Complexity Machine 1
A 3D Modeling Application
Implementing Behavioral Simulation

Aaron Westre
University of Minnesota

COMPLEXITY MACHINE 1 IS A SOFTWARE APPLICATION DEVELOPED BY THE AUTHOR AS A MASTER OF ARCHITECTURE THESIS PROJECT AT THE UNIVERSITY OF MINNESOTA. The software acts as a platform for exploring three dimensional form produced via behavioral simulation. Specifically, the behaviors are modeled on emergent group dynamics found commonly in nature such as flocking, chasing, and evading. Though various commercial softwares and numerous small-scale architectural projects exist in this area, Complexity Machine 1 is intended as a freely available and generic platform for exploring the formal implications of these emergent behaviors. The simulated behaviors are governed by a variety of parameters and a set of eight simple rules. Formal results are influenced by these parameters and rules; along with scale, color, and geometric settings. The flexibility of the software allows users to investigate a vast array of potential forms, adjust settings in real time, and export the results for further manipulation. Complexity Machine 1 continues to be refined and improved towards the goal of providing an easy to use platform to designers for exploring forms that emerge from complex group behavior.
1 Introduction

Simulation has been a powerful tool in the sciences since computer technology was first introduced into academic and laboratory environments. From medicine to meteorology, simulation is used to refine our understanding of natural systems and to predict how they will change. The technology of simulation is employed as a primary tool in what mathematician Warren Weaver called the “great middle region” of “organized complexity”. (Weaver, 1948) This is the domain of complexity theory, a science that has grown up alongside information technology. It has as its primary focus the study of aggregation; specifically, how populations of autonomous agents – be they cells or people – are almost preternaturally predisposed to move from chaotic states toward states of intricate order. (Waldrop, 1992) Complexity theory, and the simulation technology that evolved with it, have been co-opted by industries ranging from logistics to video gaming to cinematic special effects. Particularly important to the project presented here is the work of Craig Reynolds who, in the late 1980s, devised an algorithm which realistically simulated the graceful movements of flocks of birds and schools of fish. (Reynolds, 1987) Reynolds’ algorithm, using only three simple rules, produced such realistic results that it is still in use today whenever virtual flocks and swarms are needed.

An interesting inversion occurs when the scientific tool of simulation is used toward creative ends. Whereas the sciences seek to model our world as it exists, design has as its challenge envisioning our world as it could exist. Simulation let loose from the descriptive role it serves in science can function as a tool for design speculation and discovery. Diverting computational tools toward aesthetic ends is fast becoming an established method of exploration for a growing number of artists and designers. Simulation and visualization of concepts from physics, mathematics, and the biological sciences are just a few of the vast array of methods that have been borrowed by artists and designers who have discovered unexplored aesthetic and performative potential therein. The computer becomes more than a tool in these explorations. It is endowed, through the development of algorithms, with a certain amount of autonomy – becoming in essence a co-author of the work. As this emerging field of computational design grows concepts from complexity theory are becoming an important lens through which designers see their work. Though computers have been in use for decades in architecture, complexity theory and the simulation technology that comes with it offer ways to explore previously unreachable territory. It is a vast territory; one that calls out to be explored. Complexity Machine 1 is a tool to explore one small sector by diverting simulation from its standard analytic role, towards one of creative synthesis.

2 The Software

Complexity Machine 1 (CM1) is an agent-based simulation environment that implements a logic that is inverted from that found in the sciences. Whereas agent-based simulation as a scientific tool offers insight into rule-based behavior first derived through observation of natural processes, simulations in CM1 use these simple rules as a starting point for creative exploration. These rules are primitive, meaning they each represent a single, simple potential action that governs the behavior of individual agents in relation to their context. They describe behaviors related to movement such as group cohesion and separation, individual characteristics such as lifespan and reproduction, and environmental constraints such as zones of activity. Based on the rules chosen, the software simulates the behavior of the interacting agents in space, yielding three dimensional form that can be further investigated from an architectural perspective. Any defined simulation can be run multiple times and, due to the indeterminism inherent in complex-adaptive systems, every run will produce a new formal assemblage. By implementing this bottom-up approach for the creation of simulations, CM1 becomes a generic platform for the production of forms based on the emergent phenomena of group dynamics. This generic quality allows a great diversity of potential simulations and from each simulation, a great diversity of forms.

CM1 was developed using Processing, a free and open source computer programming environment that implements the Java programming language. (Reas and Fry, 2007) Pro-
Proceeding was developed by Casey Reas and Ben Fry while they were graduate students at MIT and is under continuous development by a community of visual artists, designers and programmers. As a coding application geared specifically toward producing visual effects, it allows a much more rapid development cycle than traditional programming techniques. This ease of development and the large community of code contributors were the major factors in the decision to use Processing as a platform for creating CM1. Prebuilt code libraries were incorporated into CM1 that handle 3D graphical rendering, vector mathematics, file export, user interface, and other functionality that sped up development dramatically. The dedicated community that surrounds Processing was invaluable in many ways including advice offered on discussion boards, documentation of problematic programming issues, and other help.

3 Parameters and Rules

A simulation in Complexity Machine 1 is composed of one or more groups of agents, or flocks, each of which have a variety of parameters that can be adjusted. (fig. 2) An agent is, at its most basic level, a point in space. In addition to the location, each agent keeps track of its heading and where it has been. A flock is a collection of agents that share the same settings, including maximum velocity, maximum turn radius, a bounding volume that limits where they can travel, field of view, and other factors. These collective parameters can be set, and adjusted in real time, by using a control window. However, attributes such as locations, headings, and velocities of individual agents are determined by the interaction of those agents with each other. In this way, the user has an indirect type of control over the behavior of the agents. Initial parameters are set and the particular behavior of the agents emerges based on those specific settings. As the action of the simulation proceeds, specific observed behaviors can be enhanced or limited by adjusting the parameters in real time as the simulation progresses.

In addition to parameters, each flock has a set of eight simple rules that it can follow: (fig. 3)

1. Cohese: Move toward the average location of flockmates in view.
2. Align: Conform to the average heading of flockmates in view.
3. Evade: Avoid colliding with flockmates in view.
4. Pursue: Choose a target agent and chase.
5. Consume: “Eat” any agent in the target flock that is close enough.
6. Die: Become inactive if the maximum age is exceeded.
7. Randomize: Choose a new heading randomly.
8. Reproduce: Create a new member of this flock at the current location.

CM1 users can determine how each flock in a simulation adheres to these rules. Each rule has a probability setting that governs how likely it will be used by each agent in a flock during each step of the simulation. The first four rules (cohere, align, evade, and pursue) and rule 7 (randomize) also have a weight setting that can be adjusted to fine-tune the behavior of a flock. For instance, to define a standard flocking behavior for a flock the cohese, align, and evade rules are assigned probabilities and weights of one hundred percent, with the probabilities of all other rules set to zero. If the user desires a flock behavior that creates a more diffuse distribution of agents, the weight of the cohese rule can be lowered until the behavior is satisfactory. (fig. 12) More complex interactions can be achieved by adding additional flocks and rules. A predator and prey simulation can be defined by setting one flock to pursue another that has a basic flocking behavior. By recombining rules and adjusting probabilities and weights, a wide array of behaviors can be explored. Since the rules in CM1 are based on rules found in animal behaviors, but not limited to accurate execution of those behaviors, new and entirely artificial behaviors are continuously invented through the set up of simulations. These artificial behaviors can be evaluated and refined based on their propensity to create beautiful, interesting, or useful formal arrangements.

4 Form

Agents, and the paths they trace through space, are simply collections of points in space in
Complexity Machine 1. These collections form a three dimensional armature over which a wide variety of forms can be constructed. For each flock, shapes can be specified for both the location of the agents and for the paths those agents trace through space. (fig. 4-7) Basic cubes and triangular shapes are possible, as well as more complex combinations of shapes. The scale of these shapes is specified in three dimensions so that more or less elongated shapes are possible. The shape and scale of the agent paths yield the most dramatic results visually, since paths occupy a greater and greater volume as the agents move around during a simulation.

The character of forms produced in CM1 is also influenced by the values of the various parameters and rules for each flock. A high maximum turn radius will yield a more jagged path, while a low maximum will produce smoother arcs. (fig. 10) Maximum velocity determines how long each segment of the path can be. (fig. 11) The overall density of form is influenced to a great degree on the initial population of a flock. A high initial population will precipitate a very dense weave as the paths of each agent are rendered. (fig. 8) The aggregate shape that a flock traces out can be effected by placing its starting point, or spawn origin, at different locations within the activity volume. The activity volume sets the limits of motion for a flock, so starting the flock close to the edge of that volume will cause it to move immediately away from that edge. (fig. 9) By experimenting with the spawn origin and the overall scale of the activity volume, specific types of movement can be encouraged. The aggregation of path forms resulting from these choices begin to suggest useful architectural forms such as walls, canopies, screens, and other configurations. Rules, too, have a profound impact on overall form. Adjusting cohesion, alignment, and evasion control how dense the resultant aggregation will be. (figs. 12-15) The reproduce rule has very dramatic outcomes in which a very small initial population expands into a dense mass in a relatively short period of time. (fig. 15) Branching structures are simple to achieve through the judicious use of this rule.

These are only a few of the key variables that have influence over the formal possibilities in CM1. The control window of the application contains an array of interface elements that can be used to adjust the settings for each flock in a simulation. These adjustments can be made as the simulation progresses, yielding instantaneous visual feedback about the impact of an individual setting on the form produced. By altering settings in this manner, often by only slight amounts, an incredible spectrum of form is achievable. (fig. 16-18) During the development of CM1, nearly one thousand unique formal variations were generated. A number of presets were incorporated into the software that allow users to quickly load the settings for simulations that the author identified as potentially useful in an architectural design context. These preset simulations do not correspond to specific building typologies, but rather generate forms that create interesting spatial relationships, surface textures, and overall compositions.

5 Conclusion

As the necessary tools become more accessible to designers, a growing number are turning to the production of software in order to explore new territory. Though the computer has been an object of study in architecture and other design disciplines for a number of decades, the discipline has seen an acceleration of experimentation in recent years. Several factors have contributed to this expansion including: the increasing ease of software production, the wide availability of high-performance computers, and the growing impulse to borrow new modes of research from the sciences. Couple these factors with the communicative power of the internet and a massively collaborative project emerges in which ideas and code are shared instantly, spurring new inquiries and experiments. In an effort to contribute to this collective project, the source code of Complexity Machine 1 was licensed under the GNU GPL, an open source license that will allow others to use it as a starting point for their own explorations.

Complexity Machine 1 does not fit the traditional description of computer-aided design software. Instead, it may be more accurately classified as a type of game because of the indirect nature of the user’s control over the outcome of simulations. While this mediated...
quality limits the user’s authority in a traditional sense, it opens up new possibilities for exploration. Since behaviors in CM1 are modeled on – but not limited to – those found in nature, entirely artificial scenarios can be developed that result in forms which satisfy various performance criteria such as density, arrangement, size, or aesthetics. These evaluations are left as an exercise to the user based on their own interpretive logics and the requirements of their specific design task. Because of this neutrality, CM1 acts not as a design tool geared toward a specific purpose, but rather as a general tool for the exploration emergent behaviors. This neutrality, however, also limits the usefulness of the software in design workflows. The emphasis thus far has centered on the refinement of the behavioral logics of the software, i.e. on making the correlation between the parameters of a given flock and its behavior direct and sensible to the user. Products of the software, while often evocative, are most likely to be usable in an architectural design only after further manual refinement by the designer. Further development of the software will focus on increasing the immediate usability of forms produced while assuring that the generic, exploratory nature of the software remains. While this task could be accomplished by several means, the introduction of further productive limits seems most promising. Enabling the user to specify site conditions by importing 3D models, the introduction of material and joinery logics, and
the specification of programmatic spaces are a few additions that have been identified. With the incorporation of these functionalities, CM1 could potentially evolve from its current state as a free-form, inspirational tool to a platform for guiding emergent behaviors towards very specific, though still surprising, architectural ends.

6 References