalar whose fixed value depends on only L and P values.
f) Somet !FALSE! several correct answers n are incorrect.

2) (2pts*8) The cross-product of the vector (**L - P**) and the vector **N** yields vector **S**. (e.g. (**L-P**) x **N** ==S)
The cross-product of the vector (**C - P**) and the vector **N** yields vector **T**. (e.g. (**C-P**) x **N** ==**T**)
We define S_mag as the magnitude of the **S** vector, AND T_mag as the magnitude of the **T** vector.

a) The magnitude of the vector (**P - L**) is always S_mag, for any **P** and any **L**.
b) The area of the triangle formed by points **L**, **P** and (**P+N**) is always equal to 0.5*S_mag.
c) For any non-zero S_mag, the vector **S** is parallel to a plane tangent to the surface at point **P**.
~~d) For any non-zero T_mag, the vector **S** is parallel to a plane tangent to the surface at p~~ **2d has typo: +2pts free credit for all !**
e) If both **S** and !FALSE! unrelated camera & light positions! parallel (e.g. **S** x **T** == **0**)
f) If the cross-product of the vector (**C-P**) and the vector **N** equals the vector **-S**, then the camera will always see a specular highlight that appears to be centered at point **P** on the 3D part's shiny surface.
g) If S_mag >0 and T_mag >0, then the cross-product of vector **S** and **T** yields a vector parallel to **N**.

3) (2pts*8) Using the camera positioned at point **C** and aimed to view surface-point **P**, we use Blinn-Phong lighting to correctly compute the onscreen appearance of the smooth, shiny surface of our 3D part. On the same 3D surface as 3D point P, the camera sees a sharp specular highlight whose center is at 3D point **Q**. Then:

a) For most, b !FALSE! almost ALL light or camera position changes modify highlight positions m point **Q**.
b) For most, but not all light-source movements, the specular highlight will move away from point **Q**.
c) If we move the camera in a straight line from point **C** to a new point **C2** exactly half-way between **C** and **Q**, the specular highlight will not move: it stays centered at **Q**. (Specifically: **C2** = **C** + 0.5*(**Q-C**))
d) If we know that (**C-P**) = A*N and (**L-P**) = B***N** where A>0, B>0 then we also know that **P - Q** == **0**.
e) If we !FALSE when normal points away from camera and headlight **Q** == **0**.
f) If **P - Q** == **0**, then the 3D points **L**, **C**, **P** and (**P+N**) are all in the same 3D plane.
g) If the !FALSE! in e), we know that half-vector is parallel to N vector; thus ALL are co-planar. L = C = P+(1,1,1)?
h)If the 3D points **L**,**C**,**P** and (**P+N**)are all in the same 3D plane, then point **Q** must also be in that plane.

4) (3pts*6) Using the camera positioned at point **C** and aimed to view surface-point **P**, use Blinn-Phong lighting to correctly compute the onscreen appearance of the smooth, shiny surface of our 3D part. However, we set new light source properties: Ia = (0.1, 0.1, 0.1); Id = (0.9, 0.9, 0.9), Is = (0,0,0); and new materials properties: Ke = (0.2, 0.1, 0.0); Ka=(0.0, 1.0, 0.0); Kd=(0.4, 0.6, 0.8), Is = (0,0,0); These settings eliminate any specular highlights on the surface, and attempts to mimic burning charcoal: (https://www.youtube.com/watch?v=2XsiKFrOUB4): For this camera, material, and light:

a) The on-screen color for point **P** is never black (0,0,0) unless we modify its material properties

b) For most, b !FALSE! for **ALL** camera movements! ONLY light-position (N dot L) affects diffuse term! ed.

c) For most, but not all light-source movements, the on-screen color varies for surface point **P**.

d) If we tur !FALSE! Material is emissive – you'll see red-yellow glow for all camera positions (0.2, 0.1, 0.0) e see `at surface point P is black -- (0,0,0)

e) If we turn on the light, restoring Ia, Id, and Is to their original values, and then change the material's diffuse reflectance to black (e.g. Kd=0,0,0), then any and all camera movements will have no effect on the on-screen color of point P (assume we aim the camera to ensure point P is always on-screen).

f) If we turn on the light, restoring Ia, Id, and Is to their original values, and then change the material's diffuse reflectance to black (e.g. Kd=0,0,0), then any and all light-source movements will have no effect the on-screen color of point P (assume our Blinn-Phong lighting has no attenuation term).

5. (2pts * 16) True/False: **2pts each** (See Canvas →WebGL: the Full Specifications→GLSL-ES 1.0 Spec) Copy-and-paste your choice of these highlighted answers: "True" or "False")

a)_False GLSL ES allows user-defined functions ONLY within the Vertex shader, and never in the Fragment Shader. The Fragment Shader allows the '`main()`' function but no others to ensure fast, simple, simultaneous results for all pixels.

b)_True_Unlike C/C++, GLSL ES functions that use the **`inout`** qualifier for a formal parameter can change the value(s) of a non-constant variable used as the argument for that parameter. (unsure? Look up GLSL 'argument' and 'parameter').

c)_True_In the Phong lighting model, if we change by 15 degrees the direction of the view vector for one vertex illuminated by one light source, the on-screen result from the diffuse lighting term for that vertex will not change.

d)_True_Unlike the C/C++ **`for() loop`**, the number of iterations of a GLSL ES **`for()`** loop **must** be specified at compile time, and not at run-time; for example, it cannot be specified by the value of an externally-set **`uniform`** variable.

e)_False The SIMD architecture of the GPU prevents he GLSL ES preprocessor support for conditional compiling – it must compile ALL GLSL statements, and prohibits directives such as **`#if`**, **`#ifdef`**, **`#ifndef`**, and **`#endif`**.

f)_False GLSL ES offers separately-named floating point matrix data types for each of these vector/matrix sizes: 1x2, 2x1, 2x2; 1x3, 3x1, 2x3, 3x2, 3x3; 1x4, 4x1, 2x4, 4x2, 3x4, 4x3, 4x4, and NxM matrices.

g)_False GLSL ES supports type-renaming (the '**`typedef`**' mechanism); used with GLSL **`struct`** naming.

h)_True_A change in lighting model from Phong to Blinn-Phong affects only the specular term.

j) ___True___ Suppose we build a ‘model’ matrix by combining ONLY rotate, translate, and uniform scaling (same scale factor applied to x,y, and z). If this ‘model’ matrix that correctly transforms vertex positions from model coordinates to world coordinates, then it will ALWAYS correctly transform the vertex normal vectors as well.

k)___False___ A GLSL shader program that performs all its calculations for Phong lighting and Phong shading the ‘world’ coordinate system will sometimes, but not always, give different on-screen results than a GLSL shader program that performs those same calculations in the ‘camera’ or ‘view’ coordinate system, where the camera’s COP (center of projection) is the origin and the view-frustum extends outwards in the -z direction.

m) ___False___ The Phong lighting model can sometimes computed correctly in either the ‘world’ or the ‘eye’ coordinate system, but the Blinn-Phong lighting model MUST use the ‘world’ coordinate system for correct results.

n) ___True___ When implemented correctly, specular highlights formed by the Blinn-Phong lighting method and those formed by the Phong lighting method BOTH put their brightest location (the location where H==N) at the same place.

p)___True___ When implemented correctly, specular highlights formed by the Blinn-Phong lighting method are always slightly LARGER on-screen than specular highlights formed by Phong lighting.

q)___True___ The Cook-Torrance lighting model creates far more convincing renderings of metallic surfaces because its statistical models of ‘microfacets’ more accurately capture the angle-dependence of specular-reflectances.

r)___False___ Just like OpenGL, the WebGL 1.0 texture-mapping facilities accept texture-maps of any height and any width up to 4096 pixels.

s)___True___ WebGL programs that apply texture mapping will sometimes, but not always, fail to work properly if users simply drag-and-drop their HTML file into a browser. These CORS-related difficulties can ALWAYS be eliminated by proper use of a local HTTP server, as demonstrated in class.

END!