# **INFO3111 “C++ Graphics”, Midterm Exam, Tuesday, June 4th, 2023**

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 and don’t add to 100%. Bonus questions are simply added to the total amount - as if they are "regular" questions - but bear in mind that as they are "bonus", will be marked to a higher standard.
* The exam has **eight (8)** questions and **seven (7)** pages.
* **CLEARLY** indicate which answer goes to which question.
  + If you feel that the questions “build on each other” (which they do) feel free to submit a single solution/project.)
  + If you feel it’s clearer to have the submissions separate, I’d suggest placing separate questions in separate projects and folders.
  + 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 - if you are expecting me to *edit* your source code: I'm *not* going to do that and will run the code as submitted.
* If you feel I need to know something (key mappings, that you edited the 3D models, etc.) please indicate this with a **“readme” file** or some note **in the root folder** (*not* buried in the source code/project folders somewhere).
* Submit the **entire solution(s) and project(s)** *not* just the source code files.
* Please, for all that is good and sane in the world, delete any files you don’t need as well as any temporary files Visual Studio creates (“.vs”, the debug and release folders, etc.). You won’t lose marks for submitting these, but it’s very annoying and it’s something you should know how to do.
* **If the solution does not build (and run), I will not mark it** (so y*ou 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.
* While I *might* use the Debug build while marking, your submissions **must** build and run in **Release** build.
* Unless otherwise indicated, all these solutions assume that you are creating/using a C++ project using Visual Studio 2022 using the OpenGL 4.x API (with GLFW, glad, and GLM), and build in **64-bit** **Release** using the default setting in Visual Studio (C++ 14).
* Your solution may ***not*** contain any third-party libraries (like boost), **smart pointers,** or the “**auto**” keyword.

To be clear: if you have ***any*** of these elements, you will receive a mark of zero (0) for that question.   
NO exceptions: you should be aware of what’s currently used in industry (99% C++98/2003), and what just happens to be in the newer standards.

**Some notes about the models:**

* Most of the models are same as the ones in the **Dungeon\_models.7z** file   
  (the same models that you are using for project #1 and the files that are in the github folder).
* You will also need the files in the **Additional\_Mid-term\_Models.7z** file.
* You will also need the **Terrain\_to\_Island\_Converter.7z** file.



In this exam, you are going to make a pretty garden of plants and flowers, with a path of stones.

This garden is growing on an island, surrounded by water.

You will be also mimicking the garden during the day and during the nighttime.

## The Questions:

1. (5 marks) Making the island:

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.)
* A white mountain with a blue background

  Description automatically generated with medium confidenceChange the “Max Height” to **0.5**.
* ***Use the first 4 numbers of your STUDENT NUMBER for the “Seed” value (the default is 2.0).*** Note: It seems that really large numbers (like your entire 8 digit student number, make the terrain ‘blocky’)

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

MeshLab assumes that “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).

A close-up of a mountain

Description automatically generated

* 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 something like this 🡪   
    
  *(Note that you might have to rotate the window so the Y axis (the green line) is facing upward, which isn’t the default I think...)*
* A 3d model of a mountain

  Description automatically generatedSave this model with **JUST xyz** and NOT in binary form (“File”, “Export Mesh As…”, uncheck the “Binary encoding”, and choose OK.

A screenshot of a computer

Description automatically generated

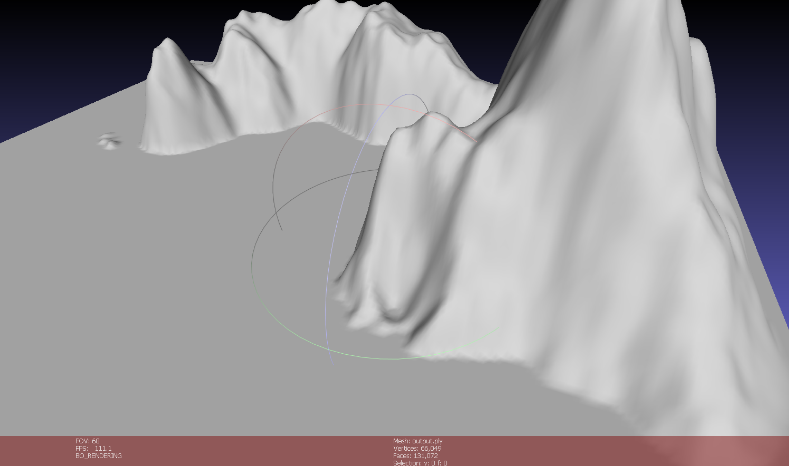
🡨 Like this, with **nothing** checked.

* Either build the “**Terrain\_to\_Island\_Converter**” project (or use the **Terrain\_to\_Island.exe** file in the x64/Release folder) and convert the mesh you made into an island.   
    
  This takes the model file name as an command line parameter and generate an “island” mesh (like the one on the right 🡪) called “output.ply”.   
    
  You can also drag the mesh file onto the exe file.
* You will use this model (output.ply) as the island your garden is on!

1. (50 marks) Pick a place for your garden and place a “stone walkway”:  
   * The island is massive compared to the garden (it’s an island after all!), so you may have to change the projection matrix to extend the “far plane”:  
       
     Line 239 of the current theMain.cpp file has:

matProjection= glm::perspective( 0.6f, ratio, 1.0f, 10’000.0f );

* + The 3rd number (1.0f) is the "near" plane. Any vertices *closer* to the camera than this (in world space) is “clipped” (not drawn).
  + The 4th number (1000.0f) is the “far” plane. Any vertices *farther* from the camera than this (in world space) is “clipped” (not drawn)
  + General rule: make the “near” plane as LARGE as possible (this is the *closest* thing to the camera), and the “far” plane as SMALL as possible (this is the farthest thing from the camera).



* + Your garden should be tiny in comparison.   
      
    Pick a “flat” location on your island. If you have a “beach” sort of area, that’s ideal, but even the top of the mountain or something.   
      
    This island (using seed 1234) has a small flat area near the “water” part (the large flat part).  
    (like the *entire garden* would fit in that little spot, is what I’m saying)  
      
    NOTE: the large flat portion is the “water”. The idea is that the the “island” is sticking out of the water.   
      
    Do NOT put your garden in the water portion.

This large flat area is the “water” part of the model.

* + Create a stone path using the “floor” models in the “dungeon” based on your “initials”:
    - Using the first letter of your first and last name (or if you have more than two, like a middle name or multiple names, choose two of them). For instance, my name is “Michael Feeney” so I would choose “M” and “F” and my garden would have stones laid out with “MF”.
    - Using the retro/pixel style characters to the right, lay out the “stones” using your initials.   
        
        
        
        
        
        
        
        
        
        
        
        
        
        
      (I got this from: https://www.vectorstock.com/royalty-free-vector/retro-game-pixel-art-font-pixelated-text-alphabet-vector-44758181)
    - I’m not looking for “perfection” here – just use this as a general guide, but the letters seem to be 5 pixels/blocks wide and 7 pixels/blocks high.
    - I’ll need to clearly see that it’s your initials.

1. (50 marks) Place some plants around:   
   * Place six (6) *different* trees around your garden. Use the tree models that have the trunk and canopy separate: The tree trunks are called “tree\_base” and the leaves are called “tree\_canopy”.
   * Choose appropriate, but slightly different colours for each tree.   
     The trunks can be grey or brown, and the canopies/leaves can be green or dark red.
   * Place ten (10) tufts of grass around the garden, hear the “stones” you placed in question 2. Choose appropriate green colours for the grass.
   * You’ll find five (5) types of flowers in the “Additional\_Mid-term\_Models.7z” file.   
       
     Place ten (10) flowers around, ensuring that you are using each type of flower at least once.
   * Again, I’m not looking for “perfection” here, but plants are sort of scattered “randomly” in nature. Like they aren’t in a line or grid or something.
2. (50 marks) Place a large, bright light above the scene to look like sunlight.
3. (50 marks) Set of some camera positions to show off your garden:   
   * Pressing the “F1” key (GLFW\_KEY\_F1) will place the camera high overhead of the island, looking slightly down. Think of a drone of helicopter approaching the island and you are looking out the window, down towards the island that you are about to land.   
       
     I should see the entire island, including your garden. It’s unlikely that you can hide the edges of the “water” but try to minimize seeing these “edges” of the mesh. Like it’s to sort of mimic that the water goes on into the distance.
   * Pressing the “F2” key (GLFW\_KEY\_F2) will pace the camera beside the garden, like you are a person standing near it, about to enter the garden.
4. (100 marks) Mimic a sunset and sunrise:
   * Choose a set of attenuation values for the light in question 4 that represents “full sun” (this is likely whatever values you have it set to in question 4)
   * Choose another set of values for “night”. This should be dark, but not completely dark, so you should still be able to see the garden.
   * Using the “F3” key (GLFW\_KEY\_F2) gradually change the light attenuation from the “full sun” values to the “night” values. This should take place over about 10 seconds. If your framerate is 60Hz (typical on laptops) then you’d want to change these values by 1/600 each frame. For example, if the difference between one of the attenuations was, say 0.12f, then you’d want to change the attenuation bot 0.12f/600.0f (quite a small number).
   * Once the value gets to the “night” values, stop.
   * Using the “F4” key (GLFW\_KEY\_F4) does the reverse, gradually changing the light attenuation from the “night” to “full sun” over around 10 seconds. When it’s at the “full sun” values, the change should stop.
5. (200 marks) Update the “night” scene so that there’s fireflies visible:
   * These “fireflies” can be around during the day (there’s a bonus question for that later where they aren’t visible).
   * If you’ve never seen a firefly, they are all around Ontario. They have some chemical in their abdomens that lights up as they fly around at night, looking for a mate:
   * Choose a very small model of your choice (a sphere, bunny, or whatever you’d like, really) for your “firefly”.
   * Place six (6) of these tiny models in the air above your garden. Fireflies like to fly about head height above grass and flowers, and they tend to fly a couple meters away from each other. So they aren’t in a tiny swarm, they aren’t spread out all over the island, and they’d be blow the canopy/leaves of the trees.
   * Place six (6) tiny lights very near these models, mimicking the light they give off.   
     You don’t want the light to be inside the models, but pretty close. Choose an appropriate colour (see the picture above).
6. (100 marks) Make the fireflies “flash”:
   * When flying around, their abdomen glows for a few seconds, then goes out for a few seconds, then goes back on. Apparently, this is very “sexy” to other fireflies, sort of like glowsticks at rave parties (I’m told).
   * Mimic this slow “flashing” but turning off and on the firefly lights in a somewhat “random” way. Like I’m looking for them to be flashing independently, *not* going off at the same time or in some predictable sequence from one another.

That’s it, but here’s some bonus questions:

* + (50 marks) Make the top of the island mountains white with snow and the area close to the water “sand coloured”. You need to do this *in the shader*, not by modifying the mesh model.
  + (100 marks) Have the fireflies slowly “fly around”. You can do this by changing the fly model/mesh and light position over time. They have to sort of “swarm” around, which you can mimic by using a trigonometric function like sin or cos, or some other way if you’d like. The movement has to look like bugs flying around, though (like bugs don’t move in straight lines).
  + (100 marks) Add a “drone shot” option the “F5” key (GLFW\_KEY\_F5). This would be like the first part of question 5, but it would slowly fly around/orbit the island, looking down.

From Stack Overflow, here's how you create a random float number:  
  
(note that you *cannot* use the C++11 <random> header, since you aren't allowed to use C++11. Also, the C++11 <random> while "all that", still isn't actually bloody random... gheesh)



(https://stackoverflow.com/questions/686353/c-random-float-number-generation)