Animal: An Agent-Based Model of Circulation logic for Dynamo

CHRISTIAN SJOBERG
University of North Carolina Charlotte

ALIREZA KARDUNI
University of North Carolina Charlotte

CHRISTOPHER BEORKREM
University of North Carolina Charlotte

JEFFERSON ELLINGER
University of North Carolina Charlotte

The variety of quantitative optimizations available in BIM software have the potential to leave more human factors to become secondary considerations in design. Computational tools can begin to embrace predictive models of human behavior in order to generate solutions which better serve their occupants, and better represent basic knowledge of human conditions. This project explores the approximation of human movement behaviors within an architectural space through the development of Animal, an agent-based model of circulation for Dynamo within the Revit environment.

INTRODUCTION

Architectural design is often driven by a complex collection of the designers learned and intuitive models of human behavior in response to space. These conceptions of human behavior influence nearly all aspects of a design. We consider the way in which occupants will use a space, move throughout a space, and perceive it. These models range from the simple understanding that a person moving through space will avoid an obstacle blocking their path, to more complex social phenomena such as pausing to engage in conversation. As humans, we find these types of situations intuitive and even simple in our everyday experience, and as designers, we anticipate and value these phenomena within spaces. While these concepts seem simple to us, they are relatively foreign within the realm of computational evaluation of space.

The increasing relevance of computational design tools to the discipline of architecture has empowered designers to rapidly simulate a multitude of quantitative design factors. Such systems provide both a simulated phenomena and a metric by which to measure the success of the system. These models rely on the objective and predictable behavior of a specific phenomena to provide a quantifiable output. Such optimizations have become nearly ubiquitous to designers within BIM applications. The use of these types of processes informs many of the decisions that designers make today. These types of tools are primarily used to solve for design aspects which can be predicted with high levels of accuracy, leaving designers to negotiate the impact that these solutions will have on more nuanced aspects of the design.

As computational models of design performance become more complex, the designer of a system may find themselves continually correcting for, or designing around the outcome of a quantitative design analysis. The designer may find that the system subverts aspects of the design which serve their intuitive response to human behaviors. Subjective behaviors, such as human movement in a space lack a singular scientific model for their prediction. In order to enable designers to generate more useful computational solutions to complex design problems, we must begin to consider the influence of subjective design considerations as well as objective simulations. This poses the question: How can we begin to translate the designer's basic intuitive understanding of human behaviors into useful computational tools?

In order to create a meaningful model of phenomena such as human movement, a system must be able to represent both the current state or location of the person it simulates, as well as the action or behavior which they exhibit as a result of this state. Agent-based modeling (ABM) provides a method for satisfying these conditions. This method models a system as a collection of autonomous decision-making entities called agents. Agents collect information about their current state and make decisions based on a predetermined set of rules.[1] In the context of the architectural models created with BIM software, we can define the agent as a simple point representing the location of a person. The behavior of the agent can then be simplified to a series of rules defining spatial navigation from one location to another. Placing such autonomous agents within the context of a BIM model can begin to provide our computational models with a simple yet practical understanding of human behavior within the spatial conditions present. This project explores the approximation of human movement behaviors within an architectural space through the development of Animal, an agent based model of circulation for Dynamo and Revit.

RELATED WORKS

Using computational methods to model emergent properties of collective behaviors of humans has been a focus of many researchers. Thomas Schelling was one of the first researchers to apply agent-based modeling to simulate complex problems. In the 1970s, he used a simple agent-based modeling methods to represent how segregation occurs in neighborhoods.[2] His analog model, using coins on graph paper, demonstrated the same methods used by more advanced agent-based models today. ABM has since been applied to simulation of many other complex systems. Another well known example is Robert Axelrod's use of agent-based models to study competition and cooperation between humans by modeling the *Prisoner's Dilemma* problem.[3]

By the mid 90s many researchers in various fields such as social sciences, earth sciences, and political sciences began using ABMs for modeling complex phenomena.[4] A variety of specialized software packages have been developed that have allowed researchers to easily employ ABM to model complex systems. *Swarm, Netlogo*, and *Repast* are examples of such specialized software. Each of these software packages allow certain amounts of freedom in programming agent-based models. *Netlogo*, for example, allows researchers to create and model using a simplified object oriented scripting language. It offers a user interface and simple visualization of models. While these methods are highly accessible, they do not easily allow the user to conduct research on a modeled architectural environment.

Many designers and architects have sought to model the behaviors of occupants within their proposed designs. There has been an extensive body of research on the movement of pedestrians in interior and exterior spaces. An example of this research is Turner's use of Gibson's model of environmental affordances to model the movement of pedestrians in an architectural setting.[5] Michael Batty et al. created a sophisticated model of pedestrian movement that takes into account destinations, directions of movement, obstacles, and the number of other pedestrians in the same space in both interior and exterior environments.[6]

Most of these models, can serve as valuable resources for designers. However, there has been little work done on creating agent based models that easily integrate into architects' digital design process. In the realm of design software for architects, one example of an available ABM tool is Quelea, the component library for the Rhinoceros visual scripting plug-in Grasshopper, that allows simple modelling of agents with physics based properties. Other libraries such as IGeo for Processing allow more technical users to apply agent based logic to geometry and export to Rhino by writing the logic in Java. These examples do not however, fit fluidly into the Revit BIM workflow. The need for incorporating these types of logic in a user friendly way is growing quickly. Especially as using BIM software such as REVIT has become an integral part of an architect's design process.

METHODOLOGY

Computational tools used by designers exist within an ecosystem of software providing packages of functionality to the user. The ultimate goal of this project is to create an agent based modeling tool which can provide designers with a means of integrating human movement into the computational design process. In order to maximize the usability of the tool, this project has been developed as a package for Dynamo, the scripting environment for the BIM software Autodesk Revit. This tool was written primarily in Python programming language using the Dynamo API with some additional functionality provided by standard Dynamo components. This allows designers to use this tool within the context of scripts or graphs which they may already be producing during the design process.

Each type of object in a designer's model may have different effects on the behavior of agents navigating the space. At their most basic level, walls serve to impede the movement of an agent, furniture impedes movement but not visual perception, and circulation elements such as stairs and elevators may serve as attractors pulling agents through a lobby towards levels above. Additionally certain programmatic elements may attract or repel the agents based on their desires. This package for Dynamo will provide designers with a method for defining and integrating the effects that these elements may have on occupants into the computational processes they use for design.

This agent based tool requires four primary inputs. First, the user specifies the quantity of occupants that they wish to simulate. Agents must

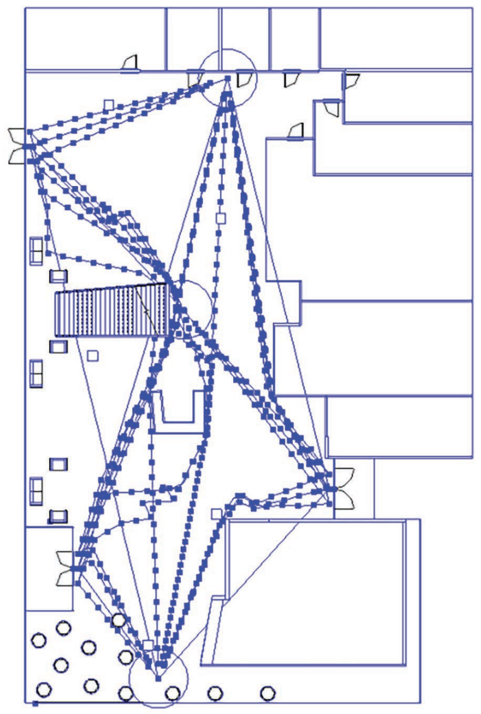
Figure 1: Agents navigate a the lobby space of a Revit model.

then receive a list of possible starting states. This is provided as an input to the tool in the form of a list of point locations. Third, the system requires a list of point locations to serve as the agent's possible destinations. Starting and ending locations are then randomly matched in order to provide a location-target pair for each agent in the simulation. Finally, the user provides the system with an array of solid geometries defining the environment for the agents to occupy. These geometries are selected by the user from their Revit model. The user must also define the appropriate response of the agent to the objects provided. Some additional settings for agent speed and other variables can be manipulated by the user as well. These inputs create the basic conditions for the simulation.

In order to simulate the movement of occupants in a space, the system relies on a set of algorithms which provide the agents with a method for handling the presence of obstacles in their path. An agent's ability to select an appropriate direction for movement relies primarily on information present within its field of vision. This field is created as an array of vectors radiating from the agent's current location. In order to approximate the human field of vision, a set number of these vectors are arrayed at even intervals from the direction of the agent's last step up to ninety degrees in either direction. Each array in the agent's field of vision is cast out to the environment and returns a list of intersections with objects. The distance from the agent to the nearest intersection in the list provides the value for that vector of the agent's vision. This calculation is repeated to form an array of distances which the agent can "see" at each interval of its vision. This creates the equivalent of two-dimensional vision consisting of a line of pixels with a brightness representing the visible distance in each direction. Using this basic vision, confined to a two dimensional plane, the agent gathers basic distance information about its environment.

Once this information is collected for the agent's current location, it must evaluate its possible movement vectors against reaching its target location. This is accomplished by scoring each vector of the view array. The view distance of the vector is multiplied by the inverse of the angle between the view vector and a vector to the agent's target location. Distance values are normalized as a value between zero and one. The values for the angle between the view vector and target vector are also scaled to a value between zero and one. This ensures calculations reflect scoring relative to possible options by normalizing the values. These values are then be multiplied by a weight coefficient. This allows for adjustment of which factor the system will prioritize when selecting the appropriate movement vector. Weights in the system include user defined variables for agent behavior and response to specific object types. This formula effectively ensures that each view vector is scored by its chance of moving the agent toward the target without coming in contact with an obstacle. (Figure 2)

The algorithm defines the agent's path by repeating this process of vector casting and scoring from each position which the agent moves to. The speed at which the agent travels during the simulation is defined by the user. This is then divided by the number of iterations that the system goes through per second to determine the appropriate distance to move the agent at each iteration.

In order to prevent the system from calculating only one possible solution for each starting point to target pair, the weight coefficients of both distance and direction factors are multiplied by a random value within a specified range. Each agent receives a specific set of weight coefficients which they will use to score at each iteration along their path. This helps to simulate the variations in behavior between individuals when navigating space.

In addition to this basic movement algorithm, agents are also repelled from objects that are less than a user specified distance from them. This helps to prevent the agent from occupying only the shortest path between the starting location and the target. For example, an agent who is rounding a corner to reach a target may see the greatest distance and angle at the edge of the obstacle corner. This would then result in a consistently high score along that vector. In order to prevent all agents from turning at the sharp corner of the object the agent will be repelled from the geometry by a value proportional to the distance between their location and the geometry. All wall geometry within the model incorporates

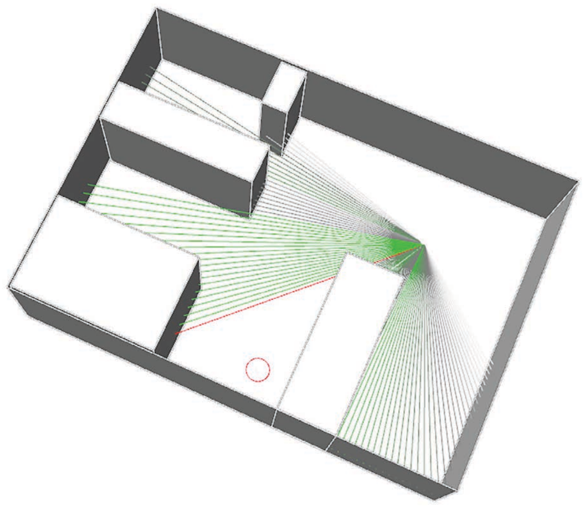
Figure 2: Scoring of agent's view vectors while navigating to the position of the red circle. Higher scores are shown in green and the highest is shown in red.

this logic in order to avoid collision.

Following this logic, the agent begins at a location provided by its location-target pair and steps iteratively toward its target while navigating the space of the model. These algorithms are written in Python within a Dynamo component. This allows for the recursive execution of the stepping function for each agent. This process repeats until the user-defined number of iterations has been reached.

The tool generates a visual output for the user by drawing a spline between all locations the agent stepped through on their path. This process is repeated for each agent in the simulation. The tool also outputs a two dimensional list of all point locations for each agent. This allows the user to easily incorporate this movement data into their script.

TESTING

In order to form a model of movement based on the scoring of possible vectors it is necessary to test the relationship of different weights on the outcome of the agent's movement. The examples in Figure 1 show multiple agents operating based on their randomly varied decision making weights.

In Figure 3, all agents share a common starting position within the large circle at the lower end of the image. This set of four agents all successfully navigated the obstacles within the space, and expressed only small differences in decision making at each step. The agents in Figure 4 were given a list of starting points to select from at random, and navigated to the target point from their selected location. Several agents beginning from starting point A exhibited a greater variation in scoring evaluation. This resulted in a divergence of navigation paths almost immediately after the first variation.

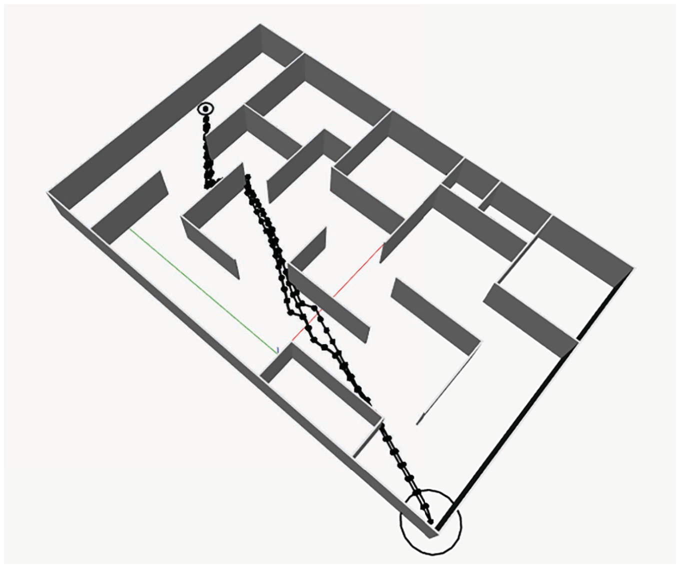

Figure 3: Multiple agents navigate toward a shared target location.

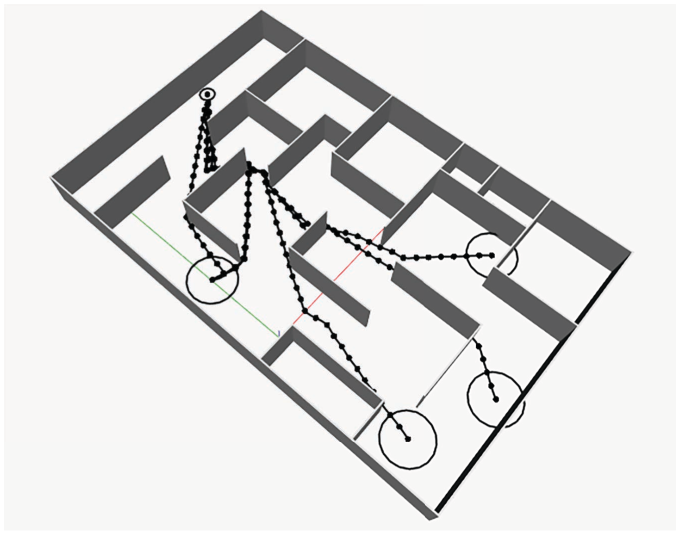

Figure 4: Multiple agents navigate toward a selection from a set of possible target locations.

During the testing process the most influential aspect of the agent's ability to navigate the space was the limitation of the maximum ratio between scoring weights. For example, an agent having a very balanced scoring criteria will navigate the space in a highly efficient manner. This is not the case however, for an agent who values the angle to the target more highly than the visible distance. Such an agent is highly likely to collide with walls, rendering the simulation invalid. On the other hand, an agent who highly values visible distance may wander into open areas which do not necessarily lead to the target. Testing of this ratio revealed that agents navigated the space the most efficiently when the ratio of weights was never greater than 1:3 in either direction. It is worth noting that the lack of efficiency resulting from valuing visible distance is not as detrimental to the overall goal of reaching the target as the collision behavior that results from highly favoring angle of travel. Understanding the limitations of this ratio on the agent's performance is important because it ensures that variations between individuals are still valid models of navigation.

Testing also revealed that when an agent is close to an obstacle, their step distance must not exceed the distance between their location and the wall along the movement vector. Limiting the step distance to slightly less than the distance to the wall allowed for the agent to stop at the wall geometry. The next iteration will then select a vector at a greater angle to the wall as long as the distance it may travel in that direction is greater.

Testing ultimately revealed that all environmental factors intended to influence the agent's movement are most effective when they are incorporated into the scoring algorithm as opposed to adjusting the agent's placement after it has selected the highest scoring vector. Once implemented, these findings dramatically decreased the amount of errors in the system and increased the possible range of individuality among agents.

FUTURE DEVELOPMENTS

The current implementation of this tool is constrained by a number of factors. The first of these limitations is the lack of three dimensional vision data and the agent's inability to leave their initial world plane. Having such ability would allow for more complex calculations of movement within multistory spaces. In addition to this, agents in this system are always in motion and fully aware of the location of their target. This limits the model to the representation of spatial navigation which is similar to that of an occupant who is highly familiar with a space. The constant movement of the agent does not currently allow for the simulation of any behaviors which might result in an occupant pausing in a space. This system would also benefit from the incorporation of interaction between agents. Currently, agents act without regard to the presence of other agents. This algorithm represents only one simple aspect of spatial awareness and can only begin to simulate the complexity of human movement in space.

Ultimately this tool is intended to provide a quantifiable simulation of the paths occupants may take through a space. This can become useful when implemented on a scale which can begin to describe areas of circulation density.

Agents within this system are generated with a predetermined understanding of the ways in which they should evaluate and navigate the space they occupy. This top down approach provides a stable and predictable model for circulation. In order for the system to begin to take on more human characteristics, it must be able to represent more than simple pre-programmed deterministic responses to its environment. In order to form a more complex computational model of human occupancy of space, the system would need to employ a model based on observance of human behavior. Future developments of this tool would be aimed at forming more complex models of movement through the introduction of machine learning methods. Training a tool like this using machine learning would open the possibility for non-deterministic emergent behaviors.

Additional features in future developments of this tool would include the ability of agents to respond to each other's presence either through simply avoiding collision or representation of chance "social" behaviors. In order to better represent the variety of occupants within a space, consideration must be given to the representation of agents who do not have explicit knowledge of their targets position. These improvements would greatly improve the validity and usability of the system as a tool for simulating human occupation and circulation.

CONCLUSION

The variety of quantitative optimizations available in BIM software often leave more human factors to become secondary considerations in design. Computational tools can begin to embrace predictive models of human behavior in order to generate solutions which better serve their occupants, and better represent basic knowledge of human conditions. Animal for Dynamo BIM represents a step toward a more human aware computational model. As designers our simplest intuitions about the way occupants use space are often kept separated from the tools we employ to design. This project seeks to provide a method for the integration of behavioral knowledge into the computational design process.

ENDNOTES

1. Bonabeau, Eric. "Agent-based modeling: Methods and techniques for simulating human systems." Proceedings of the National Academy of Sciences 99, no. suppl 3 (2002): 7280-7287. .
2. Schelling, Thomas C. "Dynamic models of segregation†." Journal of mathematical sociology 1, no. 2 (1971): 143-186.
3. Goldstone, Robert L., and Marco A. Janssen. "Computational models of collective behavior." Trends in cognitive sciences 9, no. 9 (2005): 424-430.
4. Šalamon, Tomáš. Design of agent-based models: developing computer simulations for a better understanding of social processes. Tomáš Bruckner, 2011.
5. Gibson, James J. The ecological approach to visual perception: classic edition. Psychology Press, 2014.
6. Batty, Michael, Bin Jiang, and M. Thurstain-Goodwin. "Local movement: agent-based models of pedestrian flows." (1998)

ACKNOWLEDGEMENTS

Thanks to Anthony Hauck and Phil Bernstein at Autodesk for their generous support of this project.

Thanks to Katherine Richard for her assistance in developing the ideas, logic, and representation of this project.