Self-Organizing Primitives for Automated Shape Composition

Linge Bai * Manolya Eyiyurekli † David E. Breen ‡

Department of Computer Science
Drexel University

ABSTRACT

Motivated by the ability of living cells to form into specific shapes and structures, we present a new approach to shape modeling based on self-organizing primitives whose behaviors are derived via genetic programming. The key concept of our approach is that local interactions between the primitives direct them to come together into a macroscopic shape. The interactions of the primitives, called Morphogenic Primitives (MP), are based on the chemotaxis-driven aggregation behaviors exhibited by actual living cells. Here, cells emit a chemical into their environment. Each cell responds to the stimulus by moving in the direction of the gradient of the cumulative chemical field detected at its surface. MPs, though, do not attempt to completely mimic the behavior of real cells. The chemical fields are explicitly defined as mathematical functions and are not necessarily physically accurate. The explicit mathematical form of the chemical field functions are derived via genetic programming (GP), an evolutionary computing process that evolves a population of functions. A fitness measure, based on the shape that emerges from the chemical-field-driven aggregation, determines which functions will be passed along to later generations. This paper describes the cell interactions of MPs and the GP-based method used to define the chemical field functions needed to produce user-specified shapes from simple aggregating primitives.

Keywords: Shape Composition, Morphogenesis, Chemotaxis, Genetic Programming, Self-organization, Emergent Behavior

Index Terms: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems; I.6.5 [Simulation and Modeling]: Model Development—Modeling methodologies

1 INTRODUCTION

In living things, cells aggregate and grow to create complicated structures. This process, called morphogenesis, is one of the fundamental components involved in the development of all complex organisms [15]. One of the essential processes involved in morphogenesis is chemotaxis [4]. Chemotaxis is the phenomenon where cells interact with other cells by emitting a chemical that diffuses into the surrounding environment. Neighboring cells detect the overall chemical concentration at their surfaces and respond to the chemical stimulus by moving either towards or away from the source [7]. The motions induced by chemotaxis may then produce patterns or sortings of cells [10], or even large-scale structures, e.g. cavities or vessels. These phenomena have motivated us to look to developmental biology for concepts that lead to a more organic, cell-biology-inspired approach to shape modeling.

*e-mail: lb353@cs.drexel.edu
†e-mail: me52@cs.drexel.edu
‡e-mail: david@cs.drexel.edu

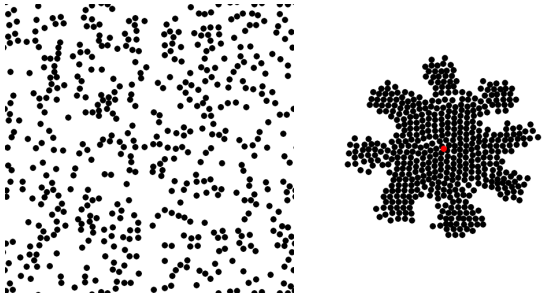

Figure 1: A set of randomly placed Morphogenic Primitives (left) aggregate to form a gear (right).

In this approach, shapes are composed from a collection of self-organizing primitives that we call morphogenic primitives (MPs). Macroscopic shapes are formed automatically by the aggregation of these simple primitives. A collection of MPs are first randomly placed in the modeling environment. The primitives emit a field, and then respond to the cumulative field by moving along its gradient. A macroscopic, user-defined shape then emerges from the combined actions of the individual primitives.

Several principles were followed when developing morphogenic primitives. 1) MPs are autonomous "agents". Each MP is an independent entity that senses the environment, responds to it, and then modifies the environment and its internal state. There is no "master designer" directing the actions/motions of the MPs. 2) Actions are based on local information. Each primitive emits a *finite* field that can be sensed only by other primitives within a certain range. The only information received by an MP is gathered at its surface, namely the concentration of the cumulative field and contact with immediate neighboring MPs. 3) MPs respond to information with prescribed behaviors. The actions performed by each MP are the same, but the specifics of the individual actions are based on information received from the environment. 4) MPs have no representation of the final, macroscopic shape to be produced. MPs do not use information about the final shape to determine what actions to take. Their actions are pre-determined by the ultimate shape to produce, but MPs do not carry or access information about the shape. The MP's final global position relative to the shape is not known ahead of time. 5) The shape emerges from the aggregation of local interactions and behaviors. Rather than follow a plan to produce the shape, MPs sense, change and respond to the cumulative field concentration. This simple behavior, when combined with somewhat complex individual fields, will direct the MPs to take individual actions based on local information that will ultimately aggregate to produce a user-defined, macroscopic shape.

A morphogenic primitive is represented by a small disk existing in a toroidal 2D environment. The environment is effectively infinite with no boundaries, since the top edge of our computational world is connected to the bottom edge, and the left edge is connected to the right edge. Each MP emits a "chemical" into the environment. An MP detects the cumulative field at eight receptors

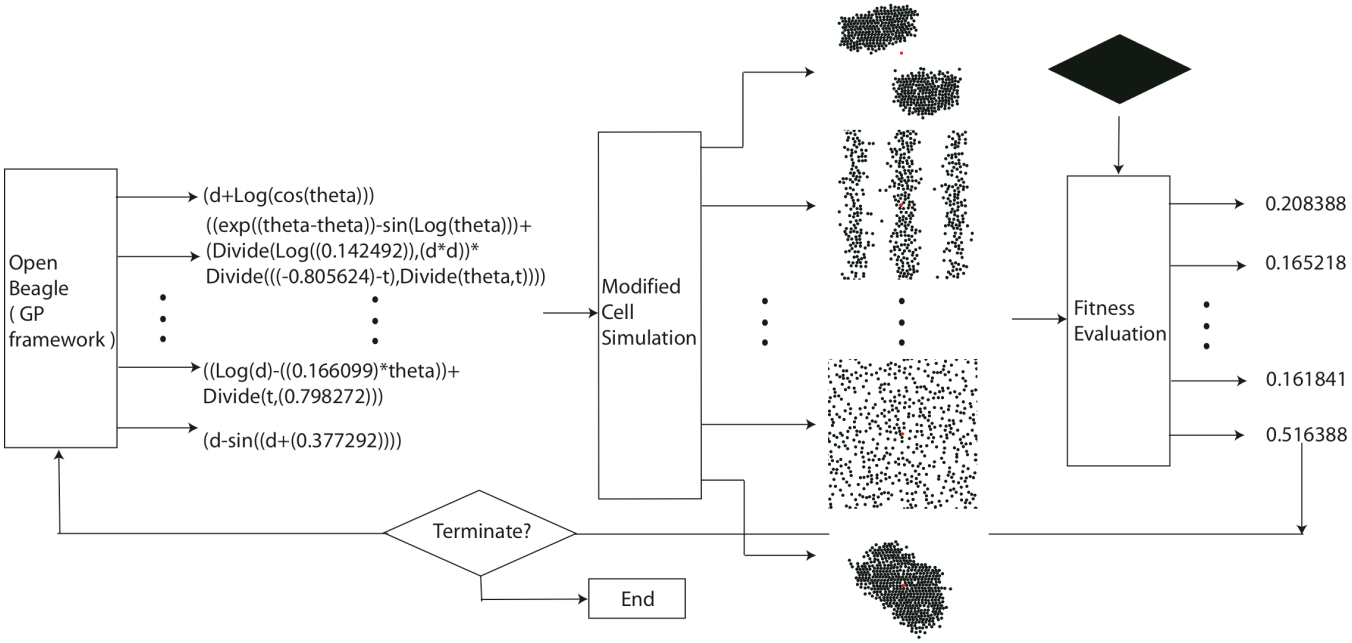

Figure 2: Overview of the genetic programming process that produces the behaviors of morphogenic primitives.

on its surface, and calculates the field gradient from this input. The gradient is used to determine the primitive's velocity. MPs move in the direction of the field gradient with a speed proportional to the magnitude of the gradient.

While MPs' fundamental interaction is based on chemotaxis, we do not limit their behaviors/properties to be physically realistic or completely consistent with biology. Instead, developmental biology provides a motivating starting point for MPs. As a way to customize chemotaxis-driven cells for shape composition, we alter the chemical concentration fields around individual cells. Instead of the chemical concentration dropping off as a function of distance (the physically accurate description), we define the concentration field with a mathematical function of distance, angle and time. Since it is extremely difficult to determine which particular local field function will direct MPs to form a specific macroscopic shape, we employ genetic programming [17] to produce the mathematical expression that explicitly specifies the field function. Genetic programming (GP) is an evolutionary computing process that evolves a population of functions. A fitness measure, based on the shape that emerges from the chemical-field-driven aggregation, determines which functions will be passed along to later generations. The genetic process stops once an individual (i.e. a mathematical expression) in the population provides the desired shape, or after a certain number of generations have been produced and evaluated.

The general approach to defining the field functions that lead to the interactions that ultimately produce the user-desired shape is presented in Figure 2. We start with a population of expressions, which is initially randomly generated. Each individual is compiled into a chemotaxis-based cell aggregation simulation, as the chemical field that surrounds the individual cells. A cell aggregation simulation is calculated for each field function, usually producing some kind of aggregated structure. Frequently the randomly generated field functions produce no discernible patterns after the aggregation simulation. The resulting structure is compared to the user-desired shape, and a scalar fitness value is calculated that quantifies how well the input shape matches the desired shape. A subset of the top candidates are then used to create the next generation of field functions. The process continues until a field function produces the desired shape or the maximum number of generations is reached.

Once the local field function has been identified for a specific shape, MPs may be randomly placed in the computing environment, with each MP surrounded by the GP-produced concentration field. A simulation is performed where the cumulative field is computed, and each MP moves along its gradient, until the MP population reaches an equilibrium, which is the desired shape.

We have utilized this new approach to define morphogenic primitives that aggregate to form a number of user-defined shapes, e.g. an ellipse, a diamond, a boomerang, and an hourglass. In the process of evolving the field functions for user-defined shapes, a few pleasant surprises materialized. Most of them included repeated patterns, e.g. stripes, spots and sine waves, but most interestingly a gear shape emerged from the evolutionary process. This approach can be extended for and applied to the control of robotic swarms, motion specification of animated crowds, generative model creation, and even the computer-aided design of real cell aggregates.

2 Related Work

Sims [20] describes one of the first applications of evolutionary techniques to computer graphics. Here, he used genetic programming to create functional representations of intricate, interesting images and solid textures. In this work, an interactive process allows a user to guide the evolution of the mathematical expressions by the selection of preferred images and results. The repeated interaction between the user and the evolutionary process leads to the mathematical expressions which define a large number of surprising and appealing images. This approach was extended for the creation of procedural models [21] and the definition of motions and behaviors of virtual creatures [22].

Fleischer explored a cell-based developmental model for self-organizing geometric structures [11, 12]. He applied his cell interaction simulation system to computer graphics to produce an approach to cellular texture generation [13]. Eggenberger Hotz proposed the use of genetic regulatory networks coupled with developmental processes for use in artificial evolution and was able to

evolve simple shapes [5, 16]. The combination of artificial evolutionary techniques and developmental processes provides a comprehensive framework for the analysis of evolutionary shape creation.

Theraulaz and Bonabeau present a modeling approach based on the swarming behavior of social insects [23, 24], a type of swarm intelligence [2]. They combine swarm techniques with 3D cellular automata to create autonomous agents that indirectly interact in order to create complex 3D structures. This indirect interaction, known as stigmergy [25], allows the agents to act cooperatively, but independently, through a stimulus-response mechanism based on modifications made to the environment. Bonabeau et al. [3] apply genetic algorithms (GA) to the stigmergic swarm-based 3D construction method in order to improve the overall process. A fitness function, chosen by human observers, is assigned to each pattern in this approach, and a GA is used to search the space of all possible patterns.

Nagpal et al. present techniques to achieve programmable self-assembly [18, 19]. Cells are identically-programmed units which are randomly distributed and communicate with each other within a local area. In this approach, global to local compilation is used to generate the program executed by each cell, which has specialized initial parameters. This work has been extended and applied to collective construction based on a swarm of autonomous robots and extended stigmergy [26].

Our work is similar to the previous work in that 1) we make use of a chemotaxis-based cell aggregation model; 2) we apply evolutionary computing for shape modeling; and 3) we develop an approach for multi-agent shape composition. However, our work is conceptually novel. Sims' and Fleischer's work cannot automatically generate user-defined images and shapes. Nagpal's work requires special global initializations and cells states in order to differentiate the cells. Our work is different from previous work in the following ways. 1) All MP primitives are randomly placed in the environment, and are identical and equally important. They require no special seed primitives or customized initialized states. 2) No initialization of spatial information is needed in the computational environment. We utilize a toroidal environment where all positions are considered identical. This is different from the sheet environment [18], which requires a global initialization. 3) MPs cooperate with each other only through local interactions and only with local information. Individual primitives do not know their location in any external/global coordinate system and have no knowledge of the predefined global shape that is being composed. 4) Genetic programming is used to explore the space of concentration field functions that surround the individual primitives. Automatic fitness evaluation is implemented in our method, which evaluates how similar an aggregate is to the final shape and produces a scalar fitness value. Genetic programming uses this fitness value to generate a field function that directs the individual chemotaxis-based MPs to form into a user-specified shape.

3 Cell Aggregation Model

MPs are based on a computational model that is capable of simulating chemotaxis-based cell aggregation in 2-D [8, 9]. Our 2-D chemotaxis-based cell aggregation model incorporates fundamental parameters involved in cell-cell aggregation, such as a cell's ability to secrete and detect chemoattractant chemicals, cell motility, proliferation, aggregation and various stages of a cell's life cycle. Using appropriate approximations / assumptions with efficient and effective algorithms, our model creates a robust and extensible simulation environment, which in turn has enabled us to study and identify the critical components of cell interaction that most influence aggregation outcomes.

In our simulations each cell is defined by a collection of physiologically relevant parameters and actions, such as the number and position of chemical receptors on the cell surface, location of the cell, age, life cycle stage, chemoattractant secretion and response rates, diffusion radius, proliferation rate and number of attached cells. Our virtual cells are able to emit chemoattractants, sense the chemoattractant gradient, move in the direction of the gradient, proliferate, adhere to other cells, age and die.

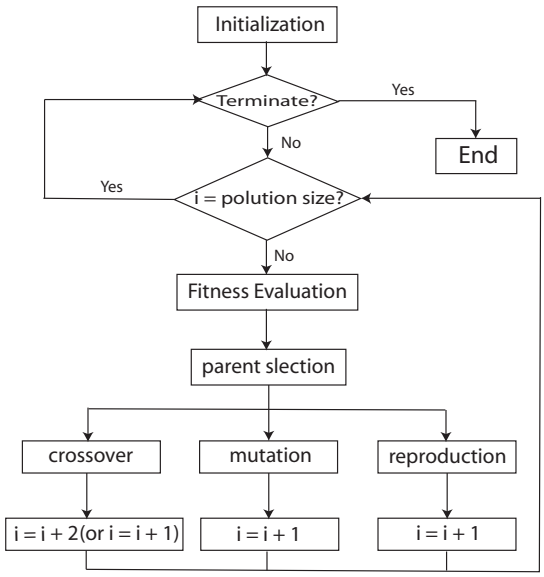

Figure 3: Genetic programming flowchart

The model's parameters were refined and simulation results were validated by comparisons with *in vitro* cell-cell aggregation data obtained from an experimental study, in which PC12 pheochromocytoma cells were allowed to aggregate over a 24-hour period. Samples were imaged at the beginning and the end of the experiments and the numbers of individual and aggregated cells were determined. Using the experimentally-derived aggregate size distributions as an outcomes measure, our optimized computational model is capable of reproducing PC12 aggregation behavior. This was accomplished by systematically modifying the parameters of the model, performing 24-hour aggregation simulations, and comparing the resulting *in silico* aggregate size distributions with the *in vitro* distributions.

4 Genetic Programming

Genetic programming (GP) is an automated approach to software and function development based on genetics, evolution and natural selection, which evolves a computer program or mathematical function based on its ability to perform a specified computational task, as evaluated by a fitness function [6, 17]. Genetic programming and other evolutionary computing methods, such as genetic algorithms, evolutionary programming and evolutionary strategies, make up the field of evolutionary computation [17]. The underlying concept of evolutionary computation is: given a population of individuals and provided an evolutionary process of variation based on the fitness of the individuals, produce a new population with higher fitness values than the previous population.

The evolutionary computation approach to a certain problem consists of the following components: individuals, fitness evaluation, selection and variation, initialization, a termination condition and control parameters. An individual is a representation in the form of a solution to the problem. For example, when GP is used to automatically generate computer programs, each individual of a GP generation is a parse tree that holds a computer program. Therefore, a population of individuals, which are going to evolve in the

evolutionary process, represents a number of candidate solutions to the problem. The fitness function defines how close a candidate solution is to the optimal solution. A threshold value for this function defines the termination condition. In other words, fitness evaluation defines a similarity metric between an individual outcome and the final goal. The more similar the two are, the better (more fit) the individual is considered to be. Fitness evaluation is crucial to evolutionary computing because it directs the entire process by providing the basis for selection.

There are two kinds of selections: parent selection and survival selection. Parent selection or mating selection provides a higher probability that the higher fitness individuals will become parents in the next generation. While survival selection is a replacement strategy that swaps newly-generated individuals with older ones in order to increase the overall fitness of the new generation [6]. Selection is responsible for the quality improvement of the population. Variation, which includes recombination and mutation procedures, is the mechanism that creates new individuals from the existing ones [6]. Also, variations can be viewed as the process of reproduction, crossover and mutation [17].

Genetic programming (GP), as outlined in Figure 3, is different from other types of evolutionary computing algorithms in that the individuals are parse trees consisting of elements from a Function Set F and a Terminal Set T. The ramped half-and-half method (one of the most common) is employed for initialization [6]. It utilizes the parameter D_{Max}, the maximum depth of the parse tree. The initial generation is generated by two methods with equal probability. 1) Full method: each branch of the tree has the same depth up to D_{Max}. 2) Grow method: the tree is constructed from a root with tree nodes chosen randomly from both the F and T sets as long as the depth of the tree does not exceed D_{Max}. The most common termination criteria either specify the maximum number of generations or the fitness threshold value. While the fitness evaluation function is specific to the particular problem, the process of variation can be abstracted from the problem domain. Crossover in GP is performed by swapping subtrees between two parents (in the most common case). Two important parameters of crossover are the probability of crossover versus mutation, and the probability of choosing an inner tree node as a crossover point. Mutation is mostly implemented by replacing a subtree of the selected individual by a randomly generated subtree, while the depth of the new individual does not exceed D_{Max}. Two important parameters of mutation are the probability of mutation versus crossover, and the probability of choosing an inner tree node as the root of the subtree to be replaced. Reproduction simply copies the parent individual so it become its offspring. Finally, during selection GP usually uses a fitness proportional method as parent selection, and uses a generational strategy as replacement with no elitism. This means that the life span of each individual is one generation and the number of offspring is equal to the population size.

5 Morphogenic Primitive Details

5.1 MP Behaviors and Simulation

The full complexity of our original chemotaxis-based cell aggregation model was not needed for the initial implementation of morphogenic primitives. The model was simplified and we focused the evolutionary process on one aspect of the possible cell interactions, the definition of the chemoattractant chemical field surrounding a single cell. Therefore several features in the cell model were disabled: aging, dying, proliferation, differentiation and attachment. MPs do not age, do not die, do not develop into different types of cells and the number of total cells remains constant.

In the original model cells would attach to each other when they collided. The cell-cell binding sites on the surface of the cells formed permanent links, that once made could never be broken. A single velocity was assigned to each aggregate (collection of attached cells) and the aggregate moves in the direction of the average gradient detected at the surfaces of all the aggregate's cells. This feature of the original model interfered with, rather than contributed to, the process of self-organization by limiting the interactions between MPs. Therefore, the attachment feature of the original model was disabled, which allows the cells to slide over one another, rather than forming fixed links.

Finally, we modified the cell aggregation simulation system to allow for a general functional description of chemical fields. Since we define the chemical fields as mathematical functions, we utilize a number of protected operators, such as protected division and protected logarithm. These protected operators safely handle invalid input values, for example divide by zero and negative logarithms. It should also be noted that the field function is truncated at a fixed distance ($R_{Max} = 200$ units) in order to keep the MP interactions finite and local.

Once the process has been initialized by randomly placing a number of MPs (500 for our examples) in the computational environment, a single aggregation simulation is comprised of a series of time steps. For each time step, each MP performs a prescribed set of actions. Each MP emits a "chemical" into the environment. It then detects the cumulative field at eight receptors on its surface, and calculates the field gradient from this input. The gradient is used to determine the primitive's velocity. We assume that MPs travel at a terminal velocity through a viscous fluid environment, therefore an MP's velocity is directly proportional to the chemical field gradient (∇C). When an MP moves in the direction of the chemical gradient, its velocity is calculated as

$$\textbf{Velocity} = \lambda * \nabla C, \tag{1}$$

where λ (1 for our examples) is a constant that determines the magnitude of an MP's response to the gradient. At each simulation time step (Δt) the displacement of the MP is

$$\Delta x = \textbf{Velocity} * \Delta t. \tag{2}$$

If the displacement makes the MP collide with another MP, a small random step is taken instead. MPs do not always follow the field gradient. 10% of the time MPs take a random small step instead of following the field gradient. This randomness injects a small amount of noise into the system, helping to prevent the collection of MPs from being stuck in local minima configurations.

5.2 Evolution via GP

5.2.1 Evolutionary Process

The genetic programming process for generating chemical field functions has been implemented in Open Beagle [14], a C++ evolutionary computing framework. The details of the morphogenic primitive genetic programming process, as illustrated in Figure 2, are the following. The ramped half-and-half method is used to generate the initial population of functions. These individuals, which are mathematical expressions, are then used as input to the modified cell aggregation simulation system. Each modified cell aggregation model employs one expression from the population as the chemical field function. A cell aggregation simulation is performed for each field function, producing a visual output of the aggregated result that emerges from the local interactions defined by the function.

Each aggregated result is then evaluated by the fitness function, which compares the aggregate image with a target image. A fitness value (a scalar between 0 and 1) is assigned to each individual that quantifies how closely the resulting aggregate matches the target shape. The fitness values are returned to the GP framework. Parent selection is then performed on the population based on the fitness value and variation takes place. Generational selection is used for survival selection, which means that the offspring replaces the parents from the previous generation. The GP process repeats until

Example	Function	Gen
ellipse	Divide((Log(Divide(d,t))-Log(cos(theta))),Divide(d,t))	7th
diamond	Divide(((Log(Divide(d,t))-Divide((0.924414),theta))-(0.363563)),Divide(d,t))	11th
hourglass	Log(d)+cos(Divide((t*theta),Divide((t+Log(Divide(t,Log(d)))),(-0.356662))))	18th
boomerang	Log((d*exp(d)))	13th

Table 1: Field functions for several shapes and the number of GP generations needed to derive them.

the termination criteria are met, either after generating a maximum number of generations or the fitness of an individual surpasses the threshold, i.e. the associated aggregate is approximately the same shape as the target.

5.2.2 GP Components

The following is a detailed description of the components in our GP process.

- **Individuals:** Each individual of the GP process is represented as a prefix tree with a maximum depth D_{Max}. The domain of leaf nodes is the terminal set and the domain of internal nodes is the function set.

- **Function and terminal set:**
  $F = \{+, -, *, /, exp, log, sin, cos\}$
  $T = \{E, d, t, \theta\}$
  In order to maintain the closure property of the GP process [6, 17], we use protected division and protected logarithm, which allows for the most general description of the mathematical expressions. Here, $E \in \Re$, d is the distance between two MPs, t is the simulation time and θ is the angle between two MPs measured in the local coordinate system of each MP.

- **Selection:** Parent selection is proportional to the fitness value and uses tournament selection, which randomly chooses k individuals out of the population and returns the one with the best fit. Generational replacement is used for survival selection.

- **Variation:** A swap subtree crossover operation and standard mutation, i.e. replacing a subtree with a stochastically generated tree are used in the variation process.

- **Fitness evaluation:** We compare an image of the resulting aggregate with an image of the target shape. The comparison produces a fitness value, a floating-point number between 0 and 1, with 1 being the maximum fitness (similarity) value.

Parameters used to control the GP process are:

- **population size:** 100,

- **maximum number of generation:** 50,

- **crossover probability:** $P_c = 0.9$, which means that 90% of the population of each generation is produced via a crossover operation,

- **mutation probability:** $P_m = 0.1$, which means that 10% of the population of each generation is produced via a mutation operation,

- $\mathbf{D_{Max}}$: 17 (maximum parse tree depth),

- **k**: 7 (tournament selection size).

5.2.3 Fitness Function

The fitness function defines the similarity between two shapes, the aggregate generated by the MPs under the influence of the chemical field function and the predefined target shape. Since the fitness function is based on comparing images, it is necessary to keep the total area of the MPs (the area of one MP times the total number of MPs) approximately equal to the area of the desired shape. Since MPs are not aware of a global coordinate system, we do not want the shape of the resulting aggregate to be tied to a specific fixed orientation. We therefore align the aggregated shape with the target shape before applying the fitness function. This is accomplished by calculating the centroid and major axis of the aggregated and target shapes. The appropriate translation and rotation is applied to a candidate MP which aligns its centroid and major axis with those of the target. This allows MPs to aggregate into the desired shape, but not necessarily in the same position and orientation of the target.

Once the alignment is completed, the fitness function calculates the ratio of the overlapping pixels of the aggregate shape and the target shape to the total pixels of the target shape.

$$f = \frac{\sum p_i, p_i \in S_a \cap S_t}{\sum p_j, p_j \in S_t}, \tag{3}$$

where f is the fitness of the individual, p_i are the shape pixels that are present in both the aggregate and target image, p_j are the shape pixels in the target image, and S_a and S_t are the sets of black pixels in the aggregate and target shape images. The floating-point number f (between 0 and 1) is assigned as the fitness of the chemical field function used to generate the aggregate shape.

6 Results

We have derived several chemical field functions that direct morphogenic primitives to aggregate into a number of predefined shapes. The cell simulations that led to the definition of the field functions consisted of 500 primitives, where each primitive has a radius of 4.5 or 5 (depending on which shape we were trying to make) units[1]. The MPs move in a toroidal computational environment of 500×500 units. Each simulation is run for 1000 time steps. The target shapes given to the MP framework include an ellipse (Figure 6 (left)), a diamond (Figure 7 (left)), an hourglass (Figure 8 (left)), a boomerang (Figure 8 (right middle)), a wave-like structure (Figure 9 (left)), and a flattened annulus (Figure 9 (right middle)). The best resulting MP-generated shapes are placed next to the target shapes in these figures. The first four shapes were the most successful, creating aggregates that reasonably resemble the target shapes. The field functions that produce these shapes are listed in Table 1, along with the number of generations needed to derive them. Figures 4 and 5 present snapshots from MP simulations. Both of these sequences began with the MPs being randomly placed in the computational environment, similar to Figure 1 (left).

Because of the stochastic nature of the field function generation process, we were not able to recreate all input shapes. The last two examples did not completely converge on the desired shape. The wave and annulus shapes are stand-alone objects. Recall that

[1]The micron was the unit of length in our original PC12 cell simulations.

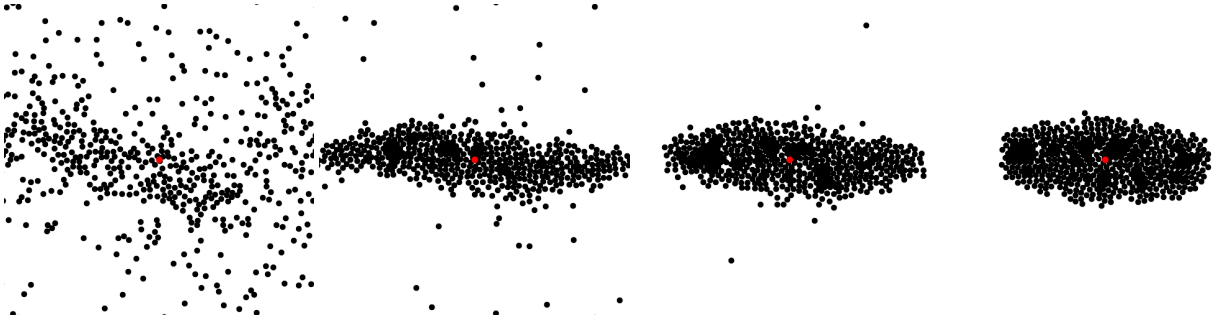

Figure 4: MPs self-organizing into an ellipse.

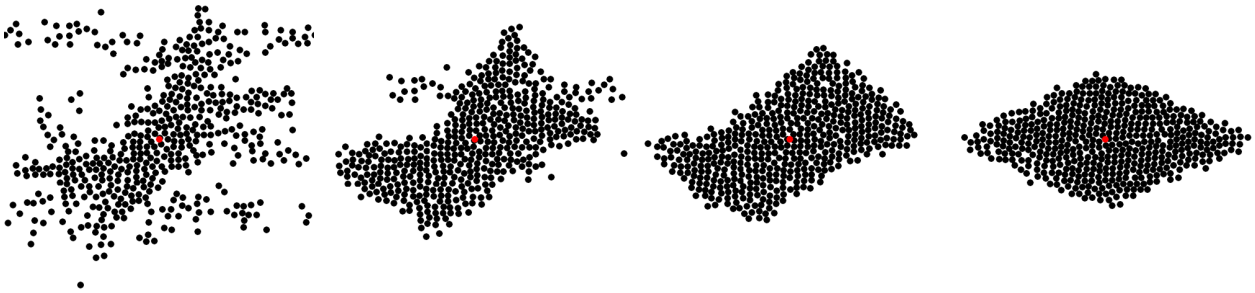

Figure 5: MPs self-organizing into a diamond.

MPs exist in a toroidal environment. Therefore the shapes that we generated are continuous objects, where their left edge is connected to their right edge.

There were a number of pleasant surprises produced during the evolution process. These are field functions that direct MPs to produce unwanted, but still quite interesting, shapes and patterns. A small set of these surprise results are included in Figure 10. The most remarkable of these "unwanted" shapes is the gear shape in Figure 1.

The computation time needed to implement the genetic process that defines the chemical field functions is quite substantial. Each 1000-time-step MP simulation requires approximately 4-CPU minutes on 1.8 GHz Opteron processor. We perform distributed GP calculations on an 8-node Linux cluster [1]. Therefore each 100-individual generation requires approximately one hour of actual time to compute on our cluster. Since the GP process usually produces the desired function by the 20th generation, we can derive a shape-specific field function in approximately one day. Of course, once we have the correct function it only requires approximately 4 CPU-minutes to simulate the desired aggregation behavior.

7 Conclusion

We have presented a new approach to shape modeling based on self-organizing primitives whose behaviors are derived via genetic programming. The key concept of our approach is that local interactions between the primitives direct them to come together into a macroscopic shape. The interactions of the primitives, called Morphogenic Primitives (MP), are based on the chemotaxis-driven aggregation behaviors exhibited by actual living cells. MPs do not attempt to completely mimic the behavior of real cells. The chemical fields that direct MPs are explicitly defined as mathematical functions and are not necessarily physically accurate. The mathematical functions are derived via genetic programming (GP), an evolutionary computing process that evolves a population of functions. We have shown that MPs may be used to define field functions that produce a number of simple shapes. The GP-based process has also generated field functions that produce a number of unexpected, but interesting, patterns and shapes.

In the future, we will study the robustness, scalability and self-repairability of morphogenic primitives. In the long term, we would like to develop more complex cell behaviors, as well as extend the cell model to three dimensions.

Acknowledgments. We would like to thank Dr. Peter Lelkes for his contributions to the cell aggregation simulation research that our current work builds upon. This work was funded by NSF grant # CCF-0636323.

References

[1] L. Bai, M. Eyiyurekli, and D. Breen. Automated shape composition based on cell biology and distributed genetic programming. In *Proc. Genetic and Evolutionary Computation Conference*, 2008.

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[3] E. Bonabeau, S. Guerin, D. Snyers, P. Kuntz, and G. Theraulaz. Three-dimensional architectures grown by simple stigmergic agents. *Biosystems*, 56(1):13–32, 2000.

[4] J. Davies. *Mechanisms of Morphogenesis*. Elsevier, Amsterdam, 2005.

[5] P. Eggenberger. Evolving morphologies of simulated 3D organisms based on differential gene expression. In *Proc. 4th European Conference on Artificial Life*, pages 205–213, 1997.

[6] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[7] E. Eisenbach et al. *Chemotaxis*. Imperial College Press, London, 2004.

[8] M. Eyiyurekli. A Computational Model of Chemotaxis-based Cell Aggregation. Master's thesis, Drexel University, 2006.

Figure 6: Ellipse: (left) Target shape. (right) Self-organized MPs.

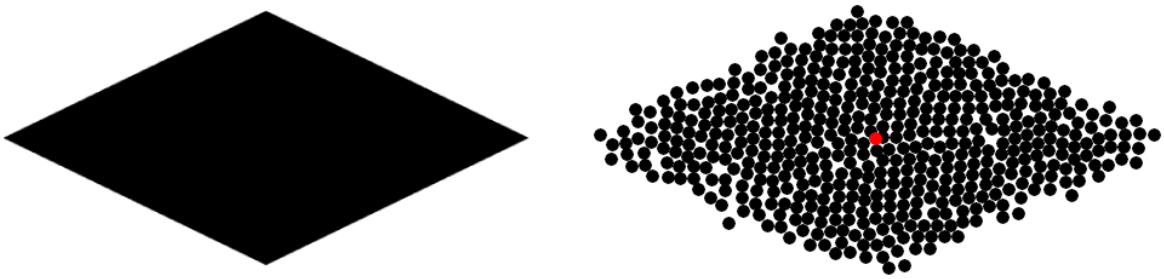

Figure 7: Diamond: (left) Target shape. (right) Self-organized MPs.

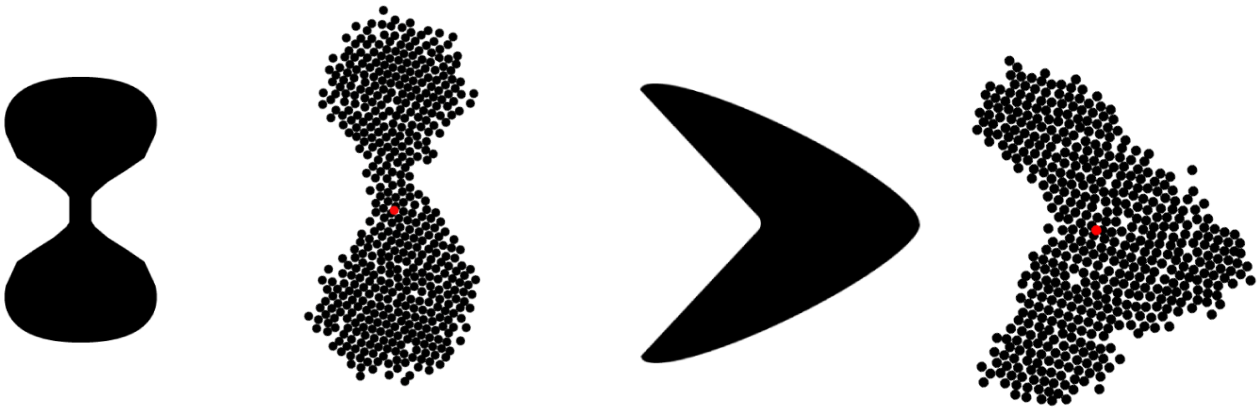

Figure 8: Hourglass: (left) Target shape. (left middle) Self-organized MPs. Boomerang: (right middle) Target shape. (right) Self-organized MPs.

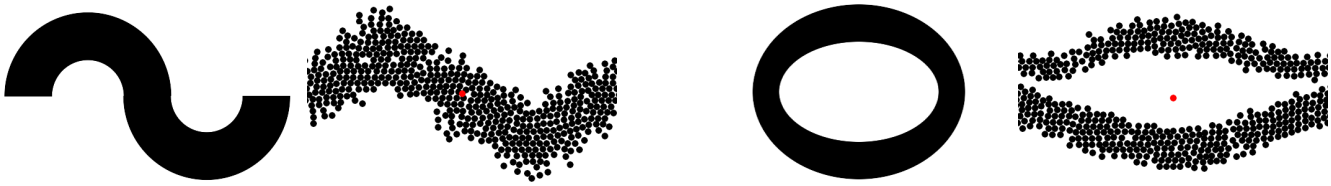

Figure 9: Approximately correct wave and annulus shapes produced by MPs.

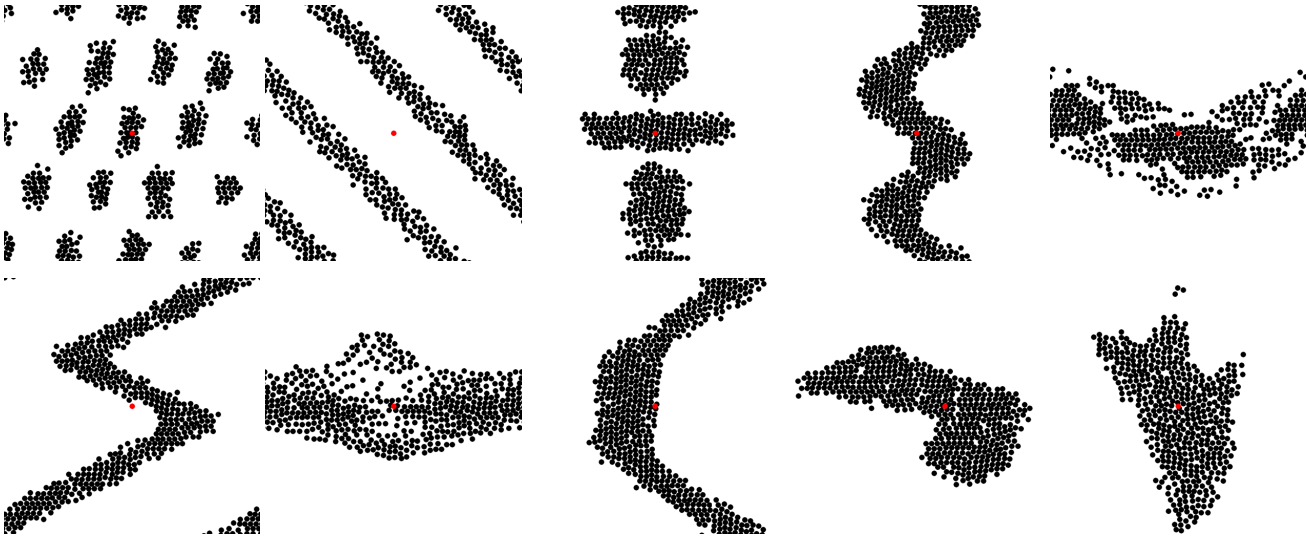

Figure 10: Unexpected patterns and shapes produced during the GP process.

[9] M. Eyiyurekli, P. Lelkes, and D. Breen. A computational system for investigating chemotaxis-based cell aggregation. In *Proc. European Conference on Artificial Life*, pages 1034–1049, 2007.

[10] M. Eyiyurekli, P. Lelkes, and D. Breen. Simulation of chemotaxis-based sorting of heterotypic cell populations. In *Proc. IEEE / NIH BISTI Life Science Systems & Applications Workshop*, pages 47–50, 2007.

[11] K. Fleischer. *A Multiple-Mechanism Developmental Model for Defining Self-Organizing Geometric Structures*. PhD thesis, California Institute of Technology, 1995.

[12] K. Fleischer and A. Barr. A simulation testbed for the study of multicellular development: The multiple mechanisms of morphogenesis. In *Artificial Life III*, pages 389–408. 1994.

[13] K. Fleischer, D. Laidlaw, B. Currin, and A. Barr. Cellular texture generation. In *Proc. SIGGRAPH*, pages 239–248, 1995.

[14] C. Gagné and M. Parizeau. Genericity in evolutionary computation software tools: Principles and case-study. *International Journal on Artificial Intelligence Tools*, 15(2):173–194, 2006.

[15] S. Gilbert. *Developmental Biology*. Sinauer Associates, Inc., Sunderland, MA, 8th edition, 2006.

[16] P. Hotz. Combining developmental processes and their physics in an artificial evolutionary system to evolve shapes. In S. Kumar and P. Bentley, editors, *On Growth, Form and Computers*, pages 302–318. Academic Press, 2003.

[17] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[18] R. Nagpal. Programmable self-assembly using biologically-inspired multiagent control. In *Proc. 1st International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1*, pages 418–425, 2002.

[19] R. Nagpal, A. Kondacs, and C. Chang. Programming methodology for biologically-inspired self-assembling systems. In *Proc. AAAI Spring Symposium on Computational Synthesis: From Basic Building Blocks to High Level Functionality*, pages 173–180, 2003.

[20] K. Sims. Artificial evolution for computer graphics. In *Proc. SIGGRAPH*, pages 319–328, 1991.

[21] K. Sims. Interactive evolution of equations for procedural models. *The Visual Computer*, 9(8):466–476, 1993.

[22] K. Sims. Evolving virtual creatures. In *Proc. SIGGRAPH*, pages 15–22, 1994.

[23] G. Theraulaz and E. Bonabeau. Coordination in distributed building. *Nature*, 269:686–688, 1995.

[24] G. Theraulaz and E. Bonabeau. Modeling the collective building of complex architectures in social insects with lattice swarms. *Journal of Theoretical Biology*, 177:381–400, 1995.

[25] G. Theraulaz and E. Bonabeau. A brief history of stigmergy. *Artificial Life*, 5:97–116, 1999.

[26] J. Werfel and R. Nagpal. Extended stigmergy in collective construction. *IEEE Intelligent Systems*, 21(2):20–28, 2006.