Polyhedron Evolver—Evolution of 3D Shapes with *Evolvica*

Christian J. Jacob and Aamer Nazir
Department of Computer Science, University of Calgary
2500 University Drive N.W., Calgary, T2N 1N4, CANADA
jacob@cpsc.ucalgary.ca

ABSTRACT

We present a programming environment for interactive, evolutionary design by genetic programming. The feasibility of using a stochastic system, that mimicks evolution through mutation and recombination, for automatic construction of design programs is demonstrated by example of evolved polyhedral objects.

Keywords: Evolutionary design, genetic programming, creativity through evolution, evolutionary optimization.

DESIGN THROUGH EVOLUTION

Nature has been creating fascinating designs of its organisms since life first appeared on our planet about 4 billion years ago. The enormous potential of nature's creative design ideas originates from a surprisingly simple experimental technique: random mutation and selection through evolution. Therefore, it is no wonder that evolutionary principles have inspired engineers and computer scientists alike, in order to find better ways to cope with increasingly complex system designs.

Genetic Programming

However, most of these "design" problems have only been considered as parameter optimization tasks. Only recently has evolutionary computation started to utilize more sophisticated ways of specifying designs through "programs." Prominent examples in this area of evolutionary computation are electronic circuit designs [7], architectural layout, or furniture design [1]. Genetic L-system Programming, which is implemented in our *Evolvica* framework, is another recent approach demonstrating how to use evolutionary techniques to breed growth programs for plant structures, encoded by biologically motivated parallel rule systems in the form of Lindenmayer systems [2]. The symbolic computation environment of *Mathematica* [9] provides an excellent platform for implementing the major state-of-the-art classes of evolutionary computation (EC): genetic programming, genetic algorithms, evolution strategies, and evolutionary programming [6]. The last three of these EC classes are mainly concerned with parameter optimization. In this paper we focus on genetic programming (GP) for developing or "breeding" design programs for 3-dimensional shapes with known mathematical descriptions, such as polyhedra.

Evolutionary Design

Evolvica is a package of notebooks for evolutionary computation in *Mathematica* [6]. In particular, *Evolvica* provides packages (libraries) for evolutionary computation on symbolic expressions. We have recently extended *Evolvica* by the *Polyhedron Evolver* package, an interactive application for the evolutionary design of 3-dimensional shapes using genetic programming techniques. The general idea behind *Polyhedron Evolver* is to support designers, artists, and engineers in their efforts to both explore new design ideas as well as optimize already accomplished designs. We demonstrate that genetic programming, through simple selection and mutation, following Darwin's principles of natural evolution, can serve designers in both of these purposes: (1) providing a tool to enhance creativity and (2) supporting fine-tuning within a set of predefined constraints.

We have chosen to use 3D shape design and, in particular, the design of polyhedra as our example application, as these very well illustrate the power of evolutionary design in combination with *Mathematica*'s sophisticated library of graphics functions, which make it relatively easy to generate a large variety of three dimensional forms.

THE POLYHEDRON EVOLVER

Exploration of Design Spaces

The Polyhedron Explorer from the *Mathematica* Help Browser is an excellent example of how to explore an extensive design space [8]. Starting from basic polyhedral shapes—such as tetrahedron, cube, dodecahedron, etc.—the Explorer provides a set of functions to manipulate these designs by adding stellating effects, or shrinking, truncating and indenting selected polygonal shapes. Through a simple button interface, the Polyhedron Explorer enables a user to step through a polyhedral design space, where each design is described by a symbolic expression that contains the graphics manipulation functions in *Mathematica* (Fig. 1). Consequently, the Explorer does not only graphically display the resulting polyhedral shape, but also provides access to an associated program that generates the particular polyhedron design.

One drawback of the Polyhedron Explorer is, however,

that all the designs have to be created by hand. Instead, wouldn't it be nice to have a tool that allows a much more efficient and faster way of browsing through a polyhedral shape space? In our view, an evolutionary system provides a perfect basis for the exploration of this and other design spaces. The *Polyhedron Evolver* is our first step in implementing such an evolutionary design system, the details of which we describe in the following sections.

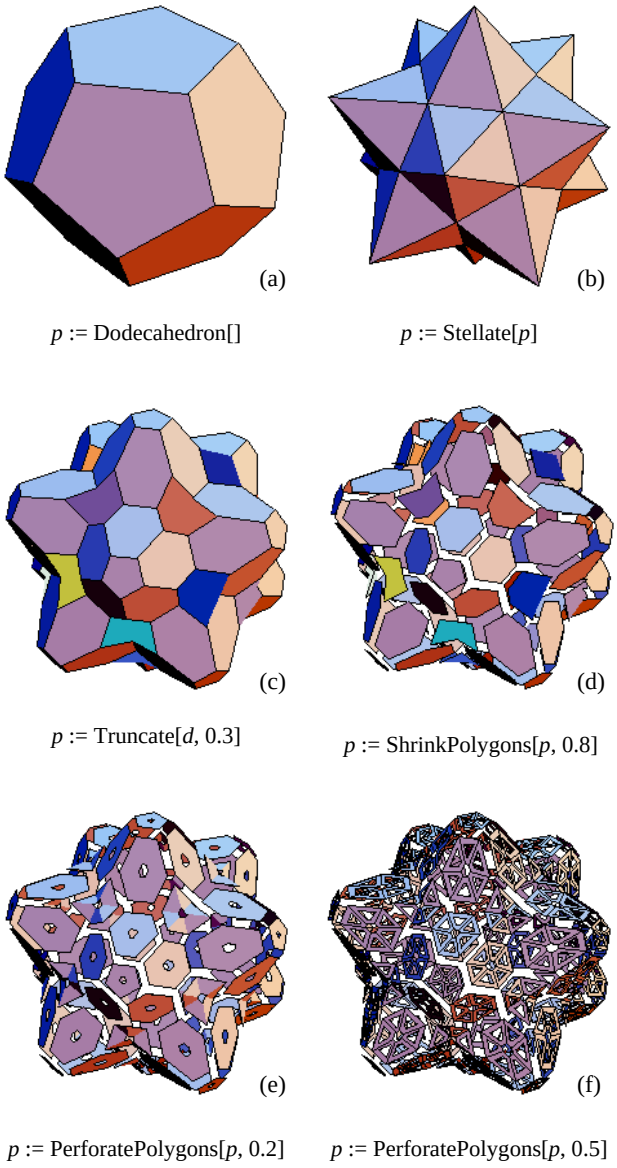


Fig. 1. Examples of a step-by-step composition of design programs describing transformations on a polyhedral object (dodecahedron).

Creation of Design Programs

An evolutionary system has to work on some sort of "genetic representation" that describes the building blocks as the entities from which design desriptions or programs can be composed. In the case of evolvable polyhedral shapes we use templates to specify both the building blocks and their possible compositions (Fig. 2).

```
poly [ _Graphics3D6| _Stellate1 | _Truncate1 |
     | _OpenTruncate1 | _PerforatePolygons2
     | _ShrinkPolygons2 | _Geodesate2 ],

Stellate[_poly, _pOpts],
Truncate[_poly, _pOpts],
...
Geodesate[_poly],

Graphics3D[_object],

object [ Tetrahedron[]1 | Cube[]1
       | Octahedron[]2 | Dodecahedron[]2
       | Icosahedron[]3 ]

pOpts[ Random[Real, {0.1, 0.5}] ]
```

Fig. 2. Building blocks for programs of polyhedral objects.

When (polyhedral) design programs (Fig. 1) have to be built from scratch, as is usually the case for the initialization of an evolutionary system, these templates are used to recursively build random compositions. Furthermore, each template has an associated weight (subscripts in Fig. 2), such that during the composition process some templates can be given higher preference over others. Starting with the template *_poly*, for example, the building blocks in Fig. 2 provide six different templates (*poly[_Graphics3D]*, ..., *poly[_Geodesate]*), where the last three are given a slightly higher rank.[1] So, the system might choose

poly[_PerforatePolygons]

as a first intermediate step. Using the only matching expression for *_PerforatePolygons*, this template would then be extended into

poly[PerforatePolygons[_poly, _pOpts]].

Now, again, we have the choice among the *_poly* expressions, which might be expanded to

poly[PerforatePolygons[Stellate[_poly, _pOpts], _pOpts]].

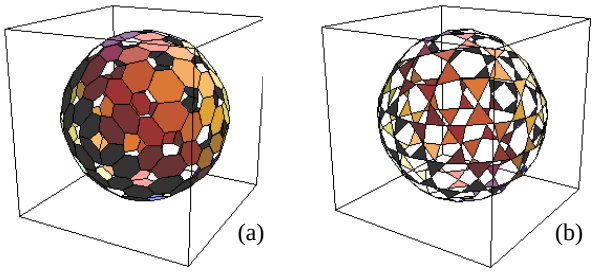
This recursive composition process only stops when the *Graphics3D* pattern is chosen, which then leads to the final insertion of a basic polyhedral shape, the selection

1. We are using *Mathematica*'s pattern notation, where '*_x*' denotes a pattern that matches any symbolic expression with a head of *x*. For example, *_Sin* would match *Sin[a]*, *Sin[Cos[t]]*, *Sin[π]*, etc. The '|' denotes alternative expressions, such that the pattern *f[a | b]* matches both *f[a]* and *f[b]*.

of which can also be controlled using the template weights. The remaining patterns denote additional parameters for the modifier functions, such that one might end up with the polyhedral design program

poly[PerforatePolygons[Stellate[Cube[], 0.3], 0.45]].

Figure 1 shows a more complicated example of such an expression that could be composed in a similar manner. This composition process is used both by the mutation operators (whenever new sub-expressions have to be created) and for generating random designs that consitute the initial set of design programs.



OpenTruncate[Geodesate[Geodesate[Octahedron[]]],0.3]

OpenTruncate[Geodesate[Geodesate[Octahedron[]]],**0.5**]

Geodesate[OpenTruncate[Geodesate[Geodesate[Octahedron[]]],**0.4**]]

Geodesate[OpenTruncate[Geodesate[**Stellate**[Octahedron[]], **0.3**],0.4]]

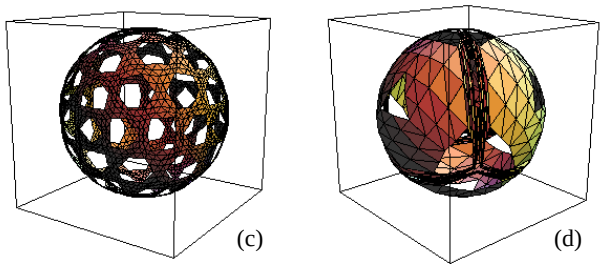


Fig. 3. Step-by-step variations involving both creative and fine-tuning mutations.

DESIGN VARIATIONS

Starting from a set of randomly generated initial design expressions, composed recursively as desribed in the previous section, the user can evaluate each shape on a scale from 0 to 10. These assigned “fitness” values determine the survival of the programs into the next iteration. Using various selection functions (fitness proportionate, elitist, rank-based, etc.), programs that receive a higher rating have a better chance of creating offspring designs [6].

Filters and Genetic Operators

Analogous to the templates that are used to define the building blocks to generate random design programs, the variational operators, such as mutation and recombination, use weighted templates as filters to identify specific subexpressions, on which the variations are allowed to occur. In the case of recombinations, pattern matching mechanisms are also used to specify the compatibility of expressions that can be interchanged without violating any syntactic constraints. Each genetic operator is associated with a weight for automatic operator selection, or can be switched on/off manually at any time during the evolution process.

In *Polyhedron Evolver* there are two categories of mutation operators for generating new variants of already evolved shapes—*creative* mutations and *fine-tuning* mutations.

Creative Mutations

Usually, during the initial phase of the breeding process, the evolutionary system attempts to present "innovative" design ideas, accomplished by major structural variations on the expressions. Mutations may (1) change the basic polyhedral shape, (2) exchange a modifier function for another, (3) delete a modifier function, (4) insert a new modifier function, or (5) replace a subexpression by a completely new expression. Some of these modifications are illustrated in Figure 3. For example, in the mutation from shape expression (b) to expression (c) another *Geodesate* modifier was added. The mutation from (c) to (d) replaces the inner *Geodesate* by *Stellate*, which also introduces a new parameter. Creative mutations like these should happen during the initial, exploratory phase of the design process.

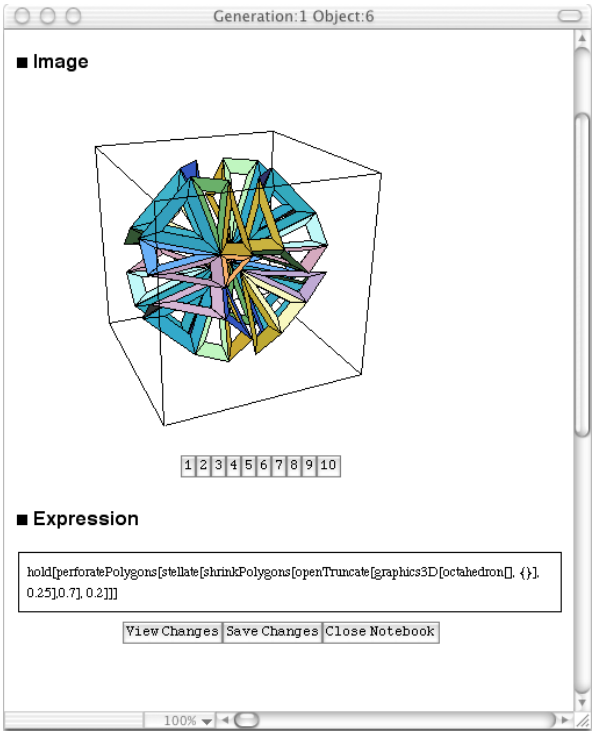


Fig. 4. The design program editor and interactive 3D display with ranking buttons.

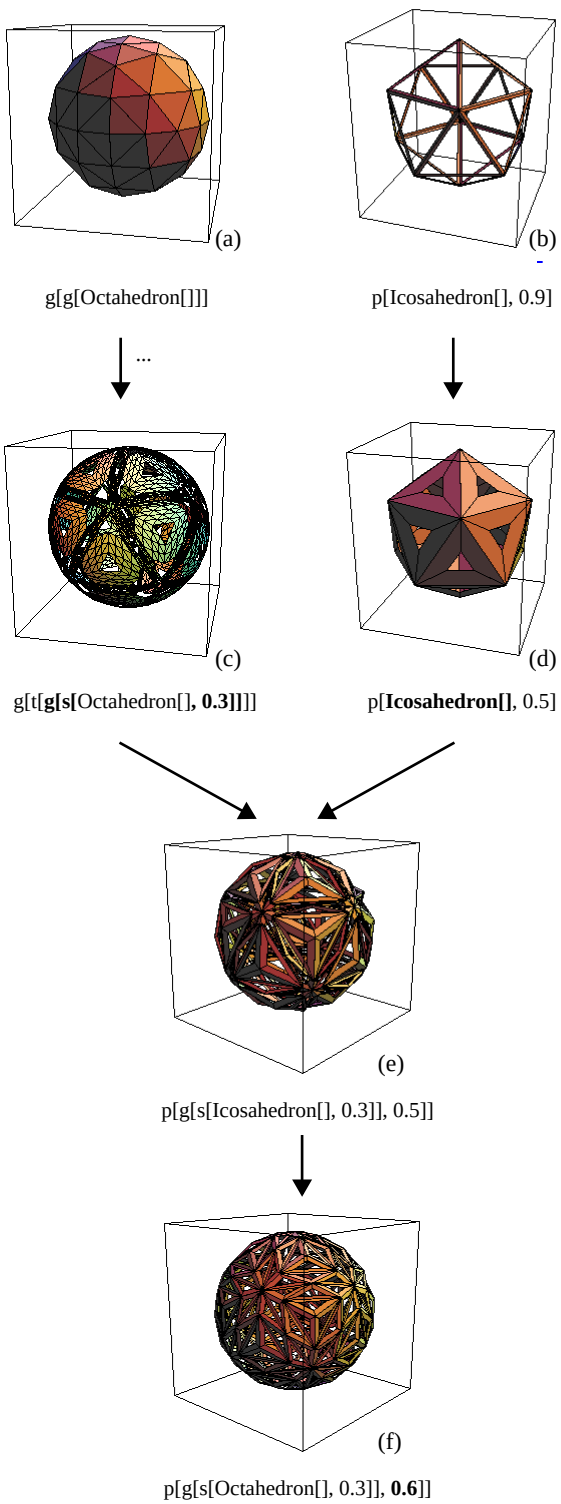


Fig. 5. Examples of mutations and recombination of two polyhedral shape programs. The following abbreviations are used: g = Geodesate, p = PerforatePolygons, s = Stellate, t = OpenTruncate.

Fine-tuning Mutations

Once a satisfactory design solution has been bred, the user might want to "optimize" or adjust a given shape. In the *Polyhedron Evolver* this can be achieved through fine-tuning mutations, which mostly modify the numeric parameters that occur in the design programs. For example, the mutations from shape (a) to (b) and (c) both involve slight changes of the numeric parameters (Fig. 3).

Interactive Design Modification

In addition to the built-in automatic mutations, any of the evolved designs can also be modified manually. For each polyhedral shape the generating program can be displayed, edited, and re-inserted into the evolution loop (Fig. 4). Thus, the *Polyhedron Evolver* utilizes the creative power of an evolutionary system, but does not exclude any direct control over and interaction with the design programs. In the same context, the *Evolver* also allows to load designs from previously performed evolution experiments.

Recombinations

The *Polyhedron Evolver* also uses a number of recombination operators which combine subexpressions from two design programs into a new program. These operators resemble the various crossover mechanisms resulting from sexual reproduction in natural organisms, where an offspring's genes are a merge of genes inherited from its mother and father. However, natural genomes are encoded by a flat, linear structure, whereas symbolic expressions have a hierarchical, tree-like structure. Consequently, recombination schemes on expressions are slightly more difficult to handle than for linear strings or number vectors.

Figure 5 illustrates one such recombination operator. Two design programs ((c), (d))—both resulting from a sequence of mutations from two initial shapes ((a), (b))—are merged into a new symbolic expression (e). The *g[s[_, 0.3]* building block of program (c) is composed into program (d), where the icosahedron base object is substituted for the '_' pattern.[1] Therefore, the recombination operators allow to create new designs through combinations of characteristics of already evolved programs. The recombination operators and further biologically inspired genetic operators are described in more detail in [3], [4], [5], and [6].

THE EVOLUTIONARY ALGORITHM

The interactive evolutionary design process can be described as follows:

1. The system generates an initial set *S* of random design programs, based on the problem-specific building blocks.
2. The designer interactively evaluates each design on a

1. Remember that in *Mathematica* the '_' pattern matches any expression.

scale from 0 to 10. Any design that is not explicitly evaluated receives a rank of zero.

(a) Design programs can be edited at any time.
(b) Design programs from previous experiments can be retrieved from files.
(c) Edited or loaded designs can either replace others or be inserted into the current design set S.

4. Based on the assigned ranks and on the mutational operators, the system selects designs from S that survive into the next iteration and—using these designs as "parent" expressions—generates a new set of mutated "offspring" design programs S_{new}.
5. The designs library is updated: $S = S \cup S_{new}$.
6. If more designs should be bred, return to step 2.

Again, more details about different selection schemes, genetic operators, etc. are described in earlier work as, e.g., [3], [5]. The evolutionary kernel functionality, including selection functions, pattern-based genetic operators, and various evolution schemes, are all provided through our evolutionary computation system *Evolvica* [6].

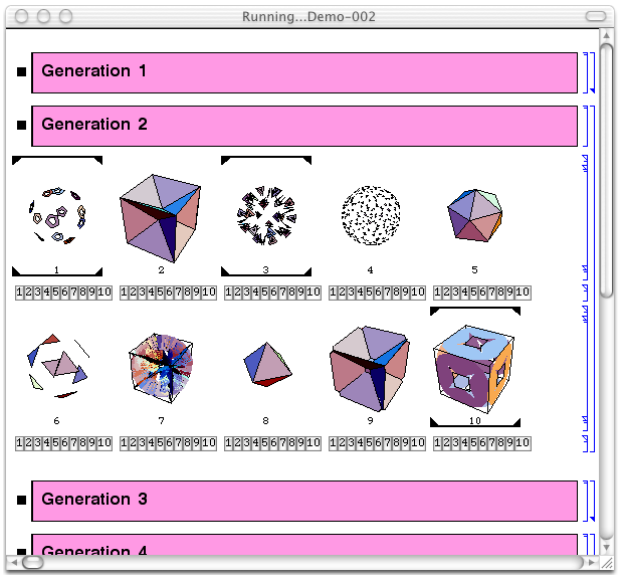


Fig. 6. The main evolution window.

Figure 6 shows an example of the main window that displays and gives access to all evolved designs during an experiment. In this notebook, a new section is automatically added for each evolved generation. Using *Mathematica*'s notebook control functionality, any of these generational sections can be expanded or collapsed, which gives easy access to the whole set of evolved designs, and can also be utilized for easy structuring and categorization. The snapshot in Figure 6 shows the expanded section for generation 2, with ten evolved polyhedral objects. A row of buttons below each graphics allows to rank the designs on a scale from 1 to 10. These ranks are then taken into account by the selection function, that determines which of the designs are going to survive into the next generation. Three of the objects are selected, depicted by the horizontal brackets above and below objects 1, 3, and 10. This selection mechanism, which works on all evolved objects across the entire notebook, is also used to access more detailed information about the design programs through the Profile and Design Editor (Fig. 4).

POLYHEDRON EVOLVER IN ACTION

Using the described evolutionary algorithms scheme, *Polyhedron Evolver* allows to generated a surprisingly rich variety of three-dimensional objects composed from basic polyhedral shapes and multiple applications of transformation functions. The set of these functions, which we have discussed here, is only a preliminary selection. However, the set of "mutation" functions can be extended easily and be made instantly accessible to the evolutionary system by adding them to the template library (Fig. 2).

Figure 7 illustrates the diversity of polyhedral designs that we have evolved so far. Of course, this is only a small selection and can not adequately represent the evolutionary creativity of this system. It demonstrates, however, that evolutionary mechanisms, combined with appropriate evaluation strategies and instant, interactive access to the designs through symbolic expressions provide a promising platform for design problems in a wide number of fields. We used polyhedra to illustrate the potential of this approach, as polyhedral objects are mathematically well defined, as are the transformations we have used in our examples. But obviously this approach can be extended to evolutionary designs in engineering, biology, virtual object and scene designs in computer games or computer-generated movie scenes, or even painting and sculpturing. The designed programs do not even have to be restricted to static images. For example, the same evolutionary approach has been used to evolve growth programs for virtual plants and flowers [4], [6].

CONCLUSION

We also use *Polyhedron Evolver* to breed designs described by 3D implicit plots, 3D plots, 3D contour plots, as well as 3D implicit surfaces. We are currently extending the system to include more basic shapes, allow more sophisticated genetic operators, extend the user interface, and parallelize the time-consuming graphical computations.

For more information on evolutionary computing and 3D shape design with *Evolvica* and genetic programming, see http://www.cpsc.ucalgary.ca/~jacob/IEC.

REFERENCES

[1] Bentley, P., ed. (1999). *Evolutionary Design by Computers*. San Francisco: Morgan Kaufmann Publishers.

[2] Jacob, C. (1995). *Genetic L-System Programming: Breeding and Evolving Artificial Flowers with Mathematica*. In *Mathematics with Vision: IMS'95, First International Mathematica Symposium*. Southampton, UK: Computational Mechanics Publications.

[3] Jacob, C. (1996a). *Evolution Programs Evolved*. In *Parallel Problem Solving from Nature—PPSN-IV*. Berlin: Springer-Verlag.

[4] Jacob, C. (1996b). *Evolving Evolution Programs: Genetic Programming and L-systems*. In *Genetic Programming 1996: First Annual Conference*. Cambridge, MA: MIT Press.

[5] Jacob, C. (1997). *Simulating Evolution with Mathematica*. In *IMS'97, Second International Mathematica Symposium*. Rovaniemi, Finland. Southampton, UK: Computational Mechanics Publications.

[6] Jacob, C. (2001). *Illustrating Evolutionary Computation with Mathematica*. San Francisco: Morgan Kaufmann Publishers.

[7] Koza, J., et al. (1999). *Genetic Programming III*. San Francisco: Morgan Kaufmann Publishers.

[8] Trott, M. (2001). *Polyhedron Explorer*. Demo notebook, *Mathematica* Help Browser. Champaign, IL: Wolfram Research, Inc.

[9] Wolfram, S. (1996). *The Mathematica Book*. Cambridge, MA: Cambridge University Press.

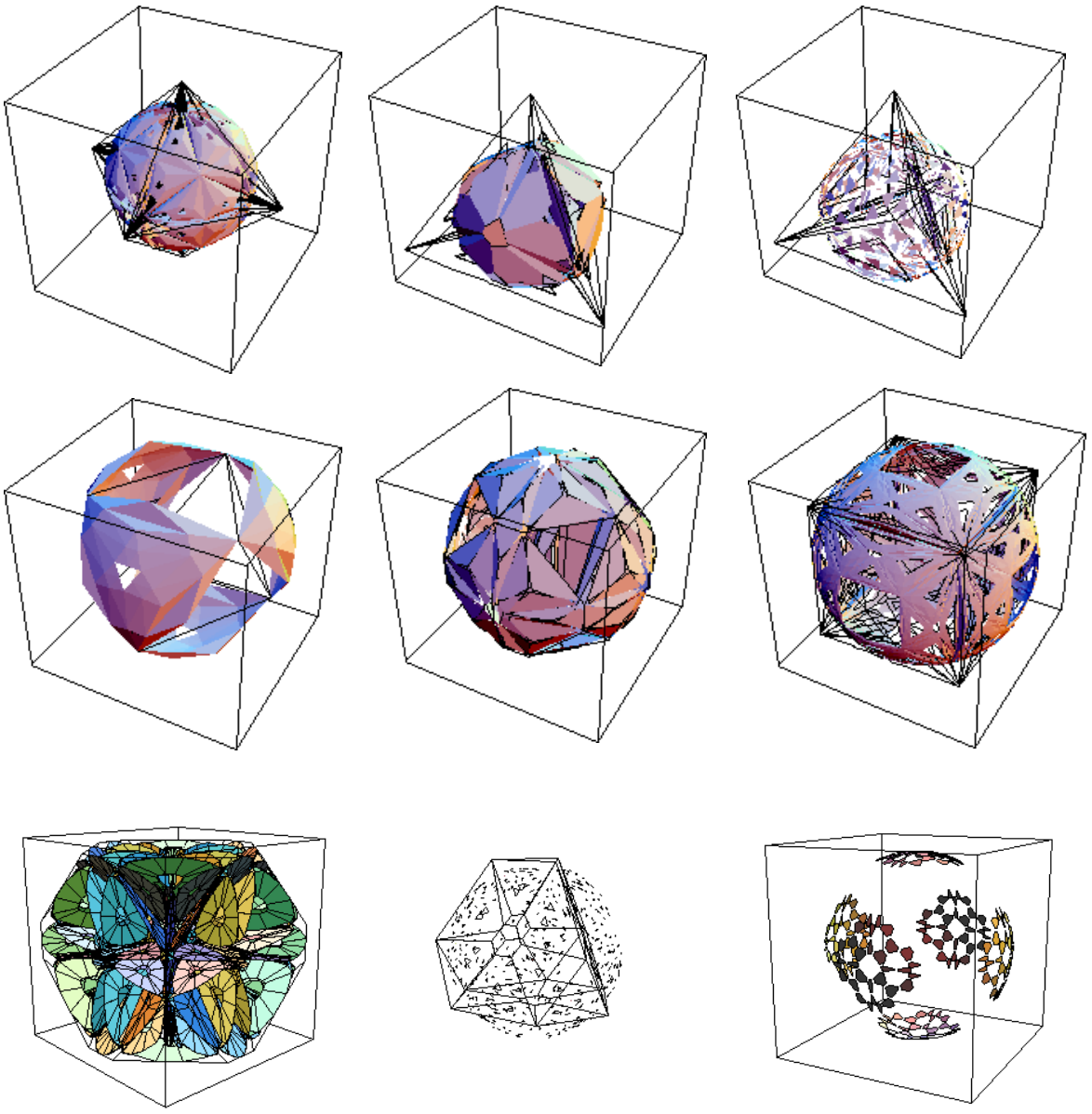


Fig. 7. A selection of evolved polyhedral objects.