Developmental Modelling with SDS

Benjamin Porter and Jon McCormack

Centre for Electronic Media Art
Monash University, Clayton 3800
Victoria, Australia

Abstract

This paper describes modelling methods based on biological development for use in computer graphics applications, specifically the automated growth and development of complex organic shapes that are difficult to model directly. We examine previous approaches, including grammar-based methods, embedded systems and cellular models. Each of these system can be classified as endogenous (internally determined) or exogenous (externally determined), with some models exhibiting features of both. We then introduce a new model, the Simplicial Developmental System (SDS), which simulates individual cells embedded in a physical environment, with cell division, movement and growth controlled by morphogenetic chemical simulation. SDS uses a tetrahedral mesh as its base representation for geometric modelling and physical simulation. Cell growth, movement and division is determined by simulating chemical morphogens that are diffused between cells according to a set of user defined rules. We discuss the advantages and disadvantages of this model in terms of the competing goals of user control, developmental complexity and open-ended development (the ability to generate new component structures without explicit specification). Examples highlighting the strengths of the model are illustrated.

Keywords: developmental model, morphogenesis, physical simulation, tetrahedral mesh

1. Introduction

One of the greatest challenges in biology is understanding the mechanisms that enable a single, fertilised cell to develop into a complex, multi-cellular organism. The developmental processes that lead to interacting functional components and self-organising, heterogeneous structures are complex and numerous, remaining the subject of on-going research. In this paper we focus on systems, inspired by biological processes of development, that begin with concise specifications for initial conditions and developmental rules, then proceed to grow and develop autonomously in simulation. Our application is in computer graphics, so we concentrate on the process of growing three-dimensional form rather than a complete simulation of biological function. Our goal is to define systems that allow the specification of complex organic shape and form, removing the tedium of trying to assemble such forms manually.

The process of simulating development in this more general context is referred to as *developmental modelling*. As a modelling methodology, developmental models[1] offer a vastly different modelling paradigm over commonly used methods in computer graphics. ‘Traditional’ modelling focuses on processes organised around the design of artefacts: the artist conceptualises a form, and then proceeds towards the goal of building the form explicitly. This is normally achieved through the direct instantiation and manipulation of geometry and texture to achieve the finished result. Procedural modelling uses the specification of procedures that automate the geometry and texture building process. *Generative models* exploit the generative properties of computation to achieve *database amplification* [2], where complex structures are built from relatively simpler (often several degrees of magnitude) specification. In this sense, developmental models are a subset of generative models.

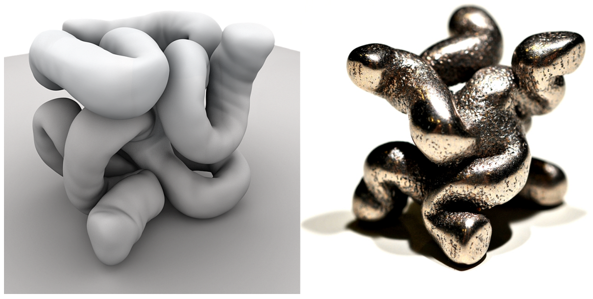
Figure 1: (left) An abstract organic form generated using the system introduced in this paper. (right) A related form manufactured in stainless steel (by `Shapeways` [1]).

There have been two complementary approaches to modelling development in computer graphics. In *endogenous* systems, development originates from, and is driven by, *internal*,

[1]The term ‘model’ has a dual, context dependent meaning in this paper: in a developmental context it means a formal mathematical or procedural specification for a system; in a computer graphics context it means a spatiotemporal geometric structure generated for image synthesis and animation purposes (e.g. a time-dependent geometric mesh). We anticipate the reader will distinguish the meaning based on context.

rule-based mechanisms. Typically, the rules are specified by the user of the system. Endogenous models work well for the procedural specification of complex, static or time-varying geometry, however there are limitations in how models generated by the system can be affected by, and interact with, their environment. For a system to develop this way – an *exogenous* system – the form and space in which it develops must be coupled.

In this paper we will first review some of the literature for both endogenous and exogenous methods, with a brief discussion on the advantages and limitations of each approach (§2). In §3 we introduce the *Simplicial Developmental System* (SDS), a tetrahedral mesh based system that supports exogenous growth through the physical simulation of a developing 3D form (e.g. Figure 1). Finally, §4 discusses implications and further development of the SDS.

2. Modelling Development

There are many established systems in computer graphics that incorporate aspects of biological development for modelling shape. The fields of theoretical biology and artificial embryology also contain models that are useful when considering shape formation from a developmental perspective. Surveys that cover these models in the literature are numerous [3, 4, 5, 6, 7, 8, 9, 10, 11] and hence only a brief overview is given here, with discussion pertinent to the model introduced later in the paper.

2.1. Grammar-based Approaches

L-systems, introduced by Lindenmayer [12] to model the development of multicellular organisms, have been extensively used to simulate the development of trees, herbaceous plants, and many other biological structures. Early L-system models were endogenous, operating at a symbolic level, with component parts and their development specified using a set of distinct symbols, and the development simulated by parallel re-writing of those symbols. As development proceeds the set of active symbols changes in discrete time steps. Symbols representing component parts are converted to geometry as a unidirectional, post-development operation, i.e., no information from the environment or built geometry flows back to the symbols to affect their development.

Researchers recognised these limitations, over subsequent years devising enhancements and additional mechanisms to circumvent them. The primary goal in computer graphics applications has been the synthesis of visually realistic models, with an emphasis on the final, rather than developing, geometric form. L-system extensions include: continuous mechanisms [6, 13]; the modelling of physical and mechanical effects [14, 15]; specifying explicit hierarchy [16] and coupling the developmental process more closely to the environment [17]. These extensions address specific issues, a number incorporating exogenous development, but they diminish the simplicity and elegance of the original formalism.

In general, L-systems simulate growth and development at an abstracted macro level (e.g. florets, meristems, leaves, branching segments), due to the complexity of defining interactions at the micro level (cells, molecules). The appropriate choice of macro-level abstraction relies on a structural understanding of the entity being modelled, along with a means of inferring developmental relationships at the level chosen. In general, this is an ill-defined and ad-hoc process, making it difficult for computer graphics users without significant experience to devise grammars that generate the desired form.

L-systems abstract the concept of development to that of *replacement of parts by other parts.* In this respect they are related to other grammar-based approaches such as shape grammars [18], graph grammars and graph-based representations [19]. They have also been generalised to grammars that operate on polygonal surface representations [20, 21, 22]. These methods address the topologically-focused bias of tree L-systems and provide generative methods for more complex geometric surfaces. Developments have enabled the application of grammar-based methods to a broader class of shape and form, such as architectural design [23] and legged animals [24].

2.2. Embedded Models

Many systems directly *embed* development in a spatial environment, an idea that dates back many years in graphics [25, 26]. A structure that is being generated within a space can be affected by its own parts, other static and dynamic objects, and environmental factors such as light and gravity. These interactions provide a mechanism for directing the growth of specific forms, and also result in an emergent complexity difficult to obtain using non-embedded systems. While embedded systems appear to be a natural and biologically realistic approach (after all, all real biological growth is embedded in a physical environment), taking full advantage of this embedded approach presents a number of challenges [27], particularly if the open-ended complexity of real biology is sought.

Cellular automata models demonstrate that exogenous growth factors, including environmental effects and spatial limitations, can contribute greatly to the complexity of a developing form [26, 28, 29, 30, 31, 32, 33]. Even if the mechanism behind the development is simple, environmental interaction can result in complex creations that far exceed the simple specification from which they emerge [34, 35].

A developmental model of accretive growth that illustrates the combination of a geometric surface-based developmental model with a physical model of nutrients and hydrodynamics is presented by Kaandorp and Kübler [36, 37]. These experiments reinforce the notion that a simple growth logic combined with a physical model can result in complex organic forms. This is further exemplified by Combaz and Neyret, who demonstrated a system that generates rich and abstract organic form through physical simulation and growth [38]. The user paints growth chemicals onto a surface, which causes that part of the surface to expand and grow, resulting in naturally wrinkled surfaces due to growth-induced deformation. Similar systems in computational biology can also be found [39, 40].

2.3. *Cellular Models*

Cellular models utilise the idea of building complex structure from a single underlying primitive – the cell – which typically divides, moves or changes based on an abstraction of chemical signalling or protein synthesis. Some of these models are embedded in restricted spatial arrangements (cartesian grids, isospatial sites), others operate at a more symbolic level. In computer graphics applications they have been used to model three-dimensional form and two-dimensional textures.

Fleischer and Barr [41] introduced a cellular programming model that could grow cellular texture elements over pre-defined surfaces [42]. Each cell consisted of a cellular program (a time-varying first order differential equation) that could control its placement over the surface based on, for example, simulation of chemical reaction-diffusion over the surface. The method offered capabilities such as cell movement, adhesion and changes in size due to cell-cell interaction.

Kumar and Bentley used a method of 'oriented cell division' controlled by a simple genetic regulatory system as a basis for an evolutionary form design system [9]. Each cell was represented by a sphere with isospatial sites for cell division. They experimented with several cell division methods, but were only able to evolve simple shapes such as a line of cells and compound sphere, partially due to the difficulty of defining fitness measures for evolving development into more complex shapes.

Miller describes an embedded, lattice-based cellular system that uses genetically evolvable feed-forward Boolean networks for each cell's program [43]. Binary state information is used as a model of 'chemical signalling' and the networks determine changes in cell state and growth. Miller's goal was to overcome the difficulty of defining the cell rules that lead to a specific pattern, inherent in systems such as that of Fleischer and Barr. Using an evolutionary algorithm, he was able to evolve a 2D cell pattern that resembled the French flag from a single zygote cell.

Some developmental models incorporate hierarchies as an explicit feature, for example, P-systems [44], Vaario's Multi-Level Interaction Simulation language (MLIS) [16] and McCormack's *Cellular Developmental Model* (CDM) [45]. To date, the predominant applications for P-systems and MLIS have not been in the creative domain (P-systems have been used primarily for studying computation and MLIS was designed to evolve artificial neural networks). CDM, however, was designed to generate complex time-varying 3D form.

CDM uses a hierarchical specification of developing cells, each with a set of predicate-action rules. In addition to these rules, cells contain a continuous state vector that is updated as the cell develops and rules are applied. Rules control changes in cell state, and if certain conditions are met, cell division, replacement or death. Rather than representing cells as simple geometric primitives, geometry in CDM is built using generalised cylinders, a fundamental design element in the functional morphology of many species [46]. Sequences of cells are interpreted as instructions to a state machine that builds the geometry. The hierarchical specification of the CDM allows low-level details of model construction to be encapsulated, parameterised, and reused by higher-level cells that specify body parts and other arrangements, overcoming the problems with single macro-level specification discussed in §2.1.

Developmental complexity is reflected not only in the number of individual parts that make up a model, but also in temporal developmental interactions. While L-systems, for example, are capable of increasing the number of developing symbols discretely over time, continuous state systems, such as CDM exploit the non-discrete, temporal nature of the development, allowing animation effects in models such as continuous growth or simulating the gaits of legged figures.

While CDM is a flexible and powerful method for developmental modelling of time-variant natural forms, it still shares with its predecessors some of the limitations of endogenous systems. That is, development is at best only partially embedded in the environment of final representation. Cells develop in structures with limited spatial relations, hence physical relationships may influence development. However, a cell may represent complex geometric development that is realised 'post-development', making physical interaction with the environment that affects development difficult. While physical parameters can be fed back into cell development (e.g. tropisms, chemical gradients), physical interaction between the development process and the geometry it generates is limited.

One the other hand, fully embedded cellular systems (such as Miller's Boolean network cells) trade developmental and visual complexity for spatial interaction. The emergent nature of spatial interaction makes it difficult to intuitively design underlying rules that will grow specific forms, hence the use of evolutionary searches to try and find the rules necessary to grow specific shapes.

3. The Simplicial Developmental System

A new system, the Simplicial Developmental System (SDS), was designed to address environmental and physical effects on a growing structure, with the goal of overcoming the limitations discussed in the previous section. SDS models a developing organism as a *simplicial complex* within a spatial environment. More specifically an organism in k dimensions is a collection of connected non-overlapping $k-$simplexes joined together by $k-1$ simplexes. In 2D this is a collection of triangles connected by their edges and in 3D it is a collection of tetrahedra connected by their faces. Driven by an internal program, the *cells* of the organism grow, divide and move – transforming the simplicial complex. Through a morphogen-based cell communication model, cells can coordinate their activity and develop coherent modules within the larger organism.

A mass-spring model defines energy minimising forces that act upon the simplicial complex, resulting in a soft-body elastic appearance. Additionally a non-overlap constraint results in a surface that interacts in space. Other spatial and physical elements such as static geometries, tropism sources, directional gravity, or even other developing organisms can be included in the simulation environment. This results in a type of *exogenous complexity* that is difficult to achieve with previous endogenous methods, such as CDM and L-systems, but still permits a rela-

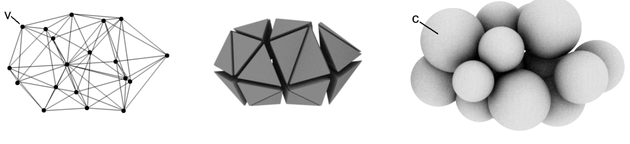

Figure 2: An example SDS organism. It has (a) vertices, edges and (b) tetrahedra. It also has (c) spherical cells. These are all views on the same structure. There is a one-to-one mapping between cells and vertices, for example, vertex v corresponds to cell c.

tively flexible degree of control that was often lacking in prior exogenous systems.

SDS was developed with three dimensional form generation in mind, however it can be instantiated in either two or three dimensions. The discussion here is limited to the three dimensional case: *SDS3*. Details on the two dimensional case, SDS2, can be found in [47]. The basic concepts in SDS are the *organism* and the *cell*. An organism is composed of cells that are spatially situated and topologically related through the geometry which transforms over time (§3.1). A physical model (§3.2) uses the geometry and a mass-spring model to give the organism a dynamic elastic behaviour. The final component of the system is the process model (§3.3), which defines the cells as individual agents, communicating with each other and performing various actions that drive the geometric transformation of the organism. Each of these aspects are now considered in turn, followed by some examples (§3.4).

3.1. Geometry

A shape in SDS3 is represented as a set of connected non-overlapping tetrahedra joined to each other by their faces. An organism consists of a set of cells, with each cell corresponding to a unique vertex of the organism's shape. Figure 2 illustrates an example organism and its shape. The edges of the shape define a topology amongst the cells – if an edge connects two cells then those cells are topological neighbours. This shape representation has many benefits including conceptual simplicity, the ability to represent detail over many scales, and the ability to model arbitrary forms.

In SDS, the starting geometry of an organism is usually very simple. This initial state is analogous to an *axiom* in L-Systems or the root system of CDM. Through a set of local transformations the geometry gains complexity. These transformations are cell division, cell movement and cell growth.

3.1.1. Cell Division

Cell division is the primary transformation; it adds new cells, edges and tetrahedra to the geometry, providing the accumulation of complexity over time. A cell may elect to divide in a specific direction, at which point it is removed and replaced with two or more cells. The tetrahedral complex is modified to accommodate them. Geometrically, the new cells occupy different positions and have an equal distribution of the mass of the original cell. Topologically, the new cells are neighbours and have the local neighbourhood of the original cell distibuted evenly amongst them. An internal cell can divide in any direction, whereas a surface cell can divide tangential to the surface or perpendicularly outwards or inwards. An ideal transformation would minimise sudden jumps in the physical model and localise changes to the topology of the system as much as possible.

Internal cell division was originally modelled in SDS3 by subdividing a tetrahedron adjacent to the dividing cell; however, this resulted in a large variation of vertex degree. This is undesirable as it decreases the stability of the physical model. The division algorithm was then modified to achieve better vertex distribution through optimal tetrahedralisation. This algorithm is as follows:

1. Let c be the dividing cell, and d be the desired direction of division
2. Let T_c be the set of all tetrahedra adjacent to c
3. Let F be the hull surrounding $\cup T_c$
4. Remove $\cup T_c$ and c
5. Add two cells a and b, with $a_x = c_x + \epsilon d$ and $b_x = c_x - \epsilon d$, where a_x, b_x, c_x are the positions of the cells and ϵ is such that a and b are contained within F
6. Tetrahedralise the structure $F \cup \{a, b\}$

This last step can be performed in a number of ways. We chose to generate the delaunay tetrahedralisation using an existing tetrahedralisation library, *Tetgen* [48].

Following the tetrahedralisation phase, the mass of the mother cell c_m is distributed evenly amongst the daughter cells, $a_m = b_m = c_m/2$. The radii of the new cells are then computed from these masses assuming a uniform density (e.g., $a_m = \frac{4}{3}\pi a_r^3$). All new tetrahedra and edges then have their rest sizes calculated (see §3.2). This allows the local volume to be preserved and existing stress the occurs in that area will be transferred into the new configuration.

Cells on the surface can be similarly divided, with the important observation that c is now on the boundary of T_c. Hence we must remove those faces of F that are adjacent to c, resulting in an *open* hull. The resulting structure $F \cup \{a, b\}$ can be tetrahedralised using a variety of different methods, and again Tetgen was used for this.

3.1.2. Cell Movement

Given a set of cells, the simplicial complex defines the topology amongst them. As cells move through space the simplexes (tetrahedra) change shape and occasionally become flat (Figure 3). When this occurs the topology changes via local adjustments to the structure. At every time step, we detect whether the signed volume of each tetrahedron has become negative. The signed volume of a tetrahedron, t, is:

$$\mathrm{vol}(t) = \frac{1}{6}(v_1 - v_0) \cdot ((v_2 - v_0) \times (v_3 - v_0)), \tag{1}$$

where the v_i are the positions of the four vertices of the tetrahedron. An *inverted tetrahedron* is one where $vol(t) \leq 0$, and the time of inversion occurs at the point when $vol(t) = 0$.

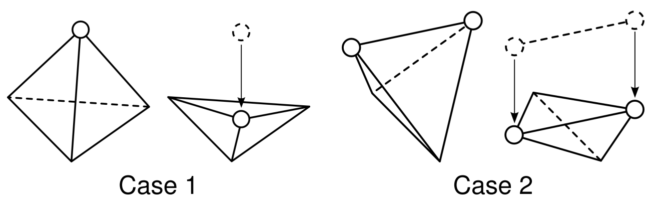

Figure 3: Two different situations that cause a tetrahedron to become flat. Case 1. A vertex enters the opposite face of the tetrahedron. Case 2. An edge intersects the opposite edge. These cases can be distinguished by observing that if any vertex of the tetrahedron lies within the face opposite then Case 1 has occurred, and if not then Case 2 has occurred.

Multiple tetrahedra may invert during one single time step. To deal with this we rewind the simulation back to the first inversion, perform a movement transformation, and then restart the simulation from that point. Given that we are at time step u_2 and the last time step was u_1, for each inverted tetrahedron t, we compute the time u that the inversion occurs by assuming linear motion of the vertices and solving $vol(t) = 0$. This equates to solving the following cubic (discarding the roots that fall outside the interval $[u_1, u_2]$):

$$(v_1(u) - v_0(u)) \cdot ((v_2(u) - v_0(u)) \times (v_3(u) - v_0(u))) = 0,$$

where:

$$v_i(u) = v_i(u_1) + \frac{u - u_1}{u_2 - u_1}(v_i(u_2) - v_i(u_1)).$$

Restarting the simulation from the first inversion is not efficient if there are many inversions occurring in one time step. This problem has also been identified in the collision literature and some systems solve it by handling multiple collisions at a time [49]. In SDS, however, handling multiple inversions simultaneously is non-trivial because of the topological modifications involved. More specifically, our movement transformations given below use the assumption that there are no other inverted tetrahedra besides the target one in order to enumerate the topological configurations that arise. We now discuss the transformations of the two inversion cases illustrated in Figure 3).

Case 1. This occurs when a vertex in a tetrahedron attempts to pass through the opposite face, and results in the transformation described in Figure 4. The figure illustrates the case where neither v nor f lie on the surface. If v lies on the surface then the transformation is exactly the same. If f is on the surface then t' doesn't exist and hence new tetrahedra aren't generated. If v and f are both on the surface then the transformation cannot be applied (as this would result in an infinitely thin section). This situation has yet to arise in the simulations performed; however, if it does then another transformation could be designed.

Case 2. This case occurs when opposite edges in a tetrahedron intersect. Let these edges be named e_u and e_l respectively. Consider the case where e_u and e_l are not on the surface, resulting in a closed hull of all tetrahedra attached to either edge (see

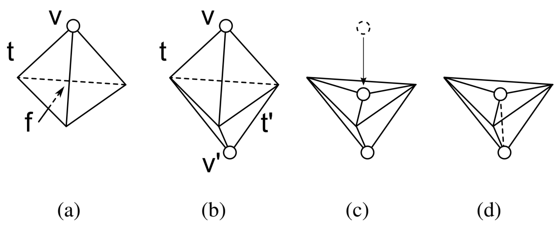

Figure 4: Case 1. (a) Consider a tetrahedron t and a vertex v. Let f be the opposite face. (b) Let t' be the tetrahedron that is joined to face f, and v' be the vertex in t' that is opposite to f. (c) v is assumed to have just intersected face f. (d) We remove f and t but keep the faces adjacent to v. We then add a new edge connecting v and v' thus implicitly tetrahedralising the hull of $t \cup t'$.

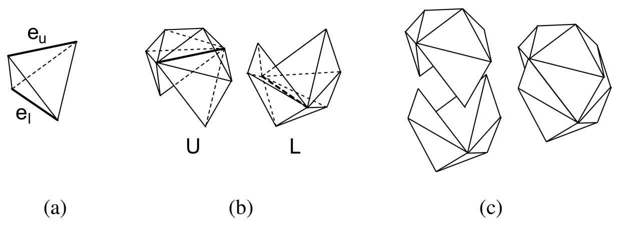

Figure 5: Case 2. (a) Consider the tetrahedron in the diagram. Call its upper edge e_u and its lower edge e_l. (b) Create the sets U and L by considering all tetrahedra that share edge e_u and e_l respectively. (c) Consider the hull around the union of those sets, $U \cup L$. When the tetrahedron's volume becomes zero the task is to tetrahedralise the hull of $U \cup L$.

Figure 5). The mesh is transformed at the time of inversion by removing all tetrahedra within the hull of $U \cup L$ and then tetrahedralising the empty space formed. Tetgen was used to generate the delaunay tetrahedralisation of the empty hull. In general any tetrahedralisation approach will work, with a trade-off existing between running speed and quality of tetrahedra. If e_u or e_l are on the surface then the hull is not closed, and additional tetrahedra must be added to cover each edge before applying the method above.

3.1.3. Cell Growth

Cells have a *radius* and *mass*. These are used by the physical model to determine the dynamics of the organism. By changing size, cells can affect the lengths of edges and the shape of tetrahedra, allowing the development of different sized regions. Hence one part of an organism can be coarsely modelled with large tetrahedra, while simultaneously another part can have many small tetrahedra modelling smaller features. This is a primary benefit of the tetrahedral complex representation, as opposed to, for example, a voxel-based representation. The relationship between cell size and the shape of an organism is discussed next.

3.2. Physical simulation

SDS incorporates a physical model in order to generate organic and natural-looking forms. The approach can be classified as a mass-spring model [50], a popular technique that has been used previously to model cellular complexes [22, 51, 52].

Each cell c has a radius c_r and a mass c_m, where $c_m \propto {c_r}^3$. Every edge is modelled as a spring. While under compression the spring simulates internal cell pressure and under tension simulates cell adhesion. Following the approach of Teschner et al. [53] tetrahedra also resist compression and tension by modelling them as *generalised springs*. The energy within such a spring s can be defined by generalizing Hooke's Law using the following potential energy equation:

$$E(s) = \frac{s_k}{2}(\frac{V(s) - R(s)}{R(s)})^2,$$

where s_k is the spring stiffness, $V(s)$ denotes the current size, and $R(s)$ denotes the *rest* size. The difference between the two sizes is normalised to make the spring stiffnesses *scale-free* as in [53], this allows us to use the same stiffnesses throughout the entire structure. For an edge, e, connecting cells a and b, $V(e) = |a_x - b_x|$ and $R(e) = a_r + b_r$. For a tetrahedron, t, we have $V(t) = vol(t)$ (Equation 1) and $R(t)$ as shown in Appendix A. At each time step the force from each spring, s, acting on each cell, c, is computed as:

$$F_s(c) = -\frac{\partial E(s)}{\partial c_x}$$

$F_s(c)$ can be understood as the direction and magnitude that c has to be pushed in one particular instant in order to minimize the energy of the spring. For each cell, all the forces from all the springs adjacent to it are summed, and then integrated to obtain an updated velocity and position.

An adaptive Verlet integration technique [54] was used as it has been shown to be efficient and sufficiently stable for this type of model [53]. An adaptive step size is necessary because the topological cell movement transformation may require the time step to be modified (§3.1.2). Damping is incorporated into the system as both damping on the tetrahedral springs and viscous damping. The spring stiffness and damping coefficients could easily be specified *per spring*, but for the experiments presented here, uniform spring stiffness and damping coefficients are used for the springs on the surface, and another set for the internal springs. The actual coefficients vary from experiment to experiment and generally have to be fine-tuned to achieve the desired results.

So far we have described a model that moves cells around space based on mass-spring equations. This correctly simulates the interactions between topologically adjacent cells, but fails to capture the interactions between cells that are spatially adjacent. If a surface cell collides with another part of the surface we would expect an interaction. The simulator that has been built uses the collision detection and response scheme of Teschner et. al. [55, 56] to prevent surface cells from penetrating another part of the surface. The same system is used to confine the organism within a physically bounded region, and to model collisions with other objects in the world.

3.2.1. *Initial Conditions and Spring Multipliers*

The SDS simulator takes as input a tetrahedral complex generated by Tetgen from a surface mesh. To convert this to an organism we compute the radii (and hence mass) of all cells. This is done based on the edge lengths, computing the radius of a cell, c_r, as:

$$c_r = \frac{1}{|N(c)|} \sum_{c' \in N(c)} \frac{1}{2} |c_x - c'_x|,$$

where $N(c)$ is the set of all neighbours of c. From the cell radii we then compute the rest sizes, $R(s)$, for all edges and tetrahedra. Initially this procedure seemed adequate, however, importing a tetrahedral complex that has non-regular tetrahedra (a common occurrence) resulted in a deformation of the initial shape. This is undesirable from a control point of view, especially as we often begin with a sphere and wish to preserve its smooth surface. To solve this issue we *could* set $R(s)$ to be equal to $V(s)$ when importing the complex; however, this solution is inadequate as cells often change size and $R(s)$ is hence recalculated. To handle this, we introduced a *spring multiplier* constant, s_m, that is initialised as $s_m = V(s)/R(s)$. Then when querying the rest size for the energy calculations, we use a modified rest size: $R'(s) = s_m R(s)$. We now discuss our model of cell behaviour.

3.3. *Process*

From the *process* perspective cells are viewed as autonomous individuals executing cell programs within the simulation, similar to CDM. Existing methods of modelling cell processes include programming languages [57], GRN-based network models [58, 59, 60], and rule-based techniques [61]. The effectiveness of the latter inspired the model used here. A simple morphogen and rule-based approach to modelling cell state and behaviour is used. Given a set of morphogens $\Phi = \{\phi_1, \ldots, \phi_n\}$, each cell, c, contains a quantity of each, c_{ϕ_i}, bounded above by the cell volume. Morphogens can be created and destroyed within cells, and isotropically diffuse between adjacent cells. Diffusion and decay of morphogens are modelled using the standard particle diffusion equation:

$$\frac{\partial c_\phi}{\partial t} = D\nabla^2 c_\phi - C c_\phi,$$

where c_ϕ is the amount of morphogen ϕ in cell c, D is the diffusion rate and C is the decay rate. Each morphogen has its own diffusion and decay rates. This equation is approximated using the discrete Laplacian:

$$\nabla^2 c_\phi = \sum_{n \in N(c)} \frac{\min(n_\phi, \mathrm{vol}(c)) - \min(c_\phi, \mathrm{vol}(n))}{|c_x - n_x|},$$

where $N(c)$ is the set of cells adjacent to c and $vol(c) = \frac{4}{3}\pi c_r^3$. For simplicity, the morphogen equations are solved in step with the physical simulator using the explicit Euler method. At the end of a simulation time step the values of the morphogens may activate a set of rules. For example, the following rule:

$$c_\phi > .5 \rightarrow \mathrm{divide}(\nabla\phi),$$

specifies that a cell, upon becoming half full of morphogen ϕ, should divide in the direction where ϕ is greatest. The specification of the rules and parameters are the user's interface to

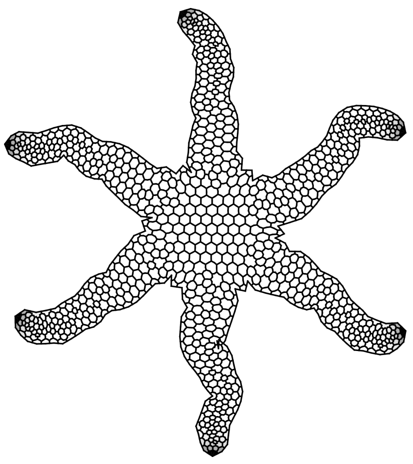
Figure 6: A starfish-like form grown with the SDS2 system. Here we are visualising the cells as polygons, where polygons are adjacent only if there is an edge between the cells in the corresponding (two-dimensional) simplicial complex.

the system. Having now discussed all the major components of SDS, we next provide some examples of how the system can be used generate organic forms.

3.4. Examples

In a previous paper [47] a model of primitive limb growth for SDS2 was presented (see Figure 6). This limb growth model has been adapted and simplified for the 3D system and the examples presented here use a derivative of this model. For a discussion of the biological inspiration for the model we refer the reader to the original paper.

There is only one morphogen: ϕ. Let c_t be a cell state variable, equal to 0 or 1, that indicates whether a cell is a growing tip or not. The rules are:

$$\begin{aligned}
r_1 &: c_t = 1 && \rightarrow c_\phi = 1 \\
r_2 &: c_t = 0 \;\&\; c_\phi > K && \rightarrow \frac{dc_r}{dt} = \triangle \\
r_3 &: c_t = 0 \;\&\; c_r > R && \rightarrow \text{divide}(\nabla\phi)
\end{aligned}$$

Rule r_1 specifies that a cell of type 1 constantly creates ϕ. Rule r_2 specifies that a cell of type 0 starts growing at a linear rate, $\triangle$, when a threshold, K, of the morphogen has been reached. Rule r_3 then states that once a cell has reached a given size, R, then it will divide towards the source of the morphogen. This basic process results in a localised growth towards a growing tip, if the tip is on the surface then we see limb-like structures forming. This entire process is possible due to the cell rules, the physical model, and the geometric transformations.

Figure 7 demonstrates the basic sequence of one growing limb. The parameters of the model can affect the rate of growth and size of the limb. The model is quite sensitive to the parameters, and some parameter ranges can result in either no limb growth, or uncontrolled tumour-like growth. Finding the appropriate parameter ranges can be done experimentally, but an ideal growth model would have a minimal set of parameters, each relating to some visual aspect of the model. In the case of limb growth, an ideal set of parameters would be *limb size*, *growth speed*, and *limb length*. Future work will look at building these kinds of models.

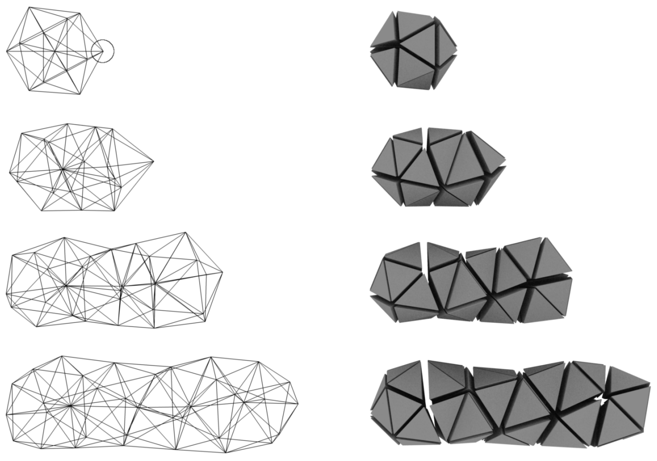
Figure 7: A visualisation of how the edges (left) or the tetrahedra (right) of a developing organism change over time. (top) An organism at the start of a simulation has a cell chosen to be a growing tip (circled). (from top to bottom) As morphogen diffuses, cells begin to grow and divide near the growing tip, forcing the tip to the right and creating a limb-like structure.

As shown in Figure 8, multiple limbs can be grown in the same organism, simply by specifying multiple growing tips. The re-use of components like this is an important feature of a procedural geometry system, as it allows us to abstract away from the level of cells to the level of *limbs* and *bodies*. However, because the limb model operates on the level of geometry we get additional benefits such as a complex boundary forming between the limb module and the body it grew from. This complexity is difficult to achieve using module-based geometric methods [62] and normally has to be achieved as a post process.

SDS forms can be influenced by spatial and environmental factors, providing another level of control over a growing geometry. For example, we can restrict the growth of a structure within a confined space (Figure 8), incorporate other objects in the space (Figure 9), or include an attraction point which affects the growth direction of the limbs (Figure 10).

Many environmental phenomenon such as phototropism or chemotaxis could be included within the simulation. The requirements of the user dictate which aspects are important for each specific case. For example a form can be made to *stick* to a geometric object as it grows, allowing vines to be grown on a wall. This embedded physical interaction provides another level of control over the system and the variety of situations it can model.

Figure 11 illustrates a variation derived from the basic limb model. In this example a *timer* (non-diffusing morphogen that slowly decays) was added to the growing tip, causing it to stop releasing the limb growth morphogen after a certain delay. It then activates another morphogen which causes nearby cells to grow slowly until they reach a specific size, resulting in buds that form at the end of the limbs.

3.5. Efficiency

SDS can be computationally intensive, particularly for com-

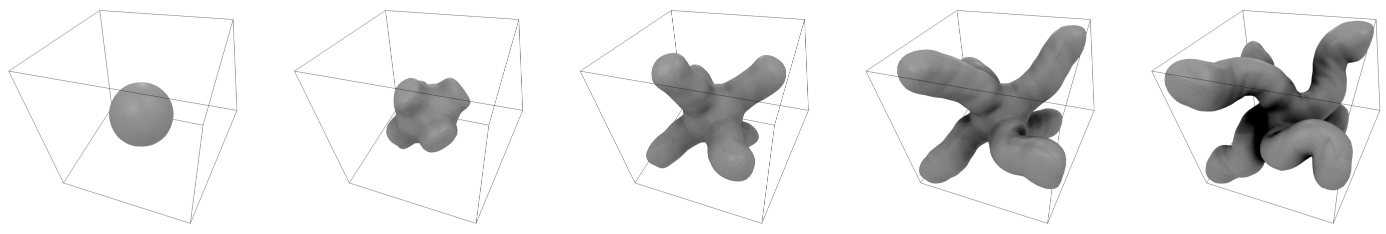

Figure 8: A time series of an organism with six limbs. The limbs eventually collide with the bounding box. They continue to grow in the constrained space and the physical model causes them to curve and bend, eventually filling the space as shown in Figure 1.

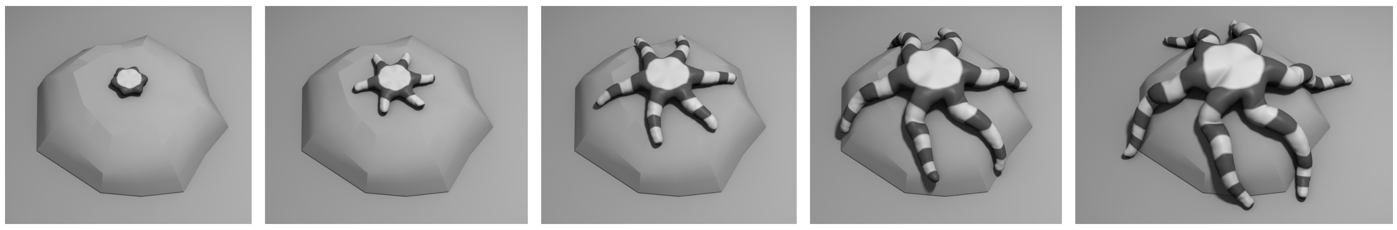

Figure 9: This simulation was initialised with a radially symmetric geometry with six growing tips placed upon a rock. Various snapshots of the simulation are shown that demonstrate the limbs growing outwards and over the rock. The growth model includes a rule to allow the body to grow slowly over time ($r_4 : c_t = 0 \,\&\, c_\phi < K \rightarrow \frac{dc_r}{dt} = \triangle'$). The stripes are generated by a morphogen *timer* that slowly decays within the growing tip. Once depleted the growing tip signals all nearby cells to become *stripe cells* and then the timer is reset. The rate of decay of the timer directly influences how apart the stripes are.

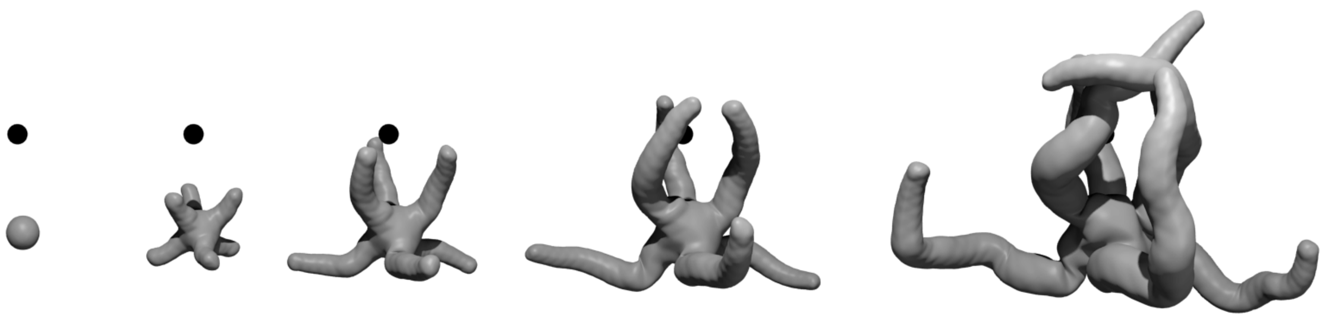

Figure 10: This simulation incorporated an *attraction point* (shown as a black dot), just above the initial organism. Starting in the same configuration as in Figure 8, the growing tips had a small attraction force applied to them, which resulted in the developmental sequence shown.

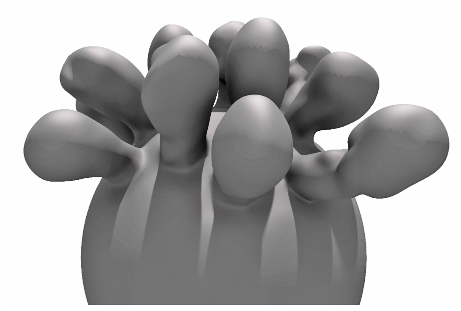

Figure 11: This budded form was created using the limb bud model with an extra rule added that caused the tentacle tips to start expanding after a short time.

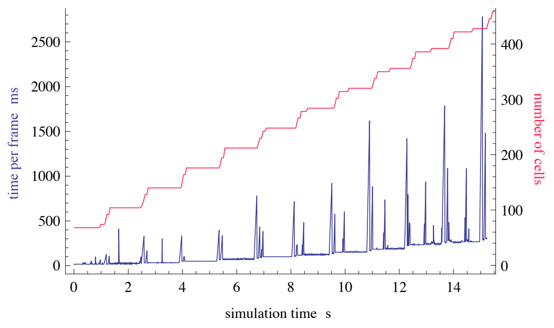

Figure 12: Timing information for the model shown in Figure 9: simulation time on the horizontal axis vs. CPU time per time step (blue) and number of cells (red). The spikes in CPU time correspond to division of cells in the growing model.

plex models with many cell divisions. As an example, the experiment shown in Figure 9 is analysed here. This model took approximately 5.7 minutes to generate to the stage shown in the figure. The simulation machine was a standard desktop PC with a dual core 2GHz processor and 2Gb RAM. Figure 12 shows plots of the virtual simulation time on the x-axis versus the real time to compute (blue) and the number of cells currently in the simulation (red). In this experiment the number of cells increases linearly over time. Cell division events for each limb occur more or less simultaneously, due to the symmetric initial conditions. The computation time plot shows the number of milliseconds the simulator takes to compute each step. The base line is punctuated by large peaks that occur when the cells divide, illustrating that the division algorithms take much longer to process that the base algorithm for physical simulation.

3.6. Current Limitations of the Limb Bud Model

The limb growth experiments presented here generate limbs that are, internally, only one cell thick (see Figure 7). Conceptually the growth model is capable of generating limbs of arbitrary thickness (as demonstrated in [47]) simply by increasing the rate of diffusion; however, generating different limb widths has not yet been explored in the 3D model. Generating these *thin* limbs results has some consequences which we briefly outline.

As discussed above, the limb structure develops because cells are dividing toward the growing tip, pushing it outwards. When generating limbs that are internally one cell thick, we need to ensure that the initial configuration has a cell directly below the growing tip, for without it there would be no internal pressure for the tip to move outwards. To guarantee this constraint, we establish the initial conditions manually by adding a vertex below all the growing tips. An important goal for future work is to eliminate this manual intervention.

4. Discussion and Future Work

This paper presents a new approach in developmental modelling for computer graphics. The Simplicial Developmental System, or SDS, addresses some of the disadvantages of previous systems, such as L-systems or the CDM. It takes a fully embedded and physically simulated approach, effectively making it capable of a more accurate and diverse set of simulation possibilities. SDS uses only a single geometric primitive (tetrahedral meshes), simplifying physical and developmental simulation, but conversely requiring far greater processing power than previous systems to simulate the development of complex structures. SDS is a new approach to generating forms and is still the subject of active research. As such the examples do not encompass all aspects of what is possible within the general framework. Combining physical and biological simulation of growth processes appears to be a very powerful methodology, opening many possibilities for further research, a number of which will be now be discussed.

Growing other sub-structures. The examples presented here are only the initial steps towards what could be a highly expressive and powerful method for synthesising complex organic form. The limb bud model was exploited in the examples to generate limb- and tentacle-like sub-forms, and adding additional processes to expand the palette of interesting sub-forms is currently under investigation. We envisage a wide range of different processes within SDS that provide the artist with a toolbox of different sub-form and pattern generating tools. It would be highly beneficial to find a set of *axiomatic processes* forming the core of this toolbox, and upon which other processes could be built. One of these, for example, might be local diffusion and directional proliferation, which forms the basis of the limb bud model. Sub-forms such as tubular growths, folds, rings and bone-like structures are amongst the kind of things we could expect to see in a toolbox, along with methods for laying them out using morphogen patterns and composing them into more complex structures.

Patterning. Morphogen creation, destruction, diffusion and decay plays the role of patterning within SDS. Distributions of morphogens can direct specific parts of a mesh to grow, for example, in the regions where limb development occurs. More complex patterns could be achieved by multiplying two different morphogen gradients together. In general, functionally composing morphogen gradients can lead to complex patterns

[63]. Another approach is to specify the equivalent of reaction-diffusion equations [64], which could be achieved through the existing rule-based framework. This latter approach, while difficult to control, would allow the specification of complex textural patterns, such as spots and stripes.

User friendliness. The procedural specification of an SDS form involves the creation of cell rules that act on, and are activated by, temporal patterns. In addition a number of different parameters, such as spring stiffness, tropism strength, viscosity, and morphogen decay rate may be adjusted. Many of these parameters are abstract or irrelevant to the end user, and the specification of rule-sets becomes more difficult for increasingly complex forms. As discussed in §2.3, exogenous systems are often difficult to control from a form design perspective, which is why past approaches have used, for example, evolutionary search techniques to find appropriate parameters and rules. In contrast, SDS allows the possibility of presenting higher-level concepts to the user, such as *Limb*, *Pattern* and *Attractor*. These building blocks could then be connected together and coupled with a geometric description of the scene built using standard 3D modelling tools. A process of translation would then take the high-level description and generate the appropriate cell programs.

Simulation method. The choice of integration schemes for the soft-body and morphogen simulations were adequate to achieve the initial results presented in this paper. However, these methods are crude and fail to correctly conserve physical properties, such as the energy within the soft-body or the concentration of morphogen in the cells. They are sometimes unstable and without the correct choice of time step the physical simulation can "blow up". This is an especially significant problem when finer geometric details are required. The use of a better integration technique, specifically for the soft body simulation, would improve the reliability and robustness of the simulation. Further work is required to compare performance of different schemes against different requirements, for example, speed versus geometric detail.

Heterogenous material. The simulations presented in this paper have all been performed using a homogeneous material (all the spring stiffnesses are the same). One exciting avenue of research would be to explore the creative possibilities of allowing different material properties within the same organism. This would allow us to have, for example, rigid bones, softer muscle mass, and a jelly-like material all within the same structure. Aside from the user-control issues (i.e., how would a user specify the formation of different materials), a major impediment to achieving this is the physical simulation of such a material. Incorporating highly stiff springs requires a large amount of computation time due to the increased instability: a commonly observed problem with mass-spring simulation [65]. Research into real-time simulation of deformable bodies for surgical simulation may provide solutions, for example the hybrid approach of Lin et al. [66] combines rigid and non-rigid materials in a conceptually simple and fast model. Another simulation issue is the refinement of heterogeneous material. For example, when a cell divides in a heterogeneous region, what is an appropriate procedure for deciding the physical properties of the new edges and tetrahedra. Recent work into numerical coarsening [67] may offer insight into a possible solution.

Transformations. Cell death is an obvious exclusion from the transformations presented here. There are numerous ways a transform analogous to cell death could be implemented – for example performing cell division in "reverse". Aside from mesh simplification it is not clear the role cell death would play in this system. Other interesting operators that could be researched are ones that allow the surface to fuse together and split apart, permitting changes in the topological quality of the surface (and would allow biological phenomenon like gastrulation and neurulation to be modelled.) In addressing some these factors, the simulation-based approach of SDS will ideally become more user-friendly and expressive.

In conclusion, computational simulation of biological morphogenesis and development holds great promise as a general purpose tool for computer graphics modelling. The idea of being able to *grow* a myriad of complex forms without the need to directly and painstakingly model them by hand has obvious appeal. The next stage in modelling development is to combine the geometric complexity possible with models like CDM with the direct embedding and full physical simulation of SDS. This raises the issues of computational power (complex physical simulation is computationally expensive) and useful user specification (the tradeoff between specification abstractions, control and complexity). Addressing these issues is an active area of on-going research.

5. Acknowledgements

The authors would like to thank the reviewers for their helpful comments and suggestions, in particular for highlighting the numerical issues that can occur with the simulation techniques used. This research was supported by an Australian Research Council Discovery Project grant DP0772667.

Appendix A. Calculating the rest size of a tetrahedron

The rest size of a tetrahedron t with cells a,b,c and d can be easily derived from Heron's formula, where the lengths of the tetrahedron's edges are the sum of the adjacent cell's radii. For the sake of completeness it is included here.

$$\begin{aligned}
R(t) &= \frac{\sqrt{ABCD}}{3(a_r + d_r)(b_r + d_r)(c_r + d_r)}, \text{ with} \\
A &= w + x + y - z \\
B &= w + x - y + z \\
C &= w - x + y + z \\
D &= -w + x + y + z \\
w &= a_r b_r c_r \\
x &= d_r \sqrt{b_r c_r (a_r + b_r + d_r)(a_r + c_r + d_r)} \\
y &= d_r \sqrt{a_r c_r (a_r + b_r + d_r)(b_r + c_r + d_r)} \\
z &= d_r \sqrt{a_r b_r (a_r + c_r + d_r)(b_r + c_r + d_r)}.
\end{aligned}$$

References

[1] Shapeways.
URL `http://www.shapeways.com/`

[2] A. Smith, Plants, fractals, and formal languages, in: Proceedings of the 11th annual conference on Computer graphics and interactive techniques (SIGGRAPH), Vol. 18, ACM, 1984, pp. 1–10.

[3] M. Lantin, Computer simulations of developmental processes, Tech. rep., SFU CMPT (1997).
URL `ftp://fas.sfu.ca/pub/cs/TR/1997/CMPT97-24.pdf`

[4] A. Sandberg, Models of development, Tech. rep., KTH, Stockholm (2006).
URL `http://www.nada.kth.se/ asa/Work/index.html`

[5] J. L. Giavitto, C. Godin, O. Michel, P. Prusinkiewicz, Computational models for integrative and developmental biology, Tech. rep. (2002).

[6] P. Prusinkiewicz, Modeling and visualization of biological structures, in: Proceeding of Graphics Interface '93, Toronto, Ontario, 1993, pp. 128–137.

[7] P. Prusinkiewicz, Modeling plant growth and development, Current Opinion in Plant Biology 7 (1) (2004) 79–83.
URL `http://algorithmicbotany.org/papers/mpg.copb2004.html`

[8] K. O. Stanley, R. Miikkulainen, A taxonomy for artificial embryogeny, Artificial Life 9 (2) (2003) 93–130. doi:http://dx.doi.org/10.1162/106454603322221487.

[9] S. Kumar, P. J. Bentley, Mechanisms of oriented cell division in computational development, in: Proceedings of the first Australian Conference on Artificial Life, Canberra, Australia, 2003.

[10] G. B. Ermentrout, L. Edelstein-Keshet, Cellular automata approaches to biological modeling, Journal of Theoretical Biology 160 (1993) 97–133.

[11] G. W. Brodland, Computational modeling of cell sorting, tissue engulfment, and related phenomena: A review, Applied Mechanics Reviews 57 (1) (2004) 47–76. doi:10.1115/1.1583758.
URL `http://link.aip.org/link/?AMR/57/47/1`

[12] A. Lindenmayer, An axiom system for the development of filamentous organisms, in: Abstracts of the III International Congress on Logic, Methodology and Philosophy of Science, Amsterdam, 1967, pp. 127–128.

[13] P. Prusinkiewicz, A. Lindenmayer, The algorithmic beauty of plants, no. xii, 228 in The virtual laboratory, Springer-Verlag, New York, 1990.

[14] C. Jirasek, P. Prusinkiewicz, B. Moulia, Integrating biomechanics into developmental plant models expressed using l-systems, in: Plant biomechanics 2000 . Proceedings of the 3rd Plant Biomechanics Conference, Freiburg-Badenweiler, August 27 to September 2, 2000., Georg Thieme Verlag, Stuttgart, 2000, pp. 615–624.

[15] Z. Lam, S. A. King, Simulating tree growth based on internal and environmental factors, in: GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia, ACM Press, New York, NY, USA, 2005, pp. 99–107.

[16] J. Vaario, From evolutionary computation to computational evolution, Informatica (Slovenia) 18 (4).

[17] R. Měch, P. Prusinkiewicz, Visual models of plants interacting with their environment, in: SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, ACM Press, New York, NY, USA, 1996, pp. 397–410. doi:http://doi.acm.org/10.1145/237170.237279.

[18] G. Stiny, Shape: Talking about Seeing and Doing, MIT Press, 2006.

[19] K. Sims, Evolving virtual creatures, in: Computer Graphics, orlando, florida Edition, ACM SIGGRAPH, 1994, pp. 15–22.

[20] J. A. Heisserman, Generative geometric design and boundary solid grammars, Ph.D. thesis, Department of Architecture, Carnegie Mellon University, Pittsburgh, Pennsylvania (May 1991).

[21] S. Maierhofer, Rule-based mesh growing and generalized subdivision meshes, Ph.D. thesis, Vienna University of Technology (2002).

[22] C. Smith, On vertex-vertex systems and their use in geometric and biological modelling, Ph.D. thesis, The University of Calgary (Apr. 2006).

[23] Y. I. Parish, P. M
"uller, Procedural modeling of cities, in: Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH), ACM, 2001, pp. 301–308.

[24] J. McCormack, Evolutionary l-systems, in: P. F. Hingston, L. C. Barone, Z. Michalewicz (Eds.), Design by Evolution: Advances in Evolutionary Design, Natural Computing Series, Springer, 2008, pp. 168–196.

[25] G. Stiny, Pictorial and formal aspects of shape and shape grammars, no. xv, 399 in ISR, Interdisciplinary systems research;, Birkhäuser, Basel; Stuttgart, 1975.

[26] N. Greene, Voxel space automata: modeling with stochastic growth processes in voxel space, in: SIGGRAPH '89: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press, New York, 1989, pp. 175–184. doi:http://doi.acm.org/10.1145/74333.74351.

[27] J. McCormack, Open problems in evolutionary music and art, in: F. Rothlauf, J. Branke, S. Cagnoni, D. W. Corne, R. Drechsler, Y. Jin, P. Machado, E. Marchiori, J. Romero, G. D. Smith, G. Squillero (Eds.), EvoWorkshops, Vol. 3449 of Lecture Notes in Computer Science, Springer, 2005, pp. 428–436.

[28] M. Eden, A Two-Dimensional Growth Process, in: J. Neyman (Ed.), Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume IV: Biology and Problems of Health, The Regents of the University of California, 1961, pp. 223–239.

[29] S. Ulam, On some mathematical problems connected with patterns of growth of figures, Proceedings of Symposia in Applied Mathematics 14 (1962) 215–224.

[30] Y. Kawaguchi, The art of the growth algorithm, in: C. G. Langton, K. Shimohara (Eds.), Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems, MIT Press, Nara, Japan, 1996, pp. 159–166.

[31] T. A. Witten, L. M. Sander, Diffusion-limited aggregation, a kinetic critical phenomenon, Phys. Rev. Letters 47 (1981) 1400–1403.

[32] P. Hogeweg, Evolving mechanisms of morphogenesis: on the interplay between differential adhesion and cell differentiation, Journal of Theoretical Biology 203 (2003) 317–333.

[33] T. M. Cickovski, C. Huang, R. Chaturvedi, T. Glimm, H. G. E. Hentschel, M. S. Alber, J. A. Glazier, S. A. Newman, J. A. Izaguirre, A framework for three-dimensional simulation of morphogenesis, IEEE/ACM Transactions on Computational Biology and Bioinformatics 2 (3).

[34] J. McCormack, Creative ecosystems, in: A. Cardoso, G. Wiggins (Eds.), Proceedings of the 4th International Joint Workshop on Computational Creativity, 2007, pp. 129–136.

[35] J. McCormack, O. Bown, Life's what you make: Niche construction and evolutionary art, in: M. Giacobini, A. Brabazon, S. Cagnoni, G. A. D. Caro, A. Ekárt, A. Esparcia-Alcázar, M. Farooq, A. Fink, P. Machado, J. McCormack, M. O'Neill, F. Neri, M. Preuss, F. Rothlauf, E. Tarantino, S. Yang (Eds.), EvoWorkshops, Vol. 5484 of Lecture Notes in Computer Science, Springer, 2009, pp. 528–537.

[36] J. A. Kaandorp, Fractal Modelling: Growth and Form in Biology, Springer-Verlag, 1994.

[37] J. A. Kaandorp, J. E. Kübler, The Algorithmic Beauty of Seaweeds, Sponges and Corals, Springer, 2001.

[38] J. Combaz, F. Neyret, Semi–interactive morphogenesis, in: Proceedings of the IEEE International Conference on Shape Modeling and Applications, 2006.

[39] C. H. Leung, M. Berzins, A computational model for organism growth based on surface mesh generation, J. Comput. Phys. 188 (1) (2003) 75–99. doi:http://dx.doi.org/10.1016/S0021-9991(03)00153-0.

[40] L. G. Harrison, S. Wehner, D. M. Holloway, Complex morphogenesis of surfaces: theory and experiment on coupling of reaction diffusion patterning to growth, Nonlinear Chemical Kinetics: Complex Dynamics and Spatiotemporal Patterns, Faraday Discuss. 120 (2001) 277–294.

[41] K. W. Fleischer, A. H. Barr, Artificial Life III, Vol. XVII, Addison-Wesley, Reading, Massachusetts, 1994, Ch. A Simulation Testbed for the Study of Multicellular Development: The Multiple Mechanisms of Morphogenesis, pp. 389–416.

[42] K. W. Fleischer, D. H. Laidlaw, B. L. Currin, A. H. Barr, Cellular texture generation, in: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques (SIGGRAPH), ACM, 1995, pp. 239–248.

[43] J. Miller, Evolving developmental programs for adaptation, morphogenesis, and self-repair, Advances in Artificial Life (2003) 256–265.

[44] G. Păun, From cells to computers: Computing with membranes (P systems), BioSystems 59 (3) (2001) 139–158.

[45] J. McCormack, Advances in Artificial Life (8th European Conference, ECAL 2005), Vol. LNAI 3630, Springer-Verlag, Berlin; Heidelberg, 2005, Ch. A Developmental Model for Generative Media, pp. 88–97.

[46] S. A. Wainwright, Axis and circumference: the cylindrical shape of plants and animals, no. viii, 132, Harvard University Press, Cambridge, Mass., 1988.

[47] B. Porter, A developmental system for organic form synthesis, in: K. B. Korb, M. Randall, T. Hendtlass (Eds.), ACAL, Vol. 5865 of Lecture Notes in Computer Science, Springer, 2009, pp. 136–148.

[48] H. Si, Tetgen: A Quality Tetrahedral Mesh Generator and a 3D Delaunay Triangulator, retrieved from `http://tetgen.berlios.de/` on 29.12.09.

[49] R. Bridson, R. Fedkiw, J. Anderson, Robust treatment of collisions, contact and friction for cloth animation (2005) 2doi:http://doi.acm.org/10.1145/1198555.1198572.

[50] M. Müller, J. Stam, D. James, N. Thürey, Real time physics: class notes, in: SIGGRAPH '08: ACM SIGGRAPH 2008 classes, ACM, New York, NY, USA, 2008, pp. 1–90. doi:http://doi.acm.org/10.1145/1401132.1401245.

[51] P. Eggenberger, Genome-physics interaction as a new concept to reduce the number of genetic parameters in artificial evolution, in: R. Sarker, R. Reynolds, H. Abbass, K.-C. Tan, R. McKay, D. Essam, T. Gedeon (Eds.), Proceedings of the IEEE 2003 Congress on Evolutionary Computation, IEEE Press, Piscataway, NJ, 2003, pp. 191–198.

[52] M. J. M. de Boer, F. D. Fracchia, P. Prusinkiewicz, Lindenmayer Systems: Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology, Springer-Verlag, 1992, Ch. A model for cellular development in morphogenetic fields, pp. 351–370.

[53] M. Teschner, B. Heidelberger, M. Müller, M. Gross, A versatile and robust model for geometrically complex deformable solids, in: Proceedings of Computer Graphics International, Heraklion, 2004, pp. 312–319.

[54] J. Dummer, A simple time-corrected verlet integration method, retrieved from `http://www.gamedev.net/reference/articles/article2200.asp` on 29.12.09. (June 2005).

[55] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, M. Gross, Optimized spatial hashing for collision detection of deformable objects, in: Proceedings of Vision, Modeling, and Visualization, Munich, Germany, 2003, pp. 47–54.

[56] B. Heidelberger, M. Teschner, R. Keiser, M. Müller, M. Gross, Consistent penetration depth estimation for deformable collision response, in: Proceedings of Vision, Modeling, Visualization, Stanford, USA, 2004, pp. 339–346.

[57] P. Agarwal, The cell programming language, Artificial Life 2 (1) (1994) 37–77.

[58] F. Stewart, T. Taylor, G. Konidaris, Metamorph: Experimenting with genetic regulatory networks for artificial development, in: Proceedings of the Eighth European Conference on Artificial Life, Springer–Verlag, 2005, pp. 108–117.

[59] F. Streichert, C. Spieth, H. Ulmer, A. Zell, Evolving the ability of limited growth and self-repair for artificial embryos, in: Proceedings of the 7th European Conference on Artificial Life, 2003, pp. 289–298.

[60] F. Dellaert, R. D. Beer, Toward an evolvable model of development for autonomous agent synthesis, in: P. Maes, R. Brooks (Eds.), Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, MIT Press, Cambridge, MA, 1994, pp. 246–257.

[61] K. Fleischer, A multiple-mechanism developmental model for defining self-organizing geometric structures, Ph.D. thesis, California Institute of Technology, Pasadena, California (1995).

[62] B. Lintermann, O. Deussen, Interactive modeling of plants, IEEE Computer Graphics & Applications 19 (1999) 2–11.

[63] K. O. Stanley, Exploiting regularity without development, in: Proceedings of the AAAI Fall Symposium on Developmental Systems, AAAI Press, Menlo Park, CA, 2006.

[64] A. M. Turing, The chemical basis of morphogenesis, Philosophical Transactions of the Royal Society 237 (1952) 37–72.

[65] D. Baraff, A. Witkin, Dynamic simulation of non-penetrating flexible bodies, SIGGRAPH Comput. Graph. 26 (2) (1992) 303–308. doi:http://doi.acm.org/10.1145/142920.134084.

[66] S. Lin, Y.-S. Lee, R. J. Narayan, Printed Biomaterials, Springer, 2010, Ch. Heterogeneous Deformable Modeling of Bio-Tissues and Haptic Force Rendering for Bio-Object Modeling, pp. 19–37.

[67] L. Kharevych, P. Mullen, H. Owhadi, M. Desbrun, Numerical coarsening of inhomogeneous elastic materials, ACM Trans. Graph. 28 (2009) 51:1–51:8.