Orbital Ecology Simulator

Executive Summary

Problem statement

Can Agent-Based Simulations be used to improve satellite maneuvering in mega-constellations to avoid propagating space junk in low-earth orbit?

Purpose

In May 2019, Space Exploration Technologies Corporation (SpaceX) deployed the first 60 satellites for the first-ever satellite mega-constellation, which will grow to 42,000 assets in orbit by 2027. The SpaceX mega-constellation, dubbed Starlink, is meant to provide a viable alternative to ground-based communications infrastructure, which is limited by terrestrial geography and national borders from providing truly unfettered access to services like the world-wide-web and international telecommunications.

While Starlink promises to improve worldwide access to information, it comes with significant drawbacks including a high potential to create space debris. Space debris are uncontrolled objects (including meteoroids, inoperable satellites, and discarded fuel tanks from rocket launches). Experts estimate that hundreds of thousands of pieces of “space junk” have already been created from collisions between satellites and other objects in orbit. However, the effects of deploying a satellite mega-constellation into low earth orbit with autonomous control into such an environment are not well understood.

To explore this problem, I built a NetLogo model called “Orbital Ecology” to simulate the interactions between a mini “mega-constellation” and space debris. The satellites have an optional capability to maneuver in attempts to avoid debris, and the debris have the ability to multiply by colliding with individual satellites.

Summary of Results

One interesting observation from this model is that a small amount of initial debris quickly proliferates in the orbital environment once an initial collision occurs. I have observed the debris count go from 300 (the maximum starting amount) up to well over 1000. Another interesting result is that limited satellite maneuvering can drastically reduce the amount of space debris proliferation, while no satellite maneuvering in such a large constellation creates many times more space debris than the model began with. An effect however is that the average lifespan of the satellites is much reduced as they use up more fuel for the maneuvers which decreases the amount of time a satellite can keep its altitude. This suggests a trade-off between keeping the orbital space clear of debris and maximizing the average lifespan of the satellites.

General Description

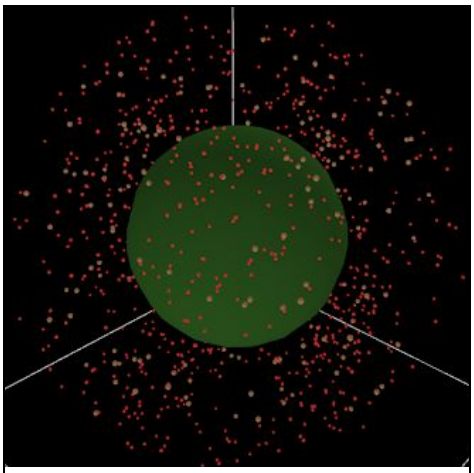
The orbital ecology model that I put together generates a planet with between 48 and 672 satellites, and from zero to 300 debris objects. The satellites are generated with half inclined at 53° from the north pole, and half inclined at 53° from the south pole. Each half also has a slightly different orbital radius. They can be evenly split from between 8 and 24 orbital planes and between 6 and 28 satellites per plane. The total number of satellites in the model is equal to the number of orbital planes times the number of satellites per plane. The satellite spacing can also be varied so that the satellites are closely lined up along their arcs, or loosely spaced along their arcs. As the satellites move along their orbits, they appear to form something that resembles the “eye of a hurricane” at the north and south poles. SpaceX uses this configuration to maximize signal coverage over the portions of the earth that are populated above a certain threshold (and reduces the number of satellites that would be needed for truly global coverage). The satellites are capable of station-keeping, or burning fuel to maintain their orbit. Once they run out of fuel, the satellite orbits will decay over time and once they get too close to the planet, the satellites will die and be removed from the world.

The number of orbital debris objects in this model can vary from a count of 0 to 300. The debris are dispersed randomly across the 3D world and have a (limited) range of sizes for visual effect. The orbital debris each follow a circular orbit around the planet which degrades over time. When a chunk of debris passes too close to a satellite, it has a chance of collision which produces between 3 and 30 additional chunks of debris. Debris created from collisions appear as red dots to show the difference from original debris (brown dots).

System Interactions and Rules

The planet regulates the orbital behavior of both other turtle-sets, querying each turtle’s distance from the origin and enforcing the circular travel. This is used in place of 3D gravitational interactions, which proved difficult to translate from a 2D example. There are a number of variable parameters built into the interface tab of the model. Varying the orbital planes, satellites per plane and spacing of the satellites have already been discussed. As has the amount of initial debris and the satellite maneuvering.

Additional parameters that I built in include changing the orbital radius, switching on/off the fuel (to observe continuous satellite maneuvering without



Debris cloud after a full simulation run (all satellites have deorbited). Red dots were created in satellite/debris collisions.

increased fuel consumption eventually killing them off), and satellite networking capabilities. The linking capabilities were to begin investigating a distributed computing mode for collision avoidance, but this part of the model is undeveloped at the time of writing this.

Model Complexity

I needed to make a lot of simplifying assumptions when developing this model. A lot of time was spent attempting to build in orbital mechanics based on a 2D example (N-Bodies in the NetLogo library). I abandoned this effort for the sake of time as the example was not translating well into 3D because of the equations used, but I was already able to simulate movement along a circular arc. The result is that all objects in this model follow a circular orbital path around the planet regardless of distance and orientation.

The next most complex element of the model was figuring out how to disperse the satellites around the planet in a way that approximates how the Starlink constellation will cover the earth. Generating two sets of satellites from the northern and southern limits of the orbits was sufficient for the specific parameters of 24 planes, 28 satellites per plane, and a spacing of 10, with the orbital radius set to 13.

Distances and other scales have not been changed from the NetLogo standard. So a distance of 1 in this model is the distance from the center of one patch to the center of its Von Neumann neighbor. This simplified model development, however I am interested in making these unit conversions as the model progresses.

The model's importance or relevance

The main relevance of the model at its current point of development is more as a visual aid to demonstrate the complexity of putting large numbers of satellites into orbit and the results of collisions with orbital debris. Ultimately I would like for it to serve as an exploratory tool for the system dynamics of satellite constellations in an orbital debris cloud. With that in mind, I must stress that without further development, testing, verification and validation activities, the results should not be taken as representative of the real world system the model is based on.

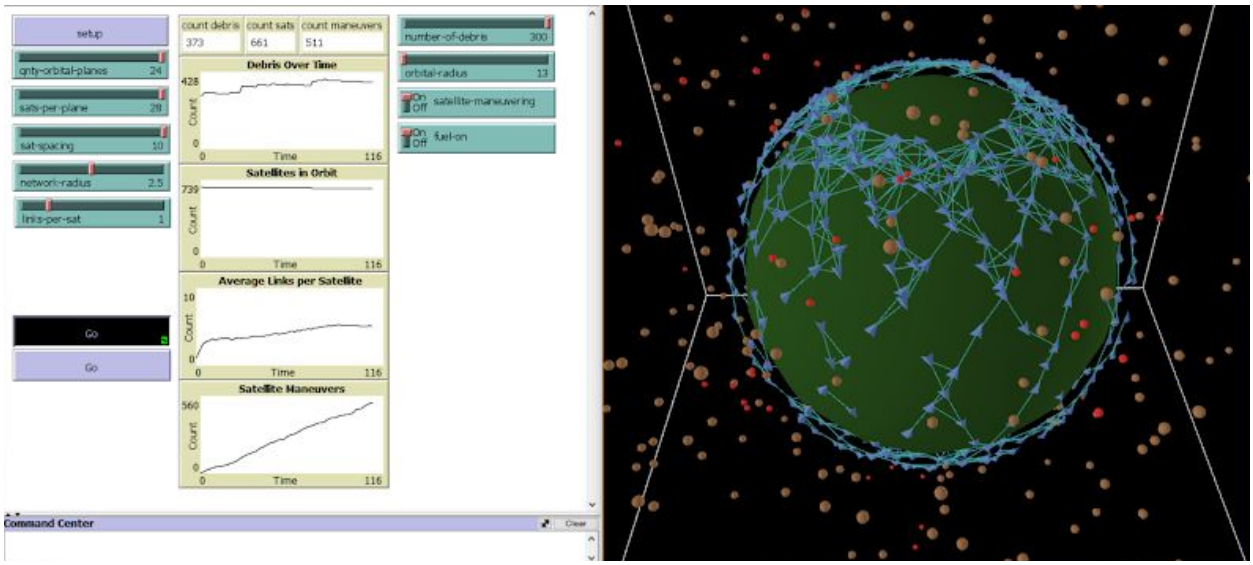
With that said, there is a growing interest in the space operations and mission planning community to have agent-based models of large-scale satellite networks to perform the same types of explorations that I am writing about in this report. While this model is not yet a good representation of a real-world system, it has stretched my understanding of what an agent-based model requires for implementation and similarly, I learned a lot about what considerations are needed for mission planning and how the two fields are related.

The Model's Aggregate Behavior

The model exhibits a combination of predictable and less predictable behavior. Satellite movement was highly predictable so long as they were only required to follow their arc around the planet. When the maneuvering capability was added, a rule as simple as changing the arc slightly to increase the orbital radius to incoming debris resulted in some satellites departing from their expected orbits entirely and in some cases taking strange paths out into space. This demonstrated that satellite maneuvering could decrease debris propagation, but also that additional consideration is needed for the input parameters and sensitivity of how the agents react.

It was also surprising to see that the debris proliferated much faster than I expected. Each collision results in a small cascade of debris creation, and with each collision the rate of debris propagation increases substantially. I expected that I would see some cases where the debris entirely destroyed the satellite constellation, however, I have not yet seen that happen in any of my runs (although it could be that insufficient time has elapsed for this to be the case).

Some additional rules that were considered for model implementation were inspired by the flocking rules in the NetLogo Flocking model. These would be considered with the satellite networking parameters to allow for distributed computing and better coordination between satellites for maneuvering. I believe this would be useful to prevent satellites from ranging too far afield when the maneuvering response is triggered repeatedly. This remains to be seen as I had insufficient time to explore this option during the academic quarter.



Screenshot of the model with all parameters set. Lines show networked satellites. Red dots are debris created from satellite/debris collisions.

Building the Model

Data Sources and Collection Methods

The model was inspired by the Starlink mega-constellation which was deployed earlier this year. Another source of inspiration has been journal articles published in recent years on the topic of using satellite constellations as distributed computing platforms in space. It seemed to me early on that it would make sense to build a model capable of exploring that feature, but as I am very new to network science I decided to focus first on orbital debris and to make networking a secondary feature.

To learn how to build the model, I did some initial research on satellite constellation concepts and mission planning. I made this the topic of my literature review, hoping to find other models that may have been built by researchers that I could learn from. While I did find papers that discussed using agent-based models to simulate satellite constellations, they held a discussion about the work and its results, but they did not provide any information detailing what steps were taken to develop the model and there was no pseudocode. The papers did discuss equations for some types of satellite constellation configurations, but to make use of them I would need to first build a physics-based model, which proved to be too challenging to accomplish during the timeframe of an academic quarter.

I was able to find several useful models in the NetLogo library which I could use for inspiration as I built my model from scratch. These were the N-Bodies, Spherical Path Example, and Flocking 3D models.

Preliminary Versions

My first model attempt was focused on recreating the gravitational interactions for the N-Bodies 2D NetLogo model in the 3D version of the program. My hope was to take an existing physical model, adapt it to my scenario, and then I could focus on coding for the satellite maneuvers. This proved to be only partially achievable because my computer found that computing the model in 3 dimensions required too much power from the CPU.

Eventually I had to scrap this idea and I shifted focus to the Spherical Path Example which was already in the NetLogo 3D library. This model already showed agents moving along a spherical path, and since the satellite configuration of interest is in low-earth orbit, the satellites will be traveling in a circular orbit. I determined that sacrificing gravitational interaction was an acceptable trade-off to simplify the model to show the movements of the agents I wished to portray. I also determined that it would be sufficient not to initially scale the model directly to the earth system it emulates. I would be satisfied with the model simply being a good representation of the complexity and dynamics of the real-world system (this was to save valuable time).

Using the Spherical Path code, I inserted the spherical “planet” agent in the center and began experimenting with ways to adjust the starting place and direction of travel for the

existing agents (renamed as satellites, or “sats”). I was very quickly able to set the direction of travel around the sphere. The real challenge in orienting the satellites was to figure out how to spread them over the surface of the planet such that while traveling along their paths, they would at least appear to provide significant coverage of the planets latitudes between 53° north and south.

My solution was to produce successive generations of satellites at 53° (north and south) and to run each forward during the setup phase until they were sufficiently spread out. At this point, I just needed to iterate through the parameters (the number of orbital planes, the number of satellites per plane, and the spacing between the satellites) until I achieved the desired coverage.

Once I accomplished this, it was a simple matter to create debris class of agents. I took again the code from Spherical Path and this time kept the stochastic distribution of the agents, but then I added stochasticity to the radius as well. It took more trial and error to adjust the code controlling the debris movement to allow for a variable radius. This was resolved by having each agent calculate its distance from the origin and pass that value to the orbit function. I added this innovation to the satellites and set them to slightly different orbits as well (more to visually resemble true orbits as the model satellites are not set to collide with each other, though this may be an added feature later on).

I also added a “Fall” procedure which incrementally decreases each agent’s orbit, and the “Maneuver” procedure to allow satellites to evade debris and reduce collisions. I came across an issue where the agents would pass through the planet and emerge on the other side. I solved this by having the planet ask every other agent to check its distance from the origin, and ask those agents to die when they got too close. I then added the “Fuel” variable to the satellites which would decrement to zero, at which point the fall procedure would go into effect for the satellites. Maneuvering would cause additional decrements of fuel.

The final step was to give the debris the ability to proliferate. Due to the Fall procedure, the debris were always falling in toward the satellites if they were at a higher altitude. To enable proliferation. I added the “Proliferate” procedure to have each piece of debris scan its immediate radius at every tick and ask any sats in that radius to hatch debris, which would then spray out within a 60°cone from the direction of the parent satellite. I set these newly generated debris to be colored red for easy visibility and distinction from the original debris.

At this point, I had reached the end of the academic quarter and ceased major development to produce end-of-project deliverables.

Model Summary

Environment: The model is set in the 3-dimensional space surrounding a simulated planet and its satellites, which consist of “natural” debris, an artificial satellite constellation, and debris created from debris collisions with satellites.

Agents: The agents in this model are laid out below in the following format:

- Agent
    - Variables
        - Rules

- Planet - a single large green spherical breed of agent which dominates the center of the model world.
    - Mass - 598
        - This variable was originally set during the attempt to implement gravitational interactions.
    - Size - sqrt mass
        - The mass variable was left because I was happy with the size as it was set here.

- Satellites - small blue arrow-shaped agents that are generated in discernible spherical patterns around the planet. Satellites move in a formation and have the ability to make small movements to avoid collisions with debris.
    - Mass - 1
        - Similar to the mass variable for Planet.
    - Size - sqrt mass
    - Local-network - an agentset of the nearest satellites if networking is enabled.
    - Fuel - 200
        - This variable is decremented with every tick. When it reaches zero, the satellite will begin calling the “Fall” procedure until it dies in close proximity to Planet.
    - Altitude - distancexyz 0 0 0
        - Each satellite’s orbital radius. This is adjusted with every tick in case of maneuvers and falling.

- Debris - small brown spherical agents that are dispersed randomly about the world.
    - Mass - between .25 and 1
    - Size - sqrt mass
    - Debris-rad - random between 13 and 30
        - Each debris object’s orbital radius.

- Sat-links - the agentset used for visualizing local area satellite networks (not fully developed).
    - Network-radius - between 0 and 5
    - Links-per-sat - the number of connections each satellite is allowed to establish.

Global Variables:

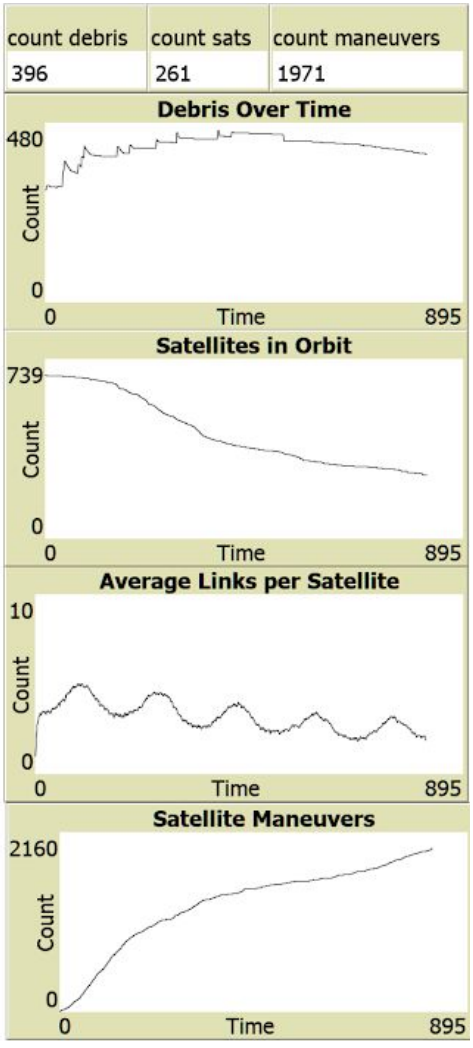
- Interface
    - Qnty-orbital-planes (slider) - set the number of paths around the planet
    - Sats-per-plane (slider) - set the number of satellites along each path
    - Sat-spacing (slider) - set the distance between each satellite
    - Number-of-debris (slider) - sets the number of starting orbital debris
    - Orbital-radius (slider) - sets the radius of the satellite constellation
    - Satellite-maneuvering (switch) - turn on/off satellite maneuvering capability
    - Fuel-on (switch) - turn on/off effects of fuel consumption for satellites
- Code
    - Count-maneuvers - plotted on the interface tab

Rules and Procedures: The rules and procedures are outlined as pseudo-code in the Appendix. The procedures are set up to dictate motion along a circular path, with rules that enforce orbital decay (the satellites and debris must fall into the planet, or burn fuel to keep its radius).

Observables: The observables for the model, aside from the 3D view, are the plots set up on the interface tab (depicted on right).

Time Units and Horizon: The time units in this model are unmodified from the standard NetLogo “ticks”, and runs until all satellites have deorbited (typically at 2800 ticks).

Spatial Boundaries: The spatial boundaries of the model extend 25 patches away from the origin in 3-dimensional space. This was important for the kind of model. An N-body simulation may be modeled in 2D, but to show objects moving along multiple planes and at various altitudes, a 3D model was necessary.



Key Simplifying Assumptions/Limitations:

As discussed in other sections of this paper, I needed to make several assumptions and impose certain limitations on the model for it to work. The key assumptions are thus:

- Circular orbits will be sufficient
- Parameters to decrease the radius incrementally can replace gravity
- 0-300 pieces of debris can sufficiently emulate an orbital debris field
- A range of less than 672 satellites can sufficiently emulate a satellite mega-constellation
- The model will still be valid as an orbital ecology demonstrator without scaling distances and time

These simplifying assumptions enabled me to build a model during the course of the academic quarter that at least appears to behave in (most) of the ways expected for the model. In keeping with these simplifications, I can continue to develop the satellite networking and maneuvering schema. The model can be further validated by converting and scaling spatial and temporal elements to the real earth/satellite system, however, I believe that is not necessary to understand the fundamental dynamics of the agent interactions for demonstration purposes.

Model Testing

Extreme Cases

My model is highly sensitive to extreme cases due to its relative size and complexity. For instance, I quickly found that I can choose to model debris collisions, or I can choose to model dynamic satellite linking, but that to do them both at the same time would cause NetLogo to crash. Indeed, an early modification that I had made to the debris proliferation procedure took many attempted runs before I figured out why it was crashing the model within seconds every time I clicked “Go”. I have restricted the starting number of debris to no more than 300 because I have found that with so many constantly moving satellites, and with all of the debris constantly falling inward, collisions are inevitable and the debris population easily climbs to over 1000 during the model run. I also set limits on the number of satellite links, the length of those links, and wrote into the “Go” sequence a rule that link agents must die each turn and only be re-established if satellites were within a certain controllable range. This is the reason for the **network-radius** and **links-per-sat** sliders on the interface tab. Because I employed this strategy, I can confidently say that the model has been verified (or at least checked for verification) at each step in the development process.

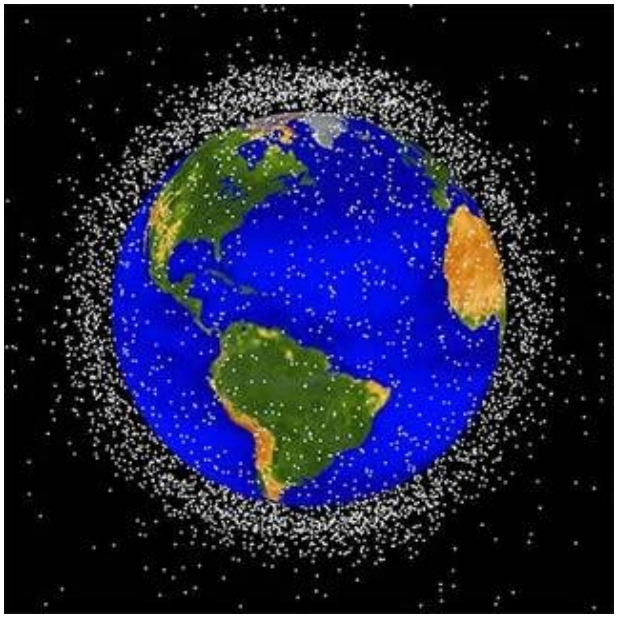
Sensitivity Analysis

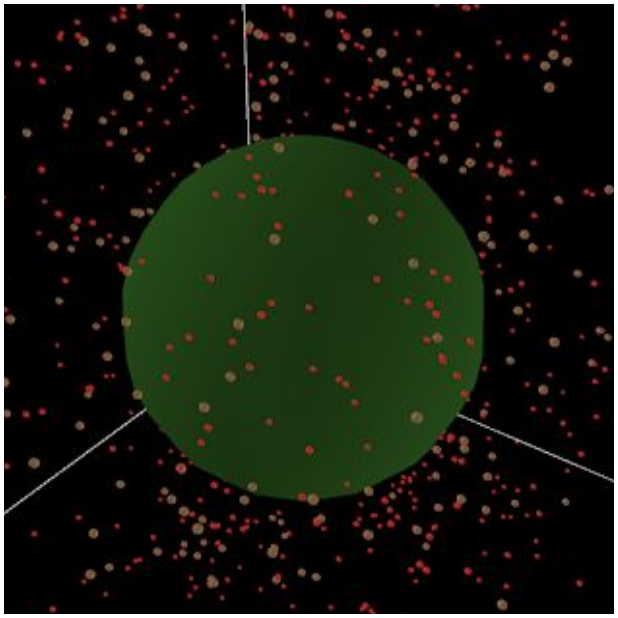
In my model the sensitivity analysis was really an extension of the extreme cases, since the same methods were used for both instances. Once I got my model to a point where it was not crashing my computer during the majority of runs, I began tweaking values incrementally until I was able to observe dynamics that seemed reasonably close to my expectations of real satellites in orbit. Every slider on the interface tab originated from this method, and I set the maximum values for the sliders based on what I had determined to be the maximum value or quantity for that parameter that the model could run without forcing a NetLogo crash or significant reduction in speed.

Comparison to Real-World Behavior

I believe that this model provides a decent representation of the low earth orbital ecology as it is today. The model satellites have been designed to follow orbital paths very similar to those of the Phase 1 Starlink orbits. And similarly the debris field that I have modeled resembles the debris field that currently exists around the earth. While the debris proliferation of the model is substantially less than in the real world (up to 30 per collision in the model vs several thousand in a real collision), the smallness of the NetLogo world has necessitated that trade-off and yet the resemblance of behaviors still seems valid.

An important but still unvalidated characteristic of the model is the satellite maneuvering procedure. This is something that I would need to conduct pointed research and further





A NASA image depicting the debris cloud around low-earth orbit	A debris cloud produced in the Netlogo simulation.

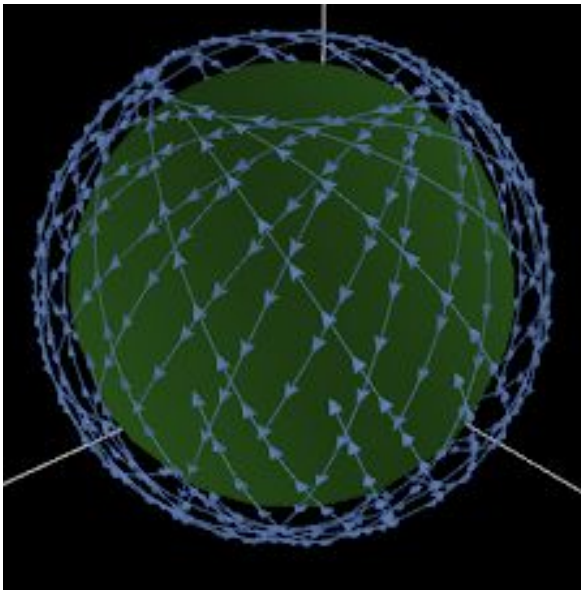
development on before I could confidently say that it is validated. Similarly, as the satellite networking feature has not been fully developed, I can not say that it matches real world satellite networking behavior, and this aspect of the model is neither verified nor validated.

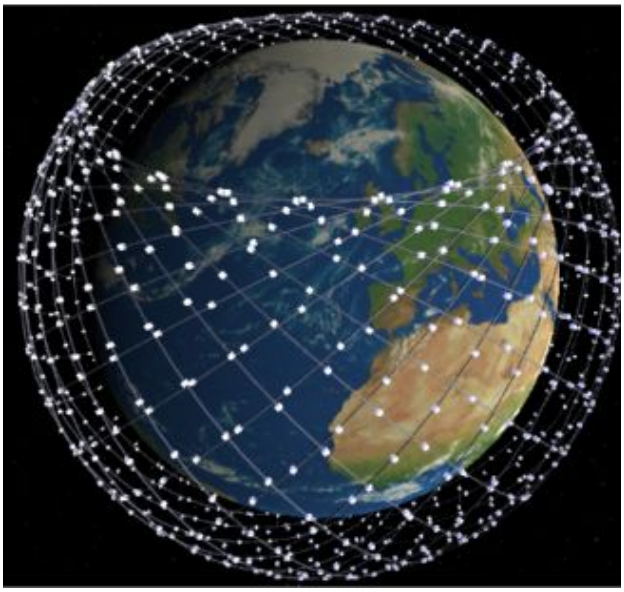
Comparison to other Models

While I have been able to find literature about other agent-based satellite constellation models, I have not been able to get access to the models themselves. Without access, I have not been able to qualitatively compare my model to any others. My model building has turned out to be very exploratory and pioneering, despite my early assumptions that I would easily find other models to compare it to.

I have been able to compare my model on a limited basis to the FreeFlyer satellite mission planning software tool, which I have a student version of the software. This program has a demonstration for satellite constellations including the SpaceX Starlink and a couple of others. Until I get to the point where I have converted the time and distance scales to real world time and distance, I can't do a full comparison, however the movement of the satellites along their circular paths match closely the paths of the real Starlink satellites. Once these conversions have been made, FreeFlyer will be a valuable tool for continued validation of the physical properties of the model. This is the long term goal: to have a physically accurate model with the capability to simulate autonomous satellite maneuvering and networking capabilities.

Model Applicability/Transfer





NetLogo simulation as each satellite traces its orbit.	A depiction of the completed Starlink Phase 1 Satellite formation.

This project began with the question: can I create an Agent-Based Model that could be used to simulate autonomous satellite maneuvering for debris avoidance? Following that, my hope was that the model could be used to explore different tactics that satellites could use to avoid debris. I was not able to accomplish this fully, but I was able to at least demonstrate that a very basic debris avoidance scheme reduces the odds of a satellite-debris collision (within the parameters of my model). This suggests that improvements to the model may be made to improve satellite maneuvering beyond what has already been achieved. I intend to continue to work on this in the hopes that it could be worth submitting for a conference presentation with the American Institute of Aeronautics and Astronautics.

Results

While the model still needs to undergo further development and testing, the results produced in its current phase are promising. Satellites in the current model exhibit much lower odds of survival to the end of a simulation run when they have very basic maneuvering capabilities than they do otherwise, however they also significantly reduce the amount of debris propagation than there is without maneuvering. Multiple runs using Behaviorspace show that with maneuvering turned on, debris propagation accounts for an average of 40-50% growth of the debris cloud, whereas without maneuvering the debris cloud has grown by as much as 350% (from a 300 debris count to above 1000). This is still not ideal, and doesn't match the real world, as real satellites being emulated have much more sophisticated maneuvering capabilities, longer lifespans, and a much

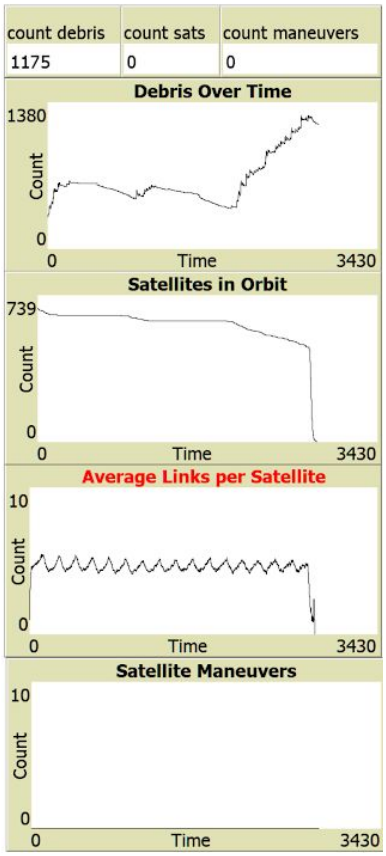
larger environment within which their overall odds of collisions with debris are much less than the model demonstrates.

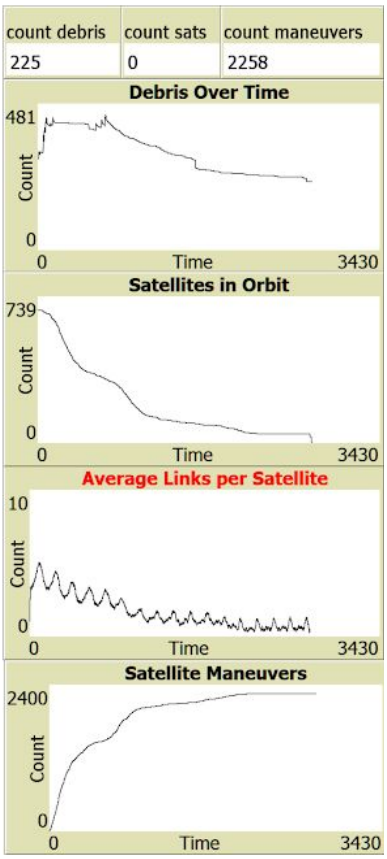
Model Limitations & Future Enhancements

As mentioned in other sections of this paper, the Orbital Ecology model's main limitation is that it does not currently emulate real physics, only an approximation of the motion that the satellites perform as they travel circular paths around a sphere. Similarly, the model lacks true units of distance and time, which will ultimately be necessary to implement real physics. While I had initially intended to include these features, it became apparent that the model would take much longer for me to build at my current knowledge level, and I instead focused on using the capabilities that I knew that I could include.

At its current state, the model is useful for demonstrations of the near-earth orbital ecology, including the nature of the debris cloud and the shape and complexity of a satellite constellation. However, the model in its current form should not be used for any meaningful analysis of how real satellites may perform.

Future versions of the Orbital Ecology model will include time and distance conversions, which will enable me to emulate real physics. This will be different from my prior attempt to emulate physics because I will be building the physical model from the ground up, rather than copy/pasting code from a 2D physical model which I've since learned uses different equations than are necessary for a 3D model.





Debris propagation without satellite maneuvering	Debris propagation with satellite maneuvering

Conclusions

The System Studied

Space is hard. That has been a recurring theme during my three years as a project manager for the Portland State Aerospace Society as it has worked on OreSat (the Oregon Satellite project) and the base 11 Space Challenge (a U.S. and Canada collegiate space race). While learning about satellite mission planning, this has been increasingly so. To plan for space missions, there are a lot of physical parameters to be studied, as well as the capabilities of the assets (satellites, space probes, robots, people) that will be deployed, and how they will respond in a variety of scenarios. Because it is very difficult to account for every possible situation, ABS is a very useful tool for learning about what scenarios are most common, and which strategies may be most likely to be successful when such situations actually occur.

My Model

Even though I cannot claim to have completed a comprehensive model that effectively models satellites and a debris field in orbit, I feel confident that the model is on the right track for someone who is only just beginning to learn how agent-based simulations can be used for mission planning. Real world systems have many levels of complexity, and through my model I was able to find ways to simplify several difficult aspects of the real world system and demonstrate debris field propagation with and without satellite maneuvering.

NetLogo

NetLogo has been an interesting program to build the model in. In many respects it is very simple to use, but in others it required me to rethink my approach that I would have used based on my (limited) prior experience in programming for Python, R, Matlab or C. One thing I would like to experiment with going forward is learning how to use the Python integration as I continue to build this model. I believe that NetLogo will be a much better tool if I am able to combine it with a python script for optimization of certain parameters that could then be passed into the NetLogo model. This is currently beyond my capacity, but later this year I will be learning python, and I have some understanding of linear programming in the context of spreadsheets.

ABS as a Research Tool

ABS is a valuable research tool in the social sciences and economics. There has also been a growing interest in the field of robotics and space mission planning to integrate ABS into mission planning for space exploration. While there have been some papers written on the subject dating back into the 1990s, I have not been able to find any publicly available ABS models with which I could compare my own. This left me with a steeper curve to get past when building my own model. With that said, the model that I have built has shown me the value of ABS in satellite mission planning and design. ABS

is a powerful tool for me to add to my portfolio of capabilities as I pursue a career in systems engineering for space systems.

Next Steps

My next steps in moving forward with this model are first to begin unit conversions so that I can emulate real physics in the model. As I bring my model further into alignment with the physics of the equation-based models used in mission planning (such as FreeFlyer), I will use those models to further validate my own model.

Once I am confident that the simulated physics are reasonably close to the real world, I will work to improve the satellite maneuvering, relate the types of maneuvers to fuel consumption, and also to build into the model the satellite networking capability. While my current model can provide a representation of these satellite features, I cannot meaningfully develop them much further without real-world parameters.

After this is done, I intend to explore an evolutionary model where I can have multiple strategies played out among many satellites and select for the most successful strategies used. This will be the ultimate goal of my model's development. It will require me to continue to build on the knowledge that I have begun to learn during this course and for subsequent courses later. If I am successful, then I will have developed an open-source tool that could be used by other university students and researchers as they pursue their own explorations of ways to improve satellite mission planning, especially in the contexts of satellite mega constellations and orbital debris fields.

Appendices

See below for the Orbital Ecology model pseudo-code

```
;; Setup planet, satellites and orbital debris
To setup
        Clear-all variables and parameters
        Create Planet
        Run procedure setup-sats
        Run procedure setup-debris, pass through number-of-debris
        Reset ticks
End

To setup-sats
        ;; Establish orbital elements
        Set number of planes (from interface)
        Set latitude and inclination 53°

        ;; Create satellites at inclination and move forward a set distance
        Run while loop (decrement down from the number of planes)
                Run procedure make-sats, pass through sats-per-plane (from interface),
                latitude and inclination
                        Ask sats to run procedure position-sats
        Repeat while loop with latitude set to -53° and inclination 143° (53° south)
        Set fuel to 200
End

To make-sats [number, latitude, inclination]
        ;; Create satellites and move out to orbit
        Use primitive procedure create-ordered-sats number / 2
        Tilt-up latitude
        Move forward orbital-radius

        ;; Set heading tangent to Planet and along orbital plane
        Tilt-down 90°
        Left turn inclination

        ;; Set physical properties
        Set satellite color, mass, size
```

```
        Set altitude distance from the origin
End

To position-sats
        Set c satellite-spacing
        While (c greater than 0)
                Run procedure Orbit, pass-through altitude
End

To setup-debris [number-of-debris]
        ;; Create and move debris to orbit
        Use primitive procedure create-debris, number-of-debris
                Tilt-up (random 180°) - 90°
                Set debris-rad 13 + (random 17)
                Move forward debris-rad

                ;; Set heading tangent to Planet along new orbital plane
                Tilt down 90°
                Turn right random 360°

                ;; Set physical properties
                Set color
                Set mass .25 + (random .75)
                Set size sqrt mass
End

To go
        Ask turtles [set orbital-decay .001]

        ;; Take stock of altitude, kill turtles that enter Planet's "atmosphere"
        Ask Planet
                Ask satellites [set altitude distance from origin]
                Ask debris [set debris-rad distance from origin]
                Ask turtles in-radius 12.5 [die]
        If no satellites or no debris remain, stop.

        ;; Run satellite procedures
        Ask satellites
                If altitude = 0 (prevent errors from division by zero)
```

```
            ;; Obstacle avoidance and stationkeeping
            If any debris in radius .5
                  If fuel = 0, run procedure fall-sats, pass through orbital-decay
                  Run procedure maneuver pass through altitude
                        Decrement fuel by .5 until 0
                  Else, run procedure orbit, pass through altitude
                        Decrement fuel by .1 until 0

      ;; Reset and re-establish network satellites
      If network-radius > 0 and link-per-sats > 0
            Ask sat-links [if link-length > network-radius, die]]
            Run procedure make-network
            Ask local-network [create-sat-link-to-myself]

      ;; Debris movement, destruction and proliferation
      Ask debris
            If debris-rad = 0 [die]
            Run procedure orbit, pass through debris-rad
            Run procedure fall-debris, pass through orbital-decay
            If satellites in-radius .1
                  Run procedure proliferate-debris
      Tick
End

;; Create local satellite network
To make-network
      Set local-network up-to-n-of links-per-sat of other satellites in-radius
      network-radius
End

;; Perform orbital travel
To orbit [radius]
      Set angle to (.25 * 180/(π * radius))
      Tilt-down angle / 2
      Move forward .25
      Tilt-down angle / 2
End
```

```
;; Parameters for orbital decay
To fall-debris [orbital-decay]
        Tilt-down 90°
        Move forward orbital-decay
        Tilt-up 90°
        Set debris-rad debris-rad - orbital-decay
End
To fall-sats [orbital-decay]
        Tilt-down 90°
        Move forward orbital-decay
        Tilt-up 90°
        Set altitude altitude - orbital-decay
End

;; Satellite maneuvering for obstacle avoidance
To maneuver [radius]
        Tilt-up 90°
        Move forward 2 * orbital-decay
        Tilt-down 90°
        Set altitude altitude + 2 * orbital-decay
End

;; Proliferate debris
To proliferate-debris
        Ask debris
                ;; Set odds of hit vs near/miss
                Ask sats in-radius (.1 + random .25)
                        ;; Proliferate up to 30 debris objects
                        Hatch-debris 3 + 27
                                Set color red
                                ;; half the average mass of starting debris
                                Set mass .1 + random .5
                                Set size sqrt mass
                                Set debris-rad distancexyz 0 0 0
                                ;; Set debris to fly out in a 60° cone
                                Right turn -30° + (random 60°)
                                Tilt-up 30° + (random 60°)
                        Die
End
```