# **INFO6028 – Graphics 1 - Final Exam – Fall 2019**

Thursday, December 12th, 2019

Instructor: Michael Feeney

## The exam format:

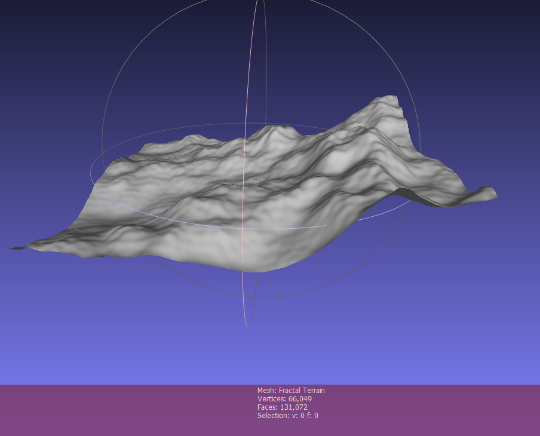
* You may use any resources you feel are necessary to complete the exam, but you are to answer the questions **on your own**. I will be looking for plagiarism (i.e. copying) very carefully. There is *no possible way* that the specific code to answer these questions, or the output to the screen, would 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 eleven (11) 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.   
  + If the questions “build” on each other, you may submit a single project.
  + If the questions “build” on each other, but you can add keyboard/mouse controls to change from one question to another, you may submit a single project.
  + If the questions *don’t* clearly build on each other, my suggestion is that you place each answer in its own folder, named “Question\_01”, “Question\_02” and so on (or something equally clear).
  + PLEASE delete any temporary files that Visual Studio generates (to reduce the upload size)
* 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 **Thursday, December 12th** 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. I will be at LDB from 1:00 – 3:00 on Thursday, December 12th.
* There is also a **fish.7z** file you will need.

1. (5 marks) Using MeshLab, generate a SINGLE island, in the following way:

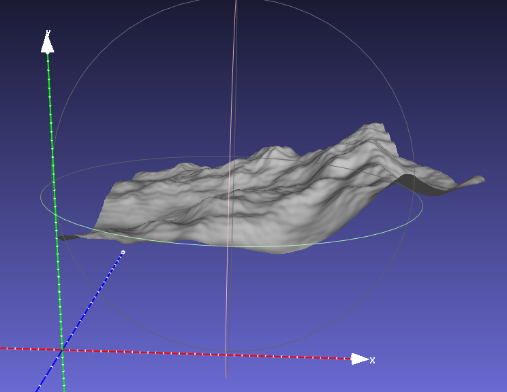
* Open MeshLab (without opening a model). This will open it with an empty “project”
* Choose “Filters”, then “Create New Mesh Layer”, then “Fractal Terrain”
* In the “Fractal Terrain” dialog box, choose “**Hybrid multifractal terrain**” (“Algorithm” dropbox.)
* Change the “Max Height” to **0.5**.
* Pick a “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”*



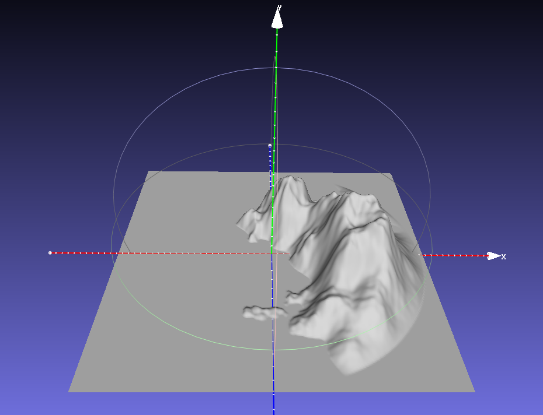
* + Note: It seems that really large numbers, like your 8 digit student number, make the terrain ‘blocky’

With a “Seed” value of 2.0, you will get this 🡪

The Island Converter assumes “up” is “z”, so we need to adjust this. Turn on the “axis” drawing by choosing “Render”, “Show Axis” to make this clear (if you want).



* Choose “Filters”, “Normals, Curvature, and Orientation”, then “Transform: Rotate”.
* Type in “-90” in the “Rotation Angle”, leaving the “Rotation on:” set to “X axis”, and click “Apply”, which will get you this 🡪
* Save this model with JUST xyz and NOT in binary form (“File”, “Export Mesh As…”, uncheck the “Binary encoding”, and choose OK.
* Download and compile the “Terrain\_to\_Island\_Converter” project and convert the mesh you made into an island. This takes the model file name as an input and generate an “island” mesh (like the one on the right 🡪) called “output.ply”.



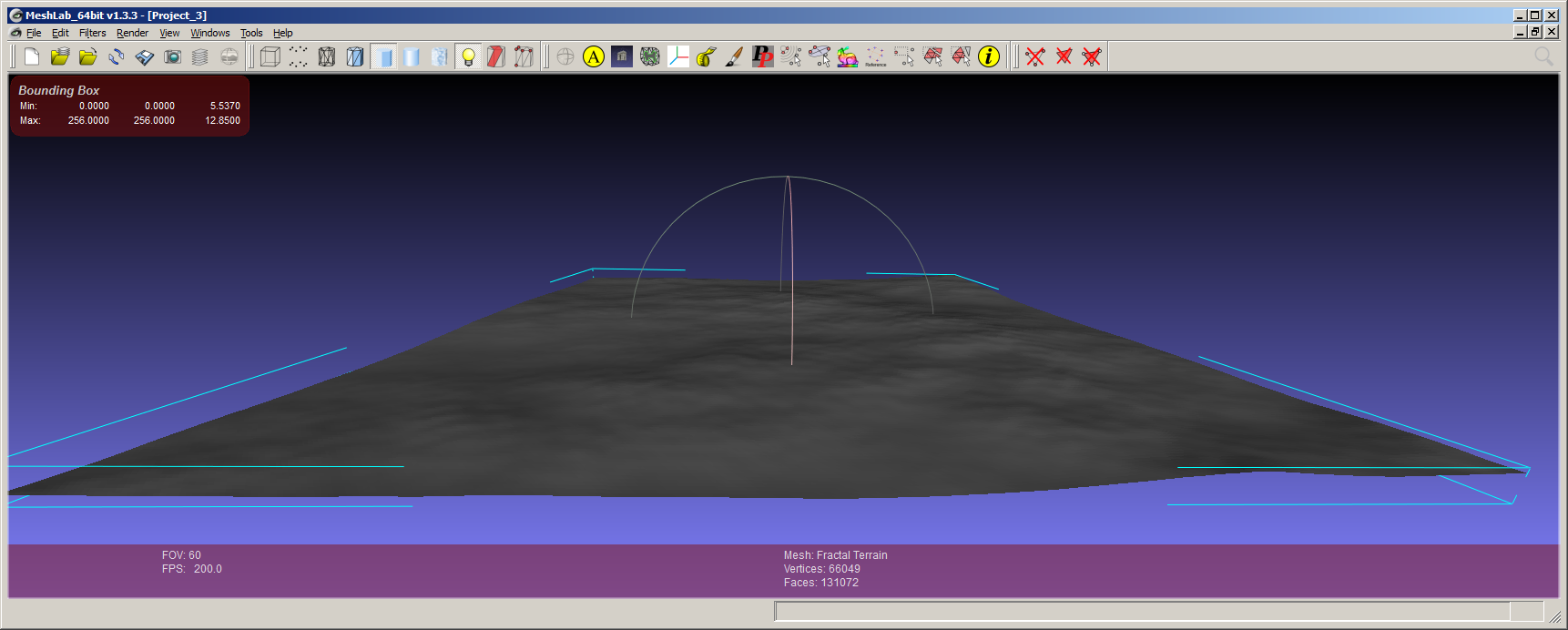
* Open this model into mesh lab, generate normals and scale it by 2.0:
  + Choose “Filters”, “Normals, Curvature, and Orientation”, then “Re-Compute Vertex Normals”
  + Choose “Filters”, “Normals, Curvature, and Orientation”, then “Transform: Scale, Normalize ”; type in 2 in the “X Axis” text box;   
    Click “Apply”
  + Save the model (BE SURE TO SAVE *WITH* NORMALS)
* You will use this model as the island your pirates live on.   
  The model should be:
  + 512 units wide in the XY axis, from -256.0 to +256
  + The “water” should be at 0.0 along the y axis.
  + The maximum height should be something like 150 (but will depend on the seed value)

NOTE: If your code requires texture coordinates, you can generate ones in MeshLab like this:

* Choose “Filters,”
* then “Textures”,
* then “Parameterization: Flat Plane”
* Choose “XY” as the “Projection Plane”
* Click “Apply”
* Next, convert to “per vertex” texture coordinates:
* Choose “Filters”,
* then “Texture”,
* then “Convert PerWedge UV to PerVertex UV”
* When you save it, be sure to *un*-check the texture coordinates in “Wedge” and *check* the one in “Vertex”

1. (5 marks) Create a “sea floor” model. This is supposed to mimic gently rippling sand that you’d see under fairly still water, near a beach.

* 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.

1. (5 marks) Create a flat “water” mesh in the following manner:

* Load the “sea floor” 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.

1. (50 marks) Draw the island scene using these three models, in the following manner:

* Draw the “island” and “sea floor” mesh over top of each other, with the “bottom” of the two meshes overlapping.
  + In other words, the “sea floor” mesh should be aligned to the “flat” portion of the “island” mesh.  
    Note that the “sea floor” mesh is “bumpy” so won’t exactly match the “flat” portion, but should be, on average, at the same “height” as the island model.
  + Place the “water” mesh parallel to the “sea floor” mesh, but at approximately 25 to 30% up the side of the “island” mesh. So that part of the island is “under water”.
  + Alter the shader so that the “flat” portion of the “island” model *isn’t* being rendered, using the “discard” feature in the **fragment** shader. Do this in the following manner:  
    - Determine the location of the “flat” portion of the island model *in world space (i.e. where it is when you render it, if you’ve moved it vertically).*
    - Add some small value to this height, and pass it to the shader, though a uniform variable.
    - Inside the fragment shader, discard any pixel that is *lower* than this small height.
    - The result is that the *island* part of the “island” mesh is drawn by the *flat* portion of the “island” model *is not*, leaving only the “sea floor” model visible where the flat portion was.
    - It should look like the “sea floor” model was part of the “island” model.

1. (50 marks) Add textures to the model:

* Find a “sand” texture online. Apply this texture to the “sea floor” model, so it looks like it’s sand.
* Find a “grass” and a “rock” texture.
* Texture the “island” model in this manner *in the shader*:  
  + Anything from the base of the island to *just* above the “water” model should be sand, and should have the “sand” texture visible.
  + Anything from the highest point of the “island” to about 15% lower should have the “rock” texture visible.
  + Anything between the “rock” and the “sand” should be textured with “grass”
  + There should be a smooth transition between the textures (you can use the mix() GLSL function to do this, or some other method)
* Find a “water caustic” texture and use it on the “water” model.

1. (50 marks bonus) Alter the “water” mesh height with the “caustic” texture:



* The “water 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:
  + 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 #1 (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).

1. (50 marks) Make the water fancier.

* Make the “water” mesh about 40% transparent.
* Create a new vec2 uniform value in the fragment shader, called "waterOffset".
* Create a matching vec2 variable on the C++ (application) side
* Set this value to all 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
  + 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 “water caustic” texture
  + make sure this new UV coordinate is altering the height of the “water” mesh *as well as* the sample from the “water caustic” texture (i.e. the colour).  
      
    What should happen is that the underlying mesh height should match the colour of the “water caustic” texture, as the texture “moves”.
* The texture should “wrap” (which is the default in the example code, so you likely don’t have to change this).

1. (30 marks) Place the water caustics on the sea floor:

* Using the same “water caustic” texture that you are using in question 7 (i.e. the one that is “moving”), apply the same colour to the “sea floor”:  
  + Combine the “sand” and “water caustic” texture so that 60% of the colour (0.60) is “sand” and the remaining 40% (0.4) is “water caustic”.
  + This should looks like the caustics are shining on the sea floor.

1. (20 marks) Place a skybox around the scene, with the “sunny tropical day” cube map we’ve used in class.   
     
   Use your own discretion as to where the skybox should be.
2. (20) Light up a couple fish (well, a fish and a shark):

* Add a fish (the “fish” model) and a shark (the “shark”) model to the scene.
* Place 1 shark and 5 fish under water and an appropriate scale.
* Make it looks like the shark is “chasing” the fish.
* Place a spot light somewhere on the side of the island mountain, shining down, into the water, illuminating the fish (but mainly the shark). This should look like someone is on the side of the island with a flash light, so make the “beam” quite narrow.
* I should be able to see the surface of the water illuminated as well as the “sea floor”.

1. (BONUS: 10%) Using a keystroke, adjust the skybox so it changes to night:

* Use the “space” skybox in the following manner:  
  + Below the surface of the water, the skybox should show the “tropical day” cube map (which is blue, like the water), so below the water surface, it should show blue.
  + Above the surface of the water, the skybox should show the “star” cube map.
  + Hint: You might want to move the entire model so that the “water” is on the x-z plane (i.e. where the “y” axis is 0.0f)
  + Note: I’m not looking for “perfection” at the transition from the “under water” to “above water” (this would be very difficult as the “water” model isn’t flat to begin with *and* it’s being altered by the “water caustic” texture, but I’m looking for an *approximate” height* where the transition occurs – “approximately” at the water line.
  + The transition between the “under water” and “star” should be smooth (like the transitions between textures in question 4)