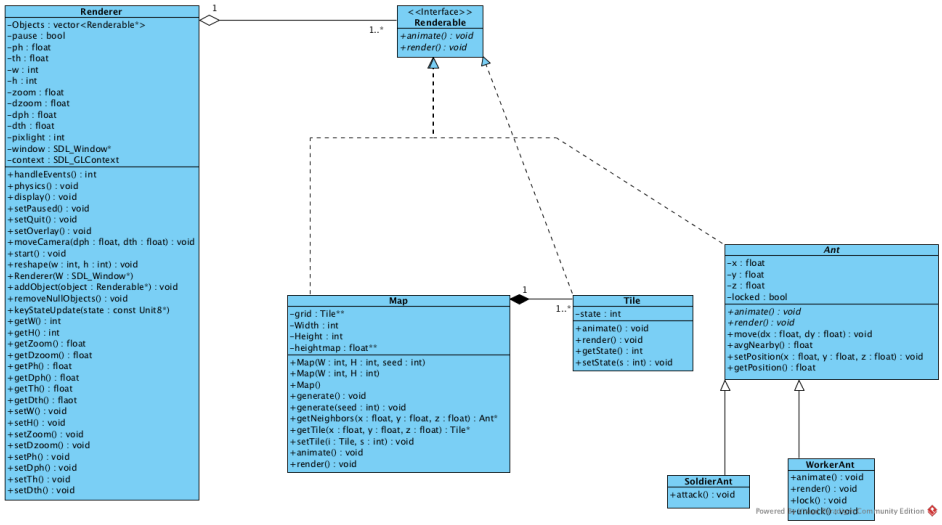
1. (a) Team Name: Terrarium

   (b) Members: Jordan Dick, Nelson Mitchell

   (c) Vision: A robust simulation of a broad range of biological-like multi-agent systems

   (d) Project Description: Terrarium will be a fully animated functional 3D model of self-assembly in ant colonies, with hopes of one day being able to simulate other, more complicated behaviors.

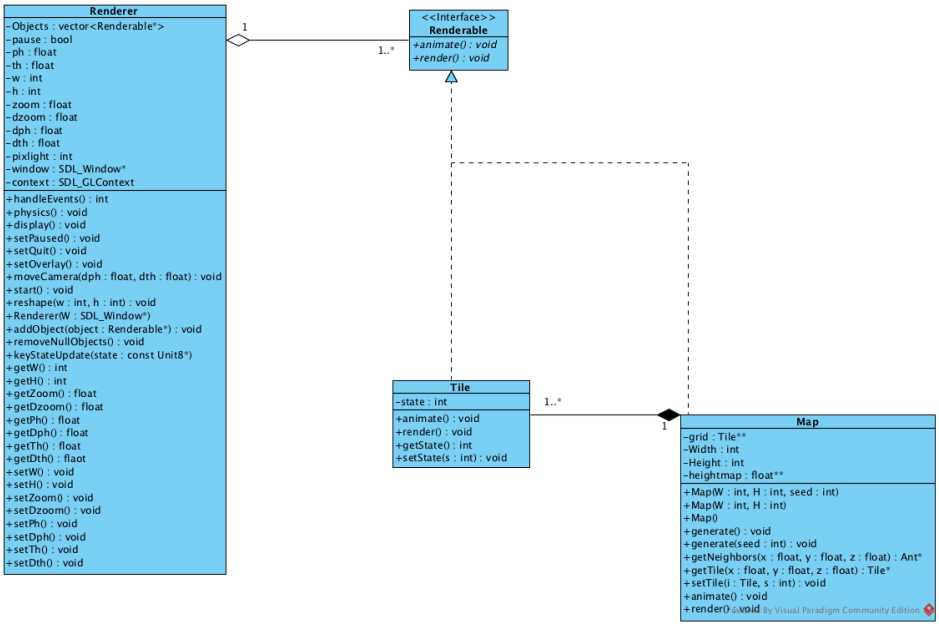
2. Updated Class Diagram:



With consideration of the following feedback:

- We noticed our project inadvertently implements the "Observer" software design pattern, with `Renderer` as the subject.
- "Fixed" attribute types, changed from C++'s `type name` descriptor to Java's `name : type`.
- Added operation return types and parameter types
- Added getters and setters where necessary, note that `Ant.move()` is a setter for `Ant.x`, `Ant.y`, and `Ant.z`. As is `Renderer.moveCamera()` for `Renderer.ph` and `Renderer.th`.
- Our project does not implement data storage in its current version.

3. Completed Class Diagram:



4. Summary: As of this progress report we have implemented the entire Map class except for the Ant-related methods, `getTile()`, `setTile()`, and `getNeighbors()`. To implement randomly generated maps, we've added a `heightmap` attribute that is set by the member function `generate()`, either given a specific `seed` or using CPU timestamps. We refactored `Renderer`, adding getters and setters for all the camera and window attributes, as well as added the necessary window attributes for SDL to run. Also, we added methods `addObject()` and `removeNullObjects()` to manage the vector of `Renderable` objects.

5. Breakdown of Work:

   - Nelson: UML updates, and this document
   - Jordan: Implemented `Map` and `Renderer`

6. Estimate of Remaining Effort:

   - Implement the entirety of `Ant`
   - Integrate `getTile`, `setTile`, `getNeighbors` in `Map`
   - Implement basic state variables like `quit`, `overlay`, etc.

7. Next Iteration: We'll have implemented working `Ant`s, as well as dynamic lighting/shading!