Jacob Biederman

The first architecture choice I made was to continue using the existing steering format for the new steering behaviors. This made it easier to code and made the code base more consistent. However, I could have added a new getSteeringAcc function that would bypass a little fluff and just return a Vector2D. I’m not sure if this would make a noticeable speed or memory difference. The second was to make the FlockingSteering class accessible from the SteeringComponent. This allowed the Data Loader to directly access it. While this adds some tight coupling, the Data Loader exists solely for FlockingSteering right now. I also chose to update each FlockingSteerings variables rather than update the data loaders variables because that seemed simpler to implement. It would probably be faster to update the data loaders variables and have each FlockingSteering check if it needs to update, but this would be more complicated. In hind sight, having tacked on a method to update the Data Loader variables to match the FlockingSteering ones, I definitely should have had Flocking Steering update from the Data Loader. The final choice I made was to set the increment amounts in the Data Loader because I didn't want to hard code them into FlockingSteering and that seemed like a reasonable place to put them. Right now they’re hard coded in, but since they are in the Data Loader could easily be loaded from a data file.

The main thing I would improve given more time is the performance. Each boid gets a vector of all the boids, converts it to locations, and iterates through it each frame. This is slow. I think centralizing this vector generation, and possibly calculation, would provide the biggest performance boost. One way would be to iterate through the vector in the FlockingSteering checking for each of the component steering radii, and then pass the relevant units to the correct steering component.