**INFO6044 – Game Engine Frameworks & Patterns**

**Midterm Exam – Monday, October 21st, 2024**

Instructor: Michael Feeney

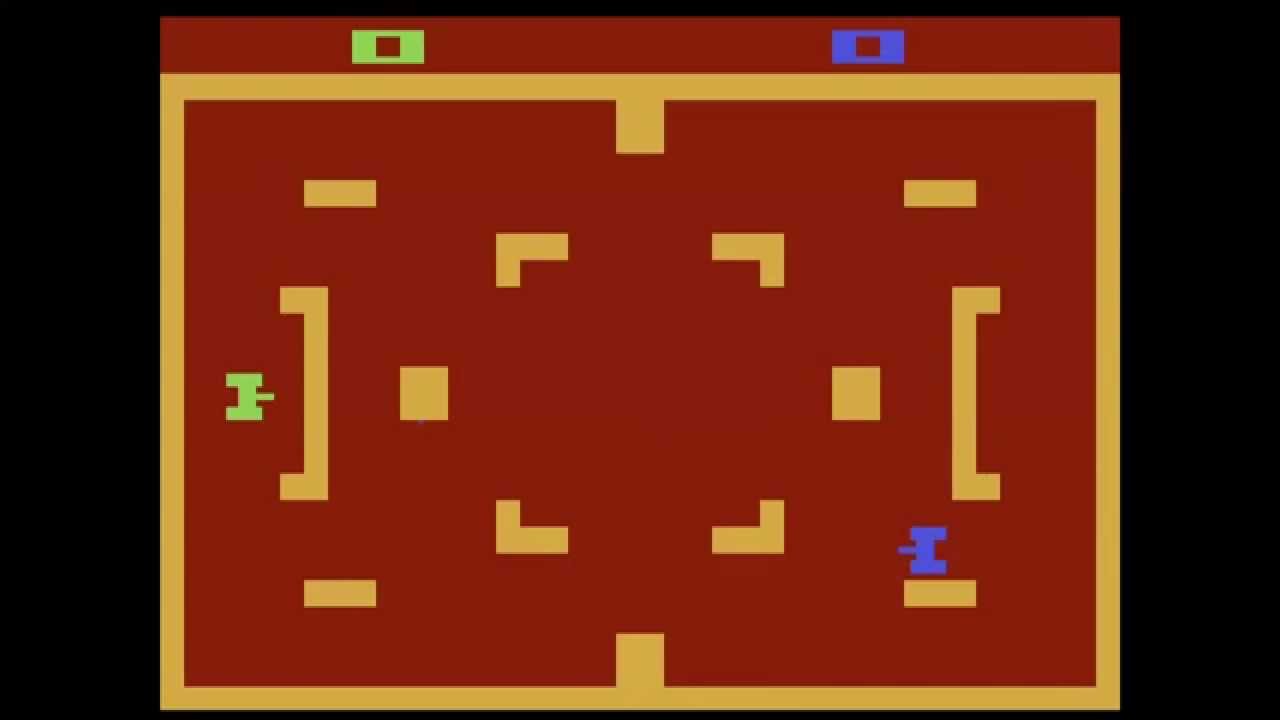
## The exam format:

* 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.   
  + <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 **nine (9)** pages.
* Your solution **must have a graphical output** (or graphical + console based if that’s helpful).  
  + Note: You are not being marked for the “beauty” of your graphical output.   
    I’m only interested in being able to see what’s going on. I’d suggest a single large point or directional light so that you can clearly see everything.  
    A camera that can move around the scene might be helpful, but is not required – so long as I can see the entire maze *and* see what’s going on.
* **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.
* Your solution may **not** contain any third party “core C++” libraries (like boost). I will not have boost installed, and will not install it; as a result, if you using boost, your solution will *not* build, and you will receive a mark of zero (0).
* You many have other “utility” libraries, like ones to load textures, models, sounds, etc. However, make sure your submission is complete so that I can build your solution.
* 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.
* solutions must be using a x64 Release library C++ project using the default settings of Visual Studio 2019 using the OpenGL 4.x API (with glfw, glad, and glm).
* **You many not use the C++ “auto” keyword. If you do, you will get a mark of zero.**
* You have until **11:59 PM** on **Monday, October 21st** 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)

**“Atari Combat 2024 edition!”**



You are going to simulate a variation of the Atari 2600 “Combat” game, specifically the “tank area” style game, along with a number of improvements.

Google the game and you will see it’s pretty simple: Each player controls a tank, which can move forward, turn (well more like aim in a few limited angles), and shoot a projectile.

Yours will be a variation of this, with more tanks, and a limitation that the tanks can only turn at 90°.



You are to use the “Low Poly Tank Model” (from TurboSquid) for all the players:

There is also an INFO6044\_Midterm\_Helper program that you are to us that will generate the “arena” that the tanks will battle in.

You use this program by passing your student number, for example, if yours was 12345, you’d type:

“INFO6044\_Midterm\_Helper.exe 12345”

(You can also edit the Run\_Me\_Your\_SN.bat file with your student number, of you’d like)

As well as the screen output, this generates two files:

* A “**Maze\_XXX.txt**” file where “XXX” is your student number. This has the same text based arena maze it displayed (but flipped along the y axis to match the ply file generated).
* A **mazeOuput.ply** file which you can use if you’d like. This is a 3D variation of the arena maze in the text file. Note that it generates an XYZ, RGBA format so if you need something like normal, you’ll have to do that yourself (in MeshLab, it’s: Filters; Normals Curvature and Orientation; Re-Compute Face Normals)

Here’s an example arena maze text output (left) and the ply output (right). This is for SN 12345 (I think?).

|  |  |
| --- | --- |
| XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  X X X X X X X X X X X X X  X X XXXXXXX X X X XXXXXXXXX XXX X XXXXXXXXXXX XXX XXX XXX X XXXXXXX XXXXX X X XXX XXX X X XXXXX X X X  X X X X X X X X X X X X X X X X X X X X X X X X X X  X X X XXX XXXXXXXXX X X X XXX XXXXXXX XXXXX XXX X XXXXX X X X XXXXX X XXXXXXXXX X X XXX XXXXX XXXXX X  X X X X X X X X X X X X X X X X X X X X X X X X X X X  X XXXXXXX X XXXXX XXXXX X X X X XXX XXX XXX X XXXXXXXXX X X XXX X XXXXX XXXXX XXXXX X X X XXX X X X X  X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X  X X X X XXXXX XXXXXXX X X XXX X XXXXXXXXX X XXXXXXXXX X XXXXX X XXXXXXXXXXX XXX X XXXXX XXX XXX X X X  X X X X X X X X X X X X X X X X X X X X  XXXXX XXXXXXXXX XXX XXXXX X XXXXX X X X X XXXXX XXX XXXXX XXX XXXXX XXXXXXX X XXXXXXX XXXXXXX X XXX X  X X X X X X X X X X X X X X X X X X X  X XXX X XX X XXXXX X XXXXX X X XXXXX XXXXXXX X X XXXXXXX X X X XXXXX X X  X X X X X X X X X X X X X X X X X X X X X  XXX X X X X X X XXXXX X X XXXXX X X XXXXXX X XXXXX XXX XXX X XXXXXXXXX X  X X X X X X X X X X X X X X X X X X X X X X X X X  X XXX XXX X X XXX X XXX X X XXX X X X X XXXXX X X X XXX X XX X X X X X X  X X X X X X X X X X X X X X X X X X X X X  X X XXXXX XXXXX XXX XXX X X X X XXX X XXXXX X XXXXX XXXXXXXX XXX X XXXXX  X X X X X X X X X X X X X X X X X X  X XXXXX X X X XXXXXXX X X XXX XXX X XXXXXX X XXX X XXXXXXXXXX X X X X X X  X X X X X X X X X X X X X X X X X X X X X X  X X XXXXXXXXXXX X XXXXXXXXXXX X XXX XXXXX XXX X XXX XXXXX X X XXX X XXX X XXXXX XXXXXXX XXXXX XXXXX X  X X X X X X X X X X X X X X X X X X X X X X X X X X X X  X XXXXX X XXX XXXXXXX X X X X XXX XXXXX XXX X X X XXX X X XXXXX XXX X X X XXX XXX X X X X X XXX X X X  X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X  XXXXX XXXXX X X XXX X X X XXXXX XXX XXXXXXX XXX XXX X XXXXX XXXXX XXX XXXXX X XXXXXXX X X XXX X XXXXX  X X X X X X X X X X X X X X X X X X X X X X X X X X X X  X XXXXXXXXXXX X XXXXXXX XXXXX XXX XXX X XXXXX XXX X X XXXXX X X X X XXX X X X X X X XXXXXXX XXX X XXX  X X X X X X X X X X X X X X X X X X X X X X X X X X X  X XXX XXX X XXXXX XXXXXXX X XXX XXX X XXX X XXX XXX XXX XXXXX XXX X X X XXX X X X XXXXXXX X XXXXXXX X  X X X X X X X X X X X X X X X X X X X X X X X  XXX XXX XXX X X XXXXXXXXXXX X XXX XXX X XXX X XXX X X XXX X X XXXXXXXXXXX XXXXXXX X XXXXX XXXXXXXXX X  X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X  X XXX XXX XXXXXXX X X X X XXX X XXX XXX XXXXX X X X X X XXX XXX X X XXX XXX X XXXXX X XXXXX X X XXXXX  X X X X X X X X X X X X X X X X X X X X X X X X X X X X X  XXX XXXXX X XXXXX X X XXXXXXXXX XXX X XXXXXXXXXXX XXX X X XXXXXXX X X X X X X X XXXXX X XXXXX X XXX X  X X X X X X X X X X X X X X X X X X X X X X X X X X X X  X XXX X XXX X XXXXX X X XXXXX X X XXX X XXX X XXX X XXX XXX X X XXX X XXXXXXXXX X X X X XXX X XXXXX X  X X X X X X X X X X X X X X X X X X X X X X X X X X X X  X X XXXXXXX XXXXXXXXX XXX X XXXXX X XXX XXXXXXXXX X X X X XXX X X XXXXXXX X XXXXX X X XXXXX XXXXX X X  X X X X X X X X X X X X X X X X X X X X  X XXXXXXX X X X XXXXX X XXX XXXXX X XXXXXXXXXXX XXX X XXXXX XXX X X X X  X X X X X X X X X X X X X X X X X X  X X XXX X XXXXX XXX XXXXX XXXXX XXX X X X X X X XXX XXX XXXX X X XXXXX X  X X X X X X X X X X X X X X X X X X X X  XXXXX XXX XXX XXXXX X XXX X X XXX X XXX XXXXX XXX X XXXXXXX X X X XXXXX  X X X X X X X X X X X X X X X X X X X  X X XXX XX X XXX XXXXXXX X X X X XXX XX XXXXX X XXXXXXX XXXXXX X X XXXXX X  X X X X X X X X X X X X X X X X X X  X XXX X X XXXXXXX XXXXXXXXXXXXX X X XXXXXX X X X XXXXXXXXX X X X X X XXX  X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X  X X XXX X X X X X X X X X X X XXX X X XXX X XXXXX XXX X XXXXXXX XXXXXXX X X XXXXXXXXX XXXXXXX XXXXX X  X X X X X X X X X X X X X X X X X X X X X X X X X X X X  X XXX X X X XXXXXXXXXXX X X XXX XXX XXXXXXXXXXX XXXXXXXXX X X X XXX X X XXXXXXX XXXXX X XXX XXXXX X X  X X X X X X X X X X X X X X X X X X X X X X X X X X  XXXXXXX X XXXXXXX X XXXXXXXXX X XXXXX X XXX X XXX X X X X XXX X X X XXXXXXXXX X X X XXXXX X XXXXXXX X  X X X X X X X X X X X X X X X X X X X X X X X X X X X  X XXX XXXXXXX X X X X X X XXXXXXX X XXX X XXXXXXXXXXXXX XXX X XXX XXX XXX X X XXX X XXXXXXXXXXXXXXXXX  X X X X X X X X X X X  XXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX |  |

1. (10 marks) Using the INFO6044\_Midterm\_Helper program, generate an arena maze text file.

Note a few things about the arena maze:

* There’s a couple exits. You can ignore these.
* There are six (6) empty regions. Your 6 tanks start in the centre of these areas.

1. (100 points): There are 6 tanks that will battle to the death, in the “arena”.   
     
   **At this point, the output should be using the 3D application.**  
   Place your six (6) tank players approximately in the centre of the six (6) regions.  
   They can face in any of the four (4) axis aligned directions: up, down, left, or right.

Choose six (6) distinct colours (so you can tell them apart).

These tanks can only “turn” to face in four (4) directions. They can also only go “forward” in the direction they are facing. They move just like Pac Man does.

Initially, they all can move at the same speed.

When moving, you can either “jump” them to the next square, or smoothly move them from one square to another. Again, think about the different variations of Pac Man: some moved smoothly, and some were more “jumpy”.

They *can’t* go through walls, though.

The maze is really a set of squares 100x60 units in size, where each grid location either has a “wall” in it, or doesn’t have a wall. When moving, you simply check what’s in the “next” grid location and decide if the tank has to stop or not – if there’s a wall, it has to stop, otherwise it can move.

Five (5) of the tanks are AI controlled and the last one is player controlled, using the keyboard.

Set up the movement of *only* the player tank in this question.

Please use the WASD or arrow keys to control the movement of the player’s tank.

* If the key is pressed *and* the tank can go in that direction, then turn the tank in that direction and start moving the tank.
* As long as the key is pressed, the tank moves.
* If no keys are pressed, the tank stops *at the next grid/cell location* it’s moving towards.
* If the player presses a key that *isn’t* in the direction of travel, the tank should turn in that direction *at the next cell/grid location.*
* If the tank can’t move forward (there’s a wall in “front” of it), then it should stop (even if the key is pressed).

1. (100 points) Now set up the movement of the AI tanks.

This is done in the following manner:

* If the tank is stopped, pick a direction.
* If the tank can move in that direction, move the tank “forward” until it hits a wall.
* If the tank *can’t* move in that direction, have it wait for a moment (you decide how long, like a second or whatever), then pick another direction.
* The tanks are pretty stupid – like they could, in theory, keep picking “bad” directions until they can move. They can also go back and forth along a corridor.
* Note: They *don’t* choose another direction until they are stopped.
* So, they can’t just “oscillate” back and forth in a possible direction – because they need to move in a straight line until they hit a wall (I *suppose* if the corridor was only 2 squares in size, this *could* happen, but would only happen briefly, until they picked another direction). In other words, they can’t move one square forward, turn around, go one square back, then repeat that “forever” – you aren’t just picking a random direction per “tick/frame”, you are picking a direction which they move over several frames, until they can’t move in that direction any more.

Suggestion: Set up 3 “states” for the tank:

* Waiting: Where it’s picked a bad direction and is waiting for a second (or so)
* Moving: Where it’s moving in whatever “good” direction it’s going
* Blocked: Where it’s hit a wall, and we pick another direction.
  + From here, it can immediately go to Moving or Waiting, depending on if it’s turned towards a wall.

Set up a keyboard shortcut to start the tanks moving around.

If the tanks “bump into each other” (i.e. they can’t move to the next grid/cell because there’s a tank there), then a new direction should be picked. In other words, they should treat another tank like they would treat a wall that they can’t go through.

You can move the tanks by “stepping” them to the next “grid” location or by gradually moving them over time.

Note if they “step” to the next location, they have to stay still for enough time that you can see them.

1. (100) Get the tanks to start attacking each other.

Tanks can have one of two (2) types of weapons:

* A bullet/shell that “flies” forwards at 2x the speed of the tank. This goes in the direction the tank is facing and stops when it hits the first wall. Each tank can only shoot one bullet/shell at a time (i.e. while the bullet/shell is moving, it can’t shoot again). The tank can move while the bullet is being fired, though.   
  + You can show this by placing a sphere model in the location of the bullet/shell and use the same “moving” logic you used to move the tank (until it hits a wall and picks another direction)
* A LASER, which takes a second or so to “charge” (there’s a delay where the tank has to stop and start firing the weapon). Then it flies instantly forward.
  + You can show this using the sphere model, and using the same logic you used above, but in this case, draw a whole bunch of spheres in a straight line.   
    In other words, with the bullet/shell, you are only drawing the current location and only drawing one sphere, but with the LASER, you draw a whole bunch of spheres at once. You can do this by either having a bunch of spheres that you place in the scene, or you can call DrawObject() many times.
  + While the tank is “charging”, you could change its colour or add another object above it (like a sphere) indicating that it’s aiming and is starting to shoot, rather than just stopped.
  + Once the LASER has shot, it can start moving again. (or shooting again if there’s a tank there)

The tanks shoot when they “see” another tank “in front” of them. In other words, if there is a tank that’s along the same axis *in the direction* they are facing, they will shoot. The “catch” is that they will shoot even if the other tank is blocked by a wall.

1. (100) Alter the player’s tank to shoot as well.

Use two keys to choose between the bullet/shell weapon or the LASER weapon.

Use the same logic for the player’s tank as for the AI players in question 4.

1. (100) Handle the tanks shots hitting the other tanks.

If a weapon “hits” another tank, do the following:

* Take 10% “damage” from the tank. Tanks start with 100% damage, and “die” when at zero (0).
* Indicate the health in some way: a suggestion would be that the colour goes from a bright colour to black/dark grey as they get closer to 0 health.
* **BONUS #1 (20):** Use a separate “health indicator” which is a cube just above the tank  
  + This cube moves with the tank. It starts out bright green, then turns more and more red as they health gets to zero. Once at zero, the cube turns dark grey.
* **BONUS #2 (20):** Also start this cube as a long bar (so not a cube, I guess...) that gets smaller as the health gets lower and lower. Once it’s at zero, it should be a little “stump” of a bar.
* They stay on the field once dead. They just don’t move, so block the way.

Use the same logic for the player’s tank as for the AI players in question 4.

|  |
| --- |
| **At this point, you should have a complete application, and if you’ve combined question 1-6 that’s OK**  **(Actually, it’s preferred)**  **Now copy the entire application and do question 7.** |

1. (200 marks) Now, once a tank has made one “kill”, it transforms into a “super tank”.

Super Tanks are just like regular tanks except that:

* They have **150% health**.
* Their **LASER** does **15% damage** rather than 10%.
* They have a “tracking missile”. This is different than the regular shell in that:
  + When it hits a wall, can make up to **TWO** turns (in any direction it *can* turn *including back the way it came if that’s the only way possible*) then continues until the next wall.
  + Note that this can use the same logic as the tank, in that if it hits a wall, it picks a possible direction, but the difference is that it *can’t* go back the way it came.
  + After the missile makes two turns, it either hits another tank or hits a wall and explodes.
  + If it reaches a “dead end” *before* the two turns, then it will also explode against the wall.  
    In other words, if it hits a wall and hasn’t turned twice, but the only valid direction is back the way it came, then it would explode *even if it hasn’t made two turns yet.*
  + The idea is that the missile can only “turn” but can’t “back track” (turn 180 degrees).
* There’s a 33% chance that the Super Tank will fire a “tracking missile” bullet/shell.   
  So each of the three weapons have the same chance of firing.
* The player can *never* be a Super Tank. Sorry.

**That’s it.**