1. Project Team Name and #. List of team members. Vision. Project description.

    Terrarium - 04. Nelson Mitchell, Jordan Dick.

    Our vision is to create a capable graphical simulation of multi-agent self-assembly. Simulated ants flock and construct themselves into towers on a 3D map.

2. List the features that were implemented (table with ID and title).

    We never made a table:

    - Ants move around map randomly
    - Ants flock when close to each other
    - Ants self-assemble into towers
    - Dynamic lighting and shading

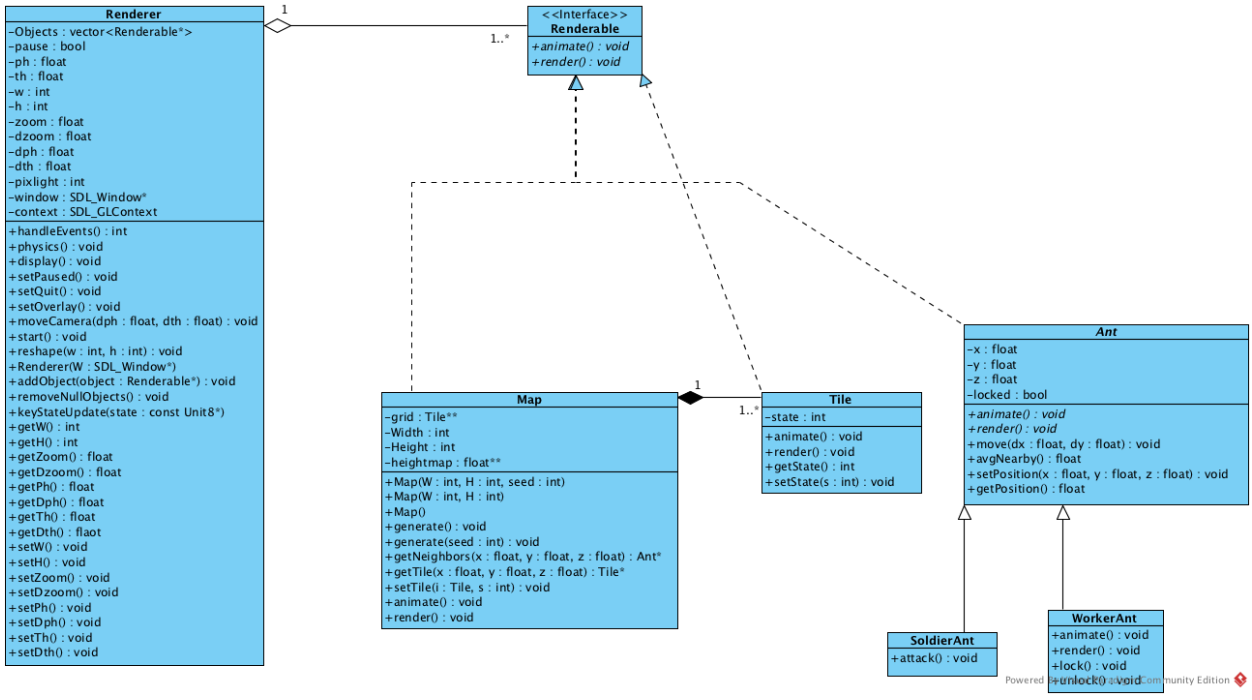
3. List the features were not implemented from Part 2 (table with ID and title).

    - Ants can be soldier or worker with different goals
    - Ants can navigate randomly-generated environment with only their local information

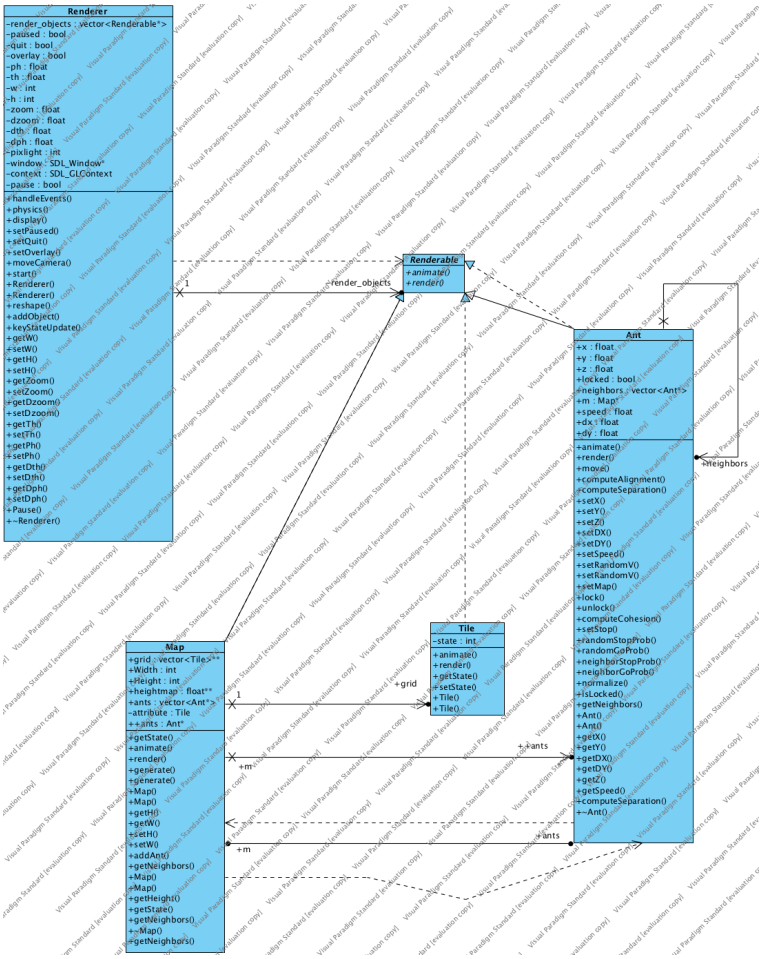
4. Show your Part 2 class diagram and your final class diagram.

What changed? Why? If it did not change much, then discuss how doing the design up front helped in the development.

Part 2 Diagram:



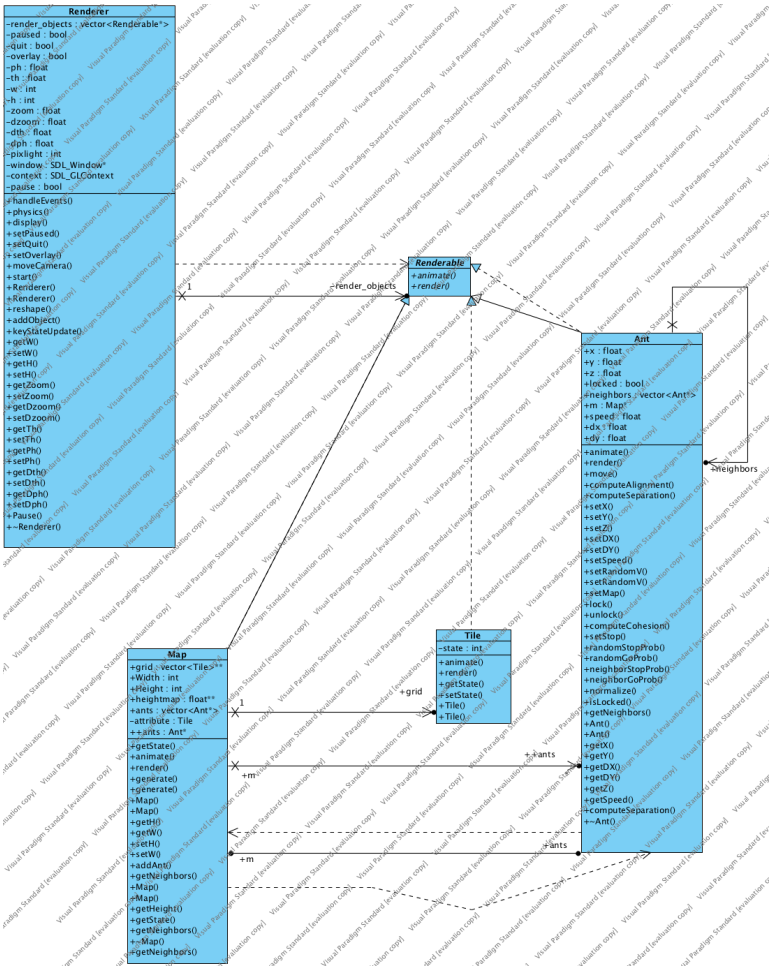
Final Class Diagram: (mind the watermark)




Not much changed in our simple program from its conception to its completion, we had to add some spaghetti coupling and cohesion to smooth some parts out, like the map holding a vector of all ants, and ants having a vector of their neighbors. We did appreciate the process of making a diagram before beginning work, as this helped us greatly in collaborating and planning our code sprints.

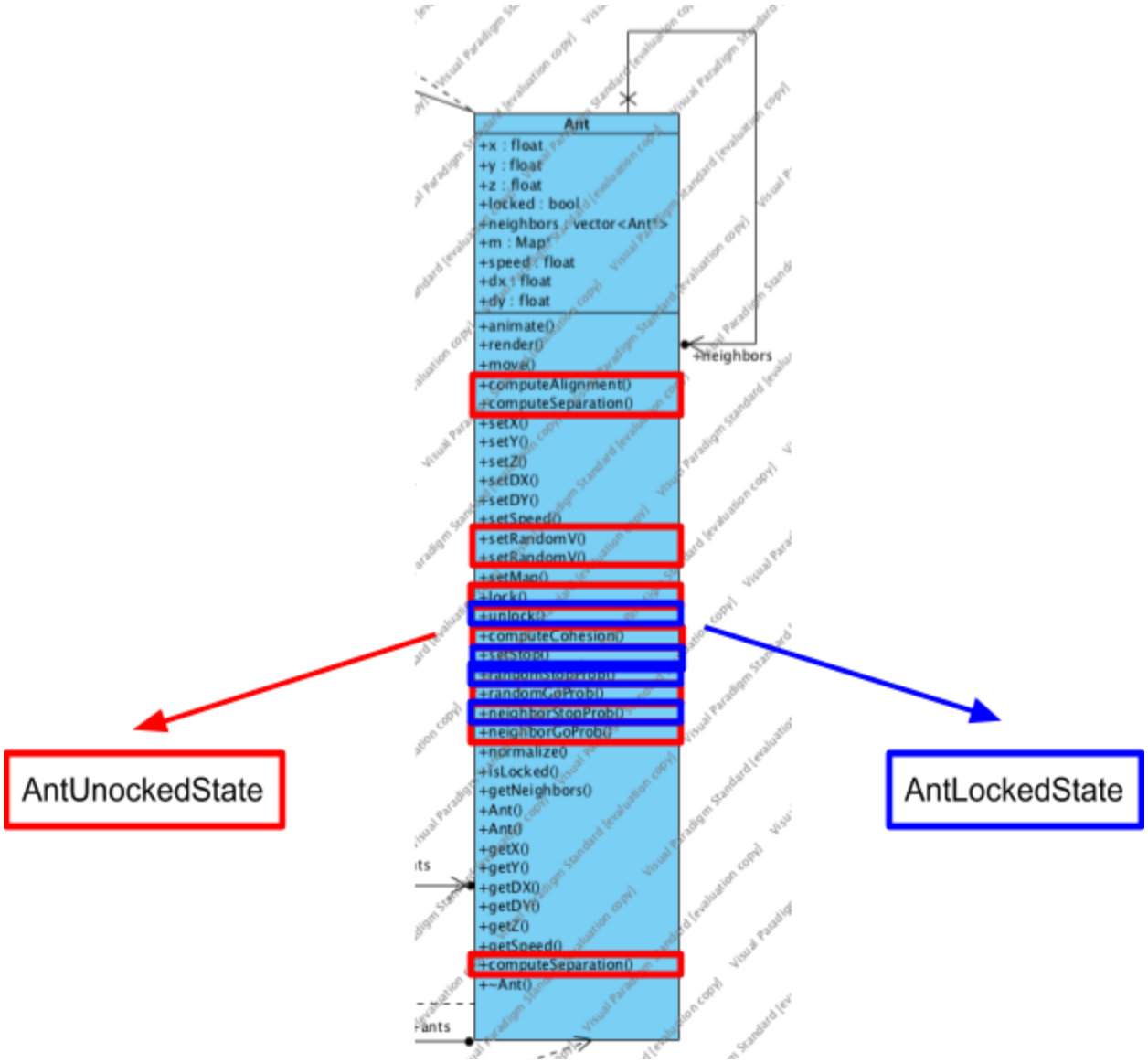
5. Show the classes from your class diagram that implement each design pattern (each design pattern as a separate image in the .PDF).

Observer design pattern:




Our entire program employs the observer pattern, as do many games/simulators. Renderer acts as the subject, or the physics engine, and updates the map and ants with their current state in the simulation.

State (-Like) Behavior:




In hindsight, our ant class could have benefitted from being split into two ant state classes, one for when it is locked in place and one for when it is not. This future refactor will de-blob the ant class, and make it easier for the user to distinguish the two on render.

6. What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?

Our most important takeaway from this class is that careful planning and lots of time spent in the design stages of the project make the implementation much simpler to code. We also realized during our time working on our project that frequently returning to and updating the design of our project would have helped greatly in avoiding some of its blobbiness and spaghettiness.