Name _____

GPU Programming
EE 4702-1
Midterm Examination

Monday, 26 October 2015 13:30–14:20 CDT

Problem 1	_____	(21 pts)
Problem 2	_____	(25 pts)
Problem 3	_____	(12 pts)
Problem 4	_____	(18 pts)
Problem 5	_____	(12 pts)
Problem 6	_____	(12 pts)
Exam Total	_____	(100 pts)

Alias _____

Good Luck!

Problem 1: [21 pts] In the code below three coordinates are defined, `ph` (P_h, head), `pt` (P_t, tail), and `ps` (P_s, surface).

(a) Complete the code below so that it renders a cylinder with axis $\overrightarrow{P_h P_t}$, with P_h at the top (flat part) of the cylinder, P_t at the bottom (the other flat part), and with P_s on the surface. Render only the curved parts, not the flat parts. The cylinder surface should be approximated by `slices` flat sides.

☐ Show code before the loop, such as for `ax` and `ay`.

☐ Show code in the loop that emits vertex ☐ coordinates and ☐ normals.

```
pCoor ph( 1.5, 0, -3.2 );
pCoor pt( 0,   5, -5 );
pCoor ps( 9,   6, -9 );
const int slices = 100;
const float delta_theta = 2 * M_PI / slices;
pNorm axis = pt - ph;
// Some of the solution goes here.




pVect ax =                                                    ; // FILL IN

pVect ay =                                                    ; // FILL IN




glBegin(GL_TRIANGLE_STRIP);
glColor3ub( 256, 89, 16 );
for ( int i=0; i<=slices; i++ )
  {
    const float theta = i * delta_theta;
    pVect n = cos(theta) * ax + sin(theta) * ay;
    // Some of the solution goes here.



    pCoor p1 =                                                ; // FILL IN
    glVertex3fv(p1);



    pCoor p2 =                                                ; // FILL IN
    glVertex3fv(p2);




  }
glEnd();
```

Problem 2: [25 pts] The code below, taken from the solution to Homework 2, renders the surface of a truss. Recall that the truss might be a part of a spinning gyroscope.

```
glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(3, GL_FLOAT, sizeof(pCoor), coord_buffer);
glEnableClientState(GL_NORMAL_ARRAY);
glNormalPointer(GL_FLOAT, 0, norm_buffer);
glColor3fv(strip->color);
glDrawArrays(GL_TRIANGLE_STRIP, 0, elts);
glDisableClientState(GL_NORMAL_ARRAY);
glDisableClientState(GL_VERTEX_ARRAY);
```

(*a*) How much data is sent from the CPU to the GPU to render a frame based on the code above. Your answer should be in terms of the variables above.

☐ The amount of data sent per frame is ... ☐ Indicate unit.

(*b*) How could someone who is not familiar with Homework 2 be reasonably sure that the code above uses client arrays and not buffer objects? What would be different *in the parts shown* if buffer objects were used?

☐ We would guess code doesn't use buffer objects because ....

☐ Show changes to parts shown if buffer objects were used.

(*c*) Explain why using buffer objects in the code above would not make a big difference, whereas using buffer objects to render the balls (used to show nodes in the truss) in the same simulation does make a big difference?

☐ Buffer objects would not make a big difference for truss surface because ...

☐ Buffer objects do make a big difference for balls (part of truss) because ...

Problem 2, continued: The Homework 2 code from the previous page is repeated below.

```
glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(3, GL_FLOAT, sizeof(pCoor), coord_buffer);
glEnableClientState(GL_NORMAL_ARRAY);
glNormalPointer(GL_FLOAT, 0, norm_buffer);
glColor3fv(strip->color);
glDrawArrays(GL_TRIANGLE_STRIP, 0, elts);
glDisableClientState(GL_NORMAL_ARRAY);
glDisableClientState(GL_VERTEX_ARRAY);
```

(*d*) Re-write the code so that it does not use client arrays, but instead uses calls like `glVertex`. *Hint: Only a few lines of code needed.*

☐ Re-write code so that calls like `glVertex` is used.

Problem 3: [12 pts] The code fragment below is from a demo-10 vertex shader.

```
void vs_main_basic() {
  gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
  // Other code removed.
}
```

(*a*) Identity the variable that uses the storage types given below.

☐ Which variable is a vertex shader input?

☐ Which variable is a uniform variable?

☐ Which variable is a vertex shader output?

(*b*) Two of these variables appear in the OpenGL shading language compatibility profile but not in the undeprecated OpenGL SL standard, meaning that they aren't truly needed but are being kept so as not to break old software. Identify the variable that *is* truly needed and explain why.

☐ The undeprecated, truly needed variable is:

☐ It is truly needed because ...

Problem 4: [18 pts] Answer each question below.

(*a*) The interface block below is used as a fragment shader input. Among other things the fragment shader does lighting calculations. Explain what the `flat` qualifier does and why it is needed.

```
in Data_GF
{
  vec3 var_normal_e;
  vec4 var_vertex_e;
  flat vec4 color;
};
```

☐ The `flat` qualifier ...

☐ The `flat` qualifier helps this particular shader program by ...

(*b*) Suppose the depth (z) test was turned off. How would that affect the rendered image?

☐ Rendered image with depth test turned of would appear ...

(*c*) Describe what textures might be used for in a graphics program.

☐ An example of what a texture might be used for is ...

Problem 5: [12 pts] Answer the following questions.

(*a*) The code below renders two triangles. Add the minimum amount of code to make the first one blue and the second one orange (RGB:1,.35,.06). *Hint: This is an easy problem, don't overthink it. Note: The shade of orange is from the New York Mets logo.*

☐ Add minimum amount of code so that the first triangle is blue and the second is orange.

```
glBegin(GL_TRIANGLES);

glVertex3f( 10,  20,  0 );
glVertex3f( 15,  20,  0 );
glVertex3f( 10,   0,  0 );

glVertex3f( 30,  20,  0 );
glVertex3f( 30,   0,  0 );
glVertex3f( 25,   0,  0 );

glend();
```

(*b*) The code below renders two triangles. Add texture coordinates so that when the two triangles are put together they show the entire texture. The texture has already been set up, just provide the coordinates.

☐ Add OpenGL calls to provide texture coordinates.

```
glBegin(GL_TRIANGLES);

glVertex3f( 10,  20,  0 );
glVertex3f( 15,  20,  0 );
glVertex3f( 10,   0,  0 );

glVertex3f( 30,  20,  0 );
glVertex3f( 30,   0,  0 );
glVertex3f( 25,   0,  0 );

glend();
```

Problem 6: [12 pts] Clipping is performed between the output of the geometry shader and the input to the rasterizer.

(*a*) Provide a simple example of a triangle that can be completely clipped. The example should include the triangle's vertex coordinates in clip space.

☐ Easily completely clipable triangle coordinates:

(*b*) Provide an example of a primitive for which clipping would be difficult or tedious.

☐ Illustration (not coordinates) of a triangle that is tedious to clip.

☐ Briefly explain what makes it tedious.

(*c*) Suppose due to some error the clipping stage doesn't clip anything. Explain why there would be no difference in the appearance of the rendered image.

☐ Image no different if clip stage doesn't clip because ...

(*d*) Again, suppose due to some error the clipping stage doesn't clip anything. Explain why performance would suffer.

☐ Performance suffers because ...