

INFO6014 – Graphics 1: Final Exam

Wednesday, December 12th, 2012, 9:00 AM to 9:00 PM

Instructor: Michael Feeney

The exam format:

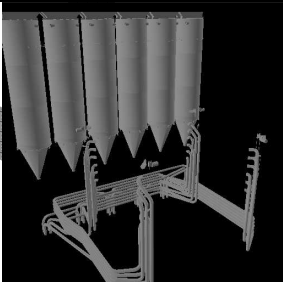
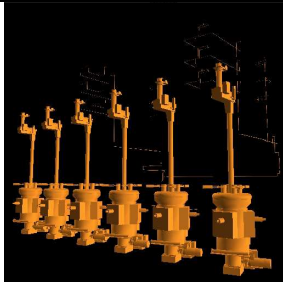
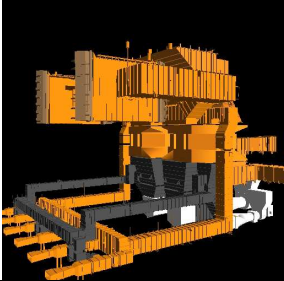
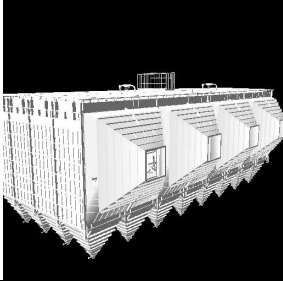
- The questions are **not** of equal weight. There are four (4) pages with eight (8) questions.
- The answers may be one or a combination of the following:
 - Short answer (in your own words)
 - Snippets of code
 - Complete running solutions
- **CLEARLY** indicate which answer goes to which question. My suggestion is that you place each answer in its own folder, named "Question_01", "Question_02" and so on (or something equally clear). Another option is to create a Visual Studio solution and add a number of projects – one per question – to it.
- Place any written answers into a Word, RTF, or text file. Again, **clearly** indicate which question you are answering.
- If you are combining answers (which is possible), please indicate this with a "readme" file or some note (not buried in the source code somewhere) indicating this.
- For applications: If it doesn't build and run; it's like you didn't answer it. I'll correct trivial, obvious problems, but you need to be sure that it compiles and/or runs.
- This is an "open computer" style exam (like "open book" with the additional resources of what's on your computer, too) but keep in mind that you are supposed to be doing this exam **completely on your own** (with *absolutely no communication or collaboration*):
 - It is almost impossible that the basic structure of your code and/or the approach you take would be virtually identical to another student taking this test. This would indicate collaboration, in other words "plagiarism" or "cheating."
 - **I will be aggressively looking** out for this kind of thing, as it is a clear violation of the college plagiarism policy.
 - Please refer to Policy 2-G-04 on Fanshawe Online or in the Student Handbook for more information.
- There may be a number of versions of the exam.

For all questions:

- Keeping in mind the plagiarism policy, you **can** use the code that was provided for you in class (the ply loader, the basic DirectX code, etc.), but keep in mind that simply submitting code we've done in class, **even if it superficially answers the question**, is not enough.
- When asked to produce a “scene,” you can ***not*** use the “sea of teapots” from class – you must create something else.

1. (10 marks) What is the specular component used for in lighting? What happens when you remove it or don't take it into account?
2. (10 marks) What version of HLSL have we been using? How did you find out?
3. (50 marks) Create a scene that uses a number of small pieces of the Power Plant (<http://gamma.cs.unc.edu/POWERPLANT/>) model, specifically parts of these sections:

(Note: the pictures on the web site are pretty screwed up, so I've tried to list the ones I'm wanting here - basically, you ****don't**** want anything with long, skinny tubes)

File: Section 4 (picture listed as "section 16")	File: Section 6 (picture listed as "section 18")
	
File: Section 10 (picture listed as "section 2")	File: Section 14 (picture listed as "section 6")
	

Ensure that you have the following present in the scene:

- You need to have at least 25 separate objects in the scene.
- There are at least 1/10th of the 500,000 triangles that are present (i.e. you can load all the models if you like, but I'm looking for more than "OK, I've loaded a single triangle, now give me my mark!" situation) - in other words, you have to have at least 50,000 triangles from the original models.
- All the objects have to be translated, rotated, and scaled. In other words, you cannot just place the models as they are in the original ply files.
- There are at least five lights: 3 point, 1 directional, and one spot.
- The lights can't be all the same colour.
- The camera is able to move around the scene (in a somewhat sensible way)
- Two of the point lights must be controllable by the user (i.e. you have to be able to move two lights around). You may tie one of the lights to the camera position if you'd like.
- You must apply texture mapping to at least one object in the scene.

4. (20 marks) Alter your solution from question 3) so that you can switch between perspective and orthographic projection (in DirectX 11) by pressing a key of your choice. Create a "2D" scene using 3D objects, so that switching between the views will display the same type of scene. Keep in mind that you will have to take into account the potential difference in size of the objects. In other words, the objects should take up essentially the same size on the screen.

5. (20 marks) Alter your example from question 3) so that there are three pixel shaders present in the scene, being applied to different objects. You need to specify the pixel shader being used "per object" (i.e. part of the properties of the object is which pixel shader to use).

- A "lights only" shader
- A "lights + textures" shader
- A "textures only" shader

6. (25) Many of our in-class examples had the following vertex layout:

```
struct SimpleVertex
{
    XMFLOAT4 Pos;
    XMFLOAT4 Normal;
    XMFLOAT2 Texture0;
    XMFLOAT2 Texture1;
};
```

Alter one of the shaders used in this exam in the following manner:

- Turn off lighting
- Turn off texturing
- Use the texture coordinates as the diffuse colour (i.e. "the colour") of the objects, specifically:
 - Texture0.u --> Red
 - Texture0.v --> Green
 - Texture1.u --> Blue
 - Texture1.v --> Not used

7. (20 marks) Use the PrntScr key to take six pictures of the files contained in each of your courses. In other words, open up Windows Explorer to the "Graphics 1" folder and take a screen shot of that. Then go to "Animation 1" and take a screen shot of that, and so on. When you have all six images, place them in a cube map and use that in an example application from this text. You can either use this a "skybox" or as a reflection.
8. (Bonus: 25 marks): Add an "off-screen" texture effect to one of the solutions from this exam.

That's it