INFO6044 – Game Engine Frameworks & Patterns

Final Exam – Monday, December 10th, 2018

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 **eleven (11)** 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. My suggestion is that you place each answer in its own folder, named "Question_01", "Question_02" and so on (or something equally clear). Another option is to create a Visual Studio solution and add a number of projects – one per question – to it. 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. 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.

- Place any written ("essay" or short answer) 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 **Monday, December 10th** 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) or C++11 "`auto`" feature. If it has either of these things, the question(s) will not be marked (because it won't build). You many have other “utility” libraries, like ones to load textures, models, sounds, etc.

- When ready to submit, please delete all the “extra” Visual Studio files before zipping it up (remember this is C++, so all I really need is the .h and .cpp files, right?), like the “Debug” and “Release” folders with the “obj” files, as well as the intellisense file (in VS2017, that’s the “.vs” folder).

- **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 2017 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.

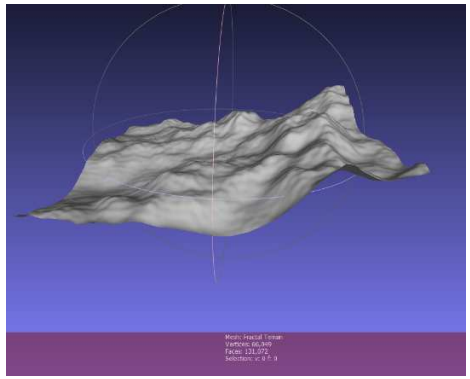
Classic “tall ships” had rows of canons along their sides. The fighting styles was to face the side of the ship towards a target, and first all the canons, shooting a volley of dozens of cannon balls.

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:

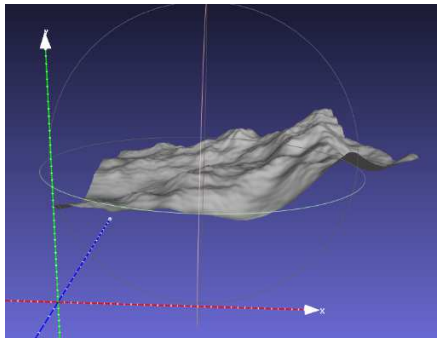
1. (5 marks) Like the mid-terms, generate an island model using the Terrain_to_Island_Converter program.

   Using MeshLab, generate at least 4 “islands”, 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”*

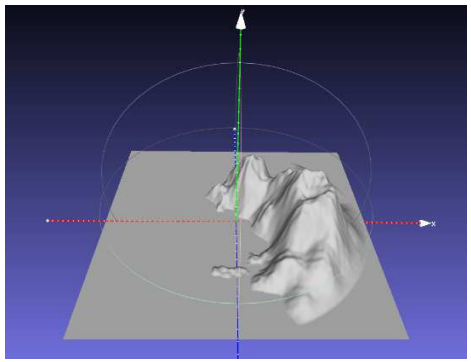


   MeshLab assumes that “up” is “z”, so we need to adjust this. Turn on the “axis” drawing by choosing “Render”, “Show Axis” to make this clear (if you want).

   - 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.

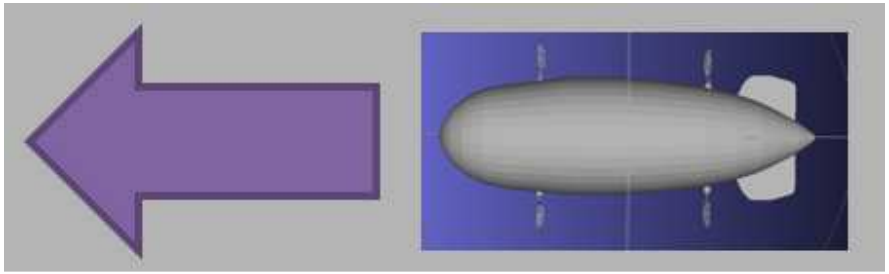


Note that I'm only interested in the "animation" portion of the demonstration, not how "beautiful" it looks. While the scene does have to be "solid" rendered (i.e. not wireframe), and I need enough lighting to see what's going on, you aren't going to be marked for lighting, texturing, etc.

A brief overview of the scene is this:

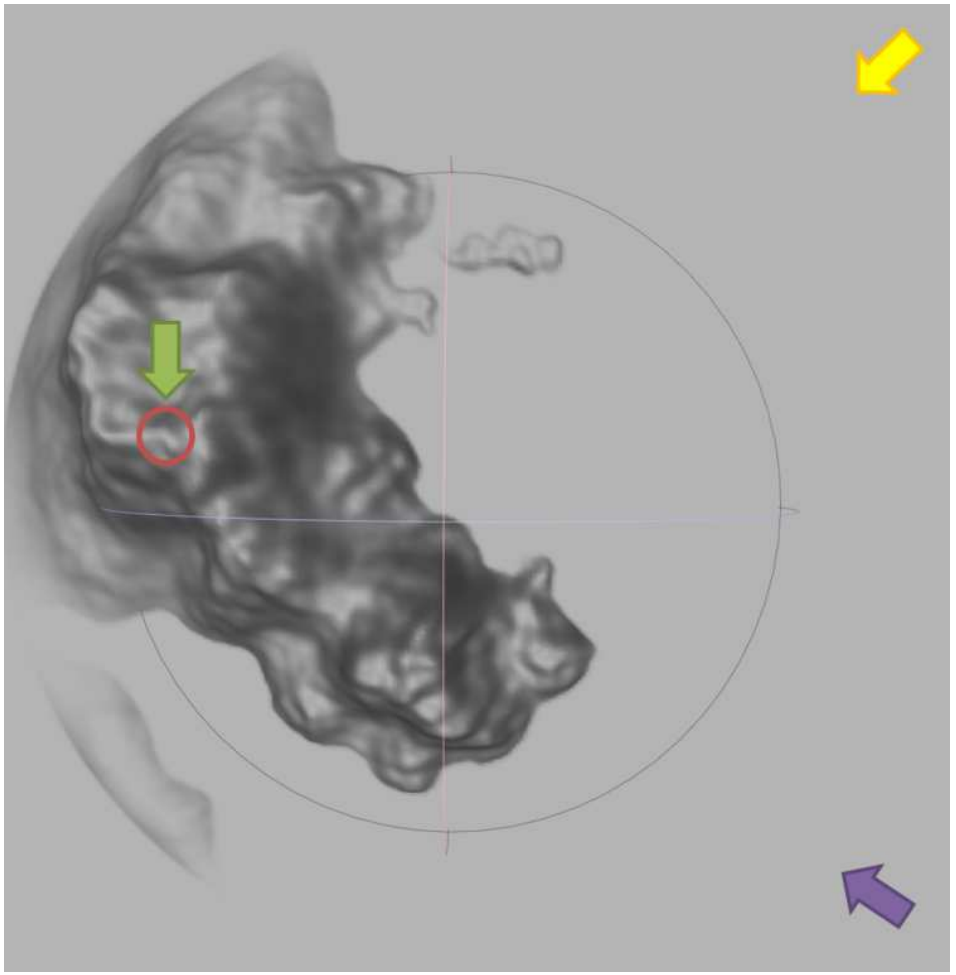
- There is a sky pirate moored near the top of one of the island. We'll call this the "defender".
- Another sky pirate coming to attack these pirates. We'll call this the "attacker".
- The "attacker" will fly over the water towards the "defender".
- As it gets closer, the "defender" will turn and start moving towards the "attacker"
- When they are close, they will both turn to set up a "broadside" attack, so they are lining up the sides of the ships. They will line up in opposite directions (i.e. one will turn right, the other left, so they end up facing opposite direction).
- Once they are lined up, they fire their cannons at each other.

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):



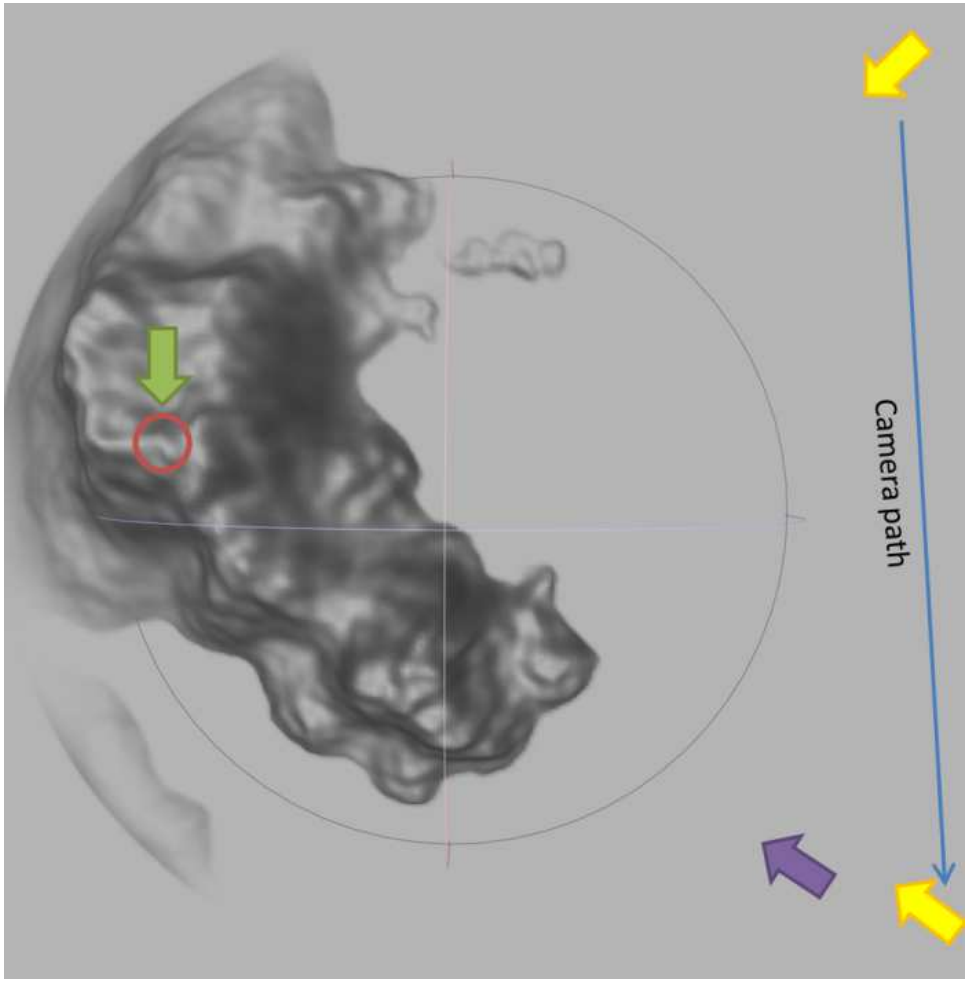
For each question, the animation needs to be "controlled" by a Lua script. This can involve the entire animation, updated every frame, or it could be that the Lua is starting the animation, or combination. However, the Lua script must be present and "control" the animation or no marks will be given.

2. (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 →



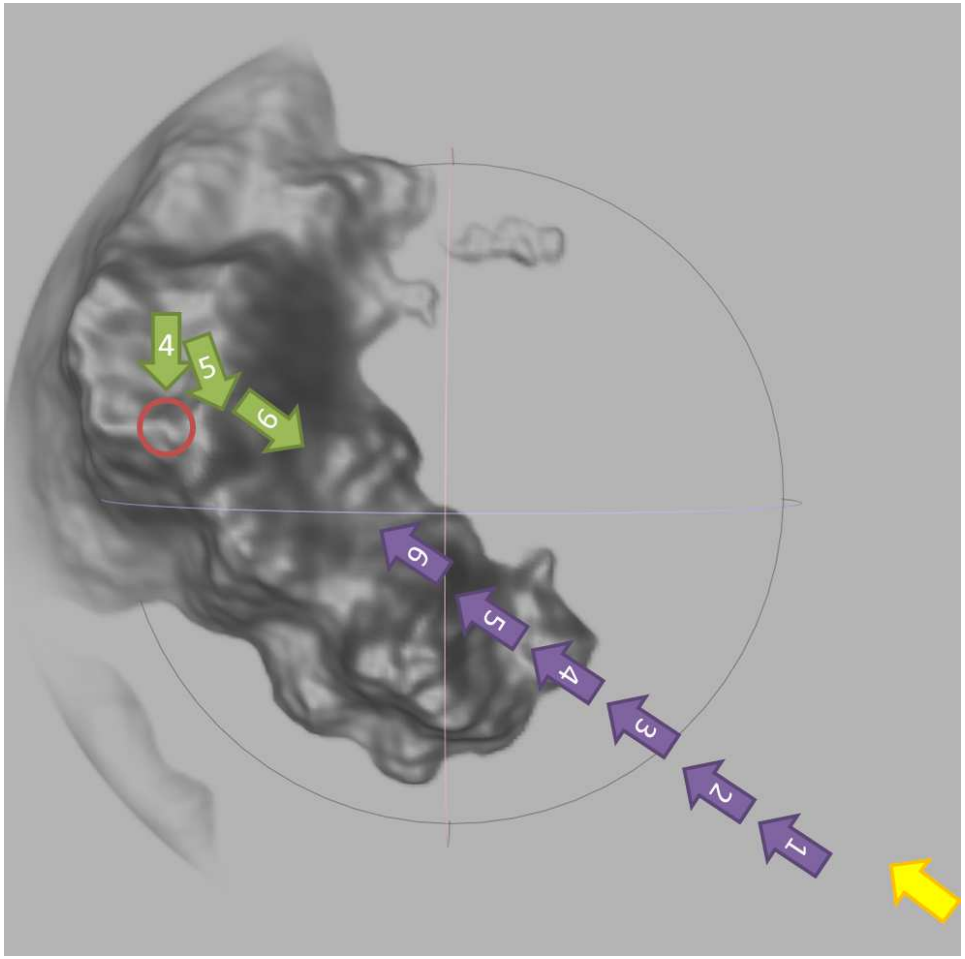
3. (30 marks) By pressing (*and releasing*) the “1” key (*once*), the camera will move towards the “attacker” ship:

   - 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.



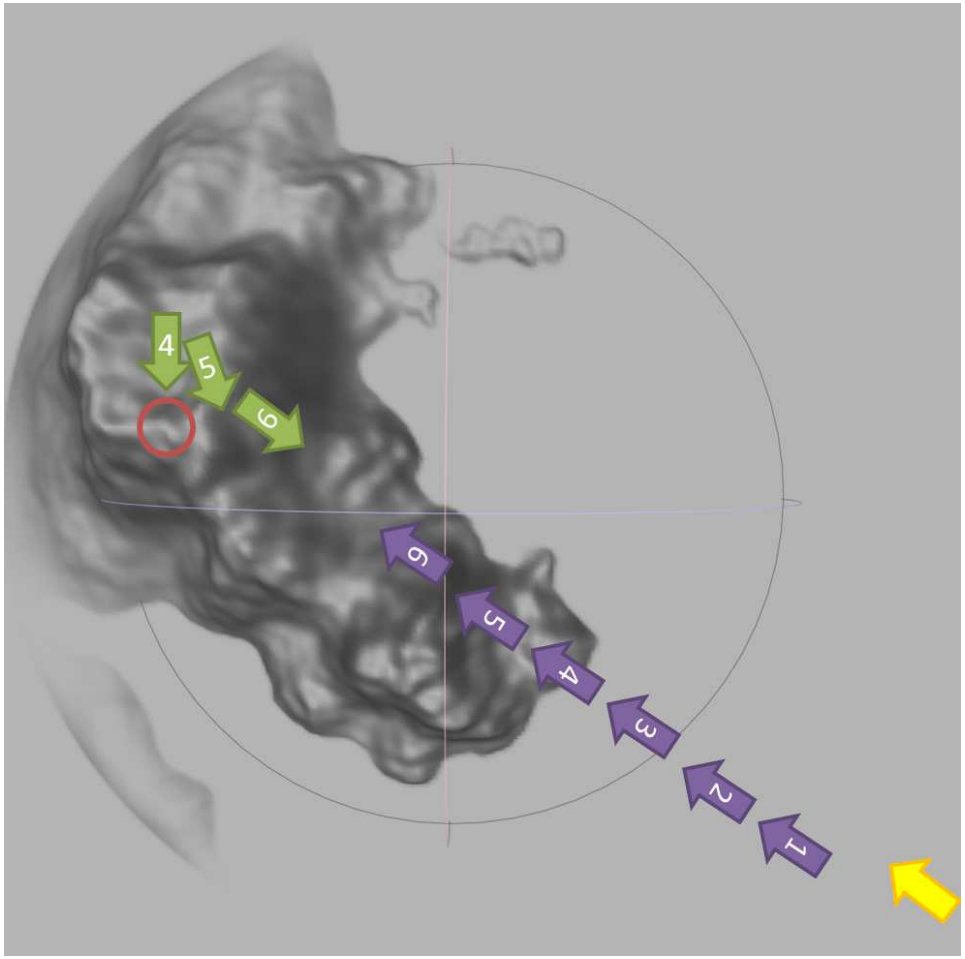
4. (40 points) By pressing (*and releasing*) the “2” key (*once*), the attacking ship flies towards the defending ship’s base. When it’s about ½ way there, the defender turns towards the attackers, and starts to fly to meet the attacker.

- This animation of this scene should take about 15-20 seconds, approximately.
- The number on the image are meant to indicate the timing:
  i. The attacker moves from 1 to 6
  ii. The defender doesn’t move until around 4 (when the attacker is a little more than ½ way along)
- Note that the defender (purple) is **both** turning to face the attacker and starting to move towards the attacker during this phase.
- The camera should stay in the same relative position behind the attacker, so a little behind and to the right (starboard) side.
- The camera should be looking at the defender (green), but needs to be enough to the right (starboard) so that the defender can always be seen; i.e. the attacker ship should never get “in front” of the camera and block the view of the camera (it should always be able to see the defending ship)



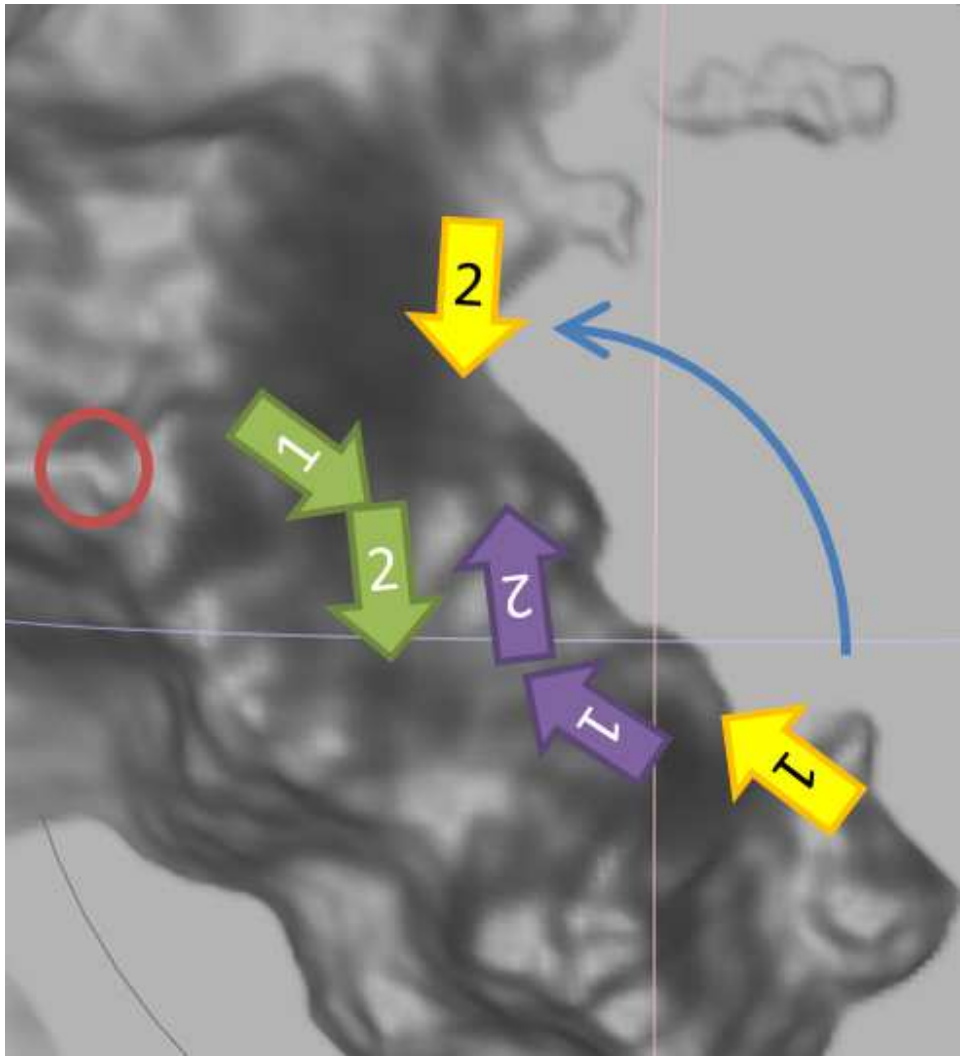
5. (40 points) By pressing (*and releasing*) the “2” key (*once*), the attacking ship flies towards the defending ship’s base. When it’s about ½ way there, the defender turns towards the attackers, and starts to fly to meet the attacker.

- This animation of this scene should take about 15-20 seconds, approximately.
- The number on the image are meant to indicate the timing:
  i. The attacker moves from 1 to 6
  ii. The defender doesn’t move until around 4 (when the attacker is a little more than ½ way along)
- Note that the defender (purple) is **both** turning to face the attacker and starting to move towards the attacker during this phase.
- The camera should stay in the same relative position behind the attacker, so a little behind and to the right (starboard) side.
- The camera should be looking at the defender (green), but needs to be enough to the right (starboard) so that the defender can always be seen; i.e. the attacker ship should never get “in front” of the camera and block the view of the camera (it should always be able to see the defending ship)



6. (40 points) Immediately after question 5, this stage should start automatically (i.e. you *don't* need a key stroke to start this, although you can; if you choose to do this, it should be the "3" key). Both ships turn to go "along side" of each other (the long sides of the ships are facing each other).

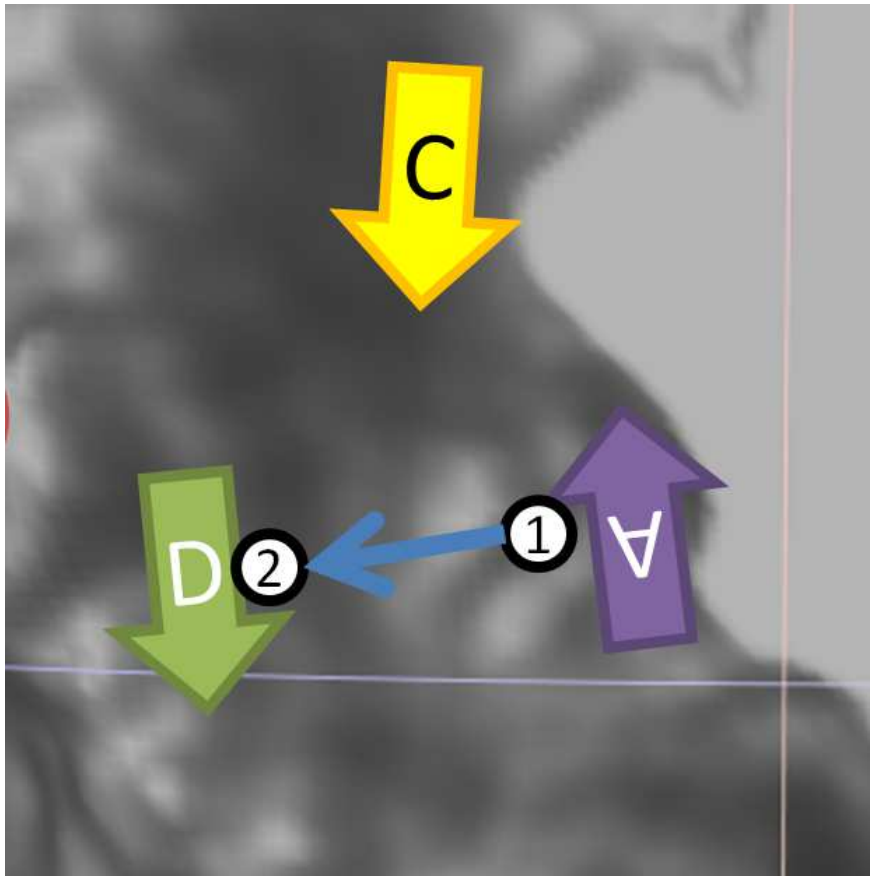
- In the image, the numbers attempt to indicate the starting and ending positions
- Both ships should continue to move forward *while* they are turning.
- Once they are alongside each other, and then stop.
- They should "ease out" (decelerate) to a stop.
- While this is happening, the camera should move:
  i. Move from position 1 to 2
  ii. It should be looking at some location mid-way between the two ships
  iii. The camera can go in a straight line ***or*** along a curve as in the image (for a **bonus** of 20 marks)



7. (40 points) By pressing (*and releasing*) the “4” key (*once*), the attack begins!

- The camera (yellow arrow with the “C”) should stay in the same location as it was at the end of question 6.
- You are showing one “barrage” of the broadside canons:
  i. There are four (4) cannons on each side of the ship, for a total of eight (8) cannons that will be fired.
  ii. Use a sphere model for the cannon balls.
  iii. The balls should fly from the cannon to the other ship *in a parabolic arc.* i.e. they *can’t* fly in a straight line from one ship to another.
  iv. Each of the cannon ball arcs has to be slightly different.
  v. The cannons fire one after the other in a random fashion, at random intervals. So while the cannons fire one after the other, the interval between shots should be a random value. i.e. they *don’t* fire all together, nor does one ship fire all at once, then the other ship, and they don’t fire in a “even” tempo.

  (I realize that if it’s actually random, it could be that one ship fires all the balls, then the other, and it’s also possible that the interval would be the same, but this is *exceptionally* unlikely).



That’s it.