CSE 40166 Final Exam, Fall 2010

Short Answer

NAME: _____

1. *(5 points)* What is the homogeneous coordinate system, and *why* does OpenGL use it?

2. *(8 points)* What are the transformations that a vertex undergoes in the OpenGL vertex transformation pipeline? Which of these transformations can the user specify, and what commands are needed to do so? Additionally, specify the space that the vertex is in at each stage.

3) *(5 points)* When using custom vertex and fragment shaders, we disable the default functionality of these stages in the graphics processing pipeline. What functionality must we replace if we want to emulate the fixed-function pipeline? (In other words: what are the simplest vertex and fragment shaders we would need to be able to replicate the default OpenGL pipeline? Actual code is not necessary.)

4) What are the major input(s) and output(s) to:

a) *(3 points)* Vertex shaders? (Give GLSL keywords and descriptions.)

b) *(3 points)* Fragment shaders? (Give GLSL keywords and descriptions.)

5) *(5 points)* What is the purpose of the rasterizer -- specifically, how is it meaningful in the context of vertex and fragment shaders?

6) *(4 points)* Suppose we have the program below:

```
void display()
{
       glClearBuffer(GL_COLOR_BUFFER_BIT);
       draw_cute_bunny(); // Draw cute bunny model
       glFlush();
}
```

It runs great on the lab machine and shows a cute little bunny. Unfortunately on the Professor's old Thinkpad it runs a little slowly, and suffers from flickering and tearing when the camera is moved too quickly. Describe two techniques or methods that could be used to improve the speed and smoothness of this interactive application.

7) *(4 points)* What features or properties of OpenGL make it unsuitable for rendering photo-realistic graphics? What alternative rendering techniques can be used to generate more realistic images?

8) *(5 points)* Describe in words how a simple ray tracer using ray casting works.

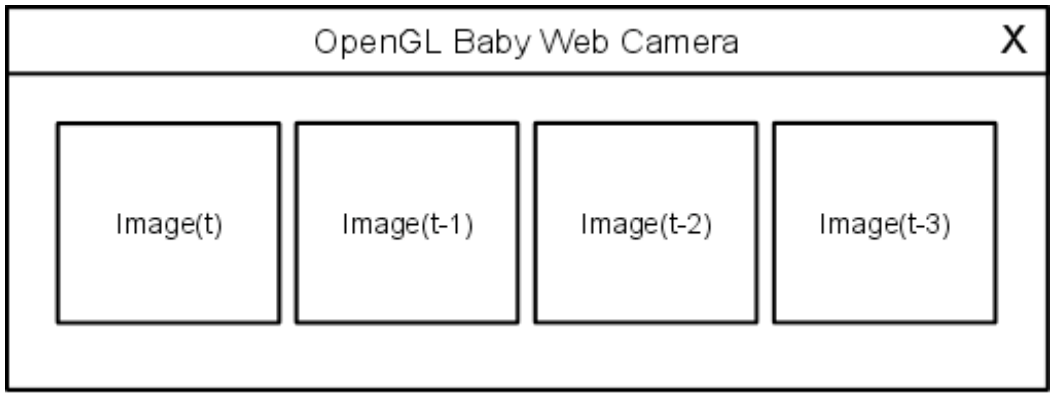
Long Answer

1) ***(15 points)*** **Both Peter Bui and Steve Kurtz are having a baby next Spring. Since being a graduate student is a full-time job with demanding work hours, each father needs to be able to monitor his bundle of joy remotely. Being the responsible fathers that they are, Peter and Steve purchase Super-Baby-Webcams. Unfortunately, the manufacturers are noobs and only provide Windows software to view the camera (Steve is an Apple cultist and Peter is a Linux zealot). Help Peter and Steve by writing an OpenGL program that allows them to view the web camera.**

The goal of the program is to do the following:

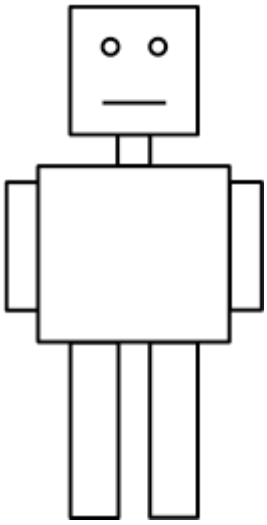
- Every second fetch the latest image from the web camera. Assume you have a function `web_camera_read(unsigned char *buffer, int *width, int *height)` that returns the image captured by the web camera as a buffer of **RGB** values.

- Display the current captured image on the screen along with the previous 3 images as shown below:



Using **pseudo-code,** write the outline of the core of application described above. *(**Hint:** include at least the **timer** and **display** callback).*

2) Consider the following animated robot, which moves its legs as it walks:



a) *(4 points)* Would you use instance modeling or a scene graph to construct the robot above? Why?

b) *(6 points)* Write scene graph pseudo-code that connects the parts of the robot (body, neck, head, arms, legs) to form the model above. Assume you have nodes for each component.

Example:

```
draw_robot()
{
       Node *body = new Body();
       Node *neck = new Neck();

       body->connect(neck);  // Connects neck node (child) to body node (parent)
       // continue …
}
```

c) *(5 points)* Add two animation nodes to your scene graph to enable the legs to rotate as the robot walks. Write pseudo-code for the animation node (hint: show the transformations used to rotate the legs).

3)

a) *(4 points)* Diagram the OpenGL rendering pipeline and describe what happens at each stage.

b) *(5 points)* Based on the diagram above describe how shaders fit it and diagram the pipelines for both vertex and fragment shaders.

c) *(6 points)* What is the difference between a uniform, attribute, and varying variable? Describe why we would use each type of variable and provide an example using each type.

4) Your somewhat troubled but well-intentioned friend has decided to make a video game starring Waddles the Penguin as he escapes from a children's book to pursue his lifelong dream of becoming a lumberjack. Naturally, for the first part of the game, your friend has elected to implement a cel-shading aesthetic, but is having some problems getting his or her code to run correctly. (You've never actually met this friend in real life so you're not sure which gender-specific pronoun to use.) The relevant C++ code, a vertex shader, and a fragment shader are shown below. Answer the questions and try to help your friend help Waddles pursue his dream.

Assume the following *C++* code compiles correctly (no undefined variables, typos, etc.), assume that the projection matrix has been appropriately setup with gluPerspective, and assume that celshader_program_id **correctly refers to the program ID for the compiled vertex and fragment shaders (if they are free of errors).**

waddles_game.cpp:

```
//this function is registered as the GLUT display callback
void renderScene()
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	gluLookAt(cameraX, cameraY, cameraZ, lookatX, lookatY, lookatZ, 0, 1, 0);

	glPushMatrix();
	glRotatef(waddles_current_rotation, 0, 1, 0);

	glPushMatrix();
	glTranslatef(0, waddles_hat_offset, 0);
	//render the lumberjack hat before we enable cel-shading; it will help
	//visually contrast waddles' storybook origins with the realism of his fate
	draw_lumberjack_hat();
	glPopMatrix();

	GLuint location = glGetUniformLocation(celshader_program_id, "time");
	glUniform1f(location, glutGet(GLUT_ELAPSED_TIME) / 1000.0f);
	glUseProgram(celshader_program_id);
	render_waddles();          //sets normals correctly, etc.

	glutSwapBuffers();
}
```

vertex shader, waddles.vert:

```
varying vec3 theNormal;
void main()
{
	theNormal = normalize(gl_NormalMatrix * gl_Normal);
	gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

fragment shader, waddles.frag:

```
uniform float time;
varying vec3 theNormal;
varying vec3 tempN;
void main()
{
	float lambertianIntensity;
	tempN = normalize(theNormal);
	lambertianIntensity = dot(vec3(gl_LightSource[0].position), tempN));
	//and vary the intensity of the light with time, just to make it look
interesting
	lambertianIntensity *= (sin(time)+1.0) * 0.5;

	vec4 color;
	... code to threshold intensity here ...
	... uses lambertianIntensity and sets 'color' appropriately ...
	gl_FragColor = color;
}
```

a) *(5 points)* The first thing that your friend notices is that Waddles doesn't appear cel-shaded at all. It's like the shader program is not being used. Why is this and how would you fix it?

b) *(5 points)* Now the shader program is working, but too well; the lumberjack hat is being cel-shaded as well and that's definitely not in keeping with your friend's artistic vision for the project. Why is this happening?

c) *(5 points)* The scene is finally beginning to come together, but the intensity of the cel-shading effect should be varying with time. It's remaining constant even though we're passing in the current time with glUniform1f**(). Why is this? (Hint: it was working just a second ago, before we fixed the problem where the lumberjack hat was being cel-shaded.)**

d) *(5 points)* Phew! Everything looks right. Unfortunately, the program crashes after running for a while. What's wrong now?

A. What do you enjoy the most and the least about the computer graphics course? What is the most useful part of the course: lecture, lab, homework? What can be improved (teaching style, course material, assignments, etc.)? Please provide comments for both the instructor and teacher's assistant.