# **INFO6028 – Graphics 1 - Final Exam – Fall 2018**

Tuesday, December 11th, 2018

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>
* The questions are ***NOT*** of equal weight. There are eight (8) pages with four (4) 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. If I can’t make heads or tails of what question is what, I probably won’t even mark 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 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 **Tuesday, December 11th** to submit all your files to 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.*

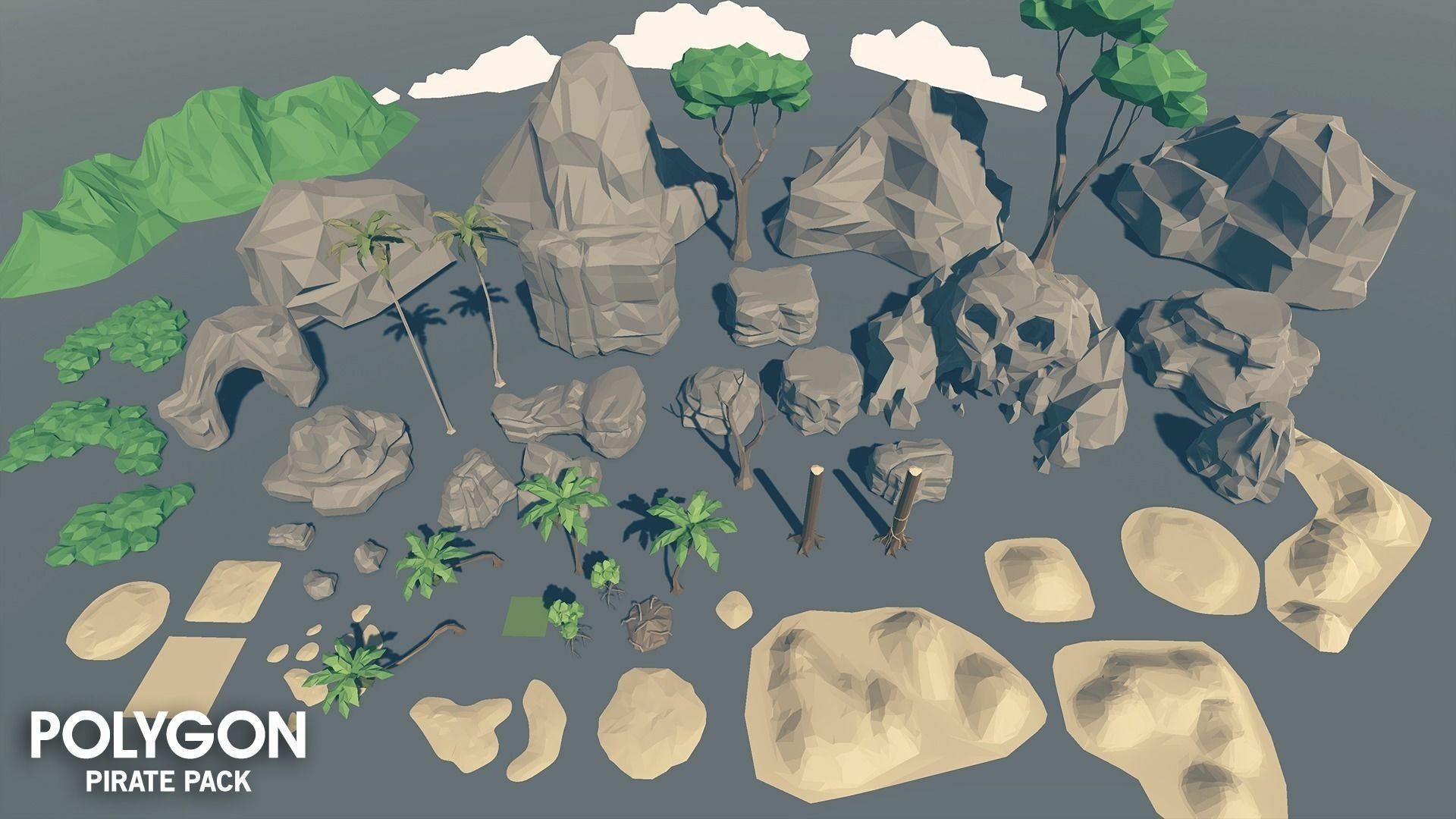
(Also be **SURE** that you are actually submitting the correct files)

* You can reach me through e-mail ([mfeeney@fanshawec.ca](mailto:mfeeney@fanshawec.ca)) or by calling the school.
* There is also a **Pirate\_Rocks\_and\_Trees.7z** file you will need. It’s available on FOL with the mid-term.

## Questions:

**“It’s a *sky* pirate life (and death) for me!”** *(one more time!)*

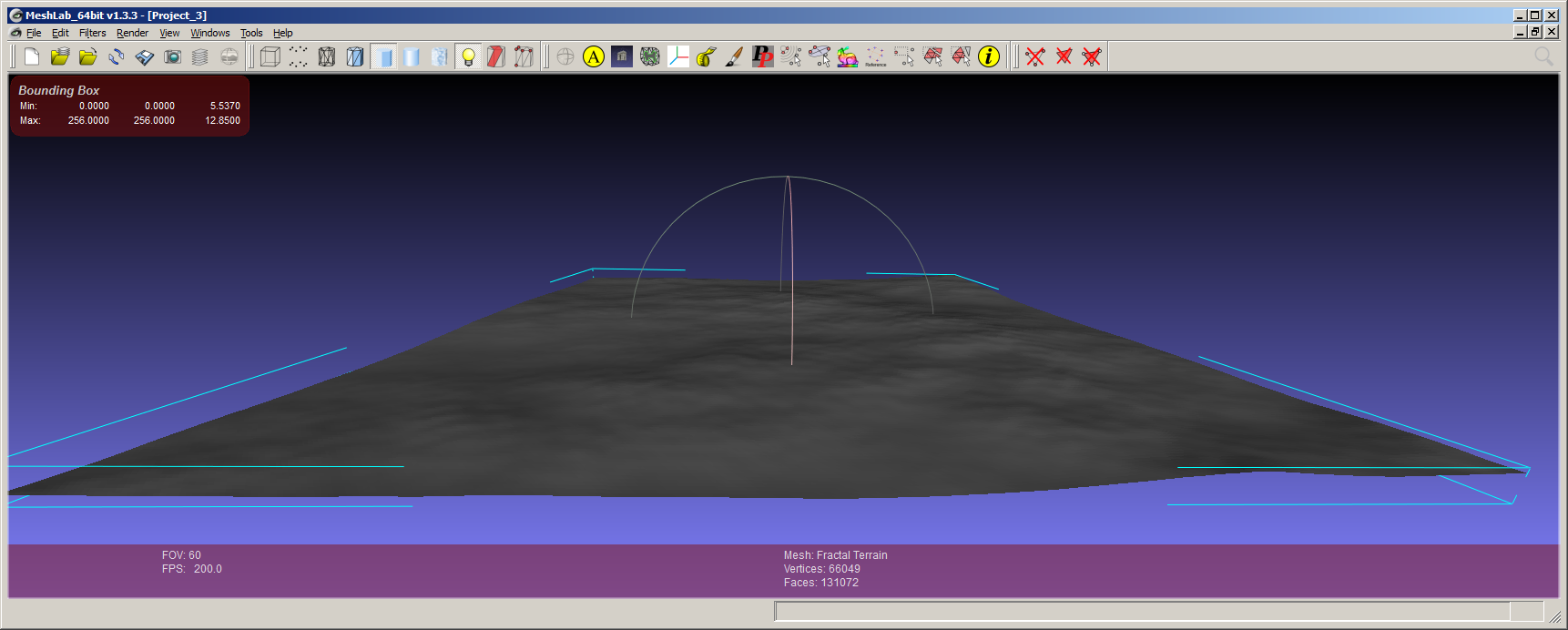




You are going to create a more detailed pirate island, using a combination of the meshlab and some of the models in the “Polygon Pirate Pack” (seen above). Here’s the link to the entire collection: <https://www.cgtrader.com/3d-models/exterior/historic/polygon-pirate-pack>

Unlike the other exams, you will *not* need to make an island with the island generator! But, you *will* have to make a couple meshes for the “water”:

The 1st “***upper*** water mesh” is created like this:

* Open MeshLab (without opening a model). This will open it with an empty “project”
* Choose “Filters”, then “Create New Mesh Layer”, then “Fractal Terrain”
* In the “Fractal Terrain” dialog box, choose “**Hybrid multifractal terrain**” (“Algorithm” dropbox.)
* Change the “Max Height” to **0.05**.
* With a “Seed” value of 2.0, you will get this:
* ***Pick a number for the “Seed” value (the default is 2.0) using the following method:***
  + Get the ASCII value for each letter of your *full* name. Add all these numbers up. Take the first three (3) numbers of the final result as your seed value.
  + For example: Michael Feeney gives: 77+105+99+104+97+101+108+ (“Michael”)  
    70+101+101+110+101+121 (“Feeney”)  
    = 1295 🡪 *so my seed would be “129”*
* Choose “Render”, “Show Box Corners” to get the summary at the top left.   
  NOTE: You will need to know these “extent” numbers later (and you can also get them programmatically by determining the minimum and maximum x, y, and z values, if you’d rather do that)
* Rotate the mesh so it’s on the XY plane by:
  + Select “Filters”, “Normals, Curvature, and Orientation”, “Transform: Scale”, and “Transform: Rotate”
  + Enter either -90 or 270 into the “Rotation Angle”, click “Apply”, then “Close”

* Note: The mesh will **NOT** be centred on the origin, so you might want to do that by choosing: “Filters”, “Normals, Curvature, and Orientation”, “Transform: Move, Translate, Centre”, then check the “translate centre of bbox to the origin” and “Apply”. *This is optional, though.*
* You will need “flat” UV texture coordinates, so do the following:
  + Choose “Filters”, “Texture”, “Parameterization: Flat Plane”
  + Choose “XZ” from the drop box, click “Apply”, then “Close”
  + This will create “per wedge” UVs, so chose: “Filters”, “Texture”, and “Convert PerWedge UV into PerVertex UV”
  + When you save the model, be sure to **un**check the TexCoord in the *Wedge* area, and ensure that there is a check in the *Vert* area.

The 2nd “***lower*** water mesh” is created like this:

* Load the “upper water mesh” into MeshLab
* Choose “Filters”, “Normals, Curvature, and Orientation”, “Transform: Scale”
* **Un**check the “Uniform Scaling”
* Enter 0.01 into the “Y Axis” value
* Click “Apply” a few times until the mesh looks super flat, then click “Close”
* The normals will likely be messed up, so fix this by:
  + Choose “Filters”, “Normals, Curvature, and Orientation”, “Compute Vertex Normals”
  + Click “Apply”, then “Close”
  + Choose “Filters”, “Normals, Curvature, and Orientation”, “Normalize Vertex Normals”

So, in the end, you will have *two* meshes, that are the same XZ size, but one is a little “bumpy” (with Perlin noise) and the other one is completely (more or less) flat.

* The slightly bumpy one is the “upper” water mesh
* The flat one is the “lower” water mesh

1. (50 marks) Create the initial pirate island:

* Use the “upper water mesh” as the water.
* Place an appropriate “water” texture on it (just get one from the internet). It should look reasonably like water (like if I showed it to a child, they would say “that’s water, duh!”)
* Place the “SM\_Env\_Background\_Hills\_01.ply” model in the water. This will be the main island. Make it pretty large (it will have rocks, explained below, on it).
  + Place a “sand” texture on it. Again, get this from the internet.
* Place a number of "rock" objects on the "island"
  + two "large" rocks (choose two of the “SM\_Env\_Rock\_Large\_XX” models)
  + the "cave" rock (“SM\_Env\_Rock\_Arch\_01” model)
  + the "skull" rock (“SM\_Env\_Rock\_Skull\_01” model)
  + the 3 small rock models (“SM\_Env\_Rock\_XX” models)
* Add a light grey “stone” or “rock” texture to these. You can use the same texture.
* Add 6 trees by combing the tree trunks (“SM\_Env\_Tree\_Base\_XX” models) with the tree top models (“SM\_Env\_Tree\_Canopy\_XX” models).
  + You need to use *both* the tree trunks and all 4 of the canopies, but there should be 6 “unique” trees (i.e. you can’t use the same trunk+canopy combination more than once)
  + Place a “leaf” or “grass” texture on the tree tops
  + Place a “wood” or “tree bark” texture on the tree trunks
* Use the “Tropical Sunny Day” cube map to make a skybox
* Place the camera near, but a little above the water, looking at your “work of art”, clearly showing all the elements. It could be sort of “personal drone height” above the water, so something like the “outside” shots here: <https://www.youtube.com/watch?v=YsYzOb3lc0w> (it’s higher than a human could reach, but it’s not WAY up in the sky like a helicopter or a balloon, either).
* Place enough light so that it looks like a bright sunny day

1. (50 marks) Make the water fancier. What you’re going for is a little like what was done for the game “Giants: Citizen Kabuto”: <https://youtu.be/MkVG3GtDGhI?t=121> Here, there are two meshes with different textures, with the top one slightly transparent.

* Change the “upper water mesh” texture to a new “water caustics” texture (google “water caustic texture” and you’ll see what I mean: it’s the pattern you see at the bottom of a swimming pool).
* Add a “lower water mesh” to the scene, a little below the “upper water mesh”, and place the *original* water texture you used on this one.
* Make the “upper water mesh” about 50% transparent.
* Create a new vec4 uniform value in the fragment shader, called "waterOffset".
* Create a matching vec4 variable on the C++ (application) side
* Set this value to all 0.0f. (i.e. 0,0,0,0, so even the 4th “w” value is 0.0f)
* In each frame, change the value by these tiny amounts *times the delta time in seconds*:
  + waterOffset.x by +0.1f / second
  + waterOffset.y by +0.017f / second
  + waterOffset.z by -0.13f / second
  + waterOffset.w by -0.013f / second
  + To be clear, these number are *per second*, meaning that the x value would take 10 seconds to go from 0.0f to 1.0f. These *aren’t* the *per frame* changes; to get those, you’d multiply these values by the “delta time” passed per frame. This “moves” the water texture a little each frame, but it shouldn’t be flying across the mesh, though (moving it at 0.1f/frame certainly would!)
  + Note they are similar values, but not the same.
* Each frame, update the "waterOffset" uniform
* In the shader:
  + add the x & y values to the UV offsets of the "higher" water texture
  + add the z & w values to the UV offset of the "lower" water texture
* The texture should “wrap” (which is the default in the example code, so you likely don’t have to change this).

1. (50 marks) Allow a transition from day to night time.

* Add two small, “flickering” point lights inside the “eyes” of the “skull” cave thing.
  + The “flickering” is done by setting the linear attenuation to some random value *each frame*.
  + It should be a “small” light, like that of a small camp fire or candle, so the difference between the brightest and dimmest “flicker” should be subtle.
  + In the “daylight”, it should be barely perceptible (or not perceptible at all)
* Load the "space" cube map in addition to the "tropical day" cube map, so:
  + You should have loaded both cube maps on the C++/Application side
  + You should have a /second/ cube map sampler in the shader
* Using the keyboard, make the "Ctrl+Alt and UpArrow" gradually make it "daytime", and the "Ctrl+Alt and DownArrow" gradually "night time". This is done in the following manner:
  + Pass another uniform float value, called "dayMix"
  + When you press Up, this value slowly increases to 1.0f
  + When you press down, this value slowly decreases to 0.0f
  + In the shader, the colour of the skybox is taken from BOTH cubemaps (tropical day AND space), and mixed together this way:
    - Assume skyBoxTropical is the RGB sampler colour from the tropical day cubeMap and skyBoxSpace is the RGB sampler colour from the space cubeMap:
  + skyBoxColour.rgb = (skyBoxTropical.rgb \* dayMix) + (skyBoxSpace.rgb \* (1.0f - dayMix));
* Use this “dayMix” value to also “dim” the “sunlight” that you used in question 1 & 2 so that it’s goes from “bright mid-day” to very dim “middle of the night/dim moonlight” level.
  + So the “moonlight” *isn’t* completely dark (i.e. *isn’t* actually 0.0f when the dayMix variable is 0.0, but scales somewhere close to zero).
  + Note: You also *can’t* simply “clamp” the light value to some value above 0.0, while the skybox continues to change more to the “space” texture, so:
    - dayMix goes from 0.0 to 1.0 to transition the skybox values...
    - ...while at the same time, the light goes from, say 0.1f to 1.0f  
      (or something “dark”: it might be 0.1f is too dark, but whatever the lowest value for the overall light in the scene)
    - It should be “dark” (as dark as it’s going to get, with a little light so we can see stuff), when the skybox has gone completely to the space texture.

1. (25 marks bonus) Make the upper water mesh move with the “caustic” texture:



* The “caustic” texture would look something like this:
* Note that it’s going from “blue” to “white”, where the “white” areas are thinner. Note that this looks, sort of, like what waves look like. In fact, the pattern is created from the surface of the moving water, so that makes some sense, right?
* Use this variation from blue to white to adjust the height of the upper mesh.
* Remember:
  + The caustic texture is moving (from question 3)
  + White is RGB: 1,1,1, while blue is RGB: 0,0,1, so think about that...
  + Where the colour is “blue-est”, the water is lowest.
  + Where the colour is “white-est”, the water is highest.
  + So you are sampling the *colour* of this texture to adjust the *height* (y value) of the upper mesh model.
  + You *also* need to keep the texture on the mesh, too (so just don’t change the colour sampling)
* Don’t worry about the normal change, as you’d need a geometry shader to do that, but...
* Bonus #2 (10%)
  + ...if you increase the specular value at the highest 5% (or so) of the height, then you’ll get an additional 10% bonus.
  + Water is a little specular all the time, so I’m looking for it to be “pretty specular” at the bottom, and “really specular” at the highest point.
    - i.e. If I move the camera around, there still should be *some* specular at the lower, curved points of the water, as the light reflects off the surface, but there should be much more at the peaks of the water).