**INFO6019 – Physics 1**

**Final Exam – Friday, December 13th, 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. 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.
* 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 **five (5)** questions and **five (5)** 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).
* You may ***not*** use the C++ **auto** keyword.
* 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 **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 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).



This large ship is a Star Destroyer from the Star Wars movies.

In one of the movies, one of the ships is destroyed when one of the large spheres (yellow arrow in the picture) at the top of the ship are destroyed: <https://youtu.be/sMqR0ANOSVM> (11 seconds in)

Apparently, the nerds have determined that these spheres are “deflector shield generator globes”, so we’ll call them that from now on (<https://starwars.fandom.com/wiki/Imperial_I-class_Star_Destroyer>)

You are going to simulate an attack on a Star Destroyer from a number of X-wing fighters. Here’s what they look like:



There are two models that you can use:

* The Star Destroyer, which has a number of different resolutions BUT *you must use the highest resolution model for your physics calculations and for the final submission.*
* An X-wing fighter.

1. (10 marks) Place the Star Destroyer in the scene. Use enough light to see it, though I’m not specifically marking the lighting, this shouldn’t be wire-frame or something – it should be a something similar to what you see in the video (which can be done with a single light).
2. (50 marks) Set up an “attack run” with the X-wing.

The idea here is that the X-wing is flying quickly towards the Star Destroyer, getting as close as they can, shooting their space bomb/whatever then flying away. From approximately 46-52 seconds in this clip shows the “Y-Wings” doing something like this: <https://youtu.be/ZkZvyfLC-LU>

When the “2” key is pressed, this functionality will happen:

* Randomly pick two locations around the Star Destroyer. At a scale of 1.0, the Star Destroyer has a bounding box of approximately -500 to +800, on the longest edge, for a length of 1,300 units. These points should be at a range of 3,000 units away from the centre of the Star Destroyer (approximately)
* If you pick a point inside the Star Destroyer, indicate this by printing something to the console (“Picked a point inside the Star Destroyer. Picking another point…”) and pick another point. Keep picking points until you have two points in space, on “either side” of the Star Destroyer (so that the X-Wing will fly “past” the Star Destroyer).
* These two points mark the path that the X-Wing will take. Show this path by drawing a large number of very small spheres along this path (simulating a line, or something like when you are using Google Maps). The starting and ending spheres should be a different colour and a different size. The spheres between these start and end points should be quite small, relative to the X-wing. (Much smaller than the “bullet” that the X-Wing will fire in question 3).  
    
  Note while the start and end points *can’t* be “inside” the Star Destroyer, the *path* that the X-wing can travel *can* take it “through” the Star Destroyer. This will happen pretty often, in face.  
  See question 3 to see how this “through” path gets handled.

**Note:** Keep in mind: the 2nd half of the course was “broad-phase collision detection”, specifically accelerating the eventual narrow phase with a broad-phase stage. You will need to have a narrow phase stage here, but it must run at interactive frame-rates and/or not completely saturate your CPU with unnecessary “brute force” “every triangle tested every frame” sort of solution.

In other words, it should run at the same frame-rate and CPU usage that is comparable to just drawing the two models and not doing any collision detection at all. If the frame-rate is very low or the CPU utilization very high (i.e. doing unnecessary detections), then you will likely lose the marks for the rest of the questions.

Similarly with simplification of the test: there eventually *must* be a ray-triangle/triangle-triangle narrow phase collision detection stage, not just a broad-phase and/or simplified geometry detection happening (like *just* a sphere/ray detection for instance).

1. (200 marks) Make the attack run:   
     
   This continues from question 2, in that when the “2” key is pressed, the steps of question 2 happen, then the steps of this question continue immediately after.   
   (i.e. *don’t* remove the spheres you drew in question 2)

* “Fly” the X-Wing along the path.
* The X-wing should be oriented along this path (worth 10 of the 50 marks).
* Move the camera with the X-wing, at some “large” distance away, but so that you can clearly see the X-wing, the direction the X-wing is headed, and some of the Star Destroyer (if it’s visible). I’m suggesting a distance of something like 10-30x the length of the X-wing.   
    
  You can just pick a fixed offset from the X-Wing and use the “look at” camera transform for this.
* The X-wing should avoid crashing into the Star Destroyer, and so will never get within approximately 10x the length of the X-wing. This can happen in two ways:  
  + Either it is flying right towards the Star Destroyer (like if it keeps going, it would crash into it) in which case it will “pull away” and fly away – explained blow.  
    (This is sort of what happens here: <https://youtu.be/sMqR0ANOSVM>)
  + If never gets close enough to fire.
* **If it gets within this length,** it should “fire” its weapons, and turn around, flying back the way it came. It can immediately turn around (it doesn’t have to “arc” back or anything like that).
* **If it *never* gets close enough**, then it will **NOT** fire its weapon, and will continue along the path, to the end, and it’s “run” is complete and it will fly off into space.
* The “bullet” that the X-wing shoots is represented with the “tear drop” shaped model.
* Move this “bullet” along the path the X-wing was originally flying. This happens at the same time that the X-wing is flying the other way.

The idea is that the X-Wing will make an “attack pass” across the Star Destroyer, but doesn’t necessarily shoot at the shield generators – if it’s not “close enough”. The pilot would only choose to shoot if the attack run is “good” (like in the video), but many of these attack runs *won’t* be good and the pilot would choose to abort and fly away or just not shoot at the shield generators.

1. (200 marks) Indicate a hit on the “deflector shield generator globes”:

* If the bullet hits the Star Destroyer, then indicate this by placing a *small* coloured sphere at the impact point.
* If the bullet hits one of the “deflector shield generator globes” (spheres at the top of the model), then indicate this by updating the title of the OpenGL window (you can do this by calling glfwSetWindowTitle() function).   
    
  The globes have a “health” of 100% each, and each hit takes out 25%. The text should indicate how much “health” is left in each globe.

1. (50 marks) “He’s dead, Jim…”

* If BOTH globes lose health, destroy the Star Destroyer in this manner:
* Print something to the console and update the OpenGL window with something like “You’ve destroyed the Star Destroyer!”
* Alter the *vertex* shader by:  
  + Adding a uniform variable called “offset” that is a single float value, like this:

**uniform float explosionOffset;**

* + Adjust the *pre-transformed* vertex location (i.e. *before* you multiply the vertex by the model\*view\*projection or MVP matrix), by multiplying it by the vertex normal and this “explosion offset” variable.   
      
    What will happen is that the vertex will move away from where it was, along the normal of the triangle, as the value of “offset” gets larger. In other words, the triangles of the model will all fly away along the direction of the normal. Since the normals face “out” the ship will “explode”.
  + Starting with 0.0f, make the “offset” value a little bigger each frame. This will cause the Star Destroyer to “break apart” on the screen. If it’s taking too long, adjust the offset value by a larger amount each frame.

That’s it. May the force be with you.