1. **Introduction**

Many concepts can be reduced to a very simple model: one composed of mobile agents that navigate and work within a space. Though it is not obvious at first, even more abstract concepts can be modeled in this way. The limitations of this system are all in how it’s implemented. Several factors of which include the autonomy and homogeneity of the agents as well as how the space of the model is defined. This report will cover the development process of one such implementation as carried out by the author under the guidance of Dr. Torben Amtoft. In describing this effort, the first areas that will be covered are the inspiration, research, and planning of the project. These will then be followed by the development process of the base library as well as a summary of its current state. Finally, this report will include the future plans for the project.

1. **Designing Process**
2. *Inspiration*

The first inspiration for this project came from human interactions with society. When one looks at the world of humans they will find a population of rather homogeneous agents navigating a space of discrete locations. Differences between these agents mostly arise from the interactions they perform in or with these locations. These actions are not just mediate by the agents, but through the environment as well. Looking beyond the world of humans, one will see that this is not the only example of such a system.

There are environments beyond our physical space. Abstract concepts, which we use to describe our universe, live within a world of their own. Take, for example, mathematics which we can separate into values and operations. In this example, the operations would make up the space that the values reside in. The values would then navigate from operation to operation; interacting with each other and their environment. While thinking about this in a philosophical sense can be stimulating, there is also value in attempting to emulate these ideas.

1. *Research*

Very little research was done in preparation for this project. This was an intentional decision in order to minimize external influence with the hope of creating something new. However, this isn’t to say that this project was developed in complete isolation.

Though it’s no longer utilized in the current state of the project, one of the first implementations of space building was via diffusion-limited aggregation (DLA). Space building, in the context of this project, is the process in which an environment is created. DLA was originally chosen to fulfill this role because of the natural dendritic structures that it generated. The process of DLA was first described by physicists Thomas Witten and Leonard Sander in *Diffusion-Limited Aggregation: A Kinetic Critical Phenomenon*, and their work acted as a stepping stone for the first implementation of this project.

1. *Design Choices*

Before any actual coding could be done, a significant amount of planning had to take place. What should the library do? What should the library leave the user to do? What should the library allow? These were just some of the questions that needed an answer before development started.

The first tenet of this project is that the user should have “the freedom to do or not to do”. That is, the library should provide base functionality for most parts of creating and running a model, but the user can also choose to override these functions with their own algorithms. This is most obvious in two functions: the graph building and the location defining functions. The first function builds tree-like graphs, and the second gives each node in the graph a function-defining object. This leaves the user free to work on other parts of the model. Still, if these functions do not meet the user’s needs, then the user can also develop their own functions for handling these parts of the modeling process. This tenet is most important when considering the parts of the model that the library cannot define itself.

The second tenet of this project concerns what the user must define for themselves. When it comes to these aspects, the largest number of options should be made available in the simplest format. The minimum of what a user must define to make a working model with this library includes an agent and three node functions. These four pieces alone offer a wide variety of outcomes in the user’s model thanks to generalization and abstraction. However, this freedom does not lack direction, and the user is hopefully aware of what is necessary to make the model function as desired. Once these decisions were made, and a few algorithms fleshed out, the actual coding could begin.

1. **Development Process**
2. *Stage Play Metaphor*

The first iteration of the library was a rough sketch of the idea without any real use cases to guiding what functionality would be necessary. A key aspect of this version of the library was the three life cycles it took to make a functioning model.

The first cycle generated a non-functional, rigging graph which acted like a kind of structural skeleton. The next cycle replaced all of the rigging nodes with functional, stage nodes which could hold props and actors. The final cycle setup the stage with props, known as “set pieces”, which gave each node a function. Once this was complete the model could run and would become populated with “actors”. This version also saw the beginnings of those design tenets outlined earlier.

At this stage the library had functions for generating graphs, defining the functions of nodes, and running the simulations. It also heavily featured the use of generic types. The graph-generation function at this time was, as discussed earlier, an adaptation of the DLA algorithm. The action-defining function simply generated appropriate set pieces at random. To see all of this in use, the next step was to actually make a model.

1. *Polynomial Expression Demonstration*

The first model was fairly simple. The actors are polynomial expressions. These expressions navigated the stage by changing the state of intersections. The expressions also change routes by becoming morphed when a dead end is reached. This model works, but it highlighted some problems with the original design.

1. *Library Remodeling*

In revisiting the original design for the library, a few unnecessary and limiting features were removed or altered. First and foremost, the DLA algorithm was slow and costly. It became apparent that similar results could be achieved by building up the tree in layers without relying on the random walks of particles. Changing how the graph was produced then lead to rigging graphs becoming obsolete. As a result, rigging graphs were completely removed, and a functional graph would be built from the very beginning.

The next major change was the removal of the stage play metaphor. The set pieces had taken on transportation-based names (i.e. junction, terminal, and port), and they no longer fit the theme. Additionally, the actors were now agents, and the graph structure was no longer a stage.

Finally, the use of generic types had greatly bogged down the readability of code and limited what agents could do in their environment. Communications were limited to fit within the types that were specified, and all types had to be listed for every piece (regardless of whether or not that piece used those types). Replacing the generic types with the superclass of object great reduced the amount of typing necessary. With these and some other minor changes, it was time to once again make a model.

1. *System of Equations Solver*

The model developed using this new version had a goal; solving systems of linear equations. In developing this model, it became apparent that there are many ways to attempt solving this problem and they each have their own set of challenges. The solution that was settled on uses the matrix theory of linear algebra. The environment is composed of the columns of the system matrix. The agents are then the rows of the solution matrix. These rows are randomly generated and tweaked towards the correct combination by being tested against each column one at a time. This method utilizes the ability to make custom graphs and takes a ring shape with the start also being the end. Creating this model showed a few features that needed more tweaking in the library.

1. *Current Library*

Something that is important to the implementation of the model environment is that certain nodes have different functionality. This is why there are three set pieces (with the ability for the user to create more). The ports, junctions, and terminals all have different base functions which the user then expands. What those base functions should be was not immediately obvious, but they have now been refined.

The port is responsible for adding and removing agents from the graph. How this is done – where the agents come from when they enter, and where they go when they leave – is dependent on the user’s implementation, but it is all functionally the same to the graph itself.

The junction controls the flow of agents through the internals of the graph. This is done using the numerical state of the graph, but how that number is translated into an exit is up to the user.

Finally, the terminal acts as a means of input and output to the graph in a way that will change how the system functions. Just as with any of the other set pieces, the agent may be modified, the agent may modify the set piece, or both could occur. What is nice about this method is that every node is an enclosed system that can act in parallel.

The .NET framework offers a simple and robust library for the parallelizing of code. The method that was chosen for this library is to parallelize the loops that update each node. Being able to update nodes in parallel offers a significant speedup when working with larger graphs and larger populations of agents. This roughly summarizes the current state of the library, but development has not finished.

1. **Future Plans**
2. *Research and Refinement*

As mentioned earlier, little research has been done on projects of a similar nature. As the library has now reached a point where the base features are mostly complete, it would be ideal to go back and see what features have already been done. Hopefully, some of these features will already have better, more optimized, implementations that can be incorporated into this library. Some features have not been implemented because they do require research to find a functional solution.

1. *Future Features*

One of the features that require research is graph visualization. This is a two-part problem. The first is something known as a graph layout. Laying out the nodes of a graph in a 2-D plane so that they will not be visually confusing turns out to be quite an involved problem. The next part is actually rendering the nodes in an application. Depending on the implementation, rendering could be rather resource or time inefficient. When one most likely wants whole-graph operations to be carried out multiple times a second (and visuals to keep pace), then time is a rather important factor to rendering.

Another feature that is important is how graphs are generated and actions are defined. Right now, only one kind of graph can be automatically generated: the dendritic structures related to the DLA algorithm. These structures are natural, but sometimes one doesn’t want natural. Other pre-made generation functions should be available to the user such as the ability to generate grids or cyclical structures. As the structures of graphs become more important so too does the placement of set pieces. Something that has been a goal from the beginning is the contextual building of environments. Set pieces should be distributed in any many of quantity and density as the user sees fit, and the contents of certain set pieces should be aware of the contents of other set pieces so that the system as a whole can behave in a more “smart” manner. All in all, it does seem that this project is nearing a state where it might be considered “done”, but development may continue as new ideas come forth.

1. **Conclusions**

Considering what has been accomplished, this project seems to be a success. If nothing else, something was learned in the making of this library and its demonstrations. However, there may be more than just an experience. There is a tool to make many kinds of models that others might find useful. Since the applications of this tool include anything from education to research, what others can create may be the best result looking forward. Regardless of the actual use, it is here.

1. **References**

Sander, L. M., & Witten, T. A., Jr. (2000). Diffusion-limited aggregation: A kinetic critical phenomenon? Contemporary Physics, 41(4), 203-218. doi:10.1080/001075100409698