# **Final Exams – Fall 2022 for BOTH:**

# **INFO-6028 “Graphics 1”**

# **INFO-6044 “Game Engine Frameworks & Patterns”**

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 **thirteen (13)** questions and **nine (9)** pages. The questions involve submitting a working Visual Studio solution.

|  |
| --- |
| * The questions build on each other, to make a complete scene. However, you may decide that a different camera angle will better show the scene, so:   + You may submit a single solution & project if you thinks it’s appropriate   + If you submit multiple solutions and/or projects, please name them in some easy to understand way like “question\_1”, etc.   + Please include any information I’ll need (keyboard controls, etc.) in a readme file, and not buried in the source code somewhere. |

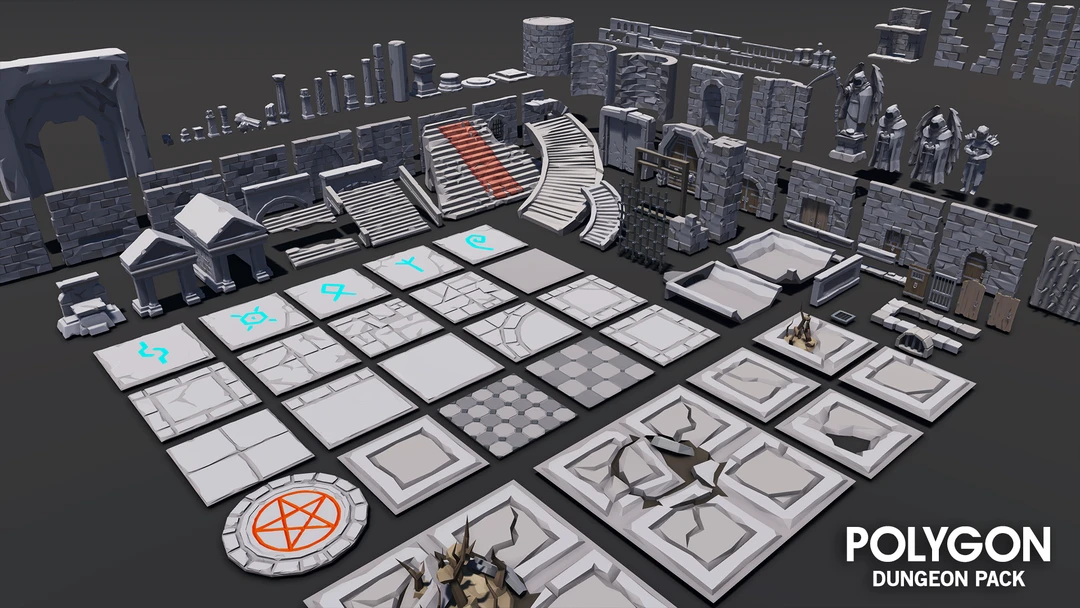
* I should not be expected to alter your code in any way to get it to build or run, so do not comment out your code, or expect me to alter it in any way at all. It should be ready to run as is.
* Your code should run under Windows 10 (i.e. a “win32 application”) with Release and x64 (“64 bit”) build.   
  I will *not* be building a “32 bit” application (note: “win32” means “windows API” and *not* “32 bit”).
* **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.
* No “**auto**” or the **boost library** (or any other 3rd party library we haven’t used or you haven’t cleared with me first).   
  If you use either, I will not mark your submission and you will receive a mark of zero.  
  No exceptions.

## The Questions:

You are to create one or more “ruined buildings” 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.



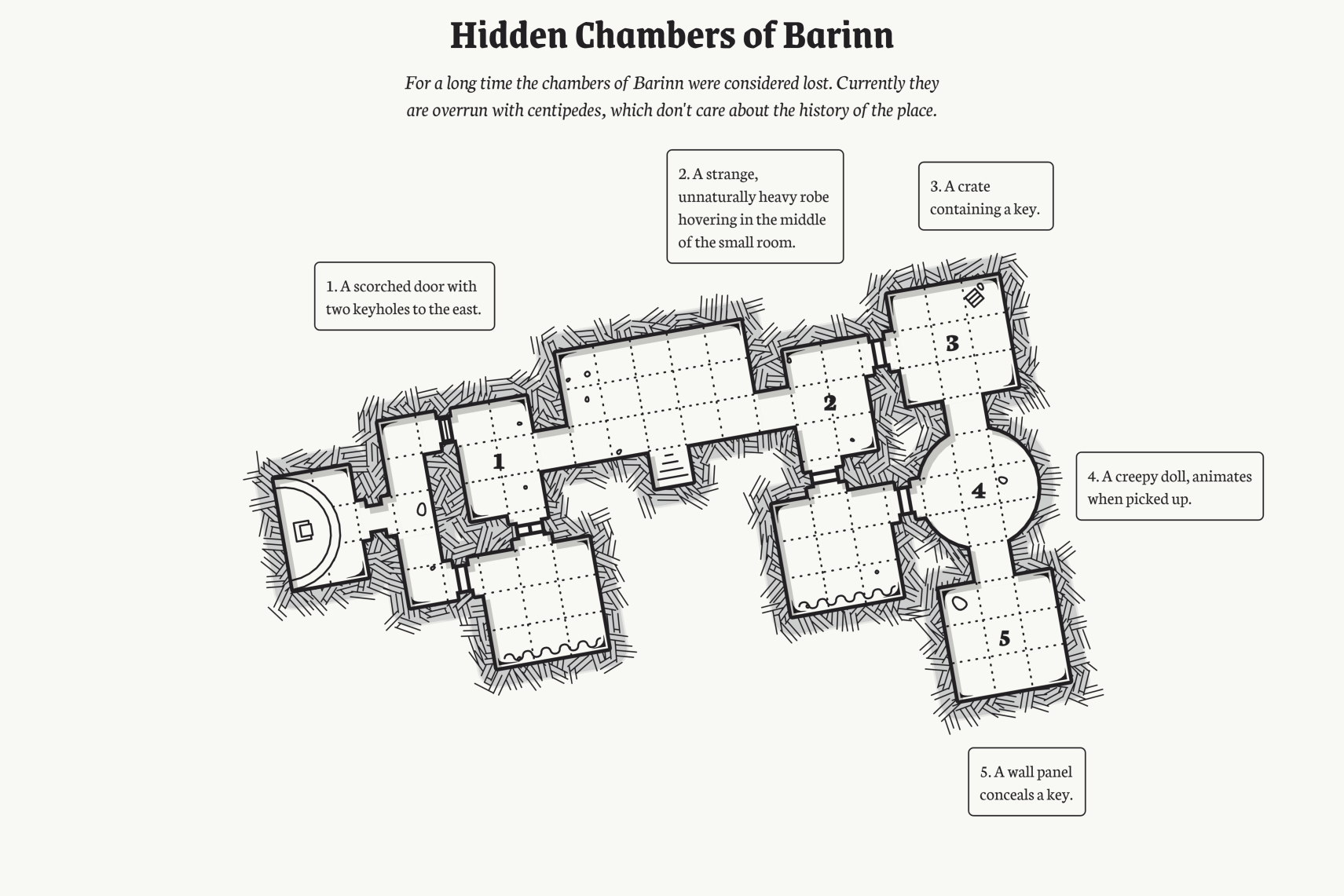
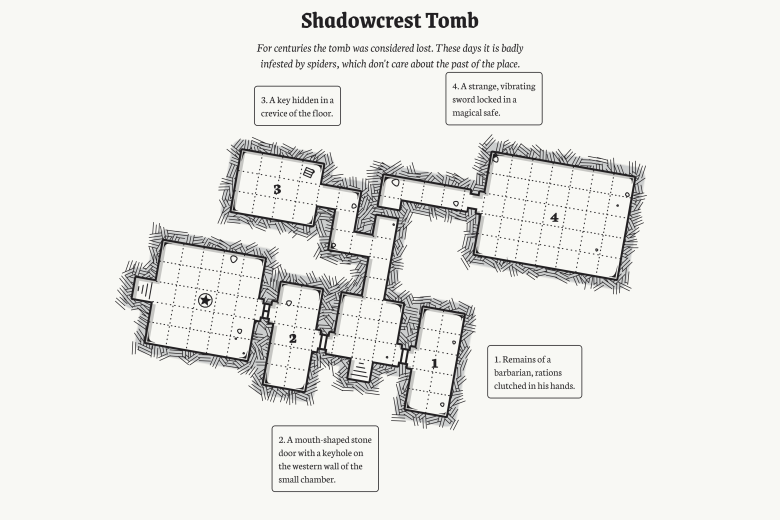
The idea is that these are *very* old building with stone walls, but there is no roof, as they were wooden and have rotted away long ago (i.e. there’s only walls, no roof anymore, so you can see the sky).

**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>

Note that some of these are quite large, so just keep refreshing the page until you get to a layout that that you’d like to make.

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.

(Interestingly the models are in 500x500x500 unit squares – coincidence?)

My point is that since we are using these as *building* floor plans rather than underground dungeons, you have a couple options:

* They could be long narrow rooms.
* They could also be outside walkways between separate buildings.

Use your own artistic discretion with this.

|  |
| --- |
| **Since this exam is for both exams, the questions indicate which course they apply to:**   * Questions 1 & 2: both exams * Questions 3 to X: INFO-6028 “Graphics 1” * Questions X to X: INFO-6044 “Game Engine Frameworks & Patterns” |

**For both INFO-6028 & INFO-6044:**

1. (5 marks) Pick a floor layout from the “One Page Dungeon” site.   
     
   If you right-click, you can:  
   * Save it as a PNG file.
   * Get the “permalink” so you regenerate it later.  
       
     (Here’s the one for “Shadowcrest Tomb”: <https://watabou.github.io/one-page-dungeon/?seed=1686096162>)
2. (200 marks) Using the “POLYGON Dungeon Realms” components, make floor and walls of your building.   
     
   For the “ground” use one of:  
   * the “Island” models (from the graphics mid-term),
   * one of the “fractal terrain generator” models from meshlab,
   * or another model you’ve found/made.

The only requirement for the “ground” the building is on is that it’s very large – I shouldn’t see the “edges” of the ground model (I mean it should be *way* off in the distance, so if the camera is anywhere near the building, the “ground” model should be HUGE in comparison, like 100x bigger or something like that.)

At this point, you don’t need any lighting. In fact, if can even be in wireframe if you’d like.

Position the camera so the entire building can be clearly seen – so that it can be compared to the floor plan you generated in question 1.

**For INFO-6028:**

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

1. (100 marks) Set up the night sky using the “Space skybox texture” (This is the same texture that we used in class).
2. (100 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).
3. (300 marks) Place a number of lit “torches” in the scene:
   * Place at least five (5) lit “torches” on the walls of your building(s). These should be spread out, with no more than one torch per “room”. 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:
     + Place an “imposter” (quad or tree or whatever) with a “fire” texture on it, in the part of the torch that would have the flame.
     + 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.
     + Modulate the length of the “fire imposter” as well as the brightness of the light quickly over time. In other words, the light should “flicker” like a fire and the imposter should *slightly* grow and shrink approximating the length of the flame.
4. (300 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:

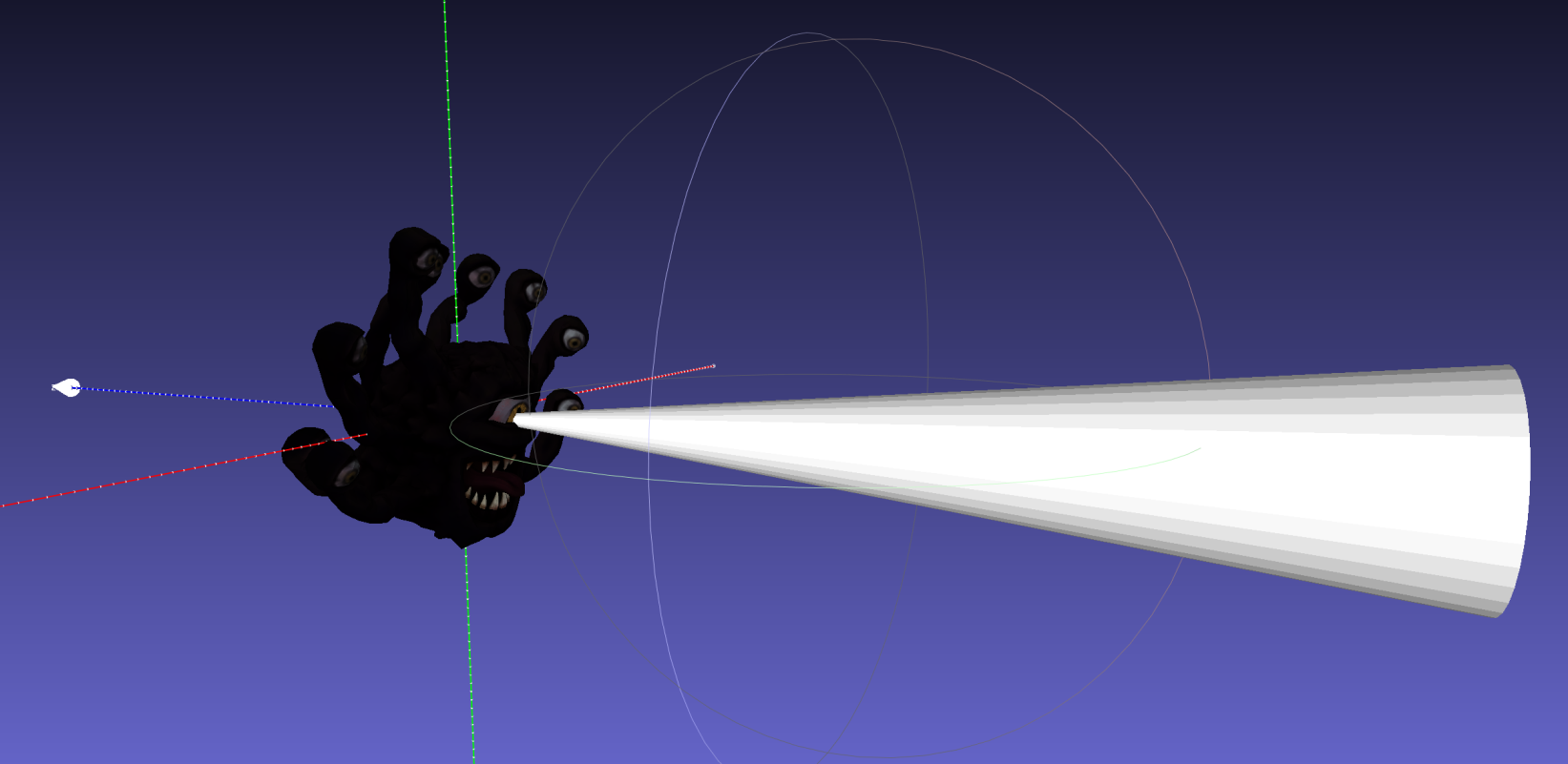
* + Place a very bright, but very “tight” spot light at each of the eyes.
  + This spot light will shine in some direction more or less where that eye is looking.   
    (Note: we’re talking about the little eyes on the stalks *not* the giant eye in the middle)
  + They are “tight” in that the spot cone 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.

**For INFO-6044:**

Note: While I’d encourage you to use what you’ve made up until question 8 (because it will look prettier) you *don’t* have the lighting/graphical portions from question 3-8 for this part of the exam, since it’s actually the INFO-6044 portion.

It can be all wireframe if you’d like – just so it’s clear what is happening.

1. (100 marks) Place three Beholders in various parts of your scene. You want these very far away from each other, so they can’t “see” each other. If your map has a single large room, then I guess they will have to all be in the same room, but we can assume they have bad eyesight or it’s dark or something (so they can’t see each other).
2. (100 marks) Add the “vision cone” to each of the Beholders, showing the direction they are looking. This cone is along the Z axis, with the “point” at the origin and it expanding along the negative Z axis, which is the same as the direction the Beholder model is looking:



1. (200 marks) Make the Beholders “patrol” the room they are in.
   * They should slowly float in a square/rectangle pattern to each corner of the room they are in.
   * The move along the walls (like parallel to, like they are moving “along” the wall)
   * When they get to a corner, they should slowly turn 90 degrees – to align with the next wall – then continue along this new wall.
   * The cones should move with the model to show what they are looking at.
   * If there is more than one Beholder in a room, space them out so they aren’t “bumping” into each other. They can all “patrol” in the same direction and pace if that helps.
2. (200 marks) Have the camera follow each Beholder for a time:
   * When the “F6” key is pressed...
   * The camera should *gradually* (i.e. somewhat slowly, but more importantly, smoothly) close to one of the Beholders. It should “follow” the Beholder (while it’s patrolling) at a “reasonable” distance (imagine you are invisible and were in the room with it, following it “pretty closely, but not *too* close so you don’t accidentally bump into it).
   * When the “F6” key is pressed again...
   * ...gradually fly over to the next Beholder.
   * And every time you press “F6” it goes to the “next one” (forever – like you can keep pressing the “F6” key to move to the next Beholder in the scene).
3. (200 marks) Oh oh, the Beholders noticed each other!
   * When the “F7” key is pressed, it’s fight time!
   * Remember that Beholders hate everything, even other Beholders.
   * When “F7” is pressed, the Beholders *gradually* move towards each other.
   * **Note this might mean that they have to move down hallways or whatever.**If they have to move outside, they *don’t* have to follow the pathways, though.
   * They *can’t* pass through walls.
   * When they are “close enough” to each other (maybe separated around 2-5’ – remember that the squares on the grid are 5x5 feet) they attack, but kill each other.   
       
     How sad.
   * To mimic this “death” you are to have them spin around and shrink in size until they are gone (pick some really small size, then make them invisible or something).

**That’s it.**