- What problem(s) do you want to investigate/solve, or what algorithm(s) do you want to implement?

Since I have a game development background, I've been looking into a project that combines my game dev interest with the GPU background from this course. But I'd like to take on a project that uses the GPU more purely – so no ray-tracing or random shader business, I'd like to have actual GPU kernels in the source.

The main idea project that interests me the most is the infinite zoom Mandelbrot, but I want to incorporate some kind of "interactive" layer on top of it, more than just a procedural zoom or viewer tool. The current idea I have is some kind of Mandelbrot walker, where you can traverse/walk about the "edge" of Mandelbrot boundaries by moving left or right, and being able to zoom in or out on boundaries by moving down or up.

- Which programming model(s) and language(s) do you want to use? Why?

As a base, I would like to use Godot, since it's an engine I'm pretty familiar with. But Godot by itself does not support the ability to write low-level GPU code, which isn't sufficient for me. Fortunately, Godot is susceptible to all kinds of crazy extensions (including language ones). So, there's a bit of investigation I'll need to do to set up a working development environment.

1. For me to use Godot at all, I'll need to pick a secondary programming language with it. This has to be a language with an existing package that provides language bindings to Godot, and one that can use low-level GPU code.
    a. Python – supports language bindings for Godot 3, and still in-development bindings for Godot 4. Even if I could run PyTorch alongside it (the goal, probably) I think Python is too high-level for this, and I don't trust it for this problem, especially still in-development
    b. C++ – has native bindings with Godot, so technically completely usable for this project. But C++ is extra-infuriating for me, so I'd rather not.
    c. Friendship – not a real language, so definitely can't use
    d. ✅Rust – Has somewhat stable language bindings for Godot 4? Like, not feature-complete, but enough to be able to sufficiently offload computations onto Rust code. So I think Rust is the most practical option here.
2. AS A FALLBACK: if I can't use Godot for whatever reason (like if the GPU libraries I use for the language bindings prevent the project from compiling, which my gut suspects is likely), I can pursue other frameworks as well – if I stick with Rust for this project, Bevy is a tried and true Rust package for game development that I suspect could do the job as well.
3. So, since Rust is likely what I'll be using for my low-level language, it comes down to knowing if Rust is viable for building GPU code. These packages I'll have to investigate throughout the project as well for viability and compatibility:

a. Rust-gpu – Runs SPIR-V shader code within Rust. Still in development, and this feels more dedicated to shaders than something that can actually run gpu kernels, despite it being in the name. Whatever
b. Ocl – OpenCL bindings for Rust. This looks really scary, but seems like it should actually work quite well.
c. Rust-CUDA – No way its CUDA!!!! Unfortunately, it also seems still seems early in development, but I get the impression it may be usable for this project, though it seems no longer maintained.

Full disclaimer: I've never actually used Rust before. But I have been studying it, and I do have friends that could help provide some consultation along the way, so I believe it to be possible for me to follow through with.

- Why is a GPU a good fit for this problem/algorithm?

Mandelbrot is already computationally expensive, and running it anywhere close to real-time for game applications will naturally require the GPU.

- How will you use the GPU(s)? Which particular parts of the code do you want to accelerate?

Primarily for rendering Mandelbrot frames out to a texture dynamically. Perhaps some simple edge detection for determining the Mandelbrot boundaries within a frame.

- How will you evaluate your work? E.g., speedup achieved over serial, amount of data analyzed, comparisons between two programming models?

Well, if I can get it working at all, that would be a success for me – the intersection between the moving parts of Godot, a language extension, and a GPU library is a dangerous one indeed, and where most of the challenge in this project lies. It's also practical for me too – writing GPU code is one thing, fitting it into a practical development environment (especially with Godot no less, which is bordering on being one of the best software development tools in general IMO) is another thing.