**INFO6044 – Game Engine Frameworks & Patterns**

**Final Exam – Friday, December 13th, 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>
* 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 **seven (7)** questions and **twelve (12)** pages.
* Your solution can be either graphical or console based (or graphical + console based if that’s helpful).
* 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)
* 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.
* 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 **Friday, December 13th** 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.*

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

* Your solution may not contain any third party “core C++” libraries (like boost).
* You many have other “utility” libraries, like ones to load textures, models, sounds, etc.
* **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.
* Unless otherwise indicated, all these solutions assume that you are creating/using a C++ project using Visual Studio 2008 through 2019 using the OpenGL 4.x API (with glfw, glad, and glm).

**Yes, it’s another Sky Pirate exam! Horary!**

****

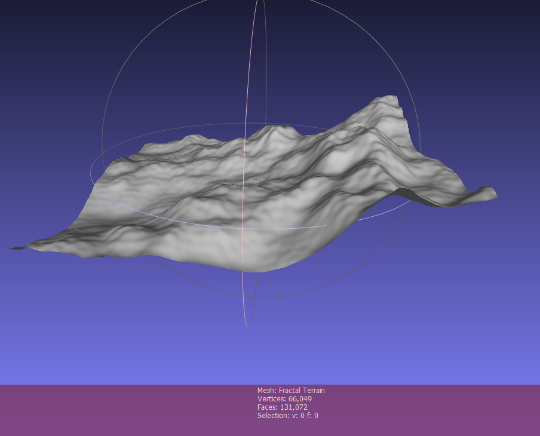


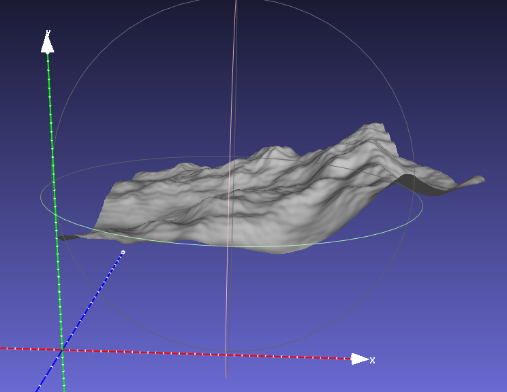
You are to create an animated scene involving a “broadside” attack between two sky pirates.

Classic “tall ships” had rows of canons along their sides. The fighting styles was to face the side of the ship towards a target, and first all the canons, shooting a volley of dozens of cannon balls.

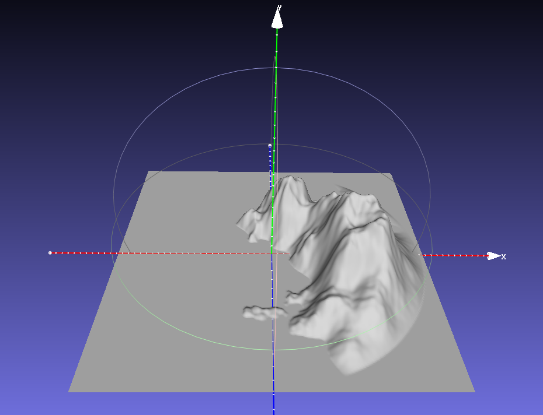
<https://unity3d.com/sites/default/files/game_articles/navalaction_3.jpg> (below) is a good image of this, as are the ship fighting scenes from the “Pirates of the Caribbean” or “Horatio Hornblower” movies:

1. (5 marks) Generate an island model using the Terrain\_to\_Island\_Converter program.   
     
   Using MeshLab, generate **one 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**.
* 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”*



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).

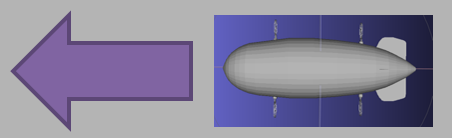
* Choose “Filters”, “Normals, Curvature, and Orientation”, then “Transform: Rotate”.
* Type in “-90” in the “Rotation Angle”, leaving the “Rotation on:” set to “X axis”, and click “Apply”, which will get you this 🡪
* Save this model with JUST xyz and NOT in binary form (“File”, “Export Mesh As…”, uncheck the “Binary encoding”, and choose OK.
* Download and compile the “Terrain\_to\_Island\_Converter” project. Drags the mesh file you made (above) onto the compiled exe file, which will generate something like the image to the right.   
    
  You should have one (or maybe two) large island shapes.

*Note that I’m only interested in the “animation” portion of the demonstration, not how “beautiful” it looks. While the scene does have to be “solid” rendered (i.e. not wireframe), and I need enough lighting to see what’s going on, you aren’t going to be marked for lighting, texturing, etc.*

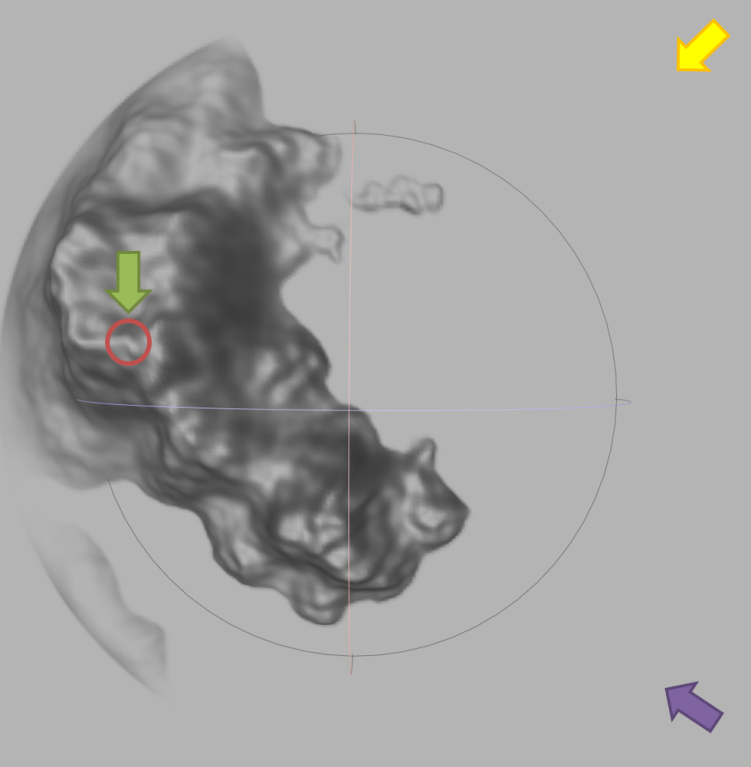
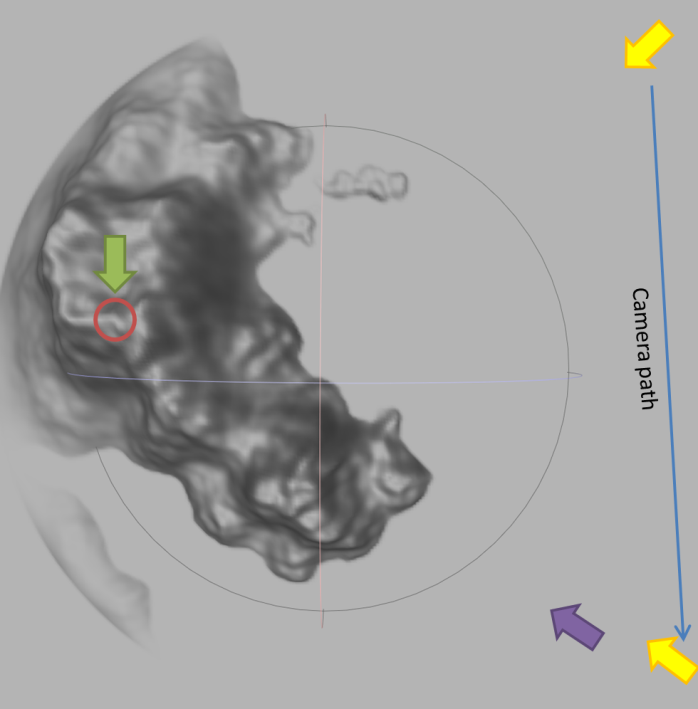
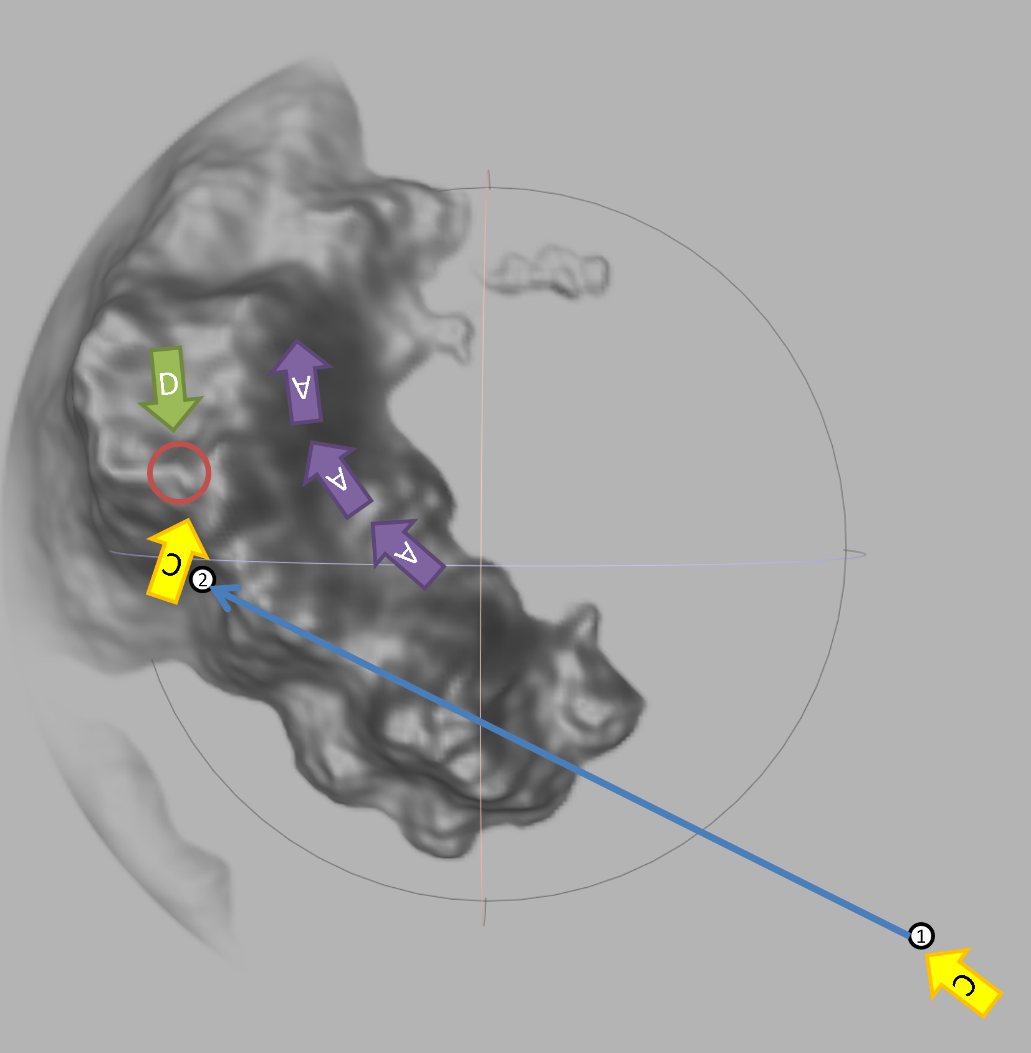
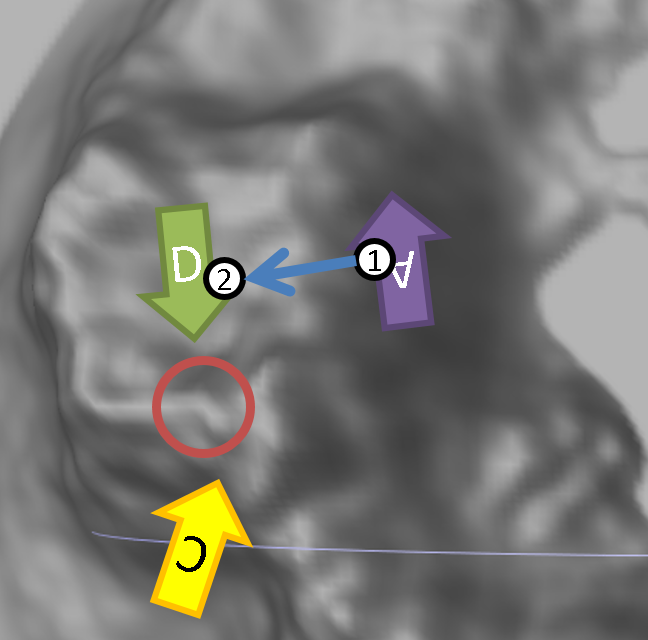
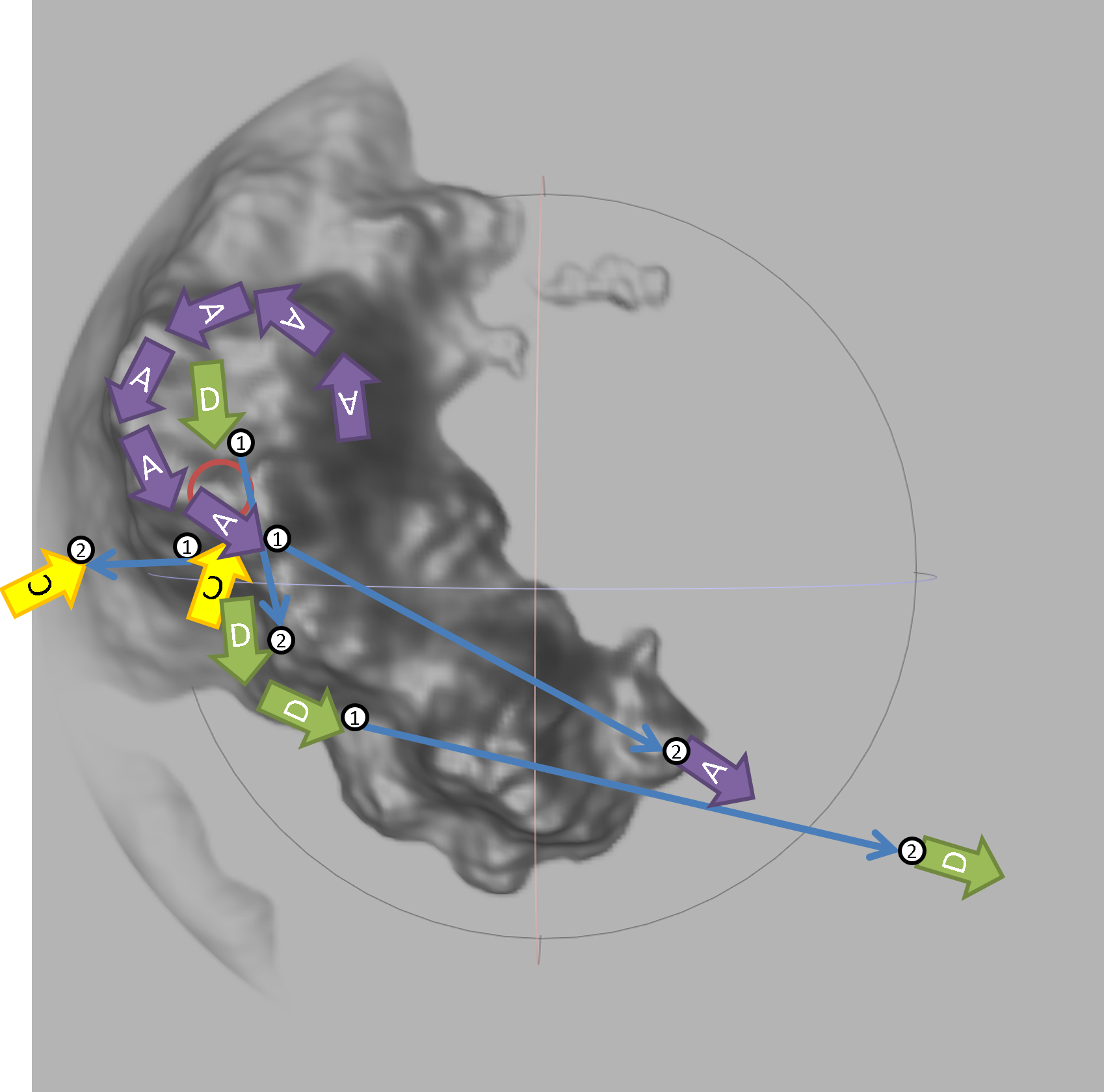
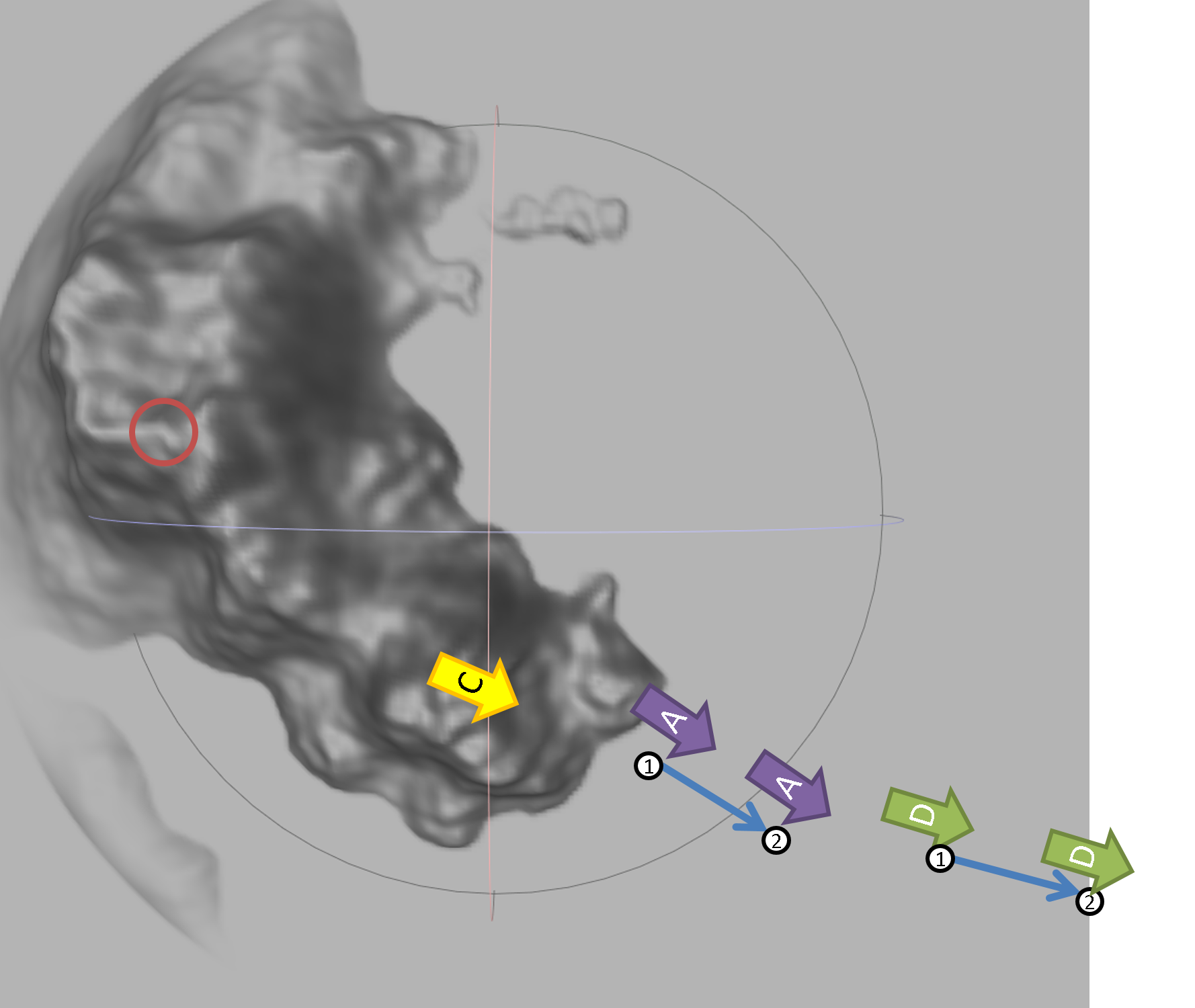
A brief overview of the scene is this:

* There is a sky pirate moored (parked) near the top of one of the island. We’ll call this the “defender”.
* Another sky pirate coming to attack these pirates. We’ll call this the “attacker”.
* The “attacker” will fly over the water towards the “defender”, which is moored (parked).
* When “close enough”, the attacker will turn to set up a “broadside” attack, so its side cannons are aiming at the stationary “defender”
* The “attacker” will shoot two salvos of cannon balls at the “defender”.
* During the 2nd salvo, the “defender” will start to flee.
* The “attacker” will turn chase the “defender”, flying in an arc to turn around, then pursue.
* Both ships will fly for a while, until the “defender” (in front of the “attacker”), shoots a single cannon ball backwards.
* This lucky shot will hit the “attacker”, which, now critically damaged, will slowly descend towards the water and sink.
* The “defender” will continue to fly away.

In the images below, the arrows represent the ships, with the “point” of the arrow indicating the front of the pirate ship, so something like this (the right image is the sky pirate ship, as viewed from above):



***For each question, the animation needs to be “controlled” by a Lua script. This can involve the entire animation, updated every frame, or it could be that the Lua is starting the animation, or combination. However, the Lua script must be present and “control” the animation or no marks will be given.***

1. (20 points): Load these island models into your OpenGL application, with two (2) copies of the pirate ships, and the “small factory” model.
   * Place the small factory model at the highest point on the island model (red circle on island); note you’re island likely looks slightly different from this image, of course.
   * Place the “defender” ship (green arrow, near the red circle) near the factory, like it’s “moored” near the top of the two towers.
   * Place the “attacker” ship (purple arrow, at the bottom right) at the bottom corner of the model, over the water.
   * Both ships should be at approximately the same height.
   * Place the camera (yellow arrow, at the top right) at another corner of them scene, facing the island. The camera should be far enough away to see both pirate ships.
   * Note the orientation of the ships and camera 🡪
2. (30 marks) By pressing (*and releasing*) the “3” key (*once*), the camera will move towards the “attacker” ship:  
   * The camera (yellow arrow) should stay at the same height as it did in question 2.
   * It should be looking at the “defender” (green) ship (i.e. It’s possible that you *can’t* see the attacker ship at some point in the movement)
   * It should move in a straight line towards the attacker ship, slowing down, and stopping *behind,* and *slightly to the right* (starboard) side of the attacking pirate ship. It needs to be about “2 pirate ship lengths” behind the attacking ship.
   * Note that the “Camera path” line in the image is a straight line “up and down”, but you need to move the camera *from where it was in question 2* to the final location *in a straight line*.
   * This movement should take about 5 seconds.
   * The camera should “ease out” (slow down) in the last 1 second of movement.
3. (40 points) By pressing (*and releasing*) the “4” key (*once*), the attacking ship flies towards the defending ship’s base. When it’s “close” (about 2x the length of the ship) it will turns towards the defender, in preparation for the broadside attack.   
   * This animation of this scene should take about 40 seconds, approximately.
   * The attacker, which had been going in a straight line, now “arcs” into position, moving forward, while turning the ship, and slightly ascending above the defender.
     1. The 3 purple “A” arrows indicate the general idea
     2. *NOTE: at the end, the attacking ship should be slightly higher than the defender – so the “boat” part of the attacker should be level with the “balloon” part of the defender.*
     3. This should take about 20 seconds (about ½ the time that this entire question takes)
     4. The rest of the time, the ship is in the final location, stationary (getting ready to “fire”)
   * About ½ way through this motion (the middle “A”, at about 10 seconds in), the camera should move from position “1” in the diagram (near the bottom, right of the image), to position “2”, which is positioned in front of the “defending” ship and base, looking slightly back at the “attacker”.   
       
     The camera should continue to move for the remaining 30 seconds (of the 40 seconds total).   
       
     The camera should end up closer to the defender, and looking somewhere in between both ships.   
       
     Both ships should be visible on screen at all times.
4. (40 points) By pressing (*and releasing*) the “5” key (*once*), the attacking ship starts its broadside attack.   
   * This animation of this scene should take about 40 seconds, approximately.
   * Each ship has four (4) cannons on each side, so each “broadside salvo” means that all four (4) cannons fire.
   * Each cannon ball should fly from the attacking ship to the defending ship. Use a sphere, teapot, cow, or bunny model for the “cannon balls”.
   * There are four “ports” (holes) in the side of the ships. This is where the cannon balls come from.
   * The cannons *don’t* shoot at the same time, but one at a time, then need to be reloaded.
   * Fire four shots, one from each of the cannons, from the attacker to the defender.
   * Make the speed quite slow (it’s a game, after all), sort of like if it was a cartoon.
   * It should be clear where the cannon “balls” are coming from, and that they are hitting the defending ship somewhere in the “boat” portion (*not* the “balloon” part).
   * After the first “salvo” (4 shots), the cannons need 10-15 seconds to reload.
   * While this reload is happening, the defender starts to fly away, *very slowly:*
     1. The defender will *slowly* start to ascend (get higher). This takes longer than the reload time of the attacker.   
        (in other words, the attacker starts another salvo *while the defending ship is still ascending*)
     2. Repeat the broadside attack, as you did above: four shots, from each cannon, hitting the defending ship, but keep in mind that the defender is no higher than it was (you are hitting the “boat” part, not the “balloon” part.
   * After the 2nd salvo attack, the defending ship continues to ascend a little higher, until its “boat” portion is at the height of the attacker’s “balloon” portion.
5. (40 points) By pressing (*and releasing*) the “6” key (*once*), defending ship starts to fly away, with the attacking ship in pursuit.   
   * This animation of this scene should take about 40 seconds, approximately.
   * The camera and both ships are moving during this portion.
   * The camera moves in a straight line from position “1” to position “2” in the image, where position “2” is away from the island. The camera is always looking at the attacking ship.
   * The defending ship moves in two straight stages, generally in two straight lines. The first stage takes the ship to *almost* over the water, then it turns (or arc while moving), and starts moving away from the island, over the water.
   * The attacking ship also moves in two “stages”, which are:
     1. A circular, arcing movement, where it is moving forwards, while turning around.   
          
        It sort of moves around the peak of the island, while its turning around.   
          
        This can be done with a series of straight lines combined with a rotation, or by following a mathematical curve (giving a bonus of 10%)
     2. At the end of this first stage, it is approximately above the defenders base
     3. The second stage is a straight line, following the defender.
   * The 1st stages of both the attacker and defender movement should take (approximately) the same length of time.
6. (40 points) By pressing (*and releasing*) the “7” key (*once*), defending ship shoots down the attacking ship.   
   * This animation of this scene should take about 20 seconds, approximately.
   * Both ships continue to travel in a straight line, with the attacking ship following the defending ship (not actually “following”, but moving in the same direction)
   * The camera should continue to look at the attacking ship, staying at approximately the same height as the attacking ship, even as it starts to sink into the water (more on this in a moment)
   * At the start of the movement, a single cannon “ball” from the defending ship flies from the back of the defender into the “balloon” portion of the attacker.   
       
     This should be slow, so you can see it clearly.
   * Right after the cannon ball “hits”, the attacking ship should start to descend into the water. It should still be moving forward, though.
   * While the attacking ship will continue to descend below the water, the camera will follow the ship, moving down, but will *not* actually go underwater, but stop somewhere close to the surface.
   * Once the attacking ship is completely submerged, the camera should slowly change to look at the defending ship, which continues to fly away.   
     1. Hint: you can do this by not *actually* looking at the attacking ship, but looking at some invisible object that moves (fairly quickly), in a straight line, from the position of the (now sunk) attacking ship, to the (flying away) defending ship.

**That’s it.**