University of California San Diego
Department of Computer Science
CSE 167: Introduction to Computer Graphics
Spring Quarter 2015
Midterm Examination #2
Thursday, May 21st, 2015
Instructor: Dr. Jürgen P. Schulze

Name:_______________________________________

Your answers must include all steps of your derivations, or points will be deducted.

This is closed book exam. You may not use electronic devices, notes, textbooks or other written materials.

Good luck!

Do not write below this line

Exercise	Max.	Points
1	10	
2	10	
3	10	
4	10	
5	8	
6	12	
7	10	
8	10	
Total	80	

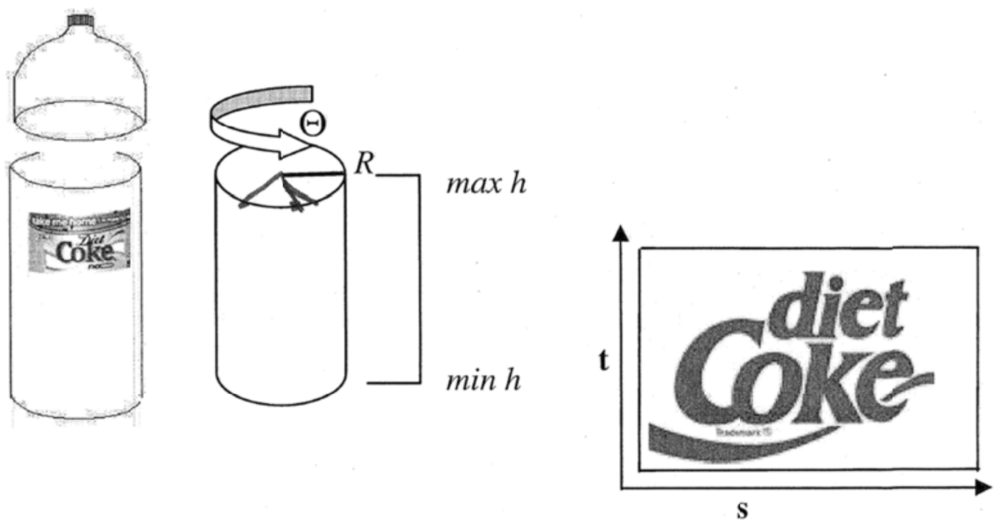
1) Texture Mapping (10 Points)

You are given the task of putting the diet Coke label on the two liter bottle as shown (it only goes 25% around the bottle and is positioned as shown).

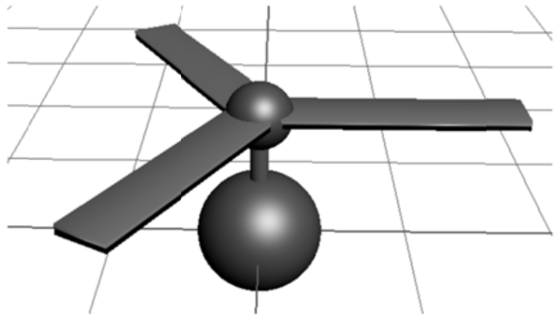
The bottle has two parts: a cylindrical bottom and a conical top. The bottom of the label should be half-way up the cylindrical part of the bottle, and its height should be 25% of the height of the cylindrical part of the bottle. What are the texture coordinates *s* and *t* which perform this operation? (In other words: express *s* and *t* as functions of *min h*, *max h*, *h* and *θ*.)

Be sure to describe any assumptions you make. Although the texture is not square, you can assume the range in *s* and *t* is [0..1].

$x = R\cos(\Theta)$
$y = R\sin(\Theta)$
$z = h$



2) Scene Graph (10 Points)



The figure above shows a simple helicopter model. It contains a body, a vertical axis, a top sphere (to hold the main propeller) and three rotor blades. The grid in the background indicates the location of the x-y plane. The z axis points up.

Assume that the following geometric objects are available as scenegraph geometry nodes, their object coordinate systems are centered in the respective centers of the shapes: **Body**, **Axis**, **Top**, **Blade**.

The following transformation nodes are available:

Tz translation along z axis (by any amount)
Txy translation in x/y plane (by any amount)
S scale operation (uniform, non-uniform, any amount)
Rz rotation about z axis (by any angle)

a) Create a scene graph for the model and draw it below. (6 points)

b) How would you use the scene graph to animate the **vertical ascent** (take-off) of the helicopter? Additionally, as the helicopter lifts, the three rotor blades should **rotate** around the vertical axis. Indicate which nodes of the scene graph need to be modified (mark in graph above) and how. (4 points)

3) Performance Optimization (10 Points)

Briefly describe each of the following rendering performance optimization strategies.

a) 2D impostors for 3D objects (2 points)

b) Adaptive mesh resolution (2 points)

c) Small object culling (2 points)

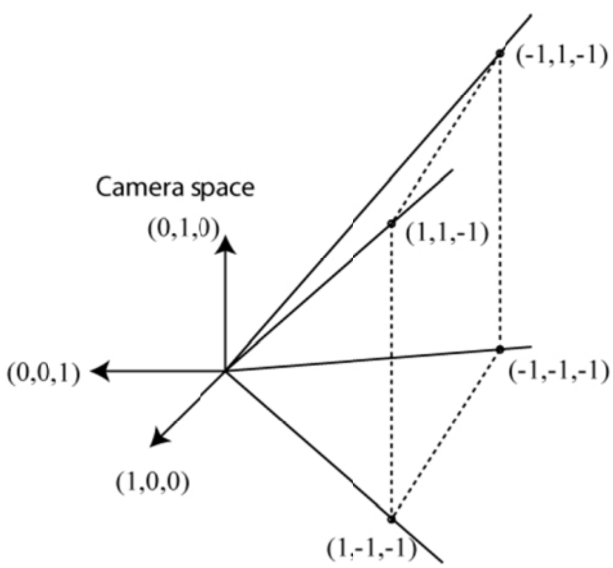
d) Backface culling (2 points)

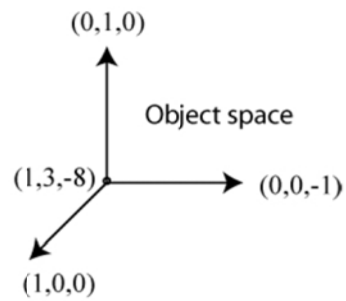
e) Degenerate culling (2 points)

4) View Frustum Culling (10 Points)

Given the perspective view frustum shown in the figure below. The top bounding plane of the view frustum is determined by the plane going through the points (0, 0, 0), (1, 1,−1), and (−1, 1,−1) in camera coordinates. Note that the other bounding planes will not be relevant to this problem. In addition, there is an object coordinate system defined by basis vectors (0, 1, 0), (1, 0, 0), (0, 0,−1) and the origin (1, 3,−8) in camera coordinates. Note that the order of the basis vectors matters!

Assume there is an object with a bounding sphere with radius 2 centered at (8, 1, 1) in object coordinates. Determine if this bounding sphere intersects with the top bounding plane of the view frustum. You should do this by transforming the center of the bounding sphere from object to camera coordinates. Then you need to compute the distance from the bounding sphere center in camera coordinates to the top bounding plane.





5) Shader Programs (8 Points)

a) GLSL has three qualifiers for inputs and outputs in a shader: attribute, uniform, varying. Explain what each of them means, and in which shader(s) they are available (vertex and/or fragment shader). (6 points)

<u>attribute:</u>

<u>uniform:</u>

<u>varying:</u>

b) Assuming that both vertex and fragment programs have been loaded for rendering a triangle. How many times will the vertex program get executed, how many times the fragment program? (2 points)

6) Light and Shade (12 Points)

For homework project 6, a fragment shader was provided which gave your scene diffuse shading using the Lambertian reflectance model. Below is that exact shader with some elements missing in some lines. Those lines are bolded. Complete the equations or formulas in the underlined areas such that the resulting shader once again gives your scene diffuse shading using the Lambertian reflectance model.

For your reference, this is the complete Blinn-Phong shading model as derived in class:

$$c = \sum_i c_{l_i} \left(k_d \left(\mathbf{L}_i \cdot \mathbf{n} \right) + k_s \left(\mathbf{h}_i \cdot \mathbf{n} \right)^s \right) + k_a c_a$$

Some useful variables, functions, and constructors:

- gl_FrontMaterial
- gl_LightSource
- max()
- dot()
- vec3()
- vec4()

```
#version 120
#extension GL_EXT_gpu_shader4 : enable

varying vec4 position;
varying vec3 normal;

void main()
{
    // Declare variables for intermediate calculations
    vec4 finalColor;
    vec3 lightDir;
    float attenuation;
    float distance;

    // Declare and initialize variables for storing lighting results
    vec3 finalLighting = vec3(gl_FrontMaterial.emission);
    vec3 ambientReflection = vec3(0.0,0.0,0.0);
    vec3 diffuseReflection = vec3(0.0,0.0,0.0);

    // Loop through all 8 lights that the OpenGL fixed function pipeline supports

    for (int i = 0; i < 8; ++i)
    {
        // Calculate the light direction

        lightDir = ________________________________________________________________;
```

```
        // Add the ambient light factor to the ambient accumulator

        ambientReflection += _______________________________________________________


          ___________________________________________________________________________;

        // If the light is behind the surface, continue

        if(________________________________________________________________________)
        {
            continue;
        }

        // Set the default attenuation to none
        // (by the multiplicative identity, this is 1.0)

        attenuation = 1.0;

        // If the light is not a directional light,
        // then apply the appropriate attenuation

        if (0.0 != gl_LightSource[i].position.w)
        {
            distance = length(lightDir);

            attenuation = ___________________________________________________________


              ___________________________________________________________________;

        }

        // Calculate the diffuse reflectance and add it to the diffuse accumulator

        diffuseReflection += _________________________________________________________


          ___________________________________________________________________________;

    }

    // Sum all of the lighting together

   finalLighting = __________________________________________________________________;

   // Return the final lighting

   gl_FragColor = vec4(finalLighting, 1.0);
}
```

7) Parametric Curves (10 Points)

Consider the two points $\mathbf{P_0}$ and $\mathbf{P_1}$ as below. There exists another point **Q** that lies on the straight line connecting $\mathbf{P_0}$ and $\mathbf{P_1}$.

a) Write an equation to represent **Q** in terms of $\mathbf{P_0}$**,** $\mathbf{P_1}$ and a parameter **t**. (4 points)

b) What are the two values of **t** at which **Q** would equal $\mathbf{P_0}$ and $\mathbf{P_1}$? (2 points)

c) From the equation above, calculate the tangent at **Q**. Does the tangent vary at every **Q**? (4 points)

8) Bezier Curves (10 Points)

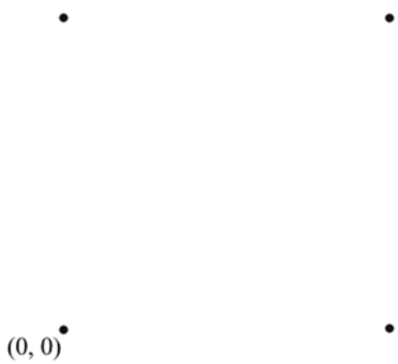
You are a fitting specialist at an upscale clothing store. But not just any clothing store, you work for Bezier Britches, a boutique shop known for selling the finest of bottoms to the busiest of Bezier business men. Your one and only job is to help busy Bezier business men find a pair of britches that best fits.

Sounds easy? In comes Bobby the Bezier Curve. Bobby needs a pair of britches, and fast. But he's so busy that he doesn't even know where his waist is! Before you can properly size him you'll need to find his waist.

Here is what he tells you. Fill in the blanks with the correct terms. (2 points):

- He's a ________ Bezier curve, which means he has 4 ______ points and is a _____ degree polynomial.
- His waist is at t = 0.5.
- His ______ points in order are: <0, 0>, <0, 8>, <8, 8>, <8, 0>

Given the above, you first make an estimate using De Casteljau's algorithm (2 points):



Estimate: (___,___)

Now you have a general idea of where his waist is, but you don't work at just any clothing store, you work at Bezier Britches. Here accuracy is top priority, so an estimate won't cut it. You'll have to solve for his waist directly. Looking at your fitter's manual, you find an equation that looks like it'll do the trick:

$$\dot{q} = \sum_{i=0}^{n}\left(\binom{n}{i}(t)^{i}(1-t)^{n-i} * \dot{p}_i\right)$$

Using the equation above, and what you know about Bobby, solve for his waist. Show work. (5 points):

Your estimate and the actual position should be quite close, why? (1 point)