NAME:

Computer Science 426 Midterm
5/3/07, 1:30PM-2:50PM

This test is 4 questions. Do all of your work on these pages (use the back for scratch space), giving the answer in the space provided. This is a closed-book exam -- you may use one-page of notes with writing on both sides during the exam. **Please write out and sign the Honor Code pledge before turning in the test.**

``I pledge my honor that I have not violated the Honor Code during this examination.''

Question	Score
1	
2	
3	
4	
Total	

Q1: Rendering Concepts (25 Points)

a) Several algorithms in computer graphics employ a strategy of performing a preliminary check ("trivial reject") to quickly determine whether a full computation is required before proceeding. Please list three examples in which this strategy is used during rendering (with any algorithm):

b) Please provide three examples of algorithms where *coherence* is leveraged during rendering. For each example, please describe clearly how coherence improves speed and/or accuracy.

c) Please provide three examples of algorithms where *integration* is used during rendering. For each example, please describe clearly *what function* is being integrated over *what domain*.

Q1: Rendering Concepts (25 Points)

d) Please provide three examples of algorithms where *discrete sampling* is used during rendering. For each example, please describe clearly *what function* is being sampled and if/how aliasing is avoided.

e) Please provide three examples of algorithms where *hierarchal representations* are used during rendering. For each example, please describe clearly how the hierarchy improves speed and/or accuracy.

e) Please provide three examples of algorithms where *adaptive refinement* is used during rendering. For each example, please describe clearly what is being refined, when it is refined, and how the adaptive algorithm improves speed and/or accuracy.

Q2: Illumination (25 Points)

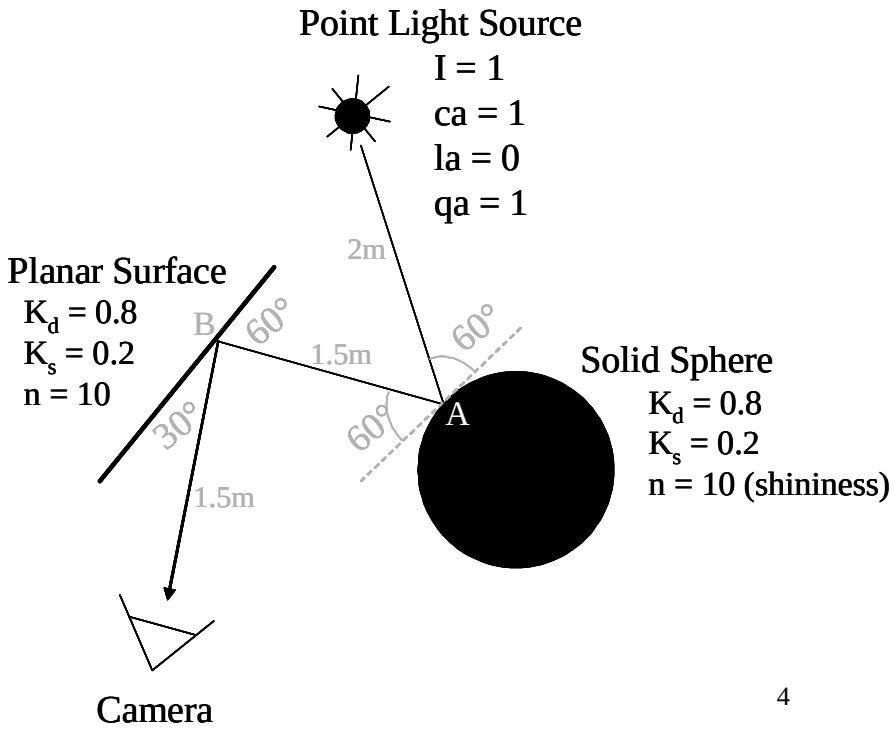
Consider the path of light shown in the figure below. The light leaves a point light source with the intensity (I) and attenuation (ca, la, qa) parameters listed for one wavelength, reflects off a solid sphere with reflectance properties shown (K_d, K_s, and n), then reflects off a planar surface with the same surface properties, and finally arrives at a camera. For this problem, please write expressions for the intensity of light at one wavelength at various points along the path – your answers may contain trigonometric and other mathematical expressions (e.g., cosine, sine, sqrt, etc.) and/or refer to expressions written for other parts of the question.

a) Please write an expression for the intensity of light traveling along this path when it *leaves the point light source*.

b) Please write an expression for the intensity of light traveling along this path when it *arrives at the solid sphere* (at the point marked 'A').

c) Please write an expression for the intensity of light traveling along this path when it *leaves the solid sphere* (at the point marked 'A').

d) Please write an expression for the intensity of light traveling along this path when it *arrives at the planar surface* (at the point marked 'B').



Q2: Illumination (cont)

d) Please write an expression for the intensity of light traveling along the path shown on the previous page when it *leaves the planar surface* (at the point marked 'B').

e) Please write an expression for the intensity of light traveling along the path shown on the previous page when it *arrives at the camera*.

f) Are there other paths through the scene that contribute to the intensity of light arriving at the camera from the same direction as the path shown on the previous page? If no, explain why. If yes, draw one such path on the diagram at the bottom of the previous page.

g) Among the following rendering algorithms, circle the ones that will model the path shown on the previous page:

Basic ray tracing (as implemented in assignment #2)

Distribution ray tracing

Radiosity

Monte Carlo path tracing

Polygon rendering pipeline (as implemented in OpenGL)

Bidirectional path tracing

h) What are the units of the intensity of light traveling along a ray path?

Q3: Scan Conversion (25 Points)

Please provide pseudo-code for the function `ScanTriangle` that uses a sweep-line algorithm to scan convert a single triangle with Gouraud Shading and texture mapping. The input arguments to the function are three vertices specified by 2D screen coordinate positions (P1, P2, and P3), colors (C1, C2, and C3), and texture coordinates (T1, T2, and T3). Within your code, you should use `SetPixel(P, C)` to draw a pixel at position P with color C. You should also use `C = GetTexture(T)` to get the color of a modulating texture at coordinates T. Your pseudo-code should be detailed enough to demonstrate that you understand how the sweep-line algorithm works and the issues encountered in Gouraud shading and texture mapping.

Q4: Rendering Pipeline (25 Points)

This question concerns the coordinate transformations performed by the polygon rendering pipeline for the scene provided below when rendered into a viewport with corners at (100,100) and (500,500). Note: the syntax for the scene file format is provided on the last page of the test.

```
begin -1
  1 0 0 0
  0 2 0 2
  0 0 1 0
  0 0 0 1
  tri   -1    0 1 0    0 0 0    0 0 1
end

camera   -4 0 0    1 0 0    0 1 0    PI/4    1 10
```

a) Please draw the scene graph hierarchy for this scene, clearly labeling the transformation matrix and the triangle.

b) What are the (x,y,z) coordinates of the triangle's first vertex (highlighted in bold) in the *modeling coordinate system*?

c) What are the (x,y,z) coordinates of the same vertex after it has been transformed into the *world coordinate system*?

d) What are the (x,y,z) coordinates of the same vertex after it has been transformed into the *camera coordinate system*?

e) What are the (x,y) coordinates of the same vertex after it has been transformed into the *image coordinate system*?

Q4: Rendering Pipeline (Cont)

f) Please draw the triangle and the camera's view frustum in the world coordinate system as seen from some other camera view.

g) Please draw the triangle in the image coordinate system as seen from the camera view given in the scene file.

COS 426 SCENE FILE SYNTAX

begin
mat_id
m11 m21 m31 m41
m12 m22 m32 m42
m13 m23 m33 m43
m14 m24 m34 m44
... commands ...
end

This pair of commands defines a node that will be added to the current scene-graph with the specified material and transformation context. All geometric primitives within the *begin* .. and associated *end* command are subject to the 4x4 transformation matrix given in the *begin* command, and any such geometric primitive that does not already have a material assigned (i.e., its *mat_id* was -1) will adopt the material indicated by *mat_id* (where *mat_id* is the index of a material in the same file).

Groups may be nested, permitting the specification of a transformation heirarchy. Shapes within nested groups are subject, in order, to the transformation contexts of all their enclosing groups. The total transformation context of a given shape is determined, then, by starting with the matrix of the root enclosing group, and concatenating additional matrices on the right as we decend into nested groups, until we reach the geometric primitives. The transformation context of a group is applicable only to its shapes and sub groups, so we must remove matrices from the right as we ascend back up the heirarchy.

---

tri
mat_id
x1 y1 z1
x2 y2 z2
x3 y3 z3

This command defines a triangle with coordinates *(x1,y1,z1)*, *(x2,y2,z2)*, and *(x3,y3,z3)*.

---

camera
eye_x eye_y eye_z
towards_x towards_y towards_z
up_x up_y up_z
xfov neardist fardist

This command defines a perspective camera in the scene. *(eye_x,eye_y,eye_z)* is the position of the camera in world coordinates, *(towards_x,towards_y,towards_z)* is a vector describing the direction the camera is pointing, and *(up_x,up_y,up_z)* is a vector in the up direction. The half-width angle of the viewing frustum is given by *xfov*, and the half-height angle is set based on the image aspect ratio according to *yfov = atan(tan(xfov)*height/width)*, where *width* and *height* are the width and height of the output image.