**Modeling Agents in Domains**

1. **Introduction**

A variety of concepts can all be modeled with the same, simple system; one composed of mobile agents that navigate and work within a domain. It is possible that the only limitations to what can be modeled with this system comes from how the system is implemented. Some factors that cause these limitations are the autonomy and homogeneity of the agents and how the domains are represented. This report focuses on a dynamic linked library for creating these kinds of models as developed by the author while under Dr. Torben Amtoft. The planning, coding, and testing of the tools necessary for this endeavor are outlined below.

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

The inspiration for this project came from observing human interactions with society. When one looks at the world of humans they will find a population of rather homogeneous agents navigating a domain 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. Look beyond the world of humans, and one will see that there are other examples of this same kind of system.

First it must be realized that there are environments beyond our physical domain. 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 domain that the values reside in. The values would then navigate from operation to operation; interacting with each other and their environment. Even though thinking about this in a philosophical sense can be stimulating, there is also value in attempting to emulate these ideas. The first step, however, is to layout blueprints to guide development.

1. *Research*

Though the first step of planning is often to research, very little research was done in preparation for this project. Avoiding this stage of planning was an intentional decision made in order to minimize external influence. It was hoped that this would lead to the creation of something new. Regardless, it is impossible to create anything in complete isolation, and at least some parts of the project were influenced by previous works.

Though it’s no longer utilized in the current state of the project, one of the first implementations of graph building used diffusion-limited aggregation (DLA). DLA was originally chosen because of the natural dendritic structures that it generates. 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*

Though the big picture of the project was clear, some finer details were not. 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 be required to implement as little as possible while still having control of as much as possible. 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 action-defining functions. On one hand, this leaves the user free to work on other parts of the model. On the other hand, 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 plays an important role with the next.

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 for the user’s model thanks to generalization and abstraction. However, this freedom does not lack direction, and the user should be 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. *Theater Metaphor*

The first iteration of the library was a rough sketch of the idea. Only a few use cases were in mind during its development. 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 as the 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 nodes their functionality in the model. Once this was complete the model could run; becoming populated with actors. This version did not completely adhere to the tenets described early, but it was instead intended as a prototype.

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. From here, models could start to be developed.

1. *Polynomial Expression Demonstration*

The first model created wass fairly simple. The actors were 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 was reached. This model worked, but it highlighted some problems with the original design.

1. *Library Remodeling*

In revisiting the original design of the library, 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.

A minor change was to move away from the theater 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 became 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 develop models.

1. *System of Equations Solver*

The model developed using this new version of the library had a real-world application; solving systems of linear equations. In developing this model, it became apparent that there are many ways that this problem could be solved; each of which each having their own set of challenges. The solution that was settled on used the matrix theory of linear algebra. The domain was composed of the columns of the system matrix. The agents were the rows of the solution matrix. These rows were randomly generated and tweaked towards the correct combination by being tested against each column one at a time. This method utilized the ability to make custom graphs and took on a ring shape. Creating this model showed a few features of the library that needed to be changed.

1. *Current Library*

Something that is important to the implementation of the model domain 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 at the time, 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 from the perspective of the domain.

The junction controls the flow of agents through the internals of the graph. This functionality is meant to be accomplished utilizing the numerical state of the graph, but how that number translates to 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 set piece may modify the agent, the agent may modify the set piece, or both could occur. Regardless of how this or any of the other types of set pieces are implemented, they all work in isolation, and thus can be made to run 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. This offers a significant speedup when working with larger graphs and larger populations of agents. Still, there is more that can be done.

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

As mentioned earlier, little research has been done on projects of a similar nature. Since the library is now at a point where the base features are mostly complete, now may be the time to look at work already exists in the literature. Most likely, there will already be more optimized solutions, or they may be features that this library lacks. For instance, 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 part 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 is a rather 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 wants whole-graph operations to be carried out multiple times a second and visuals to keep pace, time is a rather important factor in rendering.

Another feature for future work is how graphs are generated. Right now, only one style 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.

A feature that has been planned for from the beginning is the contextual building of environments. The domain that these agents inhabit is important as it guides the development of the system. The contents of nodes should be defined with locality and context in consideration. These together may make some models behave “smarter”. With this in mind, development of this project continues.

1. **Conclusions**

Considering what has been accomplished, this project seems to be a success. At minimum, something was learned in the making of this library and its demonstrations. But the results are more than that minumum. There is now a tool, which others may find useful, to make models of many different concepts. Since the applications of this tool includes anything from education to research, the biggest accomplishment may still be in what others will create.

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