A Face-Encoding Grammar for the Generation of Tetrahedral-Mesh Soft Bodies

John Rieffel[1] and Schuyler Smith[1]

[1]Union College, Schenectady, NY 12308
rieffelj@union.edu

Abstract

Many of the most profound works of artificial life have emerged through the composition of physical simulation and generative representations. And yet, while physics engines are becoming more realistic, and generative representations are growing more powerful, they are still predominantly used to simulate *rigid* objects. The natural world and its organisms are, by contrast, *soft*, and full of much more interesting (and complex) interactions than those which can be faithfully reproduced by rigid body dynamics. In this work we describe and implement a grammatical encoding capable of generating large, complex, and multi-resolution *soft structures* which can be natively simulated by the state-of-the-art hardware-accelerated physics engines. The structures generated by the encoding exhibit all the benefits (structural modularity, large-scale co-ordinated change) of more conventional rigid-body generative encodings.

Introduction

The generative encoding of morphology embedded within physical simulation has a long and rich history in artificial life, tracing back to Karl Sim's seminal work on evolved virtual creatures (1994) and to Lindenmayer and Prusinkiewicz's L-system-based plants (1990). More recent notable contributions include the evolution of satellite antennae (Lohn et al., 2005), robots (Pollack et al., 2001), and tensegrity structures (Rieffel et al., 2009).

A unifying property of these contributions is that they all produce *rigid* objects. This is largely due to the limitations imposed by popular off-the-shelf physics engines, such as the Open Dynamics Engine (ODE) which, although capable of smoothly simulating the interaction of thousands of rigid bodies , lack the ability to effectively simulate softer materials such as cloth or rubber. Finite Element Analysis (FEA) and Computational Fluid Dynamics (CFD), are incredibly accurate, but too computationally intensive to be practical for Artificial Life purposes.

Of course, most biological organisms are quite soft, and the complex dynamical interactions which arise from this softness are beyond what can be realistically reproduced by simpler rigid body dynamics. Recently, off-the-shelf hardware-accelerated physics engines, such as NVidia's PhysX, have added to ability to simulate soft shapes, opening the door to a much more dynamic range of virtual creatures.

Taking full advantage of this functionality, however, requires a grammatical encoding capable of generating large, open-ended, and incredibly complex soft structures. In this paper we introduce a face-encoding grammar which operates upon tetrahedral meshes like the one shown in Figure 1. Meshes such as these are used to describe deformable objects in computational methods such as FEA, as well as in physics engines such as PhysX. By operating directly within the representational substrate of soft bodies (avoiding post-hoc methods such as generating a more generic CAD file and then computing a near-matching mesh) we avoid design bias and have a more nuanced control over the final product.

As we show, the face-encoding grammar we introduce is able to generate arbitrarily large, and incredibly complex tetrahedral meshes. Furthermore, like other grammatical encodings, our process exhibits implicit modularity and allows small changes in the underlying grammar to produce large-scale co-ordinated changes in the final product. The results of this paper open the door to a whole new dimension of the artificial life: *soft* virtual creatures, and *soft* robots.

Generative Encodings

Generative encodings come in a variety of styles: Artificial Ontogeny (Bongard and Pfeifer, 2003), Generative and Developmental Systems (Stanley, 2008), and Lindenmeyer Systems (L-Systems) (Prusinkiewicz and Lindenmayer, 1990)(to name a few), but all have a common set of features, and all offer a variety of advantages. Using the the biological processes of growth and development as inspiration, generative encodings grow large complex objects by applying a simple set of re-write rules to an initial "seed". In the case of L-Systems, the seed is a small starting string of characters, grammatical production rules determine the order of growth. Gene Regulatory Networks Bongard and Pfeifer (2003) model the interaction between transcription factors and gene expression, and can be used to grow both morphologies and neural networks.

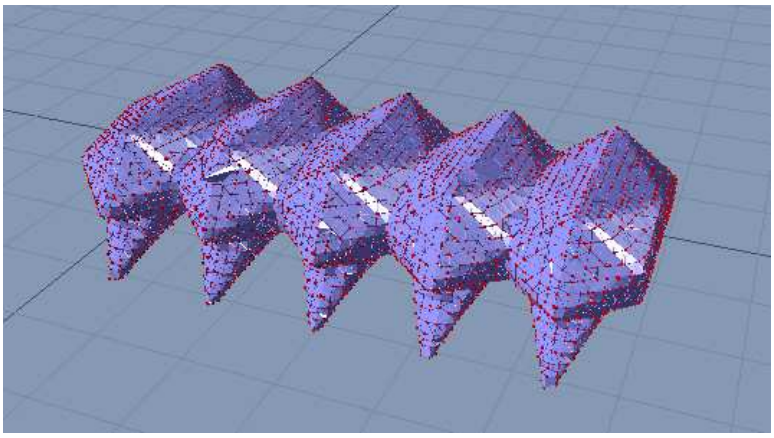

Figure 1: A large soft robot within the PhysX physics engine. Soft bodies are represented as tetrahedral meshes. This particular mesh was created in a top-down fashion: hand-designed by an engineer in CAD, and then manually converted into a tetrahedral mesh. This paper describes an alternative bottom-up approach: a grammar for automatically generating arbitrarily large and complex structured tetrahedral meshes.

Regardless of implementation, the benefits of generative representations, particularly in the context of Genetic Algorithms, stem from their ability to implicitly encode structural modularity and reuse, and the ability for small changes to the rule set to produce corresponding large-scale co-ordinated changes in the final result (Hornby and Pollack, 2001). As an example, when representing a table, unlike a direct encoding, a generative encoding is able to change the length of all four legs simultaneously.

Generative encodings are particularly popular in evolutionary design tasks, in which they are used to specify the structure (morphology) of objects and creatures. Karl Sims' early work (1994) on artificial life used a simple grammar to grow virtual creatures within a simulated environment, Lohn *et al* used L-Systems to design the satellite antennae (2005) and Hornby used a variety of L-System to develop the morphology of virtual robots (2001).

Physics Simulation

Generative encodings of morphology really come to life when they are embedded within realistic physical simulations. Karl Sim's virtual creatures were evaluated within a simple but quite effective quasi-static physical simulator (1994). Later work, such as Lipson's GOLEMs (2001) and Hornby's GenoBots (2003) also involved quasi-statics -static simulations. More recently, the advent of off-the-shelf physics engines such as the Open Dynamics Engine (ODE), has led to more dynamical simulations, such as Bongard's virtual creatures (2003) and Rieffel's tensegrity robots (2010).

Conventionally, the only means of simulating the dynamic behavior of *soft* objects was through computationally intensive tools such as Finite Element Analysis (FEA) and Computational Fluid Dynamics (CFD). While these methods are quite powerful, they are computationally intensive, and operate on small enough time scales (usually simulating only seconds at a time) as to make them impractical for common Artificial Life techniques such as evolutionary algorithms. Recently, however, following in the footsteps of modern advances in computer graphics (Jakobsen, 2001), commercial video-game physics engines, such as Intel's Havok, and NVidia's PhysX, have added the ability to simulate cloth as well as three-dimensional soft bodies. What makes these engines particularly appealing to the artificial life community is their ability to use General Purpose Computing on Graphics Processing Units (GPGPU) interfaces in order to achieve significant hardware acceleration of simulations – providing speedups of several orders of magnitude (Banzhaf and Harding, 2009).

A way of grammatically generating soft morphologies and testing them in simulation would be a valuable tool for further exploring these issues. The remainder of this paper describes one such implementation.

A Face-Encoding Grammar for Tetrahedral Meshes

Central to our approach is the use of tetrahedral meshes to represent soft bodies. While our examples below are within the context of NVidia's PhysX simulator, it is worth emphasizing that tetrahedra meshes are commonly used in other systems as well, such as Finite Element Analysis.

Figure 2 illustrates a single tetrahedron. The "softness" of a material within PhysX can be changed by varying a set of constraints placed upon the tetrahedron. The first constraint treats each edge of the tetrahedron as a spring-and-damper system, which resists both stretching and compression. A second constraint attempts to maintain each tetrahedra at

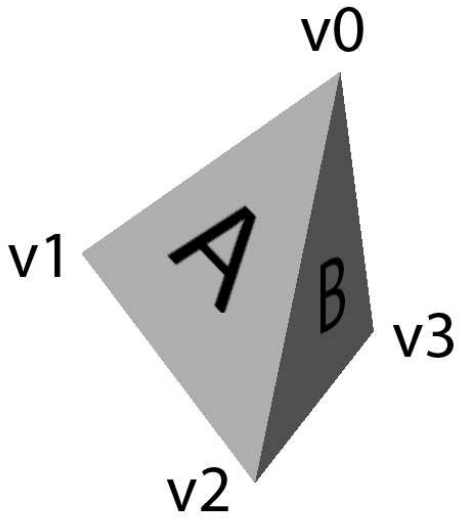

Figure 2: Soft bodies in PhysX are built out of tetrahedral meshes. Each tetrahedron is defined by four vertices and four corresponding faces

a constant volume. Changing the value these parameters changes the softness of the tetrahedron. These tetrahedra are then woven into a larger "mesh", in which neighboring tetrahedra are connected at their common vertices. By uniformly varying the parameters of the tetrahedral mesh, PhysX can simulate a wide range of soft materials, from rubbery Jell-O to semi-rigid plastics.

Since there are no known grammatical encodings which operate upon tetrahedral meshes, we will create our own. We use as inspiration the Map L-Systems, a special form of L-system whose rewrite rules operate upon the edges of 2-D graphs (Luke and Spector, 1996). Map L-Systems have been used to grow both 3-D surfaces(Hemberg and O'Reilly, 2004) and large tensegrity structures Rieffel et al. (2009).

Drawing an analogy between the edges of a graph (in 2-D) and the faces of a tetrahedron (3-D) our *face-encoding* grammar operates upon tetrahedral faces in much the same way that a Map L-system operates upon graph edges.

Assuming that each face of a tetrahedron can be given a label, there are three obvious operations which you can perform upon the faces of a tetrahedron, as illustrated in Figure 3. We will assume that operators can only be applied to *exposed* faces – that is, those which are not shared by two tetrahedra.

$A \rightarrow relabel(B)$ will replace a face labeled 'A' with a new face labeled 'B'

$A \rightarrow grow\{BCD\}$ replaces a face labeled 'A' with a new tetrahedron, labeling the new exposed faces as 'B','C', and 'D'.

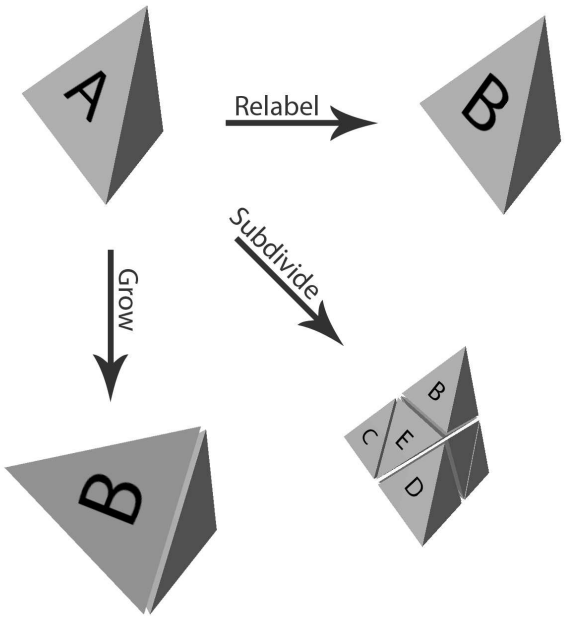

Figure 3: An illustration of the three rules which can be applied to the face of a tetrahedron. Clockwise from top left: the original tetrahedron with face labeled "A", **relabel** replaces "A" with "B", **subdivide** replaces the face with four smaller faces (this requires subdividing the entire tetrahedron), and **grow** adds a new tetrahedron with face labels "B","C","D"

$A \rightarrow divide[BCDE]$ subdivides a face 'A' into four smaller faces, 'B','C','D', and 'E'. The underlying tetrahedron must also be subdivided into eight component tetrahedra in to provide attachment points for the new faces and vertices.

Armed with these rules, we can now grow tetrahedral meshes of arbitrary size by iteratively applying them to an initial "seed" tetrahedron.

Each exposed face of the soft body kept in a queue, and is associated with three vertices (in counterclockwise order so that we can calculate surface normals) and exactly one tetrahedron. (A face can be shared by two tetrahedra, but then it wouldn't be exposed). For every generation of growth, the open faces are iteratively removed from the queue and the appropriate rule is applied. For $relabel$, a new face with the new label is enqueued. For $grow$ and $divide$, new vertices and tetrahedra are computed and added, and then the resulting three ($grow$) or four ($subdivided$) new faces are enqueued.

This entire cycle is repeated a fixed number of times to create progressively larger and more complex soft bodies. Figure 4 shows the iterative application of rewrite rules to

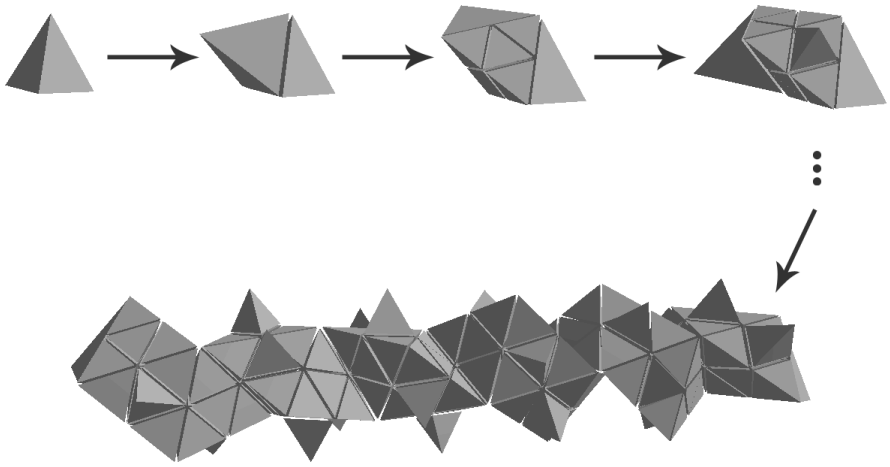

Figure 4: The growth of a larger tetrahedral mesh by iteratively applying a face-encoding grammar to an initial "seed" tetrahedron.

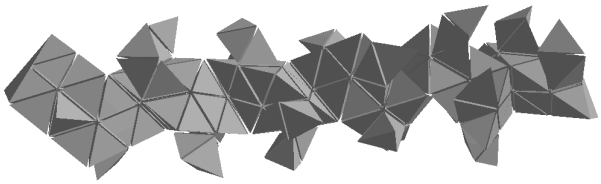

Figure 5: A small change in the grammar underlying the production of a tetrahedral mesh can produce profound and co-ordinatd change in the final result. The above figure was produced with a single mutation to the grammar which produced the mesh in Figure 4.

a single starting tetrahedron. Like in other grammatical encodings, a small change in a single production rule can have profound and co-ordinated effects upon the final product.

Technical Challenges

Although the rules may seem simple, there are several technicalities which may the implementation of a face encoding grammar difficult. First, as previously mentioned, when subdividing faces we also subdivide the associated tetrahedron. This is necessary because the new, smaller faces need new vertices and their own tetrahedra to attach to. While in principle it may be possible to subdivide less than the entire tetrahedron, during a divide, it requires more complicated bookkeeping, and the symmetry of our solution is appealing.

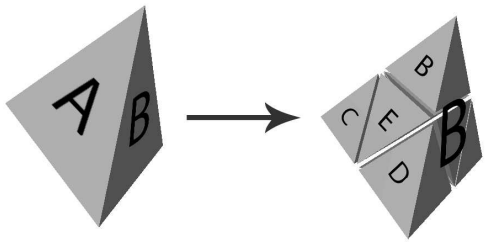

Figure 6: Subdividing a face "A" on the left hand tetrahedra actually requires splitting the entire tetrahedron. The remaining faces, such as "B" remain defined in terms of their original three vertices, and are left alone. Book-keeping must be maintained to ensure that any subsequent call to divide the original face "B" is aware that the underlying tetrahedron has already been split.

However, subdividing the full tetrahedron when a single face is divided raises the question of how to treat the remaining faces. In principle, we want to act as if the remaining faces still exist and are still associated with the original tetrahedron, even though the tetrahedron they belonged has been subdivided into smaller tetrahedra, as illustrated by Figure 6. This works fine as long as the other faces want to grow or relabel – they can proceed as usual, because to do either of those things doesn't rely on the underlying tetrahedron. A special case arises if a second original face wants to subdivide, in order to ensure the work isn't duplicated.

A similar scenario occurs when we want to subdivide two adjacent tetrahedra. Before subdivision they are connected only at their corners, but after subdivision they should be connected in the middle of each of the edges they share as well, where the smaller, subdivided tetrahedra are now adjacent to each other. We solve this problem and other similar special cases largely by ignoring them during growth, and then removing duplicate and redundant vertices during a post processing stage.

Examples

As an example of the complexity of features achievable with this grammatical encoding, consider the rule set shown in Table 1, and the soft body which results by iterating the grammar over a single "seed" tetrahedron 10 times, as shown in Figure 7.

A	$\rightarrow$	grow	$\{DBF\}$
B	$\rightarrow$	grow	$\{ADF\}$
C	$\rightarrow$	grow	$\{EDF\}$
D	$\rightarrow$	relabel	(D)
E	$\rightarrow$	grow	$\{DCF\}$
F	$\rightarrow$	divide	$[DDDG]$
G	$\rightarrow$	grow	$\{DDG\}$

Table 1: The rule set used to grow Figure 7.

At each iteration, faces labeled with an A, B, C, E, or G are grown, and the three new faces created are labeled as shown. For example, the faces of the tetrahedron grown from any A face will be labeled as D, B, and F faces. Face D, meanwhile, is relabeled as itself. This serves effectively as a no-op, and is a dead-end for growth. Face F is subdivided,into three dead-end D faces and one G face. In the final soft body, faces A, B, C, and E work together to grow the "legs" of the soft body, while the F and G faces work together to grow the smaller "tentacles" that protrude at every angle.

Figure 8 illustrates how further iterating the grammar in Table 1 20 times produces a structure which can be considered an elaboration of the smaller 10-step mesh of Figure 7

Discussion: Applications to Soft Robotics

Soft bodies, both natural and virtual, bring with them fascinating new questions about the relationship between morphology and control. Soft and deformable objects can possess near-infinite degrees of freedom, and elasticity in the system means that local perturbations can propagate to distal regions with interesting consequences. One might be inclined to think that this would create intractable control challenges, and yet the animal kingdom is full of soft and deformable animals. The *Manduca sexta* caterpillar, for instance, which might seem a relatively simple organism, is in fact rife with non-linearities and complex dynamics imposed by the interaction of hydrostatics, an elastic body wall, and nonlinear muscular behavior. New insights from biomechanics and neuro-ethology (Trimmer, 2007) suggest that rather than being hobbled by these complex dynamics, soft creatures in fact are able to exploit them as an advantage, via a form*morphological computation*(Valero-Cuevas et al., 2007; Pfeifer and Bongard, 2006).

This is particularly relevant to the budding field of soft robotics. Imagine a machine that can squeeze through holes, climb up walls, and flow around obstacles. Though it may sound like science fiction, thanks to modern advances in materials such as polymers (Huang et al., 2007), and nanocomposites (Capadona et al., 2008) such a "soft robot" is becoming an increasing possibility.

The largest outstanding problem in soft robotics is that while we possess the means to build them, no principled method exists to design or control them. There are no textbooks on soft robot design and control. And, while intuition suggests that the best way to control soft structures is, like caterpillars, to exploit their complex body dynamics via *morphological computation*, the dynamics are too complex to hand-code a solution.

The most promising approach is probably body-brain co-evolution (Pollack et al., 1999). The grammatical encoding we have presented here is a vital tool for the the co-evolution soft robotic design and control.

Conclusion

The face-encoding grammar presented in this paper provides us with a principled way of generating large and complex *structured* soft objects. The ability to generate complex and life-like soft structures (via this face encoding grammar) and to efficiently simulate them (via hardware-accelerated physics simulators) broadens the horizons of artificial life research, and provides entire new sources of bio-inspiration. Instead of mimicking (relatively) rigid vertebrates such dogs and horses, we can now begin to create artificial creatures which resemble octopii, squid, slugs and caterpillars.

References

Banzhaf, W. and Harding, S. (2009). Accelerating evolutionary computation with graphics processing units. In *GECCO '09: Proceedings of the 11th annual conference companion on Genetic and evolutionary computation conference*, pages 3237–3286, New York, NY, USA. ACM.

Bongard, J. and Pfeifer, R. (2003). *Morpho-functional Machines: The New Species (Designing Embodied Intelligence)*, chapter Evolving complete agents using artificial ontogeny, pages 237–258. Springer-Verlag, Berlin.

Capadona, J., Shanmuganathan, K., Tyler, D. J., and Rowan, S. (2008). Stimuli-responsive polymer nanocomposites inspired by the sea cucumber dermis. *Science*, 319(7).

Hemberg, M. and O'Reilly, U.-M. (2004). Extending grammatical evolution to evolve digital surfaces with genr8. In *EuroGP*.

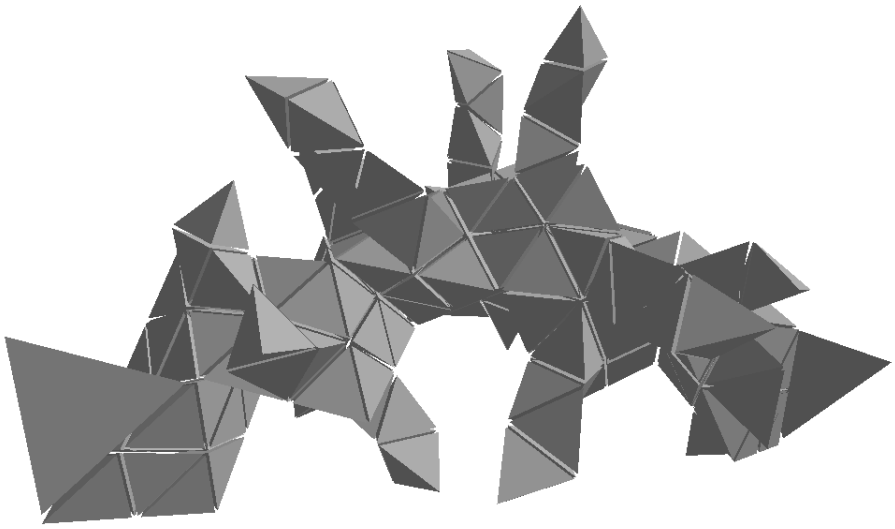

Figure 7: A larger grammatically-produced tetrahedral structure which shows several desirable features, most notably modular structure and varied tetrahedral resolution. This particular mesh was created by iterating the grammar in Table 1 10 times.

Hornby, G. S., Lipson, H., and Pollack, J. (2003). Generative encodings for the automated design of modular physical robots. *IEEE Transactions on Robotics and Automation*, 19(4):703–719.

Hornby, G. S. and Pollack, J. B. (2001). The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 600–607, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea. IEEE Press.

Huang, J., Foo, C. W. P., and Kaplan, D. (2007). Biosynthesis and applications of silk-like and collagen-like proteins. *Polymer Reviews*.

Jakobsen, T. (2001). Advanced character physics. In *Game Developer's Conference 2001*.

Lohn, J. D., Hornby, G. S., and Linden, D. S. (2005). An Evolved Antenna for Deployment on NASA's Space Technology 5 Mission. In O'Reilly, U.-M., Riolo, R. L., Yu, T., and Worzel, B., editors, *Genetic Programming Theory and Practice II*. Kluwer.

Luke, S. and Spector, L. (1996). Evolving graphs and networks with edge encoding: Preliminary report. In *Late Breaking Papers at the Genetic Programming 1996 Conference*, pages 117–124.

Pfeifer, R. and Bongard, J. C. (2006). *How the Body Shapes the Way We Think: A New View of Intelligence (Bradford Books)*. The MIT Press.

Pollack, J., Lipson, H., Funes, P., Ficici, S., and Hornby, G. (1999). Coevolutionary robotics. In *EH '99: Proceedings of the 1st NASA/DOD workshop on Evolvable Hardware*, page 208, Washington, DC, USA. IEEE Computer Society.

Pollack, J. B., Lipson, H., Hornby, G., and Funes, P. (2001). Three generations of automatically designed robots. *Artificial Life*, 7(3):215–223.

Prusinkiewicz, P. and Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Springer–Verlag, New York, USA.

Rieffel, J., Valero-Cuevas, F., and Lipson, H. (2009). Automated discovery and optimization of large irregular tensegrity structures. *Computers & Structures*, 87(5-6):368 – 379.

Rieffel, J. A., Valero-Cuevas, F. J., and Lipson, H. (2010). Morphological communication: exploiting coupled dynamics in a complex mechanical structure to achieve locomotion. *Journal of The Royal Society Interface*, 7(45):613–621.

Sims, K. (1994). Evolving 3d morphology and behavior by competition. In Brooks, R. and Maes, P., editors, *Artificial Life IV Proceedings*, pages 28–39. MIT Press.

Stanley, K. O. (2008). Generative and developmental systems. In *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, pages 2849–2864, New York, NY, USA. ACM.

Trimmer, B. (2007). New challengees in biorobotics: incorporating soft tissue into control systems. In *IEEE International Conference on Robotics and Automation*.

Valero-Cuevas, F., Yi, J., Brown, D., McNamara, R., Paul, C., and Lipson, H. (2007). The tendon network of the fingers performs anatomical computation at a macroscopic scale. *IEEE Trans Biomed Eng.*, 54:1161–6.

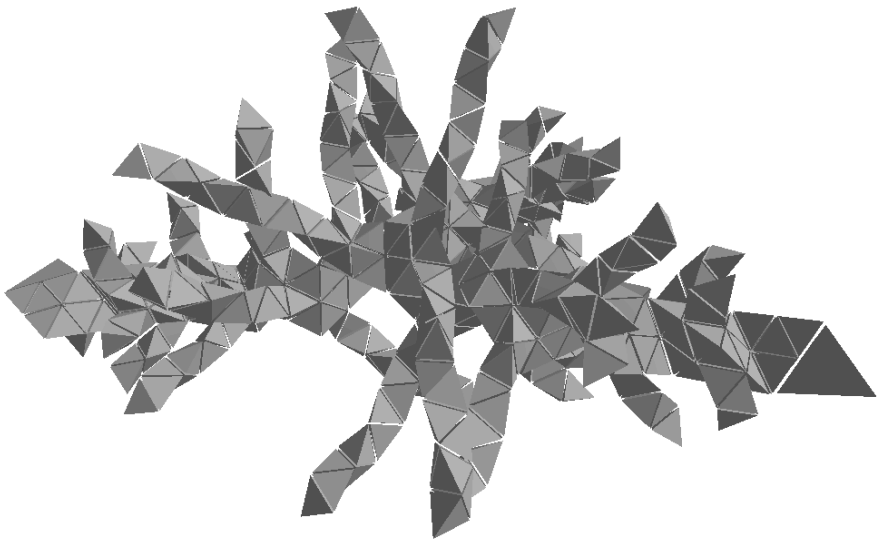

Figure 8: Continuing the growth of the grammar in Table 1 for another 10 cycles produces a mesh which is more elaborate that the earlier one in Figure 7, but which maintains much of the coarse structure.