* “Any general tips?”
  + Validate your behaviors with small swarms before validating on large swarms
  + Don’t worry about making the behaviors perfect – parameter tuning is a long process that won’t really aid understanding of the material
  + Every time there’s some variant, just make a whole new folder. It will get messy but it should make it easier to return to previous implementations

Section 1

1.1:

1.2:

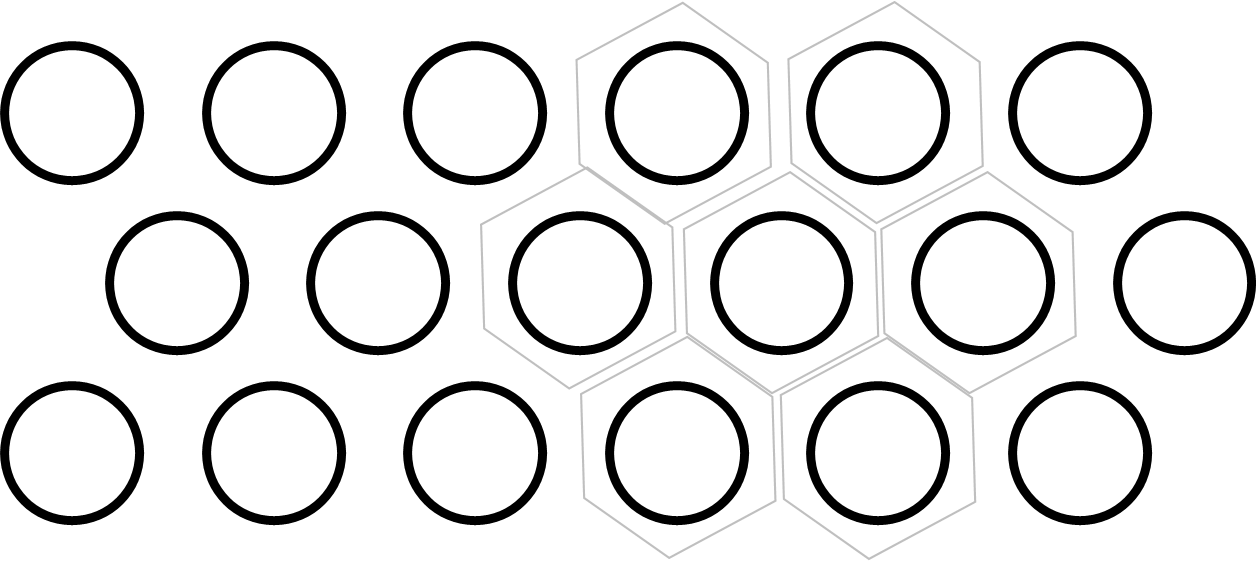
* “ What do you mean by parameter configurations?”
  + Any “magic value” you put into the behavior. Spin speed, orbit distance, forward motion speed are all good options. Just show different pictures of the paths traveled and qualitatively discuss
* “Should my orbit be smooth?”
  + It is unlikely that your orbit will be perfect, nor do I expect it to be. The goal here is to demonstrate that you are able to make the robots do a simple behavior based on inputs and outputs
* “How do I do this?”
  + A good, simple implementation would be to have the planet robot logic command a constant forward speed (self.out\_drive = 0.5, for example). And turn left or right based on whether the distance between the two robots is greater than or less than the desired orbit radius (robot logic can look at self.comm\_in(1).distance for this information)
  + You will have two roles for the robots: “sun” and “planet” assign a number that corresponds to these roles and base the robot behaviors on that (e.g. sun = 1, planet = 2). Refer to the example code for details on how to implement this.
* “Where should the robots start?”
  + Wherever is most convenient for you to place the robots
* “Where in the code do I place the robots?”
  + Robot starting locations should be defined in the function placeRobots.m

Section 2:

2.1:

* “How do I place the robots?”
  + Robot starting locations should be defined in the function placeRobots.m. You can either do this directly by defining 100 or so robots, or do this in a loop (recommended)

2.2:

* “What is a hexagonal grid?”
  + Essentially, the robots should be placed as if they were hexagons:  
    
* “How do I place the robots?”
  + Robot starting locations should be defined in the function placeRobots.m. You can either do this directly by defining 100 or so robots, or do this in a loop (recommended)

2.3:

* “How do I do this?”
  + See the end of lecture 6 for hints.

2.4:

Section 3:

* “What are ‘shells’?”
  + Different people call them different things (layers, shells, rings, etc), but this refers to the tendency of the robots to form concentric rings about the center, segregated by color. So when I’m asking for different number of shells, you can

3.1:

* “But you implemented the algorithm and gave it to us. What’s to discuss?”
  + Explain to me in text how the algorithm is working. Show me that you understand the code that was provided

3.2:

* “How / where should I implement a cost function?”
  + The easiest place to implement a cost function would be in the main loop. Some examples might be: average distance from the light source for robots in shells 1/2/3. I’m not going to be picky with this: as long as you show some qualitative evidence of the robots doing what you want

3.3:

* How do I make a video?
  + One option would be to use the functions I provided in assignment 1, but other options could be anything from using OBS to just taking a video on your phone

Section 4:

4.1:

4.2:

* “What sort of starting configurations?”
  + See the end of Lecture 6 for hints

4.3:

* “What would be a good cost function?
  + Average number of robots within communication range, Deviation from average alignment, etc might be good options. I’m not going to be picky with this. I’m basically going to look for some sort of quantitative assessment that you can show

4.4:

* How do I make a video?
  + One option would be to use the functions I provided in assignment 1, but other options could be anything from using OBS to just taking a video on your phone