# **INFO6020 – Graphics 2**

**Midterm Exam – March 2nd, 2016**

Instructor: Michael Feeney

## The exam format:

* You may use any resources you feel are necessary to complete the exam, but you are to answer the questions **on your own**. I will be looking for plagiarism (i.e. copying) very carefully. There is *no possible way* that the specific code to answer these questions, or the output to the screen, would be very similar to the look of another student’s code. Remember, this is a test and there are very clear policies about cheating on tests.   
  + <http://www.fanshawec.ca/admissions/registrars-office/policies/cheating-policy>
  + <http://www.fanshawec.ca/sites/default/files/assets/Ombuds/cheating_flowchart.pdf>
* It is an “open book” exam. You have access to anything you book or internet resource you’d like
* The questions are ***NOT*** of equal weight. The exam has **three (3)** questions and **five (5)** pages (there’s lots of “parts” to each question, though, with the Thing That Get Marked highlighted in yellow)
* 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. If I can’t make heads or tails of what question is what, I probably won’t even mark it.
* Do ***NOT*** do some clever “*oh, you just have to comment/uncomment this block of code*” nonsense. However, if the questions ***CLEARLY AND OBVIOUSLY*** build on each other, you may combine them (like if one question places objects, then the next one moves objects around with the keys) – even so, **MAKE IT 100% CLEAR** to me what questions the solution is attempting to answer.
* Place any written (“essay” or short answer) answers into a Word, RTF, or text file. Again, *clearly* indicate which question you are answering.
* If you are combining answers (which is likely), please indicate this with a “readme” file or some note (*not* buried in the source code somewhere).
* For applications: if it doesn’t build and run, *it’s like you didn’t answer it*. I’ll correct trivial, obvious problems (like you clearly missed a semicolon, etc.), but you need to be sure that it compiles and/or runs.
* You have until **11:59 PM** on **Wednesday, March 2nd** to submit all your files to the appropriate drop box on Fanshawe Online.   
    
  **NOTE:** Although this may “look and feel” like a project, it isn’t, it’s an **exam**, so there is **no concept of “late marks**”; if you don’t submit your files by 11:59 PM, you don’t get any marks at all. *Don’t Be Late submitting.*

(Also be **SURE** that you are actually submitting the correct files)

* Your solution may **not** contain any third party libraries (like boost) or any "C++11" features (auto, ranged for, initializer lists, etc.). If it has either of these things, the question(s) will not be marked (because it won't build). There has been some stress about this, so here’s a “meeting you ½ way”: if it builds in Visual Studio 2012, then you’re OK.
* When ready to submit, PLEASE for the love of all that is good on this Earth, delete all the “extra” Visual Studio files before zipping it up (remember this is C++, so all I really need is the .h and .cpp files, right?), like the “Debug” and “Release” folders with the “obj” files, as well as the “ncb” (intellisense) file – this will save a tremendous amount of space and shorten your upload time.
* **If the solution does not build (and run), I will not mark it** (so you will receive zero on questions that can't be built and/or won't run). When I say "run", I'm not speaking about some, random, unforeseen bug, but rather something that you should have obviously dealt with, like memory exceptions, etc.
* Unless otherwise indicated, all these solutions assume that you are creating/using a C++ project using Visual Studio 2008, 2010, 2012, or 2013 using the OpenGL 4.x API (with freeGLUT, GLEW, and GLM). I’ll be compiling opening them in VS2012, and using the 2008 “toolset” (or the 2012 if it won’t compile with 2008).
* I’ll be in my “main campus” office from about 9:00 until about 2:00, but since I’m at main campus, I won’t be popping down to see you. You can reach me through e-mail ([mfeeney@fanshawec.ca](mailto:mfeeney@fanshawec.ca)) or by calling the school.

## NOTE: Unless otherwise indicated, you should be displaying the ply files that were included with the exam (in the PLYFiles.7z and PLYFiles.zip folder – they are the same files, just different archives).

Some notes about the models:

* There’s a bunch of models in there. I tried to give you something to work with to make the scene, but I’m no artist. They are from TurboSquid and 3DWarehouse, so converting some of them was sort of annoying
  + 40x11 Trawler: a ship model (sorry, no texture)
  + Blue Whale
  + Ocean (by chukou): smaller, but nicer looking “wave” model
  + Ocean waves (by Maze Man): larger, but the “joins” look like garbage
* They are all random sizes, but I did centre them (did the “centre to origin” in mesh lab)
* You can find your own models if you find these really offensive or whatever

## The Questions:

**“Thar she blows!!”**

OK, that’s ***THREE*** Star Trek movies in a row, Feeney. Gheesh.   
  
Yes, nerds, you are to recreate the sequence from Star Trek 3, The Search for Spock.

Now that they’ve found Spock (On no! Spoiler alert!), they have to go into the past (well, 1980-something) to get a couple of humpback whales to take back to the “present” (their present) to save the world from an alien space probe. True story.

Honestly, I’d be afraid to even get out of bed if I were one of Kirk’s buddies.



I’m assuming the whales are somewhere under this boat....

...like here-ish?

There’s an iconic sequence at the end of the movie where the ship (yes, it’s *the same* ship as in the last movie, and so the same “Bird of Prey” model for you to use) “materializes” (well, technically it’s “cloaked” – aka “invisible”) in front of a whaling boat JUST IN TIME to rescue the whales.

The whales are then “beamed up” by Scotty, into some giant aquarium tanks they’ve installed in the ship.

You’re going to recreate some of the effects that were used, but with a modern computer, not the 1980’s whatever-the-heck-they-used-at-the-time techniques.

The edited movie clip is on FOL as well, to give you an idea of what it looks like.

1. (10 marks) Place the model of the space ship, the “water” ship (I guess), the water, and the whales (2 of them: George and Gracie). Just like the picture above!   
     
   You can either:
   * Place the camera “at the waterline” so you can see the whales, or
   * (bonus 5 marks): make the water semi-transparent so we can see the whales through the water.
2. (30 marks) “De-cloak” the Klingon ship. Here’s what it looks like in space:
   * <https://www.youtube.com/watch?v=lYvnf3pJDVs> and
   * <https://youtu.be/Lk9wSrZ0fWA?t=64> (Yes, that’s Christopher Lloyd)
   * The clip also has a (very brief) de-cloaking sequence, but you really can’t tell what’s happening very well, so refer to the clips above

It’s supposed to include the following:

* Use some kind of cube map (of the sky+water, space, whatever) to have a “refraction” effect on the space ship.
* This is combined with a transparency effect, ranging from completely transparent to fully opaque, over time.
* The normals used in the diffraction are randomly altered (very slightly, and all together) over time. My suggestion is to use something like:

normal \*= sin(someUniform) \* 0.1f;

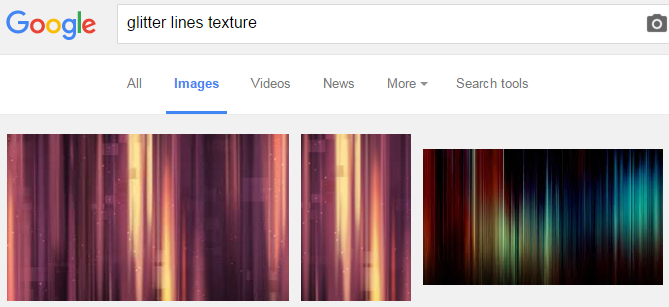
* With “someUniform” changing over time. Sine will gently change from -1.0 to 1.0, so multiplying by 0.1 will give a range of -0.1 to 0.1. Remember that the normals are only -1.0 to 1.0 (if “normalized”, which the should be).
* You could also adjust the vertex locations (and vertex normals) in the vertex shader, instead of the normals in the fragment shader – would give a similar effect
* You can use some other way if you’d like – that’s just a suggestion – but the refraction has to have that “wobble” sort of effect you see in the movie.

So what you are going for is: the ship is invisible, then starts to “appear”, while it’s doing that “heat haze” refraction wobble.

1. (50 marks): Add a “transporter” effect. Now you can do any sort of “Star Trek transporter” effect you’d like – it doesn’t *have* to be just like the video clip. Here’s some:

* <https://www.youtube.com/watch?v=jDFI87zn9t0>
* <https://www.youtube.com/watch?v=KhODlALRGoA>
* <https://www.youtube.com/watch?v=wNC5iLNguRQ> (he talks about how to actually do it in as film effect, which is interesting)

Pressing “T” (for “transport the whales!”) causes the whales to “beam” to the ship:

* Using some kind of “sparkle” or “lines” texture (Google images for “sparkle texture”/“glitter” texture” for point sparkles and “glitter lines texture” for more vertical line effect):



* + You need two textures to be applied at the same time.
  + They need to be “in motion” relative to each other, giving that “glitter-pulse” sort of effect (see the video clip). You can use the same texture – but maybe rotated 90 degrees – or a different texture (like one lines and one points, perhaps).
  + This texture should gradually appear (from transparent) until it almost completely obscures the whales.
  + When the texture is at full “brightness”, the whales need to fade away (become transparent) until they are gone.
  + Then you reverse the sequence of the transporter “glitter”, fading it out until it’s gone, too.
  + (Bonus: 15 marks): The glitter is blurred as it is appearing, then comes sharply into focus. Note that *only* the glitter is blurred.
  + (Bonus: 10 marks): The blurring of the glitter changes over time, either in how much is blurred (i.e. on screen area) or the “kernel” of the blur, giving the effect that it’s *really* blurry at the start, then not blurred at all (right when the whales start to disappear)

**That’s it**