Name

GPU Programming
EE 4702-1
Midterm Examination
Friday, 6 November 2009, 9:40–10:30 CST

Problem 1  ________  (40 pts)
Problem 2  ________  (20 pts)
Problem 3  ________  (20 pts)
Problem 4  ________  (20 pts)
Alias  __________________________
Exam Total  ________  (100 pts)

Good Luck!
Problem 1: [40 pts] Write OpenGL code to render a filled circle (a disc) of radius $r$, centered at the origin and with normal $(0,1,0)$.

- The distance between all vertices in a primitive should be approximately $\text{vert\_dist}$.
- Use triangle strips. Multiple strips are okay, but there should be no easy way to make the strips longer.
- Use `glVertex3f` calls, don’t try to construct arrays.
- Don’t specify colors, normals, or other attributes.
- Assume transformations, lighting, etc, have all been set up. Start with `glBegin`.

☐ Code rendering a circle of radius $r$, center at origin.

☐ Vertex distance about $\text{vert\_dist}$.

☐ Good use of strips.

☐ Code reasonably efficient.
Problem 2: [20 pts] Answer the normal questions below.

(a) Show an expression for the normal to triangle $ABC$, where $A$, $B$, and $C$ are the vertex coordinates.

Normal to $ABC$

(b) In the code sample below the pair of triangles is rendered using two different methods, identified as Method 1 and Method 2.

Describe the difference in appearance of the triangles rendered using Method 1 and Method 2 when diffuse lighting is used and $\text{norm}_{AB}C \neq \text{norm}_{CBD}$.

```c
pVect norm_ABC = find_normal(A,B,C); pVect norm_CBD = find_normal(C,B,D);

// Method 1
glBegin(GL_TRIANGLES);
glNormalfv(norm_ABC);
glVertex3fv(A); glVertex3fv(B); glVertex3fv(C);
glNormalfv(norm_CBD);
glVertex3fv(C); glVertex3fv(B); glVertex3fv(D);
glEnd();

// Method 2
pNorm norm_X = norm_ABC + norm_CBD; // Sum of two vectors normalized.
glBegin(GL_TRIANGLES);
glNormalfv(norm_ABC); glVertex3fv(A);
glNormalfv(norm_X); glVertex3fv(B); glVertex3fv(C);
glVertex3fv(C); glVertex3fv(B);
glNormalfv(norm_CBD); glVertex3fv(D);
glEnd();
```

Difference in appearance between Method 1 triangles and Method 2 triangles.
Problem 3: [20 pts] Consider the three methods of specifying vertices shown below.

```c
switch ( opt_method ) {

    case VM_Individual: { /// Use Individual Vertices
        glBegin(GL_TRIANGLE_STRIP);
        for ( int i=0; i<coords_size; i+=3 ) {
            glNormal3f(coords[i],coords[i+1],coords[i+2]);
            glVertex3f(coords[i],coords[i+1],coords[i+2]);
        }
        glEnd();
        break; }

    case VM_Array: { /// Use Vertex Arrays
        glNormalPointer(GL_FLOAT,0,coords);
        glEnableClientState(GL_NORMAL_ARRAY);
        glVertexPointer(3,GL_FLOAT,3*sizeof(float),coords);
        glEnableClientState(GL_VERTEX_ARRAY);
        glDrawArrays(GL_TRIANGLE_STRIP,0,coords_size/3);
        glDisableClientState(GL_NORMAL_ARRAY);
        glDisableClientState(GL_VERTEX_ARRAY);
        break; }

    case VM_Buffer: { /// Use Buffer Objects
        glBindBuffer(GL_ARRAY_BUFFER,gpu_buffer);
        glVertexPointer(3,GL_FLOAT,3*sizeof(float),NULL);
        glEnableClientState(GL_VERTEX_ARRAY);
        glNormalPointer(GL_FLOAT,0,NULL);
        glEnableClientState(GL_NORMAL_ARRAY);
        glDrawArrays(GL_TRIANGLE_STRIP,0,coords_size/3);
        glBindBuffer(GL_ARRAY_BUFFER,0);
        glDisableClientState(GL_NORMAL_ARRAY);
        glDisableClientState(GL_VERTEX_ARRAY);
        break; } }
```

(a) Why is the individual vertex method slower than the others?

☐ Reason that individual vertex method slower than the others.

(b) When used the right way the method using buffer objects is much faster than the others.

☐ Why are buffer objects faster than vertex arrays, when used the right way?

☐ Describe a situation in which the buffer object and vertex array method would have about the same performance.
Problem 4: [20 pts] Answer each question below.

(a) OpenGL allows different material property colors for ambient, diffuse, emissive, and specular lighting. However only a few of these can be changed from vertex to vertex. Why?

☐ Why can’t all material properties be changed each vertex?

(b) OpenGL lets you specify any transformation matrix for the projection, it doesn’t have to be a frustum.

☐ Describe the appearance of a scene in which the projection matrix were identity. What parts of world space would be visible?

(c) Textures are provided or used to generate multiple MIPMAP levels. Explain what a MIPMAP level is and why it is necessary.

☐ What is a MIPMAP level?

☐ It is a resolution level. Level 0 is the original image. Level 1 is an image with half the number of pixels along each axis (total number of pixels is $\frac{1}{4}$), and so on.

☐ Why is it necessary?

☐ To avoid problems when a level-0 texel is much larger or smaller than a pixel. If the level-0 texel is much smaller (the texture image will appear tiny) we would like to use some kind of average of all the texels that cover the pixel (otherwise details, such as a picket fence, can disappear or not appear properly). The pixels in a higher MIPMAP level are averages of several pixels in a lower level. Consider a texture with a white picket fence and a black background and a situation where pixels are larger than texels. If we used just the level-0 image we might choose the texels that included only the black background between the white fence, so the fence would be invisible. If we used a prepared MIPMAP, say at level $l$, we would retrieve a texel that was a blend of background and the white fence, appearing gray.