# **Final Exam: INFO-6028 “Graphics 1” Fall 2024 (Monday, December 9th, 2024)**

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.
* This also includes code that you’ve produced from some generative AI.
* You may also ***not*** use code that’s taken *mostly* from some existing framework (like downloading the code from the OpenGL book, Learn OpenGL, Open Scene Graph, or something like that)  
  + <http://www.fanshawec.ca/admissions/registrars-office/policies/cheating-policy>
  + <http://www.fanshawec.ca/sites/default/files/assets/Ombuds/cheating_flowchart.pdf>

How can I make the determination that it’s not “your” code? Simple: If I suspect it’s not yours then I’ll ask you to “defend” it one-on-one where I’ll ask you questions and see if you know what you’re doing or why the code is the way it is. If you have no idea what’s going on, then it’s pretty certain you didn’t write it yourself. Simple as that and also a tried and true technique. You *can* use any code *provided in class* or you wrote *this term.*

* You ***may*** use simple utility libraries like assimp, loadPNG, JSON/XML loaders, sound, etc. **No** boost, though.
* You may **not** use the auto keyword. Doing so will give you a mark of zero.
* You may **not** use generative AI tools to create any of the code for this exam.
* It is an “open book” exam. You have access to anything in any book, internet resource, or anything on your computer, or that has been uploaded in class, including projects you’ve already completed.
* The questions are ***NOT*** of equal weight. The exam has **eleven (11)** questions and **ten (10)** pages.
* The questions build on each other so you can put them in one solution/project.  
  If you feel you need further clarification, please include a readme file (and a video if you’d like, though not required).
* ***PLEASE*** delete any temporary files that Visual Studio generates (to reduce the upload size)
* 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. **I do NOT want to edit the code in any way.**
* 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 **Monday, December 9th** 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 the time the drop box closes, you don’t get any marks at all.

*Please don’t be late submitting and* be **SURE** that you are *actually* submitting the *correct* files.

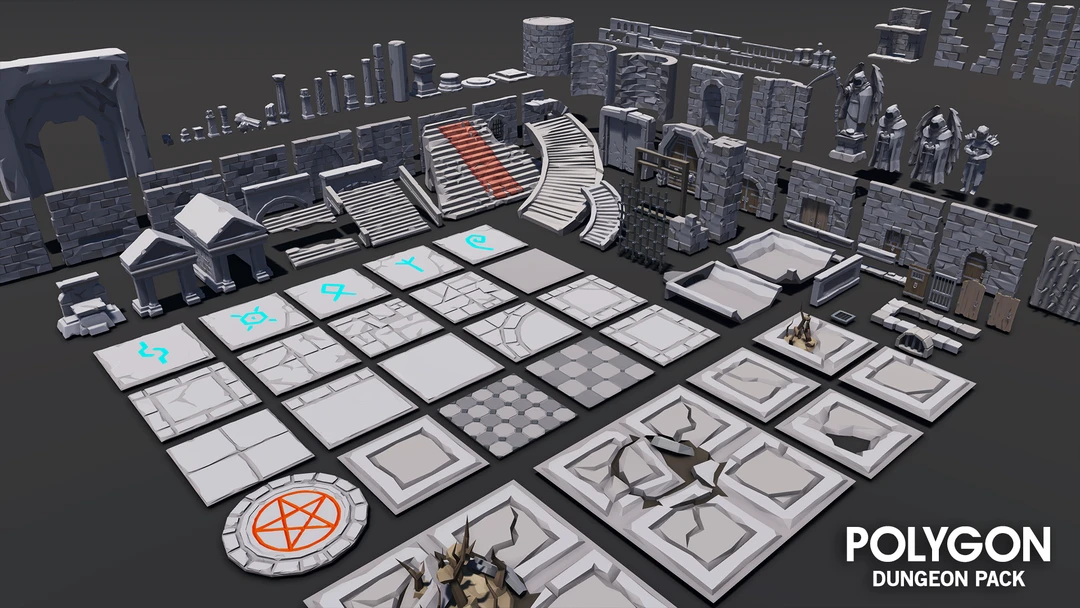
* Unless otherwise indicated, all these solutions assume that you are creating/using a C++ project using Visual Studio 2022 using the OpenGL 4.x API (with GFLW, glad, and glm).
* I will be building using the default Visual C++ settings (C++ 14).

## The Questions:

You are to create a dungeon area that’s been built into the side of an island, made up of the parts from the Synty Studios “POLYGON - Dungeon Realms” (<https://syntystore.com/products/polygon-dungeon-pack>) and “POLYGON - Pirate Pack” (<https://syntystore.com/products/polygon-pirate-pack>).

Many of the “environment” assets can be combined like LEGO bricks into any shape you’d like. They are all of similar size. In the picture below, you can see there’s a bunch of different “floor” models as well as a number of “wall” models.

These are in the “3D\_Models” folder.



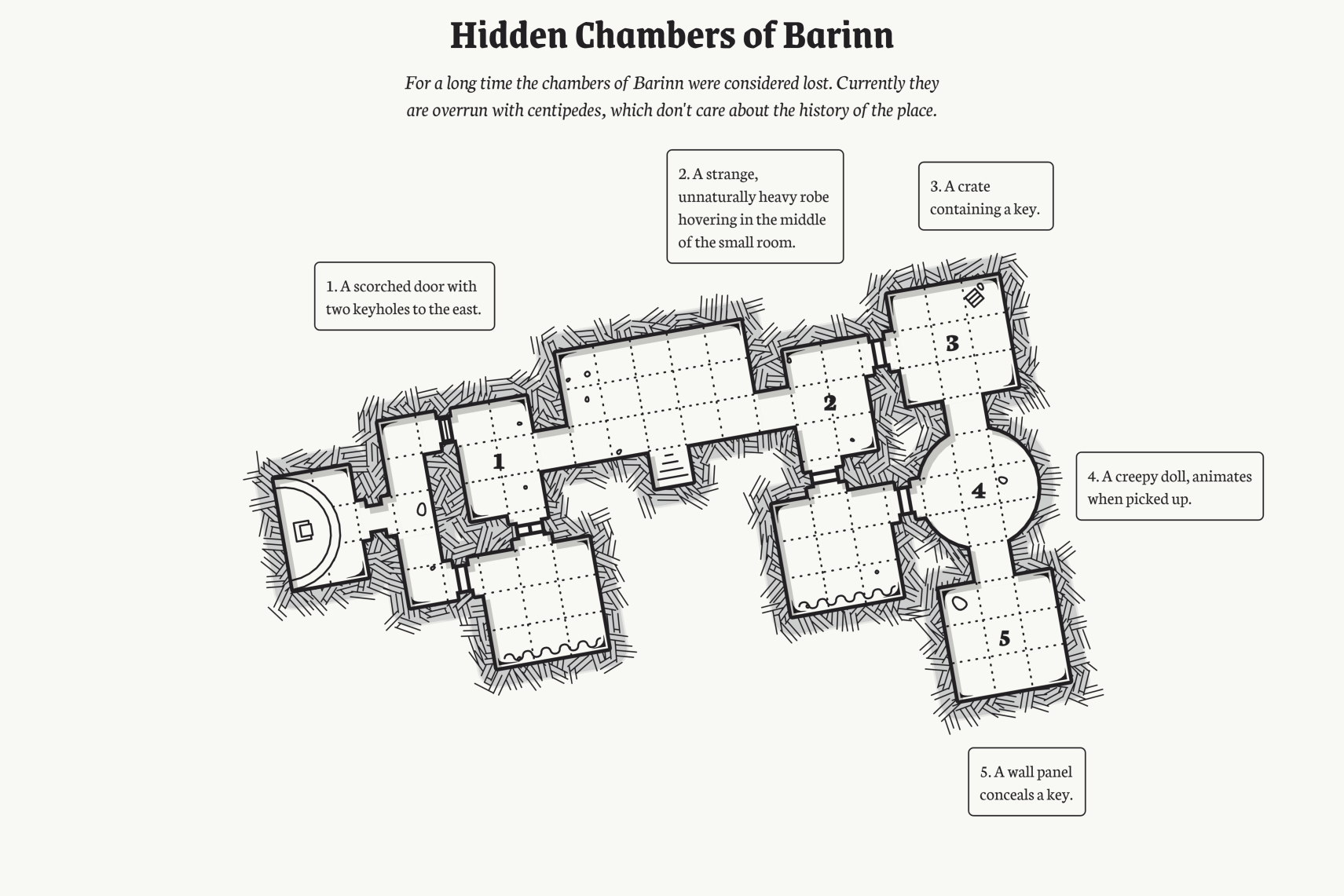
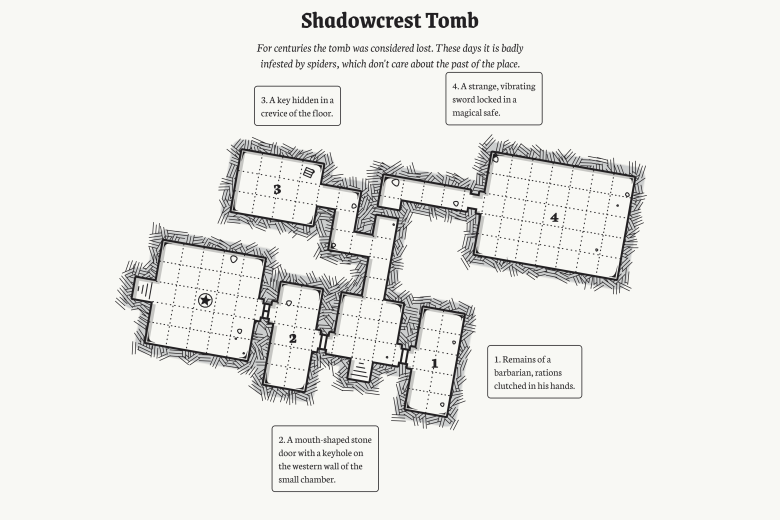
*Floor tiles:*

The idea is that these are *very* old building with stone walls, but there is no roof, as they were wooden and have rotted away long ago (i.e. there’s only walls, no roof anymore, so you can see the sky).

**Some notes about the models:**

* They are aligned in one corner of the model. This allows you to “snap” them together more easily (MeshLab “Render”, then “Show Axis” will show this.)
* The textures that correspond to the model are shown in the ply file, for example, the SM\_Env\_Floor\_Grate\_01.ply file has the following:  
    
   comment TextureFile Dungeons\_2\_Texture\_01\_A.png   
    
  This means that you need the Dungeons\_2\_Texture\_01\_A.png file as a texture.
* I’m pretty sure I’ve converted them all to an xyz+normal+rgba+uv format, but there’s 100s of files in there; if I haven’t, then you should be expected to convert and/or load them yourself.

You’ll be taking the layout of the building will be taken from the “One Page Dungeon” generator, located here: <https://watabou.itch.io/one-page-dungeon>

You are ***NOT*** to generate one yourself from the website, though.

**Instead, do the following:**

* Choose a dungeon map from the “Some One Page Dungeon models” folder (or the file “Some One Page Dungeon models.7z” which is the compressed version).
* Then, **go to this link**: <https://fanshawecca-my.sharepoint.com/:x:/g/personal/mfeeney_fanshawec_ca1/EcgdZHGLX1pDmPdylWsFPwcBy7j5JsY1nmuQt99dn04sWw> and **place your name beside the one you’d like**.   
    
  **IF THERE IS A ANTOHER STUDENT’S NAME THERE**, then **PICK ANOTHER MAP.**Do **NOT** replace the student’s name with your own.   
    
  (The idea here is that everyone does their own map)  
    
  Dire warning: If you do someone else’s dungeon, I will ***not***mark your exam, so you will get a mark of zero. Same if you didn’t put your name in the document.

This site is supposed to generate “dungeons” so something like the map on the right (Shadowcrest Tomb) would have long, narrow corridors/hallways (inside the red circle), where the one on the left doesn’t really have these (there are a couple very short ones, I suppose...)

In Dungeons and Dragons, the squares are 5x5 feet in size. So these hallways are 5 feet wide.

When making your dungeon, assume that each square on the map would match to a “floor tile” (see the arrow on page 2) in the “POLYGON - Dungeon Realms”.

This also means that the rooms would be *multiple* floor tiles. For instance, room “3” of the “Shadowcrest Tomb” (right map above) would be 4 x 3 tiles in size.

Note: unless otherwise stated, all the models need to be appropriately textured (or coloured\*). The dungeon models have various alternative textures and the beholder has both a texture as well as the vertex colours “baked” into the model.

(\* they *can’t* be just one colour, though. You can use the vertex colours instead of the texture colours if you want to “bake” the colours in, but I wouldn’t recommend that – some models won’t look good.)

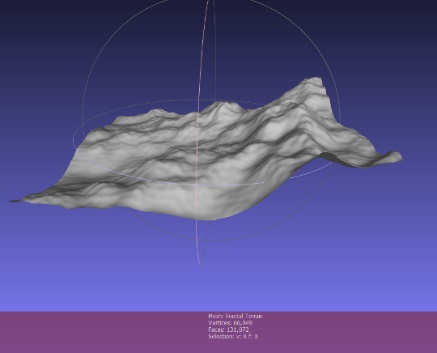
1. (5 marks) Pick a floor layout from the “One Page Dungeon” site and enter your name in the shared Excel document:   
   <https://fanshawecca-my.sharepoint.com/:x:/g/personal/mfeeney_fanshawec_ca1/EcgdZHGLX1pDmPdylWsFPwcBy7j5JsY1nmuQt99dn04sWw>
2. (5 marks) Making the island:

Using MeshLab, generate a SINGLE island, in the following way:

Open MeshLab (without opening a model). This will open it with an empty “project”

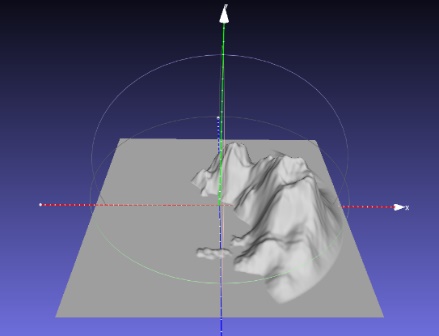
Choose “Filters”, then “Create New Mesh Layer”, then “Fractal Terrain”

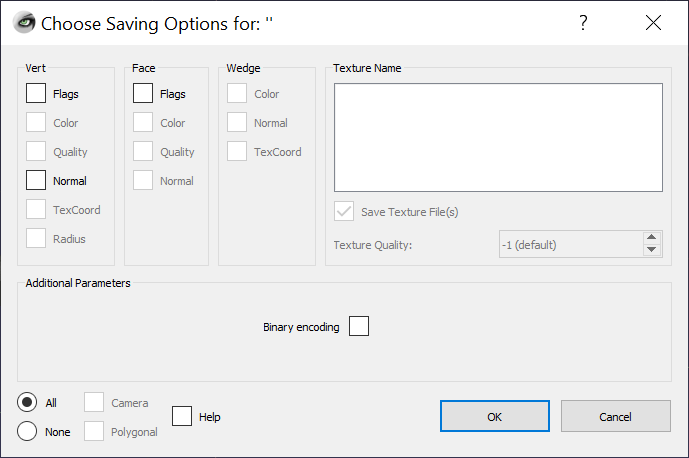
In the “Fractal Terrain” dialog box, choose “**Hybrid multifractal terrain**” (“Algorithm” dropbox.)

Change 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 🡪

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



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

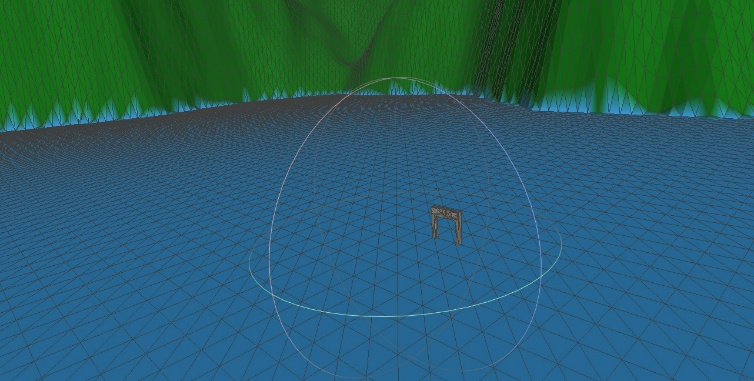
Either build the “**Terrain\_to\_Island\_Converter**” project (or use the **Terrain\_to\_Island.exe** file in the x64/Release folder) and convert the mesh you made into an island.   
  
This takes the model file name as an command line parameter and generate an “island” mesh (like the one on the right 🡪) 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 dungeon cave is in.

1. (150 marks) Choose a location for your “cave” and place the entrance.

A screenshot of a computer generated image

Description automatically generated

* + The idea is that you can enter the cave from the water or from the beach, so it is right at the waters surface (or very close). i.e. it’s not halfway up the mountain or something.
  + The circle here indicates a reasonable location.
  + Use the “SM\_Env\_Dwarf\_Wall\_Archway\_01.ply” model to mark the entrance to the cave.   
      
    Place this model near the surface of the water (or on the beach) partially into the side of the mountain. Note that this will overlap the geometry of the island model, but don’t worry about this.
  + Remember that the rest of the cave will “attach” to this entrance, so be sure to take into account the orientation of the cave.
  + The idea is that you’ve access this dungeon from a boat, with the entrance to the dungeon being through this archway, located just above the surface of the water – so you could maybe step from the boat into the dungeon cave, or if there’s a ‘beach’ area, it’s at the same level as the beach.
  + Choose a reasonable colour for this cave. I’m assuming it’s made of stone or stone blocks, so pick some grey “stone” or “dirt” colour.
  + In terms of scale/size, the dungeon is tiny in comparison to the island. Here’s is the size and scale I’m looking for (the archway and an area approximately the size of the circle in the image above):
  + Using the discard function in the fragment shader, remove a region of the island model (prevent it from being drawn) that fits the archway model.   
      
    Basically, you are removing the side of the island that matches the archway model – otherwise the side of the island would block your way into the dungeon, right?   
      
    I’d suggest you do this by passing two vec2 values into the shader (or hard coding them) that mark the locations x & z extents of the archway (like the min and max corners of the archway model, when viewed from above.   
      
    Then compare these values with the transformed world location of the vertices (which you’ll have in the fragment shader to do the lighting) and if the vertex is withing this region, call “discard”.

1. (200 marks) Place the rest of your dungeon inside the island and connect it to the archway.

* If you don’t have a clear “entrance hallway” sort of shape on your map, do the following:  
  + Add one that connects to a wall or room.
  + Place this archway on the side of a wall or room.

You don’t want to have parts of your dungeon “poking out” of the island.

* Choose *different* floor and wall tiles for each room and corridor. Like the entire room has the same floor tile style, but the next room has a different style.
* I don’t care how “thick” the walls are – I’m only interested in how the room looks when inside it.
* This room also has a ceiling.
* The room can be any “height” you’d like (1 wall tile, 2 wall tiles, 5 wall tiles – don’t care).

1. (200 marks) Place a number of lit “torches” to light up your dungeon:
   * Place at least fifteen (15) lit “torches” inside your dungeon. These should be spread out, with at least one torch per room or hallway. Use your judgement here, but I’m don’t want all the torches lumped together, so if your map has one large room (or something like that) then please spread the torches out.
   * Use one (or more) of the “torch” models in the Dungeon set. i.e. they can be all the same torch model or all different – your choice.
   * The “fire” area of the torch is set up as follows:
     + Using alpha or discard transparency, block out the “non-fire” portion of the texture.
     + Place an orange/red point light at the same location of the fire imposter object. This light will cast nearby (i.e. is the “firelight” from the torch.
     + Make sure the “fire” (textured) part of the torch is bright. This can be using a light or using an “emissive” light on the fire itself.
2. (10 marks) Set up the night sky using the “Space skybox texture” (This is the same texture that we used in class).
3. (75 marks) Set up “full moon night time” lighting in the following manner:
   * Place a dim light (point or directional) to illuminate the entire scene. Let’s assume it’s a “full moon” (more on this in a moment) and lights up everything a little bit.
   * Create a “full moon” using the assets in the “CGI\_Moon\_Kit” (the link to the NASA site where I got the textures, if you want larger ones):
     + There is a “UV Sphere” with “cylindrical UV projection” which will “wrap” the wide moon surface texture around the sphere.
     + Place the moon (the UV Sphere) object somewhere in the sky.
     + It should be bright, like a full moon, but does not necessarily need to be light with an actual light. i.e. you could use “emissive” light or shine a light on it – but the key here is that it should be evenly lit (like a full moon).
4. (200 marks) **THIS QUESTIONS HAS TWO OPTIONS DEPENDING ON YOUR MAP:**
   * If the map ***has*** water (like this one):
     + Mimic the surface of the water using a fractal terrain from MeshLab. Make sure the height of the terrain is very small (i.e. the height of the “waves” or “ripples” on the water would be very small, right?).
     + Make sure you recalculate the normals inside MeshLab before you export it.
     + To generate the UV coordinates, use “Filter”, “Texture”, “Parameterization: Flat Plane” and use the choice that matches the surface of the model (the mesh is aligned on the XY plane by default).
     + On the internet, find a “water surface” texture that you like.
     + Apply this water texture to the water surface, and make the water semi-transparent.
     + *Using the normals from the water surface,* sample the space cube map. This will give the effect that we see:
       - A semi-transparent water surface (so we can see something underneath)
       - A bit of the “water” texture
       - A reflection of the sky on the surface of the water, too.
   * If the map ***doesn’t***have water:
     + Place at least three (3) “crystal” models throughout your buildings.
     + Either set a single colour or choose a texture of a single colour (i.e. the colour of the crystal) for each crystal. They should be different colours.
     + Make them semi-transparent.
     + *Using the normals from the crustal surface,* sample the space cube map. This will give the effect that we see:
       - The coloured crystals are semi-transparent (i.e. we can see through them)
       - They reflect sky on their surface, too.

|  |
| --- |
| Note: If your map HAS water, but you’d rather do the crystal thing, then just ignore the water part.  i.e. *don’t* put the water in the scene. |

1. (**BONUS:** 100 marks) Do BOTH the water AND the crystals.
2. (200 marks) Add an attacking “Beholder” to your scene.

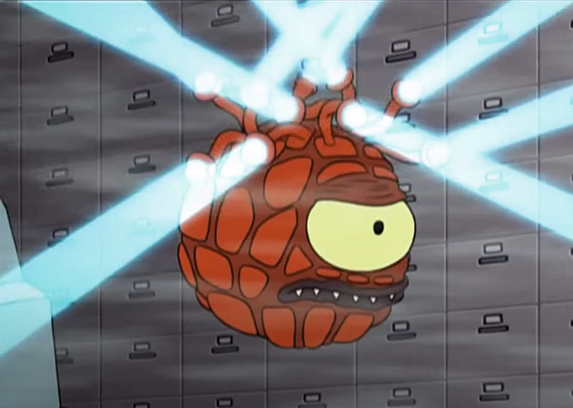
Beholders are monster characters from Dungeons and Dragons. You can read *way* more than you probably want here: <https://www.dndbeyond.com/compendium/cyclopedia/vgtm/beholders>

Here’s the part that you need to know:

* + They have lots of eyes on their stalks (like snails).
  + Each one of their eyes can shoot a different type of magical rays.
  + Beholders are sort of giant ass-holes in that they literally think they are the best thing in the universe and will attach *anything* (even other beholders).
  + We are going to adjust the size of the Beholder and assume it’s approximately 2-3 feet wide.  
    (They are normally 6 – 7 feet, but that’s too wide for the 5’ wide hallways)

The point is that they will *always* attack everything, and this one *is* attacking.

Mimic three (3) of the magical rays from three (3) of the Beholders eyes:

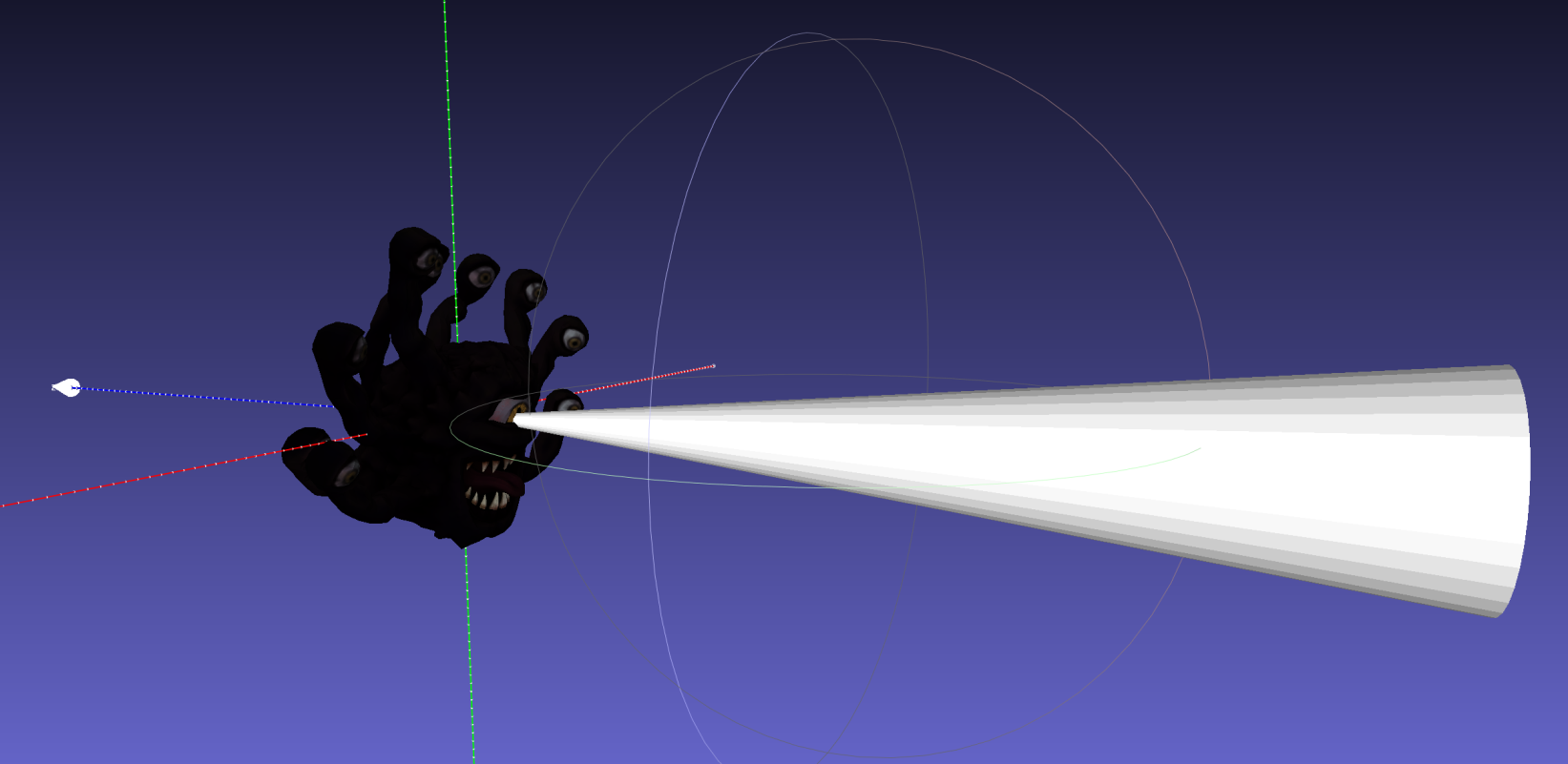
* + Place a very bright, but very “tight” spot light at each of the eyes.
  + This spot light will shine in some direction more or less where that eye is looking.   
    (Note: we’re talking about the little eyes on the stalks *not* the giant eye in the middle)
  + They are “tight” in that the spot cone angle is very small, like 5 degrees or something.   
    It should look like a “tight” flash light beam.
  + Each eye ray should have a different colour and be facing a *slightly* different direction.   
    The idea is that the Beholder thinks it’s seen something, but isn’t 100% sure *exactly* where its target is, so is shooting three rays in the general direction of where it’s facing.

Here’s an idea of what I’m talking about: <https://youtu.be/_H18JLCYj0g>   
  
Note that here the “rays” are more cylindrical than cone shaped, but not that the “rays” are coming from the little eyes in the stalks.

.

1. (**BONUS:** 100 marks) Show the outline of the Beholder’s eyes by using the “beholder\_vision\_cone” model. Make this the same colour as the lights you are using and make it semi-transparent.

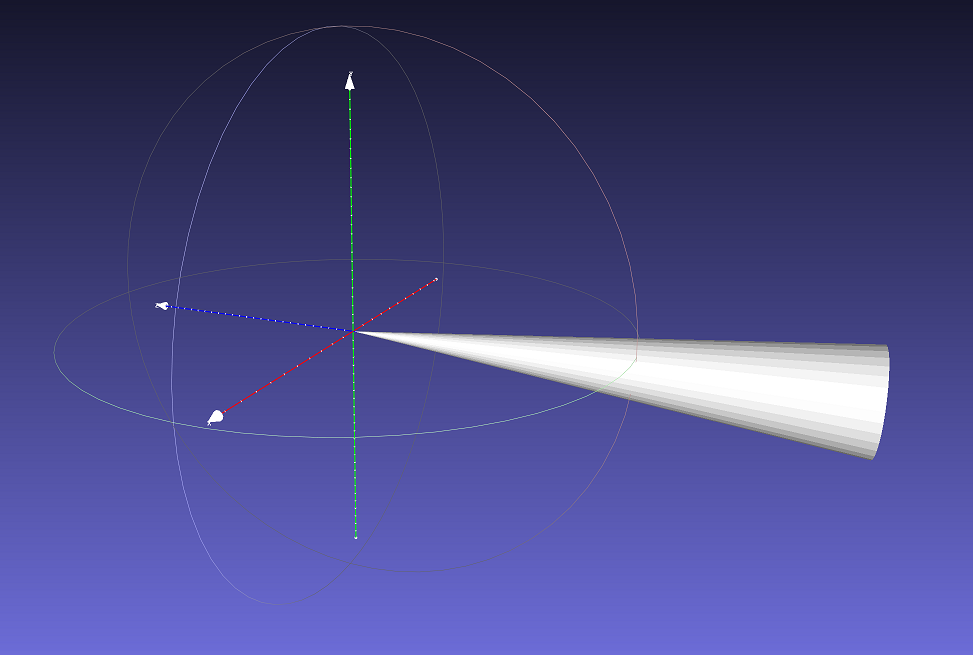
The cone lines up with the large eye of the Beholder, but you’ll need to add cones for each of the “rays” you used in question 8 (i.e. you can’t just have a single cone).



You will also have to change the “end size” of the code to reflect how “tight” you’ve chosen your spotlights in question #8. Since you don’t care about the normals here – you are just displaying this as a semi-transparent/non-lit object, you can non-uniformly scale the cone around the X and Y axes to “narrow” or “widen” the light cone.

The idea is that the large part of the code matches the size of the spot light at that point.

You will also have to “fade” the transparency of the cone over distance, just like a light would. You can do this by adjusting the transparency as it is further away from the “point” end.

****

**That’s it.**