Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)

A Framework for Analyzing and Creating Self-assembling Systems

Navneet Bhalla and Christian Jacob
Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, T2N 1N4
bhalla@cpsc.ucalgary.ca, jacob@cpsc.ucalgary.ca

Abstract - Self-assembly is an emergent property of decentralized systems, which is seen throughout nature. Understanding and applying this emergent property continues to be an important subject in the natural sciences, as well as engineering and computer science. However, only the specific principles and mechanisms of self-assembly are considered within the scope of their respective disciplines. A framework is presented here, which abstracts self-assembly to components, environment, energy, assembly protocol, spatial relationship, localized communication, and rule set. By viewing self-assembly in this manner, this framework facilitates melding the various self-assembly principles and mechanisms studied across disciplines. The benefit of this is that it aids in the pursuit of designing synthetic systems mirroring the robustness of this bottom-up construction process in nature. Several experiments are presented that exhibit this robust construction process, and demonstrate how this framework can be leveraged for analyzing and creating self-assembling systems.

I. INTRODUCTION

The realization of synthetic entities mirroring the highly adaptable and robust designs seen in nature would have tremendous benefits in a wide variety of applications. Such applications include, but are not limited to, robotics, material science, medicine, and architecture. Understanding the fundamental principles and mechanisms of self-assembly is an essential first step in the creation of such systems.

From a computational perspective, entities in nature can be viewed as the result of many interacting decentralized agents, governed by simple rules. One of the emergent properties of these systems is self-assembly. Here the term self-assembly is refined to describe a process that can be controlled through appropriately designed components and their environment, and is reversible [1]. In this context, pre-existing components interact in order to create emergent aggregate forms.

In natural systems, self-assembly is primarily dictated by the morphology of the components within an environment, including their physical and chemical properties [2] [3]. Research into understanding the properties and mechanisms of self-assembly have long been, and continue to be, an important research subject in the natural sciences, as well as engineering and computer science.

In this paper we use the self-assembly principles and mechanisms from several examples to abstract a framework for analyzing and creating self-assembling systems. This framework is the basis to the design of a virtual system. Five experiments are discussed to demonstrate how this framework can be leveraged to analyze and create self-assembling systems.

II. BACKGROUND

Natural self-assembly can be viewed as coming in four different types: static, dynamic, templated, and biological [1]. Systems that do not dissipate energy are examples of *static* self-assembly. In contrast, systems that dissipate energy are examples of *dynamic* self-assembly. Templated and biological self-assembly are both subsets of static and dynamic self-assembly (depending on the context of the system). In *templated* self-assembly, entities are created based on interacting components and regular features in the environment. The variety and complexity of biological systems is the defining characteristic of *biological* self-assembly [1].

One of the most successfully studied natural self-assemblies is the creation of artificial snow crystals [4]. In nature, the process of creating a snow crystal is initiated through the use of a seed particle, typically a dust particle, to which water molecules can attach. The morphology of a snow crystal is dependent on how water molecules selectively attach to the developing structure, based upon the hexagonal symmetry of ice crystals. This process has been leveraged to create predictable artificial snow crystals of varying form, under controlled conditions. In this case, the self-assembly process is initiated using the tip of an ice needle to act as the seed particle. The ice needle is placed in a chamber, where the supersaturation level of water vapour, temperature, and background gas can be controlled. By understanding the relationship between these three variables, it has lead to the creation of predictable artificial snow crystal morphologies [4].

The formation of snow crystals is one example of static self-assembly in natural systems. To date, most research on self-assembly in the natural sciences has focused on this type [1]. However, in recent years, a new form of self-assembly has emerged in the disciplines of engineering and computer science. In 1957, L.S. Penrose and R. Penrose showed the first mechanical analogue to self-assembly, in the form of static templated self-assembly [5]. They created two components, labelled A and B, with unique morphologies that connected in either an AB or BA configuration. Multiples of these A and B components were confined to a track in a random ordering. When the track was shaken, it allowed components to move horizontally in one-dimension and interact with one another. When an AB or a BA seed complex was placed on the track, it would cause neighbouring A and B or B and A components to self-assemble into AB and BA complexes respectively. In this example, the morphology of the components was the key factor in directing the self-assembly process.

1-4244-0708-7/07/$20.00 ©2007 IEEE

The results of this experiment showed that the mechanisms of self-assembly could be used in the creation of synthetic self-assembling systems. It is the precursor to another form of self-assembly classified as *netted systems* [1]. These systems consist of sensors and controllers that interact and self-assemble through data communication.

Templated self-assembly has been a popular form of self-assembly in netted systems. The work of L.S. and R. Penrose has been extended to netted systems [6]. In this case, programmed electromechanical square shaped components move around on a cushion of air in two-dimensions, and interact by latching and unlatching to one another. By placing a seed complex in the environment (five components arranged as a rectangle), free components are able to self-assemble and construct replicas of the seed complex. Since the components in this system have the same square form and their environment is static, data communication is required between interacting components to direct the self-assembly process.

The use of templated self-assembly has also been used in creating virtual models and physical self-assembling systems in three-dimensions [7]. In this example, programmed cubic components move around in a liquid medium and interact by attaching to one another through the use of electromagnetism. The electromagnetic regions on the faces of the components can be activated and deactivated as required. A seed complex in the environment (a three by three square, consisting of nine smaller square regions) allows for the creation of a single aggregate structure, following a grid formation. This system is capable of forming aggregate cubic and rectangular structures.

Another form of self-assembly being adopted in netted systems is biological self-assembly in the form of *swarm intelligence*. Swarm-bot [8] is the collective name to the set of programmed cubic robotic modules (s-bots), which are able to physically link together. In the previous examples of netted systems, energy was transferred to the components, in order for them to be mobile in their respective environments. In this case, each robotic module is self-powered and can move around its environment. Swarm intelligence is used to direct the robotic units in order to create aggregate structures, mimicking the formations of social insects (e.g. the formation of living bridges by *Oecophylla* longinoda worker ants) [8]. Therefore, s-bots can self-assemble into aggregate structures to move across terrain, not possible by an individual s-bot.

Netted systems rely on data communication between sensors and controllers to direct the self-assembly process. The principles of natural self-assembly can be used to simplify or enhance communication between components, as a means of physically and/or chemically encoded information [9]. Likewise, netted systems can be used to continue to study natural phenomena.

III. FRAMEWORK

The mechanisms and constraints in natural self-assembly and netted systems do not need to be viewed separately. For the purposes of analyzing and creating self-assembling systems, these mechanisms and constraints can be abstracted to seven items:

- Components
- Environment
- Energy
- Assembly Protocol
- Spatial Relationship
- Localized Communication
- Rule Set

Components are defined by their properties. Such properties include, but are not limited to, shape, mass, scale, and material properties. Depending on the context of a particular system, the relationship between components and their environment can be encoded in the components' properties and/or the other items in the framework.

The physical and chemical properties of the *environment* will influence the manner in which components interact with one another, as well as the way in which components self-assemble. For example, the environment medium can influence the motion of the components. The environment in which components are subjected to can provide various functionalities, such as a boundary to which components are confined to. The state of the environment, whether it is static or dynamic is an important factor in self-assembling systems. Controlling the parameters of a dynamic environment can be used to direct the self-assembly process of components that have limited properties or constrained interactions.

In order for the components to self-assemble, the components need to be mobile in their environment. This requires the components to have *energy*. Energy could be available internally (dynamic self-assembly) and/or transferred to the components by the environment (static self-assembly).

An *assembly protocol* defines the methods in which components can self-assemble. These methods are highly dependent on the scale of the system, as well as the physical and chemical properties of the components and the environment. For example, molecular attributes could be used for systems at the micro-scale, whereas magnetism could be used for systems at the macro-scale.

The *spatial relationship* between the components and/or elements in their environment defines the underlying pattern formations capable by a particular system. Pattern formation has a great influence in the range of achievable self-assembled forms by a system, and is seen at all scales. The influence of a spatial relationship is seen in both natural systems (e.g. the hexagonal symmetry of ice crystals) and synthetic systems (e.g. the linking points of robotic units in modular and swarm robotics).

Localized communication is an important consideration in self-assembly. This is the key factor in viewing self-assembly as an emergent property of decentralized systems. This is seen throughout nature at all scales. Localized communication dictates how components interact with one another and their environment. It can be viewed as a physical and/or chemical encoding [9], or achieved through data or other communication means.

The *rule set* can also be viewed as a physical and/or chemical encoding [9], or achieved through programmed components and/or environment. The rule set can be basic and still lead to a wide variety of self-assembled forms (e.g. the various supersaturation, temperature, and background gas settings lead to varying snow crystal morphologies). Or, the rule set can be internalized in the components and lead to a wide variation of self-assembled entities displaying many forms and performing many functions (e.g. DNA within cells).

This proposed framework facilitates the melding of the various self-assembly principles and mechanisms across disciplines. The benefit of this is that it aids in the pursuit of creating synthetic systems mirroring the robustness of this bottom-up construction process in nature.

IV. SYSTEM DESIGN

We created a software system based upon the framework presented, as a proof of concept to analyze and create self-assembling systems. The objective of the system was to determine how spherical components could self-assemble into two-dimensional and three-dimensional entities with symmetric and/or asymmetric properties. To achieve this objective, we incorporated self-assembly principles and mechanisms described in the creation of artificial snow crystals and netted systems.

Single components move around freely within the environment. One of these components is selected at random and placed in the environment, and set to remain stationary. This component acts as the seed particle to initiate the self-assembly process. When free components collide with stationary components, they either repel or assemble with them and become incorporated into the aggregate structure as stationary components. Through communication between neighbouring stationary components, it is determined where new free components are allowed to attach. It is this feature that was anticipated to allow the creation of specifically designed aggregate structures, instead of the resulting aggregate structure displaying fractal forms, as created in diffusion-limited aggregation models [10]. The details of the proposed system are as follows.

Spherical components in the system are all of unit radius and unit mass. The components are considered as solid entities and follow the principles of elastic collisions. Components are confined to a spherical environment. The surface of the environment is also treated as a solid structure in order to serve as a boundary to confine the components. The radius of the environment is specified at the beginning, and then remains fixed for the duration of a simulation run. Components are assigned random velocities within a user specified range and move around in three-dimensions within the environment. When free moving components collide and attach to stationary components, their velocity is set to zero, and they become stationary themselves.

For algorithmic reasons, the components' and environment's shape was chosen to be spherical. Computing collision detection and response between components and with the environment is much simpler for spheres, in contrast to other three-dimensional forms (e.g. polyhedra). Component morphology can be used as a physical or geometric means of encoding information to direct the self-assembly process [9]. Since the components in this system are all unit spheres, additional component attributes are required. These additional attributes are described in terms of the assembly protocol, spatial relationship, localized communication method, and rule set.

The assembly protocol in this system is based on the concept of *stickiness*. When two components collide, and each is in a particular state, the two components stick together to form an aggregate structure. One factor affecting a component's state is its location in the aggregate structure. The location of a component in the aggregate structure is determined by the spatial relationship between components and a form of localized message passing.

The underlying pattern formation or spatial relationship between components defines the set of resulting forms of a self-assembling system. Locations on a component, referred to as *sticky sites*, determine the patterns achievable by the aggregate structure. The user defines these sticky sites, in three-dimensions on the surface of the components. The locations of the sticky sites are the same for every component present in the system. Figure 1 shows two example components with different sticky sites and their resulting pattern formations.

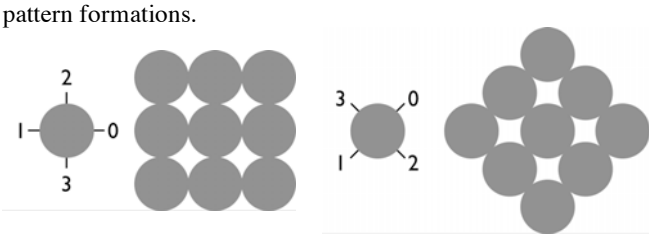

Figure 1: 2D square grid pattern (left) and diamond grid pattern (right)

For computational reasons, sticky sites are defined in a pairwise fashion. In Figure 1, the pairs of sticky sites are the ones labelled (0, 1) and (2, 3). When a component collides with a stationary component, it attaches to the closest available sticky site on the stationary component. If no sticky site is available, the colliding component reflects off the stationary component. The manner in which two components stick together is determined by the pairwise relationship of their sticky sites. For example, if the available sticky site on the stationary component is at position zero, then the colliding component will attach via its sticky site at position one. This eliminates the need to calculate the orientation of each component. It is also used to allow for two forms of localized communication.

Components are able to communicate with their local neighbours. The user-defined spatial relationship determines the neighbourhood relationship of the components. In the context of this system, components can only communicate with their immediate neighbours, i.e. the ones they are directly connected to.

Communication between components is used to update two types of information. These two types of information are encoded into every component present in the system. Both types are used as a means for the components to infer geometric information from their neighbours in regards to their position in the aggregate structure.

This information is based on the axes that are created through the pairwise relationship of the sticky sites. The number of axes present determines the number of values used to represent this information. One axis is created for each sticky site pair. For a two-dimensional square grid and a three-dimensional cubic grid this can be viewed as x-y and x-y-z Cartesian coordinates, respectively.

The first type of information is referred to as *global axes information*. Each component at the beginning of a simulation run starts with their global axes information set to (0, 0) for the two-dimensional square grid and to (0, 0, 0) for the three-dimensional cubic grid examples. Each number represents a component's location in relation to the x-y or x-y-z axis. Furthermore, the pairwise relationship of a sticky site set can be interpreted as the resulting axis having a positive and negative direction. The selection of which sticky site within a pair is negative or positive is arbitrary; its selection only needs to be consistently referred to in the manner in which it is selected.

When a component attaches to another component that is part of a self-assembled aggregate structure, it updates its global axes information. It does this by first requesting the global axes information from the component it has attached to. The component then updates its own global axes information by first incrementing or decrementing the value corresponding to the axes through which it is attached. Secondly, it either copies, increments, or decrements the rest of the values appropriately. The increment or decrement value is dependent on the type of spatial relationship used. For example, an increment and decrement value of one is used for two-dimensional square grid patterns, as shown in Figure 2.

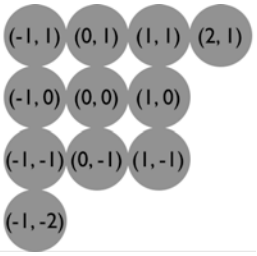

Figure 2: Global axes information for 2D square grid pattern, with component (0, 0) positioned at the centre of the grid

The second type of information is referred to as *axes count information*. Initially, all components start with their axes count values set to (0, 0, 0, 0) for the two-dimensional and to (0, 0, 0, 0, 0, 0) for the three-dimensional Cartesian coordinates examples. In this case, two values are used to represent a component's location in relation to each axis. Axes are used in a way to allow components to infer the number of components that are located in both directions of each axis present. For a specific component that is part of an aggregate structure, this is the number of components to its left and to its right, for example.

Again, when a component attaches to the aggregate structure, the axes count information is updated. However, in this case, since a newly attaching component can affect the axes count information of other components in the self-assembled aggregate structure, multiple components need to update this information when a new component attaches.

Two cases arise for this type of information. The first case is when a component attaches to the aggregate structure, and it only has one neighbour. In this case, the axes count information only needs to be updated for one axis in one direction. Figure 3 shows an example of this case. The second case is when a component attaches to the aggregate structure, and it has more than one neighbour. In this case, axes count information is updated for multiple axes and/or in both directions of an axis. Figure 3 also shows an example of this second case. For both cases, updated information and communication is done locally as a propagation of information. This maintains the decentralized characteristic of natural self-assembling systems.

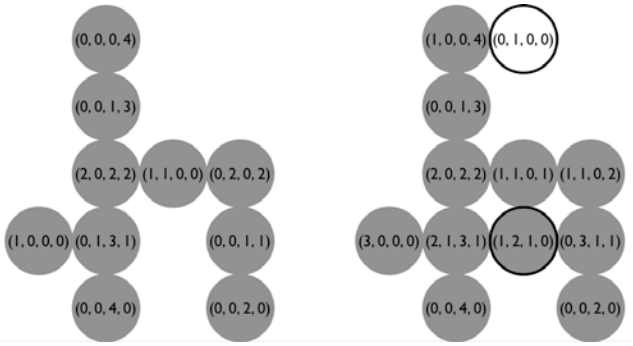

Figure 3: Axes count information (original configuration, left), with case one indicated with the black-outlined white-filed component, and case two indicated with the black-outlined grey-filled component

These two types of information were incorporated into a rule set. By doing so, it was anticipated that the system would result in being able to achieve the objective of designing self-assembled structures with symmetric and/or asymmetric properties.

The rule set, which is specified by the user, is the same for all components that are present during a simulation run. A number and a type is used to reference rules. Rule type one is in reference to the global axes information, and rule type two is in reference to the axes count information. When a component attaches to the aggregate structure, it goes through each rule in the set and executes each rule that applies.

Each rule is defined as a state-action pair. The state comprises of two parts. The first part depends on whether the component is mobile or stationary. The second part is in reference to either of the two types of information. If both parts are satisfied, then an action is performed. The action in this case is the activation or deactivation of sticky sites. By doing so, it allows for the emergence of self-assembled structures with defined boundaries.

Both rule types allow for the emergence of symmetric structures. However, the axes count information could possibly be the same for multiple components in the aggregate structure. This gives the potential for using one rule that is applicable to more than one component in the aggregate structure. Figure 4 (left) shows an example of this condition. In contrast, only the global axes information allows for the emergence of asymmetric structures. This is achieved because components can be identified uniquely, and thus a rule can be specified for a component in a unique state. Figure 4 (right) also shows an example of this condition.

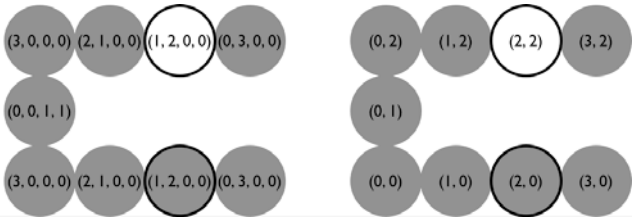

Figure 4: The two black-outlined components with the same axes count information (left); and the same two with unique global axes information (right)

Table 1 provides a summary of the system in the context of the framework presented here. The objective of the system is to determine how spherical components can self-assemble into two-dimensional and three-dimensional entities, with symmetric and/or asymmetric properties.

TABLE 1: SYSTEM SUMMARY

Framework	System Design
Components	spheres of unit radius and unit mass
Environment	spherical boundary to contain components
Energy	components are set to a random velocity
Assembly Protocol	stickiness
Spatial Relationship	pairwise relationship of sticky sites
Localized Communication	global axes information and axes count information
Rule Set	state-action pair (state: mobile vs. stationary, and global axes information vs. axes count information; action: activation and/or deactivation of sticky sites)

V. EXPERIMENTS

The software was written using BREVE [11], a software package for visualizing decentralized multi-agent systems. To test the conceptual foundation of the proposed system, four entities were designed as the end target structures as the result of self-assembly: a cube, the letters N and B, a chair, and a mug, as shown in Figure 5.

These structures were chosen because the have symmetric and/or asymmetric properties. The forms of these structures cannot be created through pattern formation alone. They require additional information, and therefore were excellent candidates to test the system.

In order to create these structures, three spatial relationships were needed. These three formations are a three-dimensional cubic grid, two-dimensional hexagonal grid, and a three-dimensional hexagonal grid (a layered version of the two-dimensional hexagonal grid). All three are shown in Figure 6.

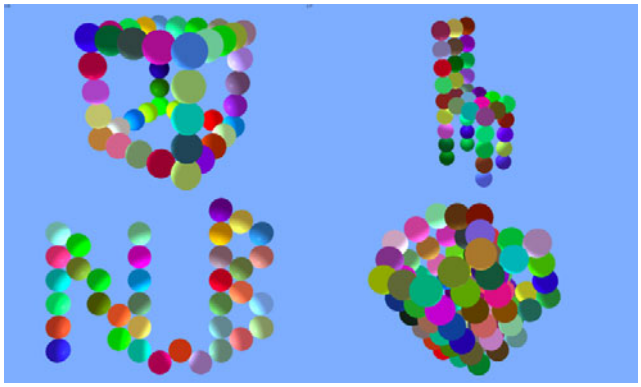

Figure 5: Four desired entities: cube (top left), chair (top right), NB (bottom left), and mug (bottom right)

Figure 6: 3D cubic grid (left), 2D hexagonal grid (centre), and 3D hexagonal grid (right)

For the three-dimensional cubic grid, the update scheme used for the global axes information was the one as described in the previous section. An increment or decrement of one was used depending on the positive or negative position of the axis the new component was self-assembling on, and all other values were copied.

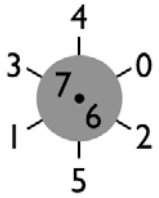

Figure 7: Sticky site locations for 3D hexagonal grid, with positions 6 and 7 located at the centre-front and centre-back of the component indicated by the black circle

The update scheme developed for hexagonal grids was developed from the idea of fitting a hexagonal grid to a square grid. With this idea, three cases arise for updating the global axes information. Each case is described in reference to the sticky sites positions shown in Figure 7. When a free component attaches to a stationary component at either position four or five, an increment or decrement of two is used to update the vertical information. All other information is copied. The second case arises when a free component attaches to a stationary component in either the zero, one, two, or three positions. This is a special case, and both the (0, 1) and the (2, 3) axes are considered as horizontal information. As a simplification of the resulting geometry,

for each component the values for these two axes are the same. Hence, the vertical information corresponding to axis (4, 5) is incremented or decremented by one, and the horizontal information corresponding to (0, 1) and (2, 3) is incremented or decremented by one, depending on the resulting position of the free component in relation to the quadrant on a Cartesian plane. In the three-dimensional case, an update to the depth information corresponding to the (6, 7) axis is incremented or decremented by one, and all other information is copied. As a result, this update scheme for the global axes information allows each component in the aggregate structure to have a uniquely identified position. Figure 8 gives an example of this idea for the two-dimensional case.

The two-dimensional and three-dimensional hexagonal grids were used to create the letters N and B, and the mug, respectively. The three-dimensional cubic grid spatial relationship was used to create the cube and the chair entities. These spatial relationships were used in five experiments.

The first four experiments corresponded to the four desired entities. For these experiment, rules utilizing the global axes information were used. This was done because it was a simplified approach to test the conceptual basis of the system, since these experiments are a proof of concept. The parameter settings corresponding to these four experiments are summarized in Table 2. The number of rules for each experiment equals the number of components needed to create its respective desired entity. A fifth experiment was conducted which utilizes rules based on both the global axes information and the axes count information. This was done as an attempt to reduce the number of rules required by taking advantage of the symmetric properties of a desired entity. This experiment was done to create the cube entity. The parameter settings corresponding to this experiment are also included in Table 2.

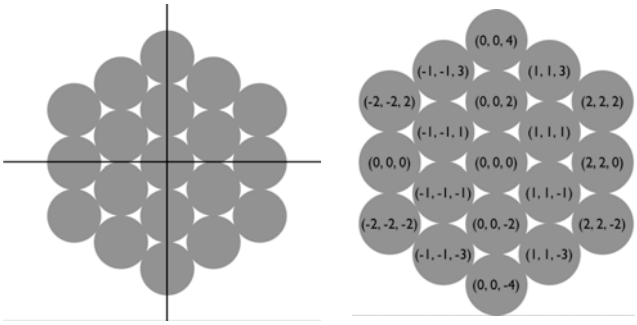

Figure 8: Global axes information for a 2D hexagonal grid shown as a 2D square grid (left), and the global axes information values (right)

VI. RESULTS

Each of the five experiments was successful in having a subset of the overall components self-assemble into their respective desired entities. The robustness of the bottom-up construction process of self-assembly was demonstrated in each of these five experiments.

Figure 9 shows the results of the first four experiments, where components were assigned random colours. Each time a particular system was executed, the self-assembly process was unique, but the final desired entity was always achieved. For example, in one simulation run, the base of the mug self-assembles, followed by the cylinder, and finally followed by the handle. In another simulation run, each of the three main features of the mug could be partially self-assembled at some time point, and continue to self-assemble in parallel, as shown in Figure 10. This was achieved because the actions associated with the rules specified all the specific sticky locations that should be activated or deactivated for each component comprising the desired entity.

Another qualitative observation from running these four experiments showed that even the mug (which consists of approximately twice as many components as the other three models) was able to self-assemble easily. This was due to the fact that most of the components present in the mug are part of the cylinder. As a result, there are a greater number of locations and a much larger surface area (in comparison to the other three models) where free components can self-assemble to. For example, the legs of the chair were on occasion difficult to self-assemble, because free components would take a long time to collide with the stationary components comprising the legs, due to the stochastic behaviour of the system.

TABLE 2: EXPERIMENTS SUMMARY

	Parameter Settings					Desired Entity	
Experiments	Number of Simulation Components	Component Velocity	Environment Radius	Rule Type	Number of Rules	Number of Components	Symmetric vs. Asymmetric
Cube - 1	150	10 units/sec	30 units	global axes information	44	44	symmetric
Chair	150	10 units/sec	30 units	global axes information	40	40	symmetric and asymmetric
NB	150	10 units/sec	30 units	global axes information	36	36	Asymmetric
Mug	300	10 units/sec	30 units	global axes information	87	87	symmetric and asymmetric
Cube - 2	150	10 units/sec	30 units	global axes information and axes count information	35	44	symmetric

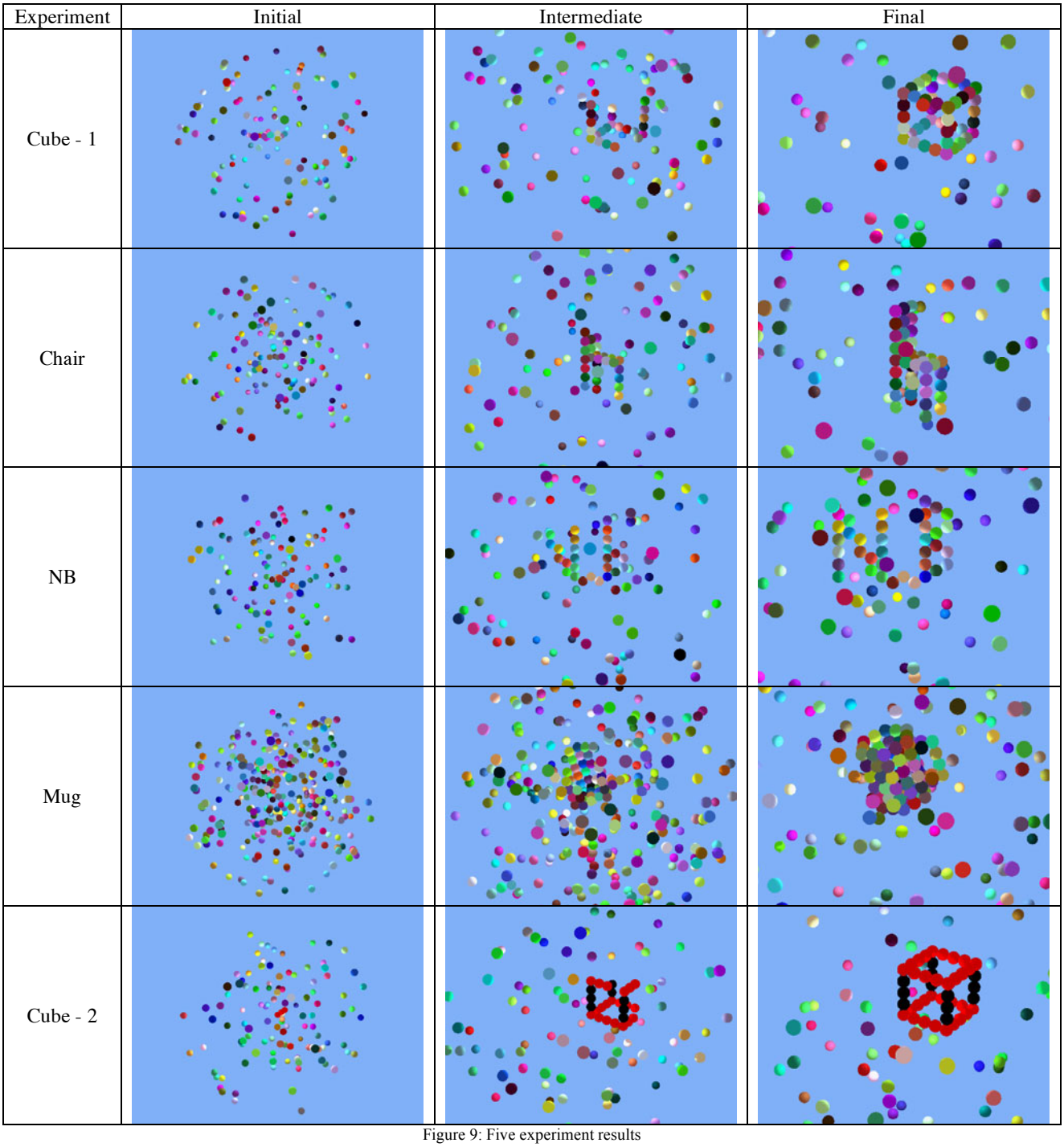

Figure 9: Five experiment results

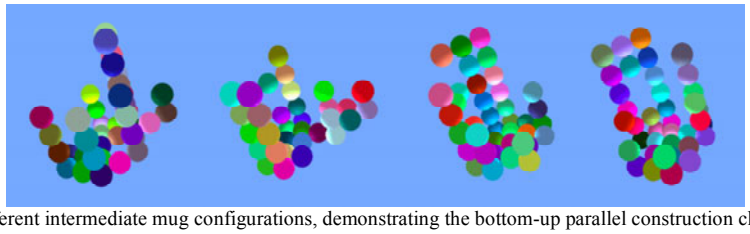

Figure 10: Four examples of different intermediate mug configurations, demonstrating the bottom-up parallel construction characteristic of self-assembly

The above observations were also present in the fifth experiment. In this experiment, the use of rules utilizing the axes count information reduced the number of rules needed to create the cube entity. This is an important result, because it shows that there is potential in creating systems that are scalable, as the number of components comprising the desired entity increases. Figure 9 also shows the results of this experiment (Cube – 2). Components present in the bottom and top two square elements change their respective colours to red, as an indication of the execution of rules associated with the global axes information. The components present in the vertical structures change their respective colours to black. This indicates the execution of rules associated with the axes count rules. Only three rules were needed to define the twelve components present in the vertical elements of the cube entity.

Research on exploiting parallel features in desired entities, as a method to reduce the number of rules required for scalability purposes, is ongoing by the authors. Since the planar surfaces present in each of the four desired entities is either very small or essentially non-existent, the spatial relationship between components could not be exploited as a method to reduce the number of rules. For example, a system utilizing a two-dimensional square grid spatial relationship in order to create a large two-dimensional filled square only needs rules to specify the outer perimeter or boundary of the desired filled square. The components in the interior of the square can self-assemble purely based on pattern formation. Research into exploiting the spatial relationship of a system is also being conducted by the authors, as another method to reduce the number of rules required.

The combination of compact rule sets, exploiting the spatial relationship of a system, and more advanced assembly protocols (e.g. swarm intelligence) are also being explored to create systems that can self-assemble in more predictable time frames. The desired entities in the five experiments here would each self-assemble approximately between forty and three hundred seconds. The focus in these experiments was not on construction time and instead on demonstrating that the self-assembly of entities with specific desired forms was feasible. However in the future, creating systems with predictable time frames is important in realizing new self-assembling technologies.

VII. CONCLUSIONS

It is important to note that in the system presented here, once a component attaches to the self-assembled aggregate structure, it is not capable of detaching. In other words, the system is not reversible. This is important, because it conflicts with the definition of self-assembly. However, such capabilities could be added to the system. For example, if a component collides with a component in the aggregate structure with enough force, it could dislodge that component. This is not incorporated into the current system for computational reasons. By not including such capabilities, it simplifies the number of calculations needed at each stage of a simulation, and eliminates the need for such algorithms. This system should be viewed as a proof of concept. Therefore, by not including such capabilities, it is felt that it does not compromise the results and the knowledge gained from these experiments.

In the future, this framework could be used to analyze and create both natural and synthetic self-assembling systems. Exploiting dynamic environments and assembly protocols could be used to create natural self-assembling systems using components with limited properties at the nanoscale and microscale. The framework could also be used to design new configurations of natural materials to create components with original properties, such as new DNA tile configurations for DNA computing.

As well, the framework could be used to design components inspired by various natural and artificial self-assembling systems to create innovative technologies. Here, the framework was leveraged to design a virtual system by melding the concepts of a seed particle to initiate the self-assembly process in artificial snow crystals, DNA serving as a shared internalized rule set, and data communication through specifically located communication channels currently used in modular and swarm robotics.

As a result, this virtual system was capable of creating self-assembled aggregate structures, displaying both symmetric and asymmetric properties. Furthermore, the experiments demonstrate the robustness of bottom-up parallel construction, displayed in nature, and how it can be achieved through self-assembly. As a corollary, the successful results of these experiments demonstrate how this framework can be leveraged to analyze and create self-assembling systems.

VIII. REFERENCES

[1] G.M. Whitesides and B. Grzybowski, "Self-Assembly at All Scales", Science, vol. 295, no. 5564, pp. 2418-2421, 2002.

[2] D. W. Thompson, *On Growth and Form*, Cambridge University Press, 1917.

[3] P. Ball, *The Self-made Tapestry: Pattern Formation in nature*, Oxford University Press, 1999.

[4] K.G. Libbercht, "Morphogenesis on Ice: The Physics of Snow Crystals", Engineering and Science LXIV, 1, 2001.

[5] L.S. Penrose and R. Penrose, "A Self-reproducing Analogue", Nature, 4571:1183, 1957.

[6] S. Grifth, D. Goldwater, and J. Jacobson, "Robotics: self-replication from random parts", nature, vol. 437, vol. 7059, pp. 636, 2005.

[7] P. White, V. Zykov, J. Bongard, and H. Lipson, "Three Dimensional Stochastic Reconfiguration of Modular Robots", Proc. of Robotics Science and Systems, 2005.

[8] R. Gross, M. Dorigo, and M. Yamakita, "Self-assembly of Mobile Robots: From Swarm-bot to Super-mechano Colony", In Proc. of the 9th Int. Conf. on Intelligent Autonomous Systesm (IAS-9), pp. 487-496, 2006.

[9] N. Bhalla and P.J. Bentley, "Working Towards Self-assembling Robots at All Scales", In Proc. Of the 3rd Int. Conf. on Autonomous Robots and Agents, pp. 617-622, 2006.

[10] T.A. Witten Jr. and L.M. Sander, "Diffusion-Limited Aggregation, a Kinetic Critical Problem", Phys. Rev. Lett. 47, 1400-1403, 1981.

[11] J. Klein, "BREVE: a 3D simulation environment for the simulation of decentralized systems and artificial life", In Artificial Life VIII, 8th Int. Conf. on the Simulation and Synthesis of Living Systems, 2002.