Using DNA to Generate 3D Organic Art Forms

William Latham[1], Miki Shaw[1], Stephen Todd[1], Frederic Fol Leymarie[1], Ben Jefferys[2], and Lawrence Kelley[2]

[1] Computing, Goldsmiths College, University of London, UK
[w.latham,ffl]@gold.ac.uk
[2] Bioinformatics, Imperial College, London, UK

Abstract A novel biological software approach to define and evolve 3D computer art forms is described based on a re-implementation of the *FormGrow* system produced by Latham and Todd at IBM in the early 1990's. This original work is extended by using DNA sequences as the input to generate complex organic-like forms. The translation of the DNA data to 3D graphic form is performed by two contrasting processes, one intuitive and one informed by the biochemistry. The former involves the development of novel, but simple, look-up tables to generate a code list of functions such as the twisting, bending, stacking, and scaling and their associated parametric values such as angle and scale. The latter involvesan analysis of the biochemical properties of the proteins encoded by genes in DNA, which are used to control the parameters of a fixed *FormGrow* structure. The resulting 3D data sets are then rendered using conventional techniques to create visually appealing art forms. The system maps DNA data into an alternative multi-dimensional space with strong graphic visual features such as intricate branching structures and complex folding. The potential use in scientific visualisation is illustrated by two examples. Forms representing the sickle cell anaemia mutation demonstrate how a point mutation can have a dramatic effect. An animation illustrating the divergent evolution of two proteins with a common ancestor provides a compelling view of an evolutionary process lost in millions of years of natural history.

1 Introduction

We present a novel biological approach to define and evolve 3D art forms. The work combines a re-implementation of the *FormGrow* system of Todd and Latham [1] with an external source to define the shapes: DNA sequences. *FormGrow* is a virtual machine producing 3D computer art forms or designs. It embodies the particular organic aesthetics favored by Latham together with a shape grammar made of primitives (horn-like structures), transforms and assembly rules, and a number of parameters encoding color, scale, texture. We have re-visited the *FormGrow* system of Latham and Todd adapting it to a modern implementation taking advantage of standard graphics libraries and portable coding, and putting the emphasis on bringing this system closer to the realm of biology.

Two methods for using real DNA data, in the form of nucleotide sequences, were devised. In the first, sequences are directly transformed via a series of empirically designed tables to become readable by *FormGrow*. These tables process nucleotides as

codon triplets of data as would ribosomes in a live cell. Notions of "start," "stop," and "junk" DNA code are also embedded in our system. We explore the application of our novel method to generate 3D organic art forms in the visualisation of particular genetic defects, and present as a case study the well-known sickle cell anaemia mutation. The second method for interpreting DNA sequences is to look at the biochemical properties of the amino acids which are encoded by the codons in a gene. Simple counts of the amino acids with certain properties, or of a certain type, are turned into parameters for a fixed *FormGrow* structure. We demonstrate the use of such an interpretation with an animation illustrating the evolution of a pair of related proteins over billions of years. The precise sequence and structure of the ancestral proteins is unknown, but sophisticated tools and an artistic intepretation of the data gives a glimpse of a process which is lost in eons of evolutionary history.

2 Background

In the 1980's. while at the Royal College of Art in London, Latham devised a rule-based hand-drawn evolution system called *FormSynth* [2]. He then joined forces with Stephen Todd at IBM to develop the *FormGrow* and *Mutator* systems from 1987 to 1993 [3,4,1,5,6]. Our work started from an open-ended aim to revisit this project which had been untouched for about twelve years.

FormGrow is a kind of building blocks kit for creating organic-style 3D computer generated forms. It uses a hierarchical system, building up complex forms from primitive shapes. The central *FormGrow* construct is a horn which consists of n ribs: repeated primitive shapes. Variants of the basic horn are made by applying elementary transforms: stack, bend, twist, and grow. *Mutator* allows forms to be grown using life-like techniques such as cross-fertilisation (marriage) and mutation. A form — as obtained via the *FormGrow* system — is expressed as a sequential set of instructions, which constitute its encoding. *Mutator* reads *FormGrow* instructions to be combined (if coming from various parents) or modified (simulating mutations). In the original work, the survival of a form was governed by human selection — typically embodied by the artist Latham seen as a kind of gardener of art-forms — or by closeness to some pre-defined measure. Latham and Todd's work during the period of 1987–93 coincided in particular with the works of Sims on 2D and 3D forms [7,8], of Prusinkiewicz and Lindenmayer on plants [9], and of Leyton on process grammars [10,11]. Discussion of the differences and similarities of such systems are covered, *e.g.*, in [6,12].

A mathematical and computational formalism which unites these shape generative systems is that of *shape grammars* whereby objects of various complexities can be generated by an iteration of a finite number of simple outline transformation instructions, such as *FormGrow*'s bends, twists and stacks. Various shape grammars have been developed in the literature. For example, the generation of self–similar fractal objects is possible with very simple grammars [13, §8.1]. Selection rules, which forbid the addition of a sub-unit under certain conditions, have been used by Ulam to generate less regular patterns [13, §8.2]. Trees and river systems, crystals, tessellations and space filling organisations are other examples of domain of applications of shape grammars as object generators [13].

An important example of early work is to be found in *L-systems*, also called Lindenmayer systems or *parallel string-rewrite systems*, which are made from productions rules used to define a tracing of piecewise linear segments with joints parameterised by rotation angles [14,9]. These rules also are a compact way to iteratively repeat constructive sequences in the description of fractals, often used to model groups of plants, flowers, leaves, and so on [15].

One can generalise shape grammars within the context of cellular automata where some randomisation is introduced in the manifestation of the rules leading to *dynamic* shapes; for example see the works of Wolfram *et al.* [16] and more recent studies in biological pattern genesis [17]. The combination of dynamical L-systems with cellular automata has been considered, in particular in the works of Jon McCormack with application to art form genesis [18].

Another possible generalisation is in the context of genetic programming where mutations and the natural mixing of a pool of genes (possibly representing shape components or features) is used to obtain evolving *natural* or organic shapes; for example see the early works of Dawkins on biomorphs [19], and, again, of Latham and Todd on genetic art [1], and more recent works in art, design [6] and biological sciences [20].

Our motivation for re-visiting Latham and Todd's work is that it is a powerful system which offers the possibility of generating organic-like shapes and which from its origins was meant as a metaphor to nature's way of evolving forms. In re-visiting this work, on the one hand we bring up-to-date the technology developed in [1] in the context of recent advances in graphics and computational geometry, and on the other hand we bring it much closer to biology via the recent advances made in understanding the working of nature in the fields of genomics and proteomics, the focus of this paper.

3 Use of DNA in *FormGrow*

DNA can be thought of as a shape-specification language residing in the cells of every living organism, encoding proteins which constitute the body's key builders and building blocks. The DNA molecule is essentially a very long string of much smaller molecules, the nucleotides, which come in four varieties (A, C, T, G).

How does this apparently simple string of nucleotides encode the complex form of a protein? A protein is also a string of simpler molecules: the amino acids. As there are 20 types of amino acids and only 4 types of nucleotides, the DNA translation mechanism looks at nucleotides in groups of three, triplets called "codons;" every codon translates to a single amino acid [20]. Working down the chain of DNA generates the corresponding chain of amino acids, yielding a protein. The codon-amino acid equivalences can be represented in a translation table (Tbl.1).

Following this model, we created an analogous translation system to convert DNA sequences into *FormGrow* code. At a coarse level, *FormGrow* code can be viewed as a series of function calls, with each function requiring a small number of arguments (this number varies from 0 to 3 depending on the particular function). Thus, we created 2 translation tables: the "transform table," which translates from codons to transformational functions (Tbl.2); and the "number table" (Tbl.3), which translates from codons to numerical arguments (integers in the range 0 to 63). Given our input sequence, we

Second letter of codon

First letter of codon	T	C	A	G	Third letter of codon
T	Phe Phe Leu Leu	Ser Ser Ser Ser	Tyr Tyr Stop Stop	Cys Cys Stop Trp	T C A G
C	Leu Leu Leu Leu	Pro Pro Pro Pro	His His Gln Gln	Arg Arg Arg Arg	T C A G
A	Ile Ile Ile Met (start)	Thr Thr Thr Thr	Asn Asn Lys Lys	Ser Ser Arg Arg	T C A G
G	Val Val Val Val	Ala Ala Ala Ala	Asp Asp Glu Glu	Gly Gly Gly Gly	T C A G

E.g. GCA => Ala (Alanine)

Table 1. Translating Codons to Amino Acids.

Second letter of codon

First letter of codon	A	C	G	T
A	Add Horn	Recurse Horn	Grow	
C		Stop		
G	Stack		Bend	
T		TCA TCC TCG TCT	Twist	

E.g. TCG => Stack

Table 2. Codons to *FormGrow* transforms.

Second letter of codon

First letter of codon	A	C	G	T
A	AAA = 0 AAC = 1 AAG = 2 AAT = 3	ACA = 4 ...	ACG = 8 ...	ACT = 12 ...
C	CAA = 16 ...	CCA = 20 ...	CGA = 24 ...	CTA = 28 ...
G	GAA = 32 ...	GCA = 36 ...	GGA = 40 ...	GTA = 44 ...
T	TAA = 48 ...	TCA = 52 TCC = 53 TCG = 54 TCT = 55	TGA = 56 ...	TTA = 60 TTC = 61 TTG = 62 TTT = 63

E.g. TCG => 54

Table 3. Translating Codons to numbers.

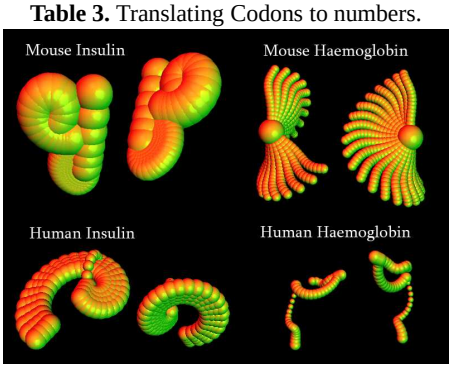

Figure 1. Images generated from real DNA sequences.

1. Get nucleotide sequence — caggcctcatcgacctct...
2. Group into 3-letter codons — cag gcc tca tcg acc tct
3. Translate codon into a transform — Add Horn(,)
4. Translate the next codons into numerical arguments, until enough inputs are obtained for the transform — Add Horn(37,52)
5. Repeat steps 3 and 4 to translate the entire nucleotide sequence — Add Horn(30,52) Stack (5) Twist (14) ...
6. Translate to JavaFormGrow code

```
Horn h1 = new Horn(30,52);
h1.stack(stackVal(5));
h1.twist(twistVal(14));
...
```

Table 4. Translating a Codon Sequence to *FormGrow* code.

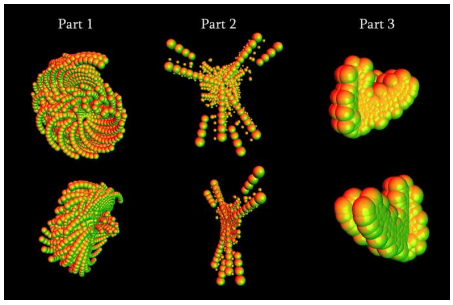

Figure 2. Images generated from sections of the mouse keratin DNA.

translate the first codon into a function using the transform table, and then generate numerical arguments for that function by translating the following codons into numbers, using the number table. Once we have sufficient arguments, we return to the transform table to generate our next function, and so the cycle continues (Tbl.4). Finally we render the generated *FormGrow* code to produce a 3D shape. Figures 1 and 2 show some images generated from genuine DNA sequences.

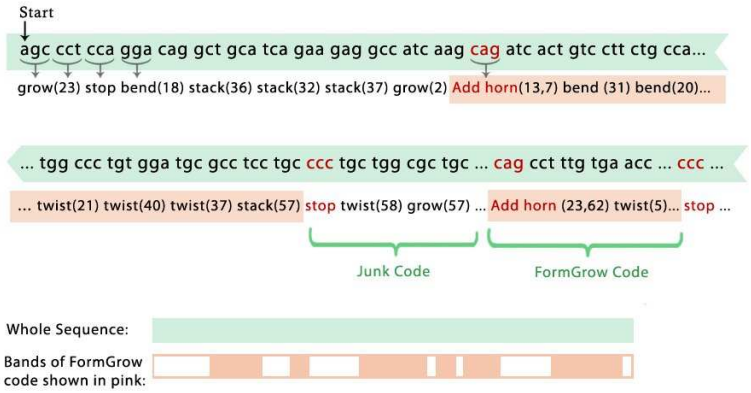

Figure 3. Translating an example DNA sequence (Human Insulin).

It is interesting to note some similarities between nature's translation method and ours, these being features which we needed in our system and realised that the biological precedents were well worth adopting. In the original translation table there is a "start" codon (AUG) which signals that a new protein is being specified. Likewise, in our transform table, the "add horn" transform flags the beginning of a new shape. The "stop" codon is also mirrored in our system. This instructs termination of the current protein or shape. A side effect of adopting the "start" and "stop" mechanism is that we end up with large sections of "junk code," *i.e.*, code which generates no proteins or shapes because it lies in a non-coding section of the sequence (Fig.3). By changing the layout of the transform table we could affect the proportion of junk code produced. We experimented with producing a few different iterations of the transform table in order to get a balance of functions that would produce a visually interesting variety of shapes.

3.1 Case study: Sickle Cell Anaemia

Using this novel method of converting DNA into 3D shapes, we wondered if we could compare different DNA sequences. We selected as our case study the gene for sickle cell anaemia. This inherited disease affects millions of people worldwide. It damages the red blood cells which deliver oxygen to vital organs, resulting in anaemia and further complications. It is particularly common in malarial regions, because it offers some protection against malaria. All this is caused by one faulty gene. The problem appears as a single point mutation in the beta haemoglobin gene: a single "A" nucleotide is changed

to a “T” (Fig.4). The reason that this single nucleotide substitution is so influential is because “GTG” encodes a different amino acid to “GAG.” And this amino acid switch changes the physical behaviour of haemoglobin in the body. In our initial transform table, there is no difference between “GAG” and “GTG,” so both normal and sickle cell forms of the DNA sequence generated identical shapes. It is not unusual for a single mutation to go unregistered. Both the amino acid table and the transform table exhibit some redundancy — in fact, there is an evolutionary advantage in this redundancy, as it makes DNA more resistant to minor changes. However the transform table only produces 7 different output functions (unlike the amino acid table which has 21), so more repetition is inevitable. Rather than adjusting the table further by hand, we applied a procedure to randomise it.

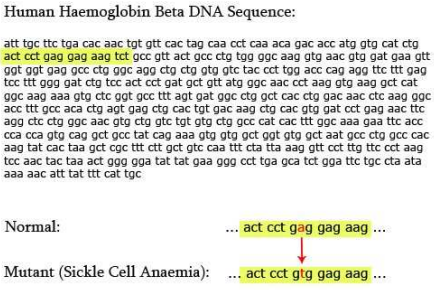

Figure 4. Sickle Cell Anaemia Mutation.

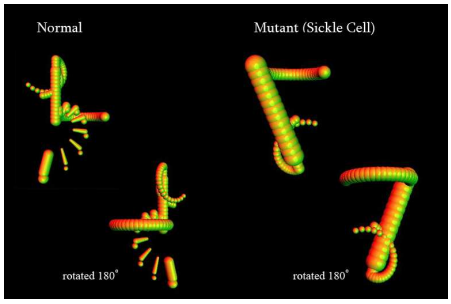

Figure 5. Forms generated from Normal and Sickle Cell Beta Haemoglobin.

After a small number of randomisation runs, a table was produced which translated “GTG” and “GAG” differently. This new table generated the images shown in Fig.5. The two forms are easily distinguished, though many similarities can be seen in their component parts. Effectively we had produced an alternative way of visualising this genetic mutation. It is an artistic impression of how a point mutation can have such a dramatic effect on phenotype. Ideally we would like to optimise the table such that visualisation reflects the sequence in some expected or sensible manner for a large set of proteins and mutations. This optimisation may take place using any standard algorithm, for example an evolutionary method might be appropriate, making stepwise changes in the table and asking a human (artist or biologist) if the resulting visualisations are an improvement on those produced by the parent transform table.

4 Use of amino acid biochemistry in *FormGrow*

The approach of directly using DNA sequences (interpreted as codons) to generate *FormGrow* shapes ignores the biochemical characteristics of the amino acids which the codons represent. Amino acids are the building blocks of proteins, which are the basic product of the genes encoded in DNA. A protein, in the form of a chains of amino

acids, folds into a specific shape, governed by the properties of the amino acids. The relationship between the amino acid sequence of a protein and the way the protein folds is complex and is probably the most fundamental unsolved problem in biology. Sometimes a point mutation, such as that in sickle cell anaemia, has a fundamental and devestating effect on the protein structure. But sometimes it has no effect, or a very small effect. It is often impossible to predict which will happen, and therefore adjust the *FormGrow* output to reflect the importance of a mutation. But we can say that the general nature of the fold depends upon the general make-up of the amino acid chain. Therefore if we summarise the amino acids content of a protein into a set of numbers, this provides a reasonable overview of the nature of the protein, and how it is related to other proteins. The 20 amino acids used in proteins can be grouped many ways according to their biochemical characteristics. These groupings are illustrated in Fig.6 (after [21]). This approach leads to a two-step process for creating *FormGrow* structures from genes. First, the DNA sequence for the gene is converted into a *histogram* denoting the relative frequencies of each amino acid type and grouping. We summarise the protein by counting how many of its amino acids fall into each group and also count the amino acids of each type. This produces a histogram "profile" of the protein's amino acid content. Secondly, these values are used as input into a fixed *FormGrow* structure. The process is illustrated in Fig.7.

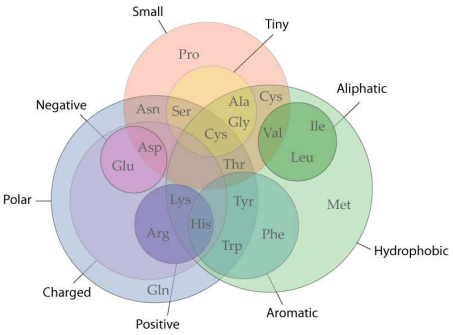

Figure 6. Venn diagram.

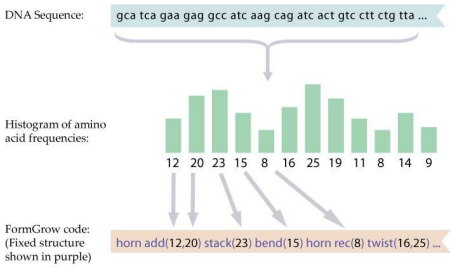

Figure 7. Histogram of biochemical properties derived from the DNA sequence for a gene is used to control parameters of a fixed *FormGrow* structure

The advantages of this approach are twofold. Firstly, the most reliable significant information from the protein is captured in the histogram which captures features of the entire sequence. Any further interpretation of the sequence would rely on predictive techniques which carry some uncertainty. This technique is used in bioinformatics and is related to the "spectral decomposition" approach [22]. It is alike a Fourier transform in that the amino acid frequencies are similar to first order terms of a Fourier transform, the dipeptide frequencies are akin to second order terms, etc. Furthermore, we do not need to worry about frame-shifts of the sequence using this technique, which was one problem with the original simpler codon model. Secondly, using a fixed structure means two shapes can be directly compared visually, and can be morphed in-between

to create a smooth animation. Therefore, the differences in amino acid composition of two proteins can be shown in a compelling new way.

4.1 Case study: Evolution of related proteins

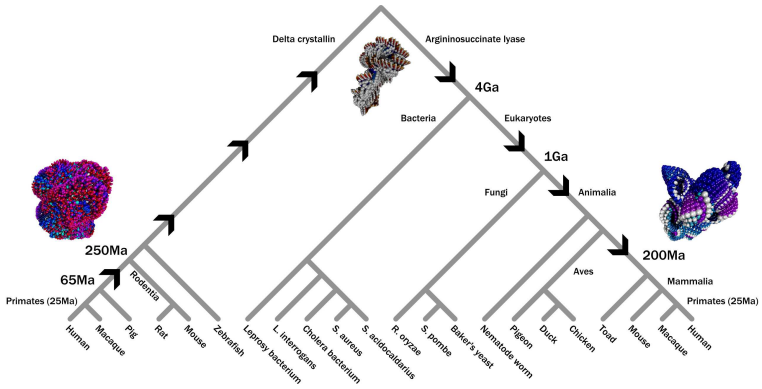

Figure 8. Divergent evolution of two proteins. This is not a conventional evolutioary tree. It shows how *ancescon* thinks the proteins are related, based upon the sequence of the proteins in modern organisms. This does not necessarily correspond to the true evolutionary tree, since the signal of evolution in protein sequence is sometimes lost in the “noise” of mutations over billions of years. The arrows show the route of the History of the Species film in visualising the devolution of one protein then the evolution of another.

Proteins evolve within organisms, with random mutations in DNA either causing the death of the organism, or increasing its chances of survival, or (usually) having no particular effect. A protein with one function can evolve into one with a different function through accumulation of mutations. Sometimes a whole gene is duplicated, meaning that one copy can continue to perform the original function, whilst the other can evolve to do something else. We looked at one such case as an interesting test of our histogram-based translation. Argininosuccinate lyase is a protein involved in producing argnine, which is one of the twenty amino acids from which proteins are made. In producing arginine, it consumes nitrogen, a product of many activities in the cell which can be toxic to an organism if it turns into ammonia. This protein exists in most organisms, from single-celled bacteria to humans and other apes. Approximately 450 million years ago, this protein was duplicated in a common ancestor of all animals, possibly some mobile multi-cellular organism. The duplicate accumulated mutations which eventually turned the protein into a structure, Delta crystallin, which, when fitted together in a specific pattern with many other proteins, forms part of the lens of the eye. The original protein has become important in removing nitrogen in the liver. The outline of this evolutionary history is given in Fig.8. We wanted to visualise this history using *FormGrow*. We chose to trace the development of the protein in the eye, backwards in evolutionary history, to the common ancestor from which it evolved. From there, we

wanted to trace the evolution of this into the protein used for removing nitrogen in the liver. The path through evolution we took is shown by the arrows in Fig.8.

The ancestral sequence are unknown — we only have the protein sequences from modern organisms. However, the ancestral sequences can be reconstructed, with some uncertainty, from the sequences in modern day organisms. For example, the protein sequence of 25 million years ago can be reconstructed by combining the sequence from humans with the sequence from the crab-eating macaque. The result is the sequence as it may have existed in a common ancestor of the primates. The sequence from 65 million years ago can be reconstructed from that sequence by combining it with the sequence from the pig. We used *ancescon* [23] to construct the ancestral sequences. Initial sequences came from a number of sources, including Pfam [24], and UniProt [25].

We can then morph through forms created for all the ancestral sequences from human eye, back to the earliest ancestral sequence, and then forwards to the human liver. Stills from the resulting film are shown in Fig.9. The film is available to view at [http://hos.mrg-gold.com] and covers up to 50 million years per second. The soundscape for the film was generated from the forms themselves, therefore both the audio and visual elements of the film are inspired by biology.

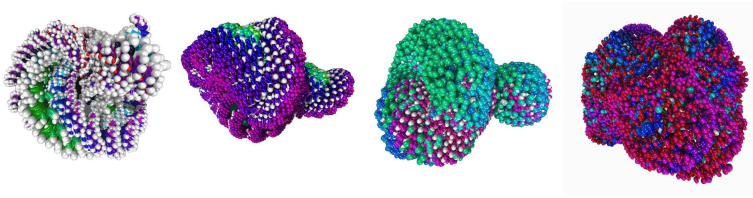

Figure 9. Stills from the animation “History of the Species.”

5 Discussion/Conclusion

At the core of this paper is a simple idea of feeding DNA data sequences into a rich 3D form generator called *FormGrow*, to generate organic-looking 3D growth structure, creating an equivalence of the DNA mapped into an alternative multi-dimensional space. How useful this mapped equivalence may be will become clearer as we work closer with biologists and engage in further cross-fertilization of ideas. Could this methodology have more direct and short-term scientific applications as well? While our shapes bear no resemblance to the proteins that the genes encode, they are still being driven by the same initial DNA sequences. So it is possible that we could use our system as a visualisation tool. Conceivably, our tool could enable users to identify whether two given sequences are similar or identical. The advantage of this tool being that it is faster and easier for the human eye to compare shapes than repetitive string sequences. The primary method used by researchers in bioinformatics is to look at a multiple sequence alignment, *i.e.*, all the sequences are simply lined up one beneath the other according to

an empirical scoring function. This alignment is often turned into a statistical profile or hidden Markov model which are useful for sequence matching and structure prediction, but there is no attempt at visualisation.

Additionally, our system is deterministic. Thus, given a sequence and transform, the same shape will result every time. But, in the case of direct transformation of a DNA sequences using a table, redundancy means that some small changes may go undetected. To rectify this we could produce more transform variants and copy the layout of the amino acid table, so that the redundancy locations are the same. *FormGrow* produces shapes which are nothing like the proteins which are actually encoded by genes. Proteins inhabit a completely different "shape space" to the multicellular organisms which *FormGrow* is inspired by. Could shapes inspired by proteins rather than entire organisms be as rich a tool for artistry and visualisation as *FormGrow*? We intend to answer this question in future work.

References

1. Todd, S., Latham, W.: Evolutionary Art and Computers. Academic Press (1992)
2. Latham, W.: Form Synth. In: Computers in Art, Design and Animation. Springer (1989)
3. Latham, W., Todd, S.: Computer sculpture. IBM Systems Journal **28**(4) (1989) 682–688
4. Burridge, J.M., et al.: The WINSOM solid modeller. IBM Systems Journal **28**(4) (1989)
5. Todd, S., Latham, W.: Artificial life or surreal art? In Varela, F.J., Bourgine, P., eds.: Toward a Practice of Autonomous Systems. MIT Press (A Bradford Book) (1992) 504–513
6. Bentley, P.J., ed.: Evolutionary Design by Computers. Morgan Kaufmann (1999)
7. Sims, K.: Artificial evolution for computer graphics. Computer Graphics **25**(4) (1991)
8. Sims, K.: Evolving 3D morphology and behavior. In: Proc. of Artificial Life IV. (1994)
9. Prusinkiewicz, P., Lindenmayer, A.: The Algorithmics Beauty of Plants. Springer (1990)
10. Leyton, M.: A process grammar for shape. A.I. Journal **34**(2) (1988) 213–247
11. Leyton, M.: A Generative Theory of Shape. Number LNCS 2145. Springer-Verlag (2001)
12. Whitelaw, M.: Metacreation — Art and Artificial Life. MIT Press (2004)
13. Lord, E.A., Wilson, C.B.: Math. Description of Shape and Form. Halsted Press (1984)
14. Lindenmayer, A.: Mathematical models for cellular interactions in development: Parts I and II. Journal of Theoretical Biology **18** (1968) 280–315
15. Ferraro, P., et al.: Toward a quantification of self-similarity in plants. Fractals **13**(2) (2005)
16. Wolfram, S.: Cellular Automata and Complexity: Collected Papers. Addison-Wesley (1994)
17. Deutsch, A., Dormann, S.: Cellular Automaton Modeling of Biological Pattern Formation. Modeling and Simulation in Science, Engineering and Technology. Birkhäuser (2005)
18. McCormack, J.: Aesthetic evolution of L-systems revisited. In: EvoMUSART Contributions. Volume 3005 of Lecture Notes in Computer Science. Springer (2004) 477–488
19. Dawkins, R.: The Blind Watchmaker. (1986)
20. Kumar, S., Bentley, P.J., eds.: On Growth, Form and Computers. Elsevier (2003)
21. Taylor, W.R.: The classification of amino acid conservation. J. Theor. Biology **119** (1986)
22. Shamim, M.T.A., et al.: Support vector machine-based classification of protein folds. Bioinformatics **23**(24) (2007) 3320–3327
23. Cai, W., Pei, J., Grishin, N.V.: Reconstruction of ancestral protein sequences and its applications. BMC Evolutionary Biology **4**(33) (2004)
24. Finn, R.F., et al.: Pfam: clans, web tools and services. Nucleic Acids Research **34** (2006) D247–51
25. Wu, C.H., et al.: The universal protein resource (uniprot). Nucleic Acids Research **34** (2006) D187–91