Genetic algorithms for evolutionary product design

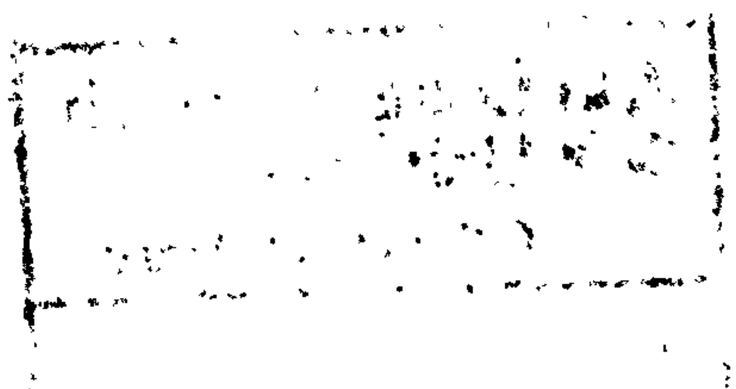
Genetic Algorithms for Evolutionary Product Design

By

Ian J Graham B.Eng. (Hons) DIS

A doctoral thesis submitted in partial fulfilment of the requirements
for the award of Doctor of Philosophy of Loughborough University

September 2002



Wolfson School of Mechanical and Manufacturing Engineering
Loughborough University

CONTAINS DISKETTE

UNABLE TO COPY

CONTACT UNIVERSITY

IF YOU WISH TO SEE

THIS MATERIAL

ABSTRACT

This thesis describes research into the development of a Computer Aided Design (CAD) tool that uses a Genetic Algorithm (GA) to *generate* and *evolve original* design concepts through human *interaction*.

CAD technologies are firmly established in the later stages of design, and include many applications of Evolutionary Algorithms (EAs). The use of EAs as generative and search tools for conceptual design is less evident in fields other than abstract art, architecture and styling. This research gains its originality in aiming to assist designers early in the design process, by *creating* and *evolving* aesthetically interesting forms (objects).

The integration of GA software with a solid modelling system has enabled the development of a prototype 'Evolutionary Form Design' (EFD) system. Objects are defined using a genetic data structure and constructed from various geometric primitives and combinations of Boolean operators. The primitives interact in ways that are not easily predicted, often creating novel shapes that are unlikely to have been discovered through conventional means. Edge blending further adds to objects' complexity and visual appeal. Populations of objects are subjected to a 'selective breeding' programme, directed through the user's allocation of scores, and may also be guided by simple geometric targets. These factors determine which objects are 'fittest' and most likely to parent a new, hopefully improved generation of objects. The challenge has been to turn the concept into a genuinely useful tool, ensuring that desirable features are reproduced in subsequent populations. The key to achieving this is the way objects are recombined during reproduction. Work has included developing a novel routine for grouping the individual primitives that form objects using a Teamforming algorithm.

Innovative, aesthetically interesting forms can be evolved intuitively and efficiently, providing inspiration and the initial models for original design concepts. Examples are given where the system is used by undergraduates to generate seating designs, and by the author, to create virtual sculptures and a range of consumer product concepts.

Keywords: Genetic Algorithms Interactive Evolutionary Design Conceptual Form Aesthetic

To Bonnie, for her love & support

With thanks to:

Mum & Dad, for everything,

My brother, Rob, for something,

Good friends, for welcome distractions,

Keith Case & Bob Wood, for their patience & wisdom,

Wolfson School support staff, for always being there,

British taxpayers & EPSRC, for research funding,

The fern collection, for their calming influence,

Charles Darwin, for an amazing discovery,

&

Evolution, for inspiration & existence.

CONTENTS

LIST OF FIGURES

CHAPTER ONE – INTRODUCTION

1.1 The Need for Evolutionary Product Design

The inspiration for this research stems from an interest in evolutionary computer programming; Genetic Algorithms specifically, and in Computer Aided Design; especially as a concept modelling and development tool for consumer products. At present, the use of CAD is concentrated around the later stages of design following on from conceptual design, but there is increasing interest in ways in which CAD can support earlier design processes.

The initial stages of product design are a rather intangible set of processes, especially concerning the way original form ideas are conceived. The industrial designer uses a sketchpad, a practised hand and a selection of pencils and markers, to externalise the shapes of product concepts. Although a designer will usually have a vague image of a shape in mind, often finding inspiration in other objects (man-made or natural), it is very much down to the individual to create pleasing forms for products by drawing on knowledge, experience and 'artistic ability'.

Ideas are formed through a combination of inspiration and mapping out thought processes on paper. Ideas then evolve through the use of development sketches, cardboard models, clay forms and other physical media. An analogy with Darwinian evolution is frequently drawn; referring to the combination of existing ideas and the process of refinement a design goes through as it is developed.

Existing CAD modelling systems are of limited use during this process, not allowing a designer to experiment with ideas freely. If CAD is to broaden its role, then there is a need to assist designers to a greater extent during the conceptual stages of form design. There is a significant gap in the market for a tool that actually generates form, and better supports form development. To exploit this gap, CAD tools should attempt to emulate the methods described above, and can provide a source of inspiration through a system that evolves forms.

1.2 Evolution and Genetic Algorithms

Evolution can be described as the continuous production of new variations with no particular intent, except some variations will be more successful than others. Evolutionary biology has shown us that the intricate 'designs' found in nature can arise through gradual, mindless improvement. Computers are incapable of conscious thought, but, by translating our knowledge of evolution into a computer program, the core innovative properties of evolution can be achieved and exploited.

Genetic Algorithms are the most well known and well used evolutionary computational technique and have been applied to optimisation, problem solving and simulation, across a range of fields including biology, engineering, computer science, sociology and finance. They include processes taken from biology, such as reproduction, parent selection and genetic data structuring, and work by maintaining populations of members (i.e. solutions to the problem), the fittest (best) of which are selected to create a new generation.

Usually, after initialisation, a GA is left to run for a number of generations until, hopefully, a solution emerges. In some applications a human operator is required to contribute to the assessment of members during the evolution process. This interactive method is most often used for producing computer art. With the user as the sole source of evaluation the system is analogous to selective breeding (of farm animals, garden plants etc.).

1.3 Product Form

It is understood that studying form in isolation is not in keeping with effective design practices, as is evident from the quotation below from 'Products as Representations' [Vihma]:

> *"In many definitions of design, the use of the word 'form' is avoided so that the outer form, appearance or surface of the product will not be given too much attention in people's conceptions of design. The avoidance and understatement of 'form' can be seen as anxiety towards a too superficial conception of design as mere styling"*

It is also acknowledged that the design of products should be concurrent, in that the areas of industrial design (e.g. styling and user interaction) should not be considered separately, after other aspects of product design have been completed [Ulrich].

1.4 Research Scope

The overall theme of the research discussed here is the integration of evolutionary techniques with a CAD modelling system, combined with human interaction to guide the evolution process. It has not been the intention to produce an 'independent' system that runs through a number of generations, with only the pre-set objective functions for reference, then presenting its solution(s) to the problem. Nor has it been the intention for the user to be the sole driving force during 'evolution'. A balance has been achieved through the development and integration of the following concepts:

- Offering the user the chance to rate the objects presented, or explicitly select which objects are allowed or discarded as parents for the next generation.
- Allowing objects' geometric properties to contribute to their fitness.
- Establishing which GA operators are most appropriate for the application.
- Designing a novel technique to maximise desirable feature propagation and distribution, thus increasing the efficiency of the evolution process.

The desire to create aesthetically interesting forms has strongly influenced the focus of this research, and for the software to be considered as a tool, rather than just an investigation into the above concepts, the created objects should be capable of representing actual products. The use of surface modelling would seem more conducive to the complex curved products with which this research is concerned. Nonetheless, it is felt that a Constructive Solid Geometry approach, with edge blending, is more suitable for providing greater object variety and a natural interpretability (as products), whilst maintaining high data efficiency.

This research is primarily concerned with the outside appearance of objects, concentrating on aesthetic qualities over functional characteristics. There is scope for the parameterisation of aesthetics qualities, such as cohesiveness, compactness, proportion and unity, to be investigated. It is also suggested that mechanical considerations could be included within the system. For example, by providing functional elements to be integrated within each design.

1.5 Aims and Objectives

Aim

To develop an interactive evolutionary CAD tool focusing on the conceptual phase of product design.

Key issues being:

- Forms should be aesthetically interesting and able to represent products.
- User interactions should have a predictable outcome.
- The evolutionary process should be efficient and sensitive to a designer's input.
- The system has to actually assist designers.

Objectives

1. To review the current state of research and understanding in the fields of Evolutionary Computation (especially Genetic Algorithms applied to engineering design and form generation), Computer Aided Design, Design Methodologies and Aesthetic Theory. To substantiate the demand for a CAD tool that enables the generation and evolution of product form and to provide an overview of existing evolutionary design systems.
2. To develop a unique interactive CAD system that utilises evolutionary techniques and has the ability to represent aesthetically interesting products. To demonstrate other software capabilities including functional considerations and aesthetic assessment functions.
3. To develop the novel GA technique of forming 'teams' of interacting individuals to produce the output, subject to a set of evolving interaction rules.
4. To validate the system through informal case studies and by allowing potential users to create a range of virtual consumer products.

CHAPTER TWO

LITERATURE SURVEY

2.1 Applications of Evolutionary Algorithms

Genetic and Evolutionary Computation

Evolutionary Algorithms (EAs) are search methods inspired by and based upon the Darwinian theory of evolution observed in nature – natural selection through survival of the fittest. The fundamental feature of modern EAs is that they work with a collection, or population, of solutions concurrently. All of the solutions in a population are evaluated, with the best solutions then contributing in some way to forming new solutions. This process is iterated, causing evolution to occur. An intriguing feature about these methods is that the evolution displayed is not explicitly programmed, or simulated, but an emergent property of the algorithm, and very real.

An offshoot of the early work on Artificial Intelligence, the concept of Evolutionary Computation was realised in the late 1950's [Levy]. John Holland played a central role in this work, and went on to create Genetic Algorithms [Holland1]. There are three other main evolutionary algorithms in use today: Evolutionary Programming, Evolution Strategies and, more recently, Genetic Programming. Subjects with close links to, and often used in conjunction with, Evolutionary Algorithms include Cellular Automata, Neural Networks, Simulated Annealing, Fuzzy Systems, Immune Networks and Machine Learning. An astonishing number of related and inter-related subjects have developed in recent years, as is demonstrated in the following list of topics represented at GECCO-2002[1].

[1] The Genetic and Evolutionary Computation Conference, a recombination of the Seventh Annual Genetic Programming Conference (GP-2002) and the Eleventh International Conference on Genetic Algorithms (ICGA-2002), presented by the International Society for Genetic and Evolutionary Computation (ISGEC)

"...genetic algorithms (GA); genetic programming (GP); evolution strategies (ES); evolutionary programming (EP); evolvable hardware (EH); evolutionary robotics (ER); real-world applications (RWA); classifier systems (CS); DNA, molecular and quantum computing (DNA); artificial life, adaptive behaviour and agents (AAA); ant colony optimisation (ACO); optimal design of engineered structures (ODES); methodology, pedagogy, and philosophy (MPP); evolutionary scheduling and routing (ESR)..."

Genetic Algorithms

Genetic Algorithms were initially developed by John Holland in the early 1970's [Holland1, Holland2], and popularised by David Goldberg's textbook 'Genetic Algorithms in Search, Optimisation, and Machine Learning', first published in 1989 [Goldberg]. GAs are good for solving problems where the range of combinations of parameters is so large that it is unfeasible to search exhaustively [Forrest]. For example, the 'problem' of finding 'aesthetically satisfying' 3 dimensional shapes for consumer products is infinite, even if it can, to some degree, be parameterised.

The remainder of this chapter, after briefly acknowledging some interesting biological applications, concentrates on GA applications in engineering, product design, aesthetics and interactive art. Genetic Algorithms have also been successfully applied elsewhere, to the modelling of social, economic and political systems [Forrest, Goldberg], and financial systems, such as stock market trends and patterns [Levy]. They are also often employed as theoretical and experimental tools for investigating the phenomena generated by complex adaptive systems (adaptive agents), such as ecologies, immune systems, developing embryos and the brain. Chapter 3 introduces Genetic Algorithms in more detail.

Artificial Life

Genetic Algorithms, being based on evolutionary principles, are firmly established within the field of Artificial Life (AL). An overview of the fascinating history of AL is described in the popular science book, 'Artificial Life – The Quest for a New Creation', by Steven Levy [Levy]. The subject is covered in greater detail by one of the pioneers in the field, Chris Langton, in his opening essay to the Proceedings of the First Workshop on Artificial Life, in 1986 [Langton1]. Two further publications, Artificial Life II and III [Langton2, Langton3] bring together descriptions of the work at the Santa Fe Institute, over the subsequent 6 year period.

Reading around this remarkable subject greatly influenced the author's decision to take a final year undergraduate degree project in Genetic Algorithms [Graham1], which led to the research described in this thesis. Three examples of particularly interesting applications of GAs in Artificial Life, taken from the Artificial Life series mentioned above, include: An artificial ant population which developed foraging strategies [Collins]; the detailed simulation of food webs [Lindgren]; and PolyWorld, the simulation of an entire ecosystem, containing autonomous agents that displayed complex learning capabilities and behavioural patterns through evolution of a neural network within each agent [Yeager]. Another, slightly earlier study, which also created an artificial ecology, used a mixed species population which displayed emergent colonisation patterns dependant on food availability and interaction between other species [Assad].

2.1.1 Engineering Applications

Over the last two decades, a vast number of computer-based tools for engineers have been developed. In more recent years, there has been a shift in emphasis towards Artificial Intelligence based methods [Garrett], of which evolutionary techniques are a major part. One of the most popular evolutionary tools within the engineering field is the Genetic Algorithm, which is most commonly applied in the fields of; scheduling, control, route and network planning, layout design, component optimisation and robotics. Many applications can also be found in manufacturing systems, material design, integrated circuit design, modelling and simulation, signal processing and image processing [Dasgupta, Gen, Grierson, GALESIA95, GALESIA97].

The variety of applications of evolutionary computation techniques in engineering can be seen through the diversity of papers presented at the Second International Conference on *Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*, held in 1997. Applications include; airfoil design, gas turbine engine controller design, switched reluctance motor control, ship auto-pilot control systems, greenhouse climate control, traffic signal timing, hot rolling of steel strip, vibration data compression, scheduling in cellular manufacturing, local access network design, resource redistribution in the developing world, aircraft route planning, transistor-level digital logic synthesis, modelling a fermentation process, image restoration, recognition of object shapes with movable parts, missile-target simulation, intelligent user interface design, fire detection technology and development of walking strategies for an 8-legged robot [GALESIA97].

Contributors were not restricted to universities and research organisations. Several large companies presented research such as Rolls Royce, on active magnetic bearing controllers [Schroder] and BT, on dynamic data searching [Amin]. The presentations ranged from the introduction of preliminary ideas and research, to post-application reports on industrially applied case studies.

A good example of real world application is the work at Politecnico di Torino with Bottero SpA, Cuneo, Italy, where a GA has been used for optimising area loss in flat glass cutting [Corno]. The GA is used to find an efficient cutting pattern, where the stock glass sheets are typically in the order of $10m^2$, and benchmark processes used for evaluation involved cutting specifications of between 12 and 161 pieces, with between 2 and 35 different sizes of piece. In this case, the GA has to find an acceptable solution in real time, in the period between starting the previous job and the loading of the next glass sheet. Other constraints arise from the cutting technology used, including the need for the cutting pattern to be composed of a series of end to end cuts, in the horizontal or vertical direction. There was also the need for the program to run on the PC-based machine controller. Commercial systems run on separate PCs 'in the office', at the time of receiving a customer order, and as such are a), under no time constraints and b), often out of the reach of smaller companies who may not have these separate facilities. The initial results were very good – comparable, and in some cases out performing the three commercial systems available. This led to the immediate installation of the software onto every flat glass-cutting machine sold by the company.

A number of papers address the different ways problems can be genetically represented. For example, work described in [Cordón], on designing fuzzy rule-based systems for surface representation, investigates, amongst other things, the alternative ways chromosomes are used to encode the solutions. Chromosomes are either treated as individual fuzzy rules and the entire population represents the knowledge base, or, alternatively, each chromosome represents an entire knowledge base. The findings are similar to those on the validity of the Teamforming technique in the research discussed here, in that each representation has its own virtues. In the paper, Cordón recommends a third model, an adaptation of the first 'each chromosome is a fuzzy rule' approach, where only the best individuals are chosen to form the knowledge base – potentially, this idea could be investigated during further development of the EFD Teamforming method.

Manufacturing Engineering applications of GAs fall into the two general categories of planning functions, and shape design [Mill]. The more prevalent manufacturing applications of GAs are systems-based, concerning scheduling, such as job-shop scheduling [Corne], flow-shop scheduling [Gonzalez], assembly line balancing [Byrne], and production planning [Ono]. Other work in the field includes object recognition, route and network planning, and robotics [Zalzala]. Applications concerning component optimisation are most relevant to this research.

Component Optimisation

GA optimisation is often applied to aerodynamic components, in combination with Computational Fluid Dynamics (CFD) software [Obayashi1], often within turbines, where thermodynamics [Pearce, Wood1], or other mechanical functions, such as stress and mass are analysed using Finite Element Analysis (FEA) software [Smith].

The airfoil design application listed above involves aerodynamic optimisation of compressor blade shape (A 2D cross section is used), where multi-objective design seeks high pressure rise, high flow turning angle, and low pressure loss [Obayashi2]. The airfoil is represented two dimensionally, by 2 B-spline polygons, involving 21 variables per polygon. Leading and trailing edges, and one further point, are frozen, in order to maintain correct representation. Evaluation is carried out using the Numerical Wind Tunnel[1] at the National Aerospace Laboratory in Japan. The solutions obtained were better in all three objectives, than current designs of airfoil.

Other examples of evolutionary component optimisation in the literature include the optimisation of flywheels, for the maximisation of specific energy density [Eby], optimisation of vibration and noise response in satellite booms and load cells [Robinson], and jet engine annulus design optimisation using a voxel-based representation [Baron], which is discussed later.

[1] Parallel vector machine (a dedicated FEA installation on a purpose built super-computer)

2.1.2 Applications in form generation, aesthetics and art

The utilisation of evolutionary computing technologies for optimisation is well established, and has been discussed. There appears to be much less recognition of the design exploration and search capabilities of EAs [Parmee]. Research into form generation and aesthetics is generally confined to the visual arts, and will be summarised shortly. There are, however, some notable exceptions. The work carried out over the last 10 years by David Wallace, on developing a computer model of aesthetic product design, is covered in the following section on evolutionary design systems. Aesthetic issues are strongly represented in civil engineering, particularly in bridge design and will also be covered briefly. Lastly, although not directly related to aesthetic design, the work on free-form shape features discussed below is distinctive, in that it initially considers the surface form in isolation. Similar work can also be found in the literature [Guo].

Artificial Embryogeny

Recently there has been increased interest in embryogeny, originally summarised by [Kumar]. Embryogenies, or growth processes, tackle the issues of increasingly ambitious and complex problems being set for evolutionary systems. They copy the way nature grows designs, rather than the more conventional direct mapping of genetic data to the solution.

Naturally, one of the first applications was in free-form shape generation, investigated at the 'Research into Artefacts, Centre for Engineering' (RACE) at the University of Tokyo. The aim of supporting the designer in the early stages of the design process is stated in [Taura1] - the intention being to develop a technique that overcomes some of the problems associated with the two principle representations: Constructive Solid Geometry (CSG) and Boundary Representation (B-rep). These problems are identified as the loss of design data during Boolean operations, and the difficulty in preserving features after combination when using the mathematical data of the surface geometry as the representation. The shape representation is described in the following chapter.

Evolutionary programming techniques allow populations of free-form shapes to be generated and explored. Genetic recombination creates new shapes through the combination of pairs of existing shapes. As with all work applying genetic operators to objects (including the GA application in the EFD research), much attention is given to the preservation of features when combining shapes. This careful attention to detail pays off, new shapes clearly exhibiting features of the two contributing shapes, whilst also displaying a useful degree of variation and exaggeration. Figure 2.1.1 shows the progressive evolution of shape to a target (left), and the results of the combination operator (right). As acknowledged in the paper, there are limitations in the capacity to represent products with this technique. However, as is evident from the continuing work by the RACE group (the adaptive-growth 3D representation has been used, more recently, for configuration design [Taura2]) and through its inspiration to other research groups, the early work on embryogeny summarised here is a valuable contribution to the field.

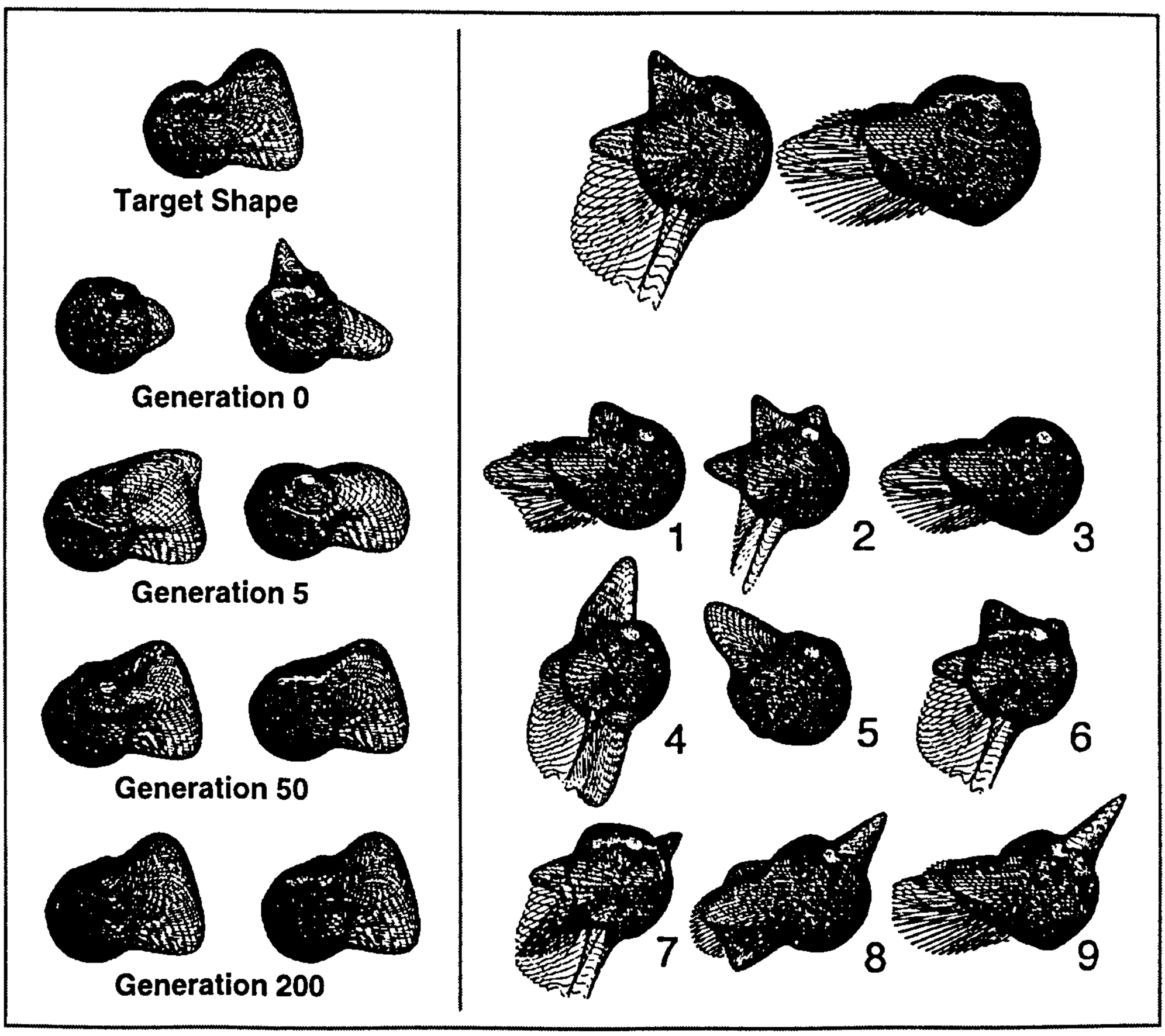


Figure 2.1.1 - Free-form shape representation - Taura

Bridge Design

Research surrounding the aesthetic design of bridges is well documented in the literature. In one case, a Genetic Algorithm is combined with fuzzy reasoning to develop a decision support system for aesthetic design of arch bridges [Furuta]. Arch type, chord type, rise rate, number of hangers, cross sections and colours of arch ribs, girders and hangers are all considered. Except for colour, variation is only between two or three alternatives in each case; i.e. 'number of hangers' can either be 'Many' (20), or 'Few' (15). Solution bridge designs are presented at the end of optimisation, after between about 150 and 250 iterations. Three examples of aesthetic designs are described (but not illustrated). The fuzzy reasoning of aesthetic functions is established by asking designers to indicate preferences for design images they are shown. These are structured around the degree of satisfaction a designer has with a list of 'design concepts' such as 'stability', 'elegance', 'reliability', 'uniqueness' and 'friendly'. These are then linked with pairs of adjectives to establish fuzzy if-then rules, i.e. 'stable / unstable', 'rigid / flexible', 'elegant / unrefined', 'strong / weak', 'unique / normal', 'lively / lonely'. No indication is given as to any mechanical considerations.

This bridge design system shares some common characteristics with the ideas presented in this research, particularly the intended further development of the work to include aesthetic assessment. The lists of 'design concepts' and adjective pairs look like the kind of criteria that would be involved with aesthetic performance of evolved objects.

Art

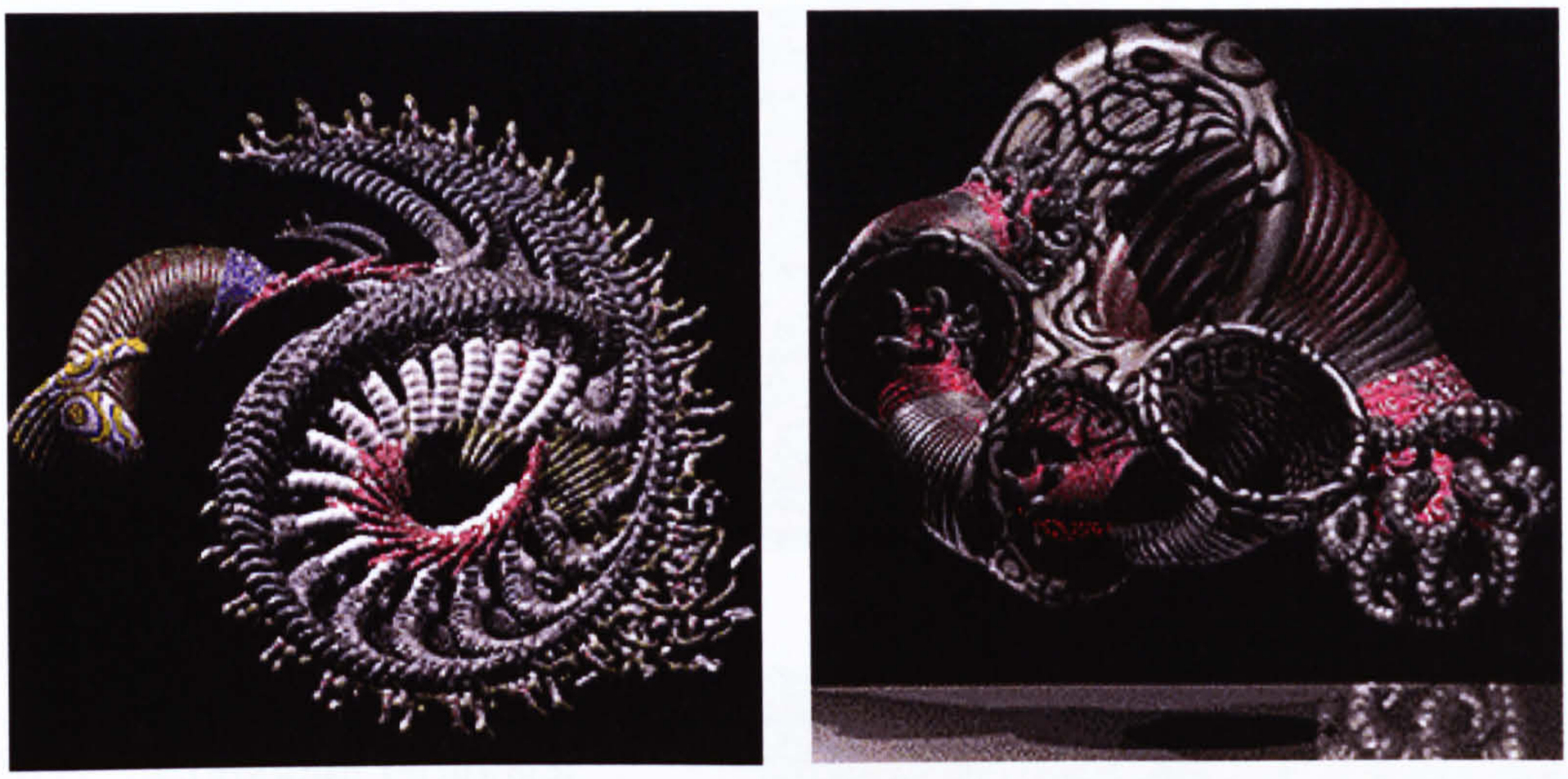
The applications described previously demonstrate the two distinct methods used in evolutionary design: Interactive, and non-interactive[1]. In the domain of evolutionary art, the interactive method is by far the most popular. In fact, this is essentially the whole point – the interactive process is enjoyable, partly explaining the occurrence of so many commercial and web-based systems [Rowbottom, Das].

The most well known products in the field are the combined works of William Latham of *Computer Artworks Ltd.* and Prof. Stephen Todd of *IBM Research Labs.* *'Organic Art'* and *'Mutator'* are both capable of spectacular results and have been well documented in both academic and popular media [Todd]. The types of 3D objects produced by the systems are abstract, but very distinctive, and are characterised by horns, shells, pumpkins, and other mathematical shapes, 'grown' using an artificial embryogeny technique. Because of the way in which the objects are encoded in chromosomes, they can be manipulated in many different ways, combined with other objects and mutated. Figure 2.1.2 shows some examples of *Mutator* and *Organic Art* objects, as well as similar work by Andrew Rowbottom [Rowbottom].

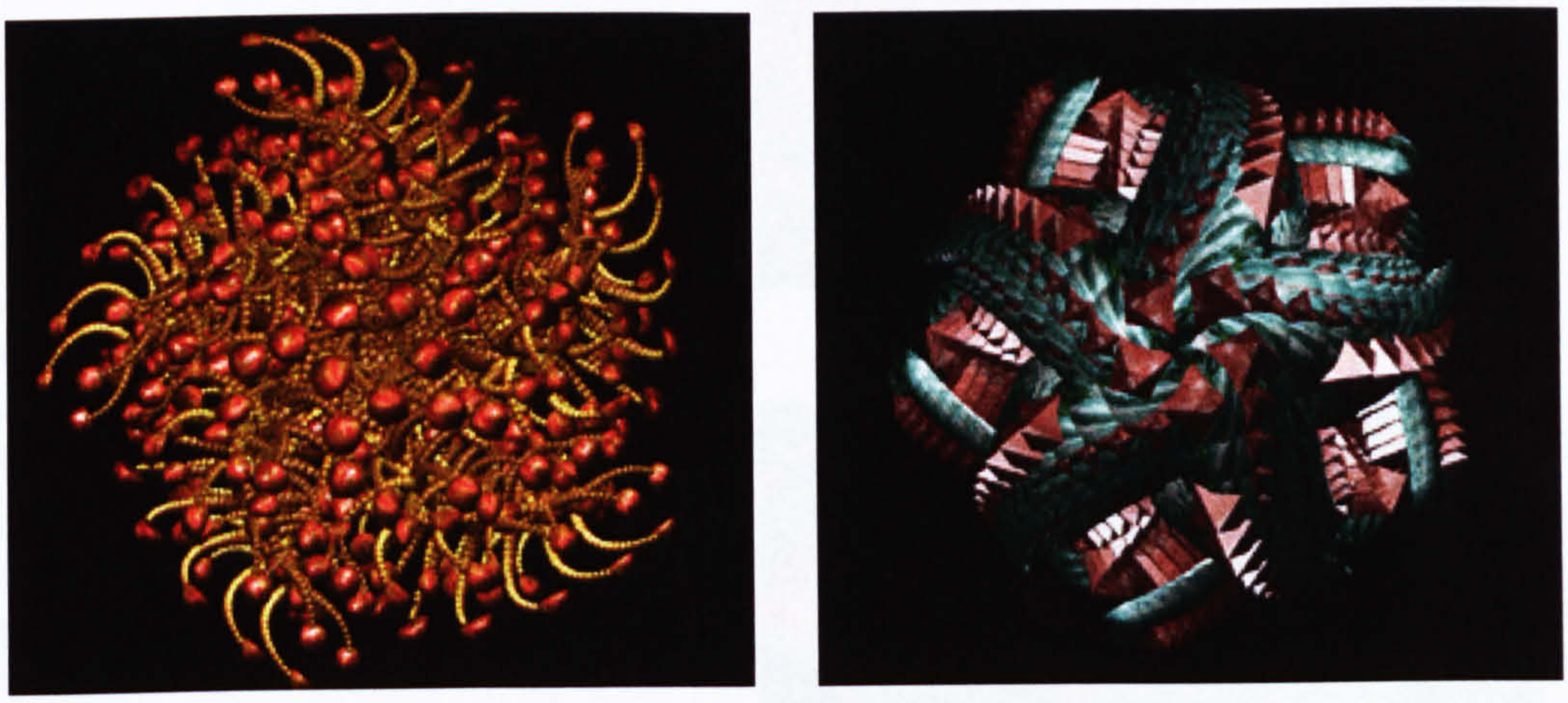
There is a clear relationship between the Mutator work and this research, in that genetic techniques are used to produce and display 3-dimensional computer models. Another consideration is that, during an active *Mutator* session, the user controls the development of objects. This is the primary approach that the EFD research uses, with the user making subjective decisions about the appearance of objects in allocating fitness values. 2D evolutionary art (Figure 2.1.3) follows a similar approach, the user being presented with an array of objects for appraisal, that are then subjected to mutation or breeding. Reproduced in Figure 2.1.4 are examples of two user interfaces [Witbrock, Rowley].

---

[1] Interactive systems depend on the user's input once the evolutionary process has started. In non-interactive systems, the user can still be involved, by setting up the system beforehand.



Evolutionary Art, William Latham



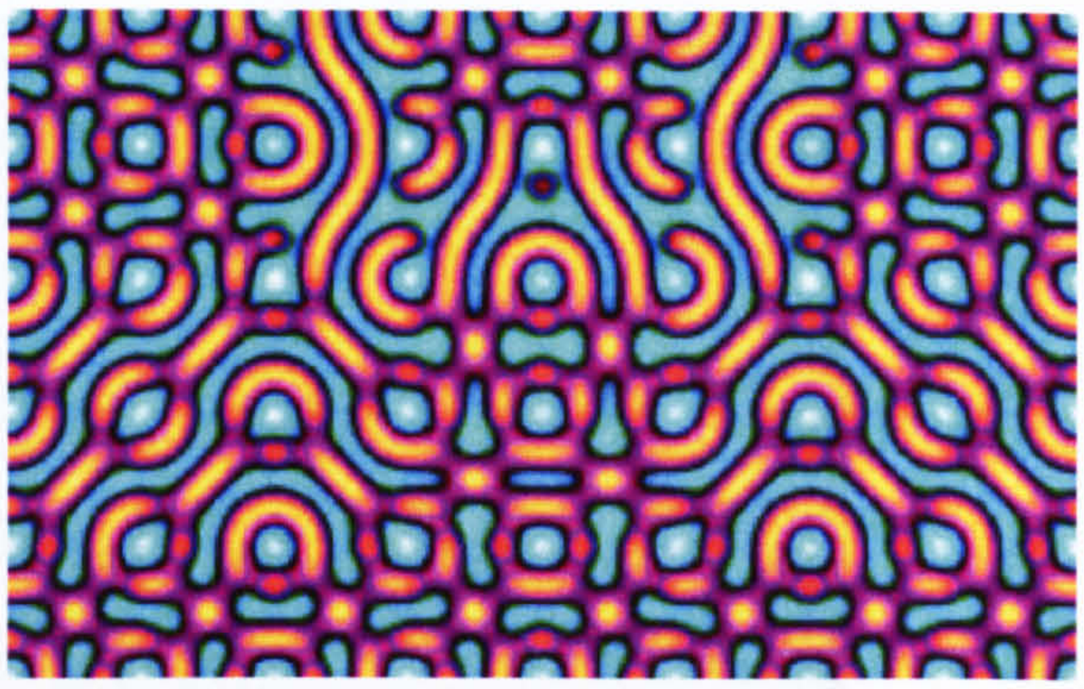
Organic Art, William Latham

Forms, Andrew Rowbottom

Figure 2.1.2 - 3D Evolutionary Art

Cellular Automata Art,

Paul Brown



Artificial Evolution for Computer Graphics,

Karl Sims

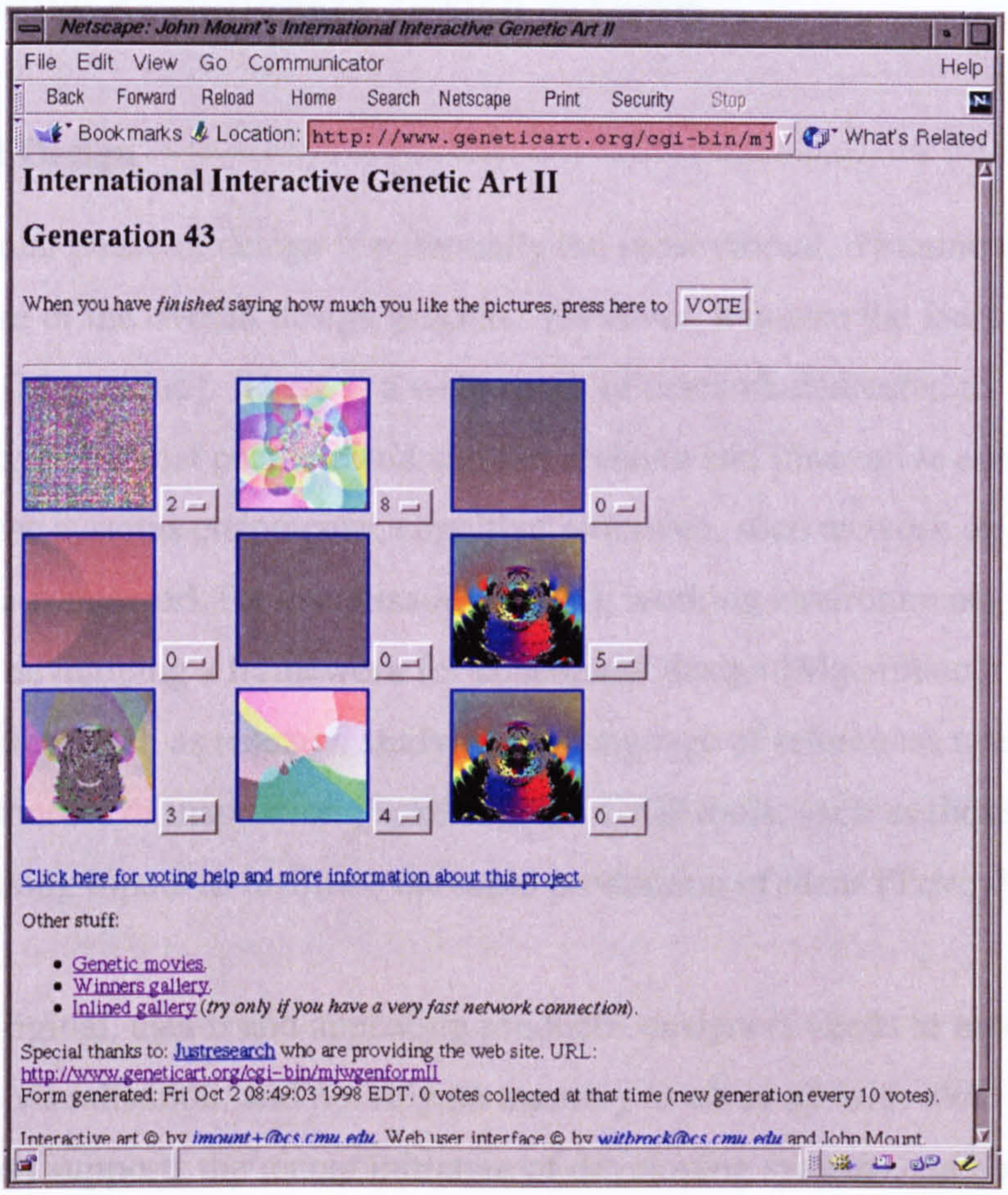
Neuro Evolutionary Art,

Penousal Machado

Visual-Musical Instrument (snapshot),

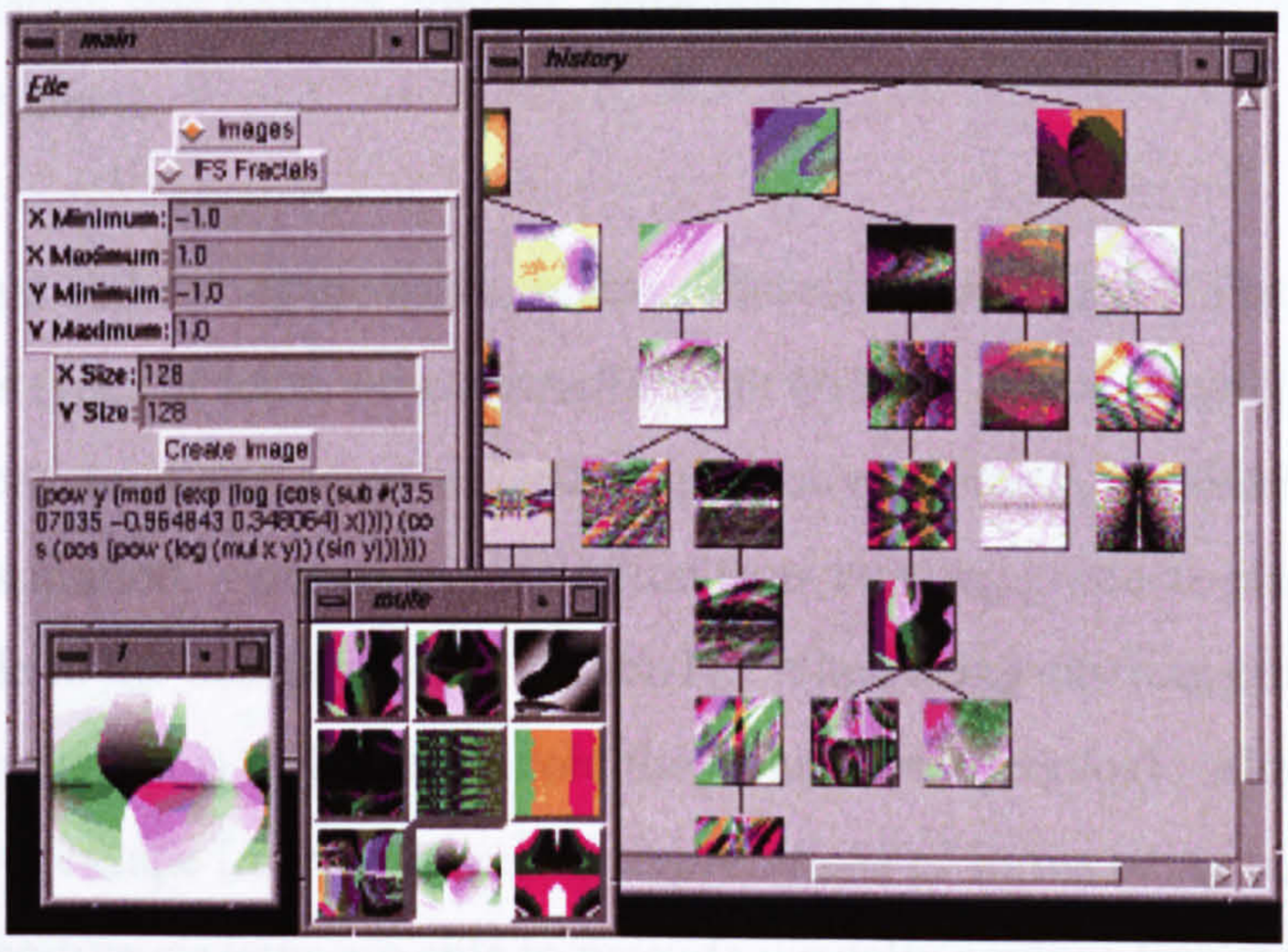
Scott Draves

Figure 2.1.3 - 2D Evolutionary Art



Quaternion-based Interactive Genetic Art System,

Michael Witbrock, John Mount



Toolkit for Visual Genetic Programming,

Tim Rowley

Figure 2.1.4 - User Interfaces for Interactive Evolutionary Art

2.1.3 Creative Evolutionary Design Systems

Conceptual Design

The conceptual phase of design is potentially the most vibrant, dynamic and creative stage of the overall design process. However, it is also the least understood [Macmillan]. There is a wide range of research dedicated to developing systems that promote and support creative and innovative conceptual design. These systems encompass; cognitive processes, such as work on developing a framework for visualisation [Dahl]; working environments, as with research on developing a framework for conceptual design [Macmillan]; communication, such as research studying the language of references used when describing sources of inspiration [Eckert1]; and CAD tools, such as those that utilise sketching inputs to facilitate the rapid production of ideas [Tovey].

To design original, useful and appealing products, designers needs to incorporate imaginative visualisation, and not rely on memory to come up with ideas [Dahl]. This assertion supports the recent initiative of developing systems, such as in the EFD research, that provide the designer with a multitude of 'imaginative' starting points for a design. It also affirms that these initial concepts need to be combined with visualisation (the product context, especially the user) if they are to become successful products.

The perceptual abilities of the brain, enable relatively basic shapes to be interpreted as types of form we are familiar with [Atick]. Consequently, the use of simple representations, often utilised by generative systems, is valid in terms of providing inspiration. Sources of inspiration from existing products play an important role in the design process [Eckert1]. When using existing products as inspiration though, it is difficult to avoid the influences of context. Although it is not possible to escape from designers' *interpretations* of forms being tainted by context, generative systems are able to provide virtually context-free inspiration for product forms.

The recent emergence of creative evolutionary design has been well documented in the academic and popular media [Graham-Rowe]. An informal introduction to generative and evolutionary design systems is available in the position paper 'Ten Steps to Make a Perfect Creative Evolutionary Design System' [Bentley1], which also includes discussion on applicability, and brief descriptions of two of the systems discussed here. There are numerous examples of creative evolutionary design systems, some of which are summarised later. Discussed first are three systems that offer much more than styling exploration, and genuinely belong in the field of design.

A computer model of aesthetic product design

Research on computer aided integration of engineering and industrial design, has been reported from 1991 onwards [Wallace1, Wallace 2]. The goal of the early work was to develop a design tool to help designers produce preliminary designs that are correct for their intended market, manufacture, and use. The expert system developed generates a design concept based upon manufacturing, ergonomic, aesthetic, and style considerations. User interaction is confined to the beginning of each of four stages of the process.

The first task is to select elements from a library of standard components and sub-assemblies (loudspeakers, keypads, microphones, jacks, displays and mechanisms, e.g. cassette drives). For spatial positioning of external components, four aesthetic characteristics are considered: *stability*, *rhythm*, *balance* and *organisation*. Each external component is positioned on a structure called the *product matrix*, and has *class*, *features* and *characteristics* attributes assigned. Six classes of component are used to position the components correctly: acoustic components, keypads, switches, jacks, visual displays, and mechanisms. The features of a component enable a visual representation of the item (e.g. a *speaker* is represented by a *grill*) to be produced. Special characteristics identify the needs of a component; e.g. a power supply jack is labelled as *service* whereas a headphone jack is labelled as *interface*. This insures that components are positioned appropriately within the matrix. Finally, a cubic bounding hull is generated around the entire assembly, ready for the surface design.

A prototype is chosen from a surface style library to provide a housing for the components, compatible with size, shape, and the intended use of the product. A mapping function preserves the style and manufacturability information encoded in a prototype. The surface styles are primarily defined by the edge treatments, i.e. rounded, chamfered etc. The surface detailing stage builds upon the product housing created in the previous stage, and finally surface applications are applied. These are limited to colour and graphics, but other surface treatments, such screening, logos, hot stampings, and textures are suggested.

The impressive aspect of this work is its holistic approach. The system considers manufacturing capability of designs (by injection moulding), and considers rules relating to aesthetics and ergonomics during layout design, producing products that are suitable for their purpose. Obviously the cubic representation restricts creativity and range application. More importantly perhaps, the system ignores the refining process often carried out by designers on their initial concepts, since the system creates just one finished product according to the criteria specified. A designer may wish for a range of product concepts to chose from, in which case having to repeat the process, which appears to be quite lengthy. The drawbacks of this non-interactive process are mentioned in the report however, with the conclusion that if the system were to be developed for practical use, it would be essential for the designer to interact more, and modify results.

Recent continuation of this work [Smyth], concentrates on issues of synthesising 'Brand DNA' (product styles adopted by corporations to distinguish brands and create a common brand identity) through aesthetic product form. The issues raised above have been addressed by incorporating evolutionary techniques that are used to generate variations on an archetype supplied by the user. A skeleton of the desired form, based on existing product geometry, is used to generate a population of skins (forms), from which the user selects appealing examples for further evolution. The method lacks genuine creativity, in that established ideas/designs are used to seed the evolutionary process. However, this process does have its place in industrial design methodology, and has actually been applied commercially (the design consultancy company *affinnova* use a similar approach, discussed at the end of this section).

GADES

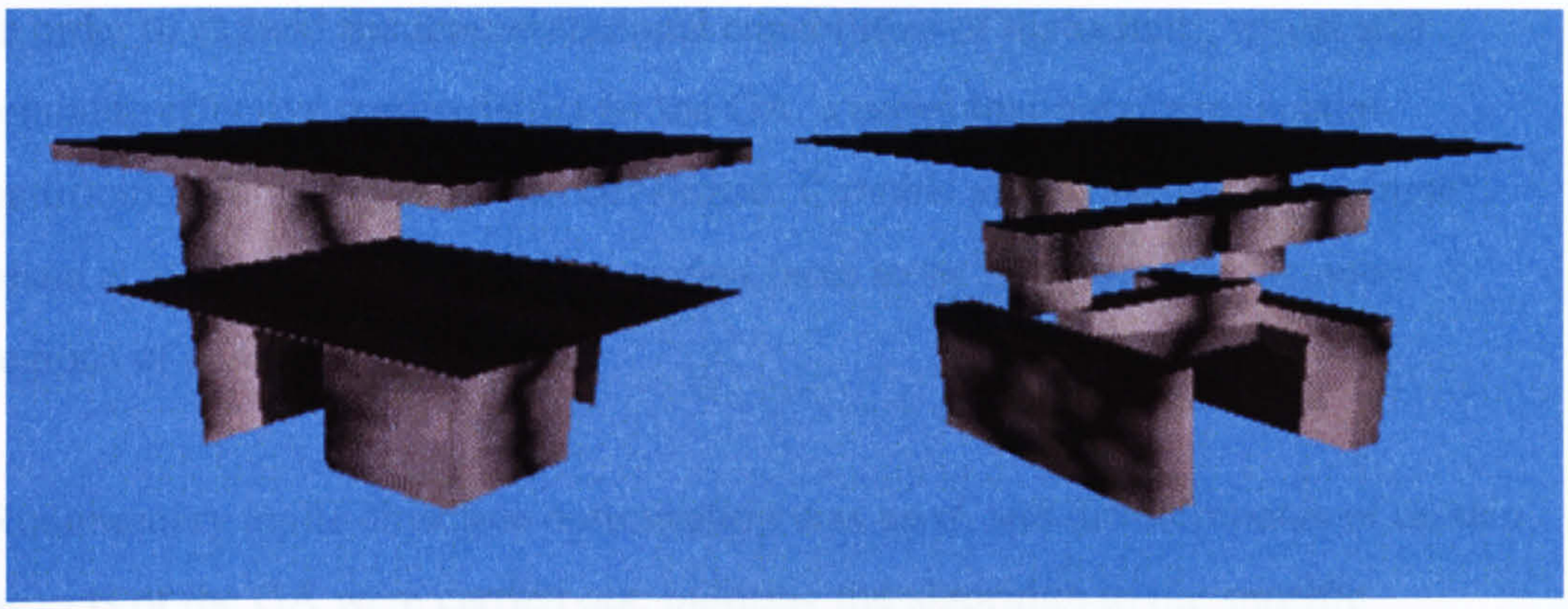
The earliest example of creative evolutionary design of products found in the literature is Peter Bentley's doctoral research into generic evolutionary design, 1993 onwards. This well cited research is introduced in an early conference paper [Bentley2], fully described in the Ph.D. thesis [Bentley3], and summarised in the final chapter of the book, "Evolutionary Design by Computers", [Bentley4].

The design system, known as 'GADES' (Genetic Algorithm DESign), has 'designed from scratch', objects such as tables, heat sinks and optical prisms (Figure 2.1.5), and a variety of streamlined designs. The designs are constructed from 'clipped stretched cuboids[1]', and, more recently, include a limited capacity for manufacturing considerations. The GADES system has also recently been applied to architectural problems – the latest example being a hospital layout design [Bentley4].

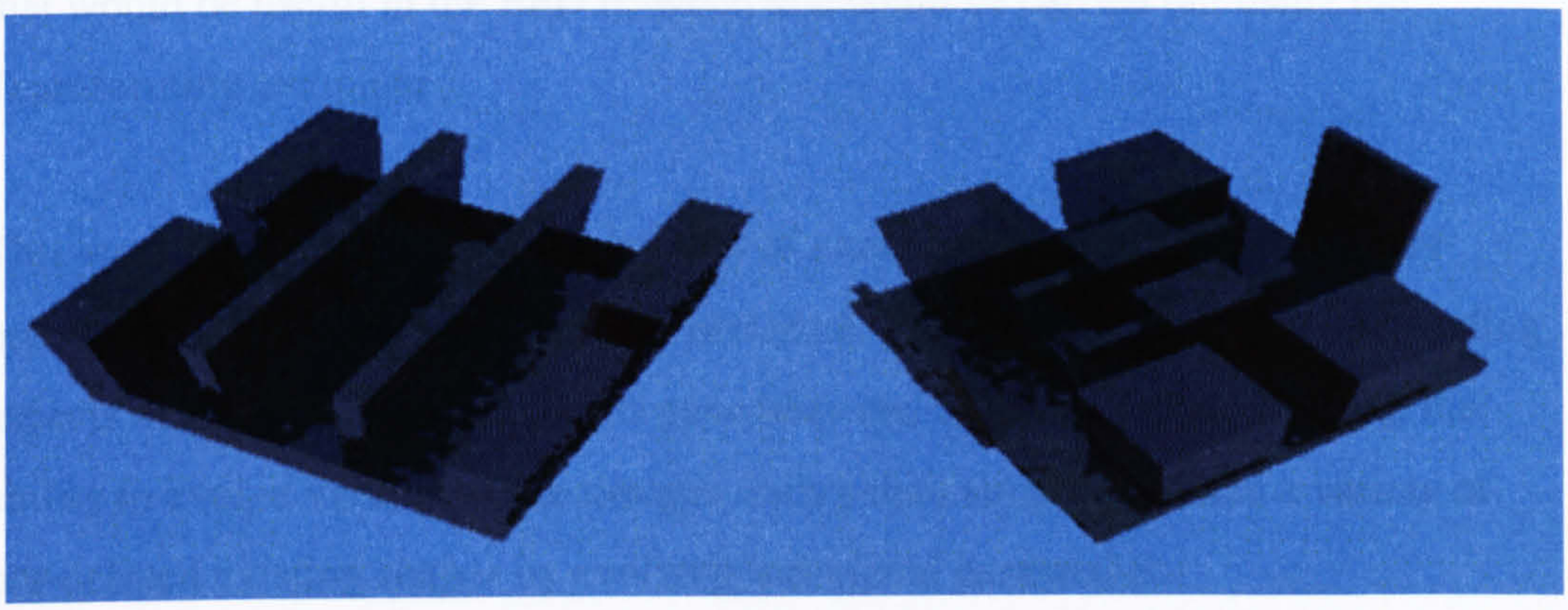
The system operates as follows: Appropriate analytical functions are selected from a library of evaluation modules, and a Genetic Algorithm is used to generate populations of candidate designs, which are assessed against these criteria. The system therefore, once initialised, needs no further prompting from the user, and presents a selection of optimised designs on completion.

Using the evolution of a table as an example, five evaluation modules are specified: size, mass, flat upper surface, supportiveness and 'unfragmented'. The system demonstrated consistent evolution of fit table designs, often with surprising creativity. A variety of different approaches emerged to increase the stability of the tables, without any specific instructions: Some designs have a very low centre of mass, some use a wide base, and by enforcing symmetry about two planes, the traditional four legged approach was 'discovered'

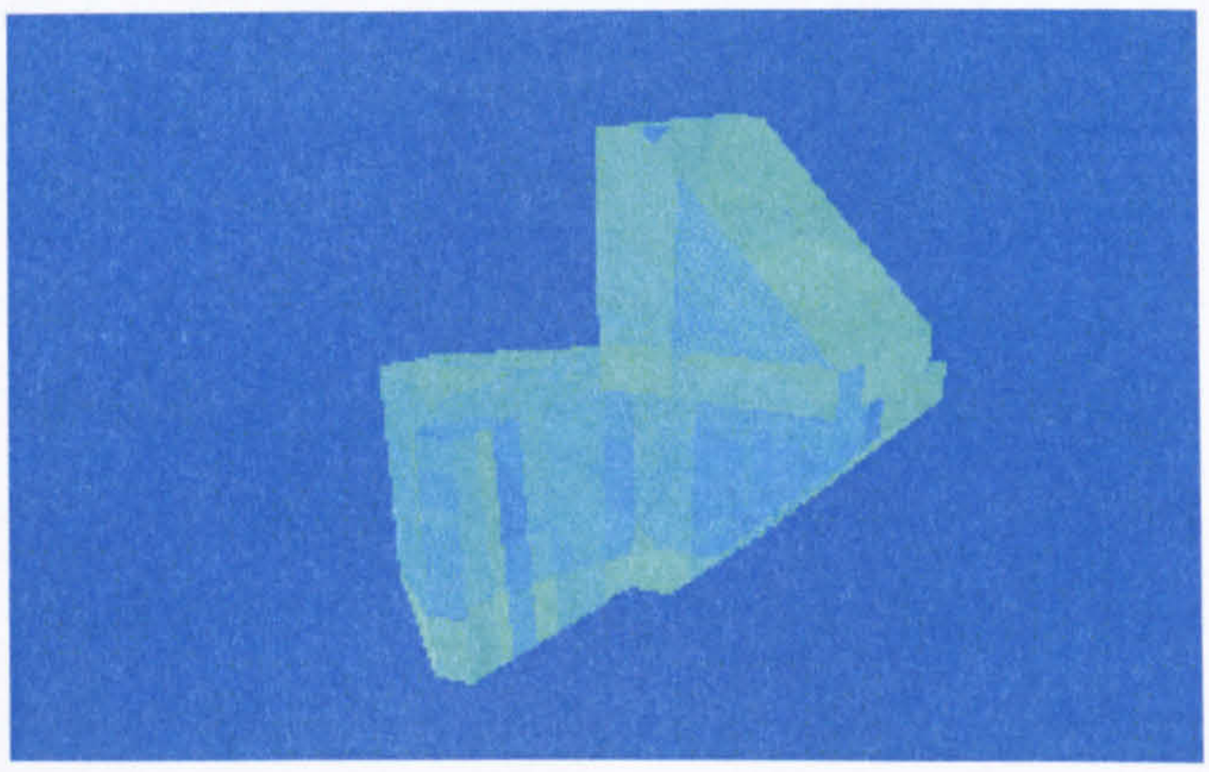
[1] Introduced overleaf and described in detail in the following chapter



Tables evolved for size, mass, stability,
supportiveness and flat upper surface



Heatsinks evolved to fit over a CPU, dissipating heat
from the processor by maximising surface area



An example of an evolved optical prism, in this case, a porro prism.
This application uses a ray-tracing module, and demonstrates the use
of 'clipped stretched cuboids' enabling the angled surfaces

Figure 2.1.5 - GADES evolved designs

In order to expand the representational capabilities of the system, whilst still retaining effective manipulation by the GA, a novel low-parameter spatial-partitioning representation was developed: Cuboids are intersected by a single plane at any orientation, allowing angled faces to be introduced to the outer regions of designs.

Rudimentary multi-objective optimisation was used, and developments of certain aspects of the GA were carried out, tailoring the algorithm to the specific application. These developments included variable-length chromosomes, enabling designs to be constructed from variable numbers of blocks (a capability that would be useful in the EFD research and could be carried out using the Teamforming technique).

This work is one of only a small number of examples of generic evolutionary product design found throughout the literature. The most original and outstanding aspects of the GADES work are its generality and creativity, demonstrating the ability to evolve various whole designs, rather than simply optimise a variety of pre-defined existing shapes by evolving individual parameters.

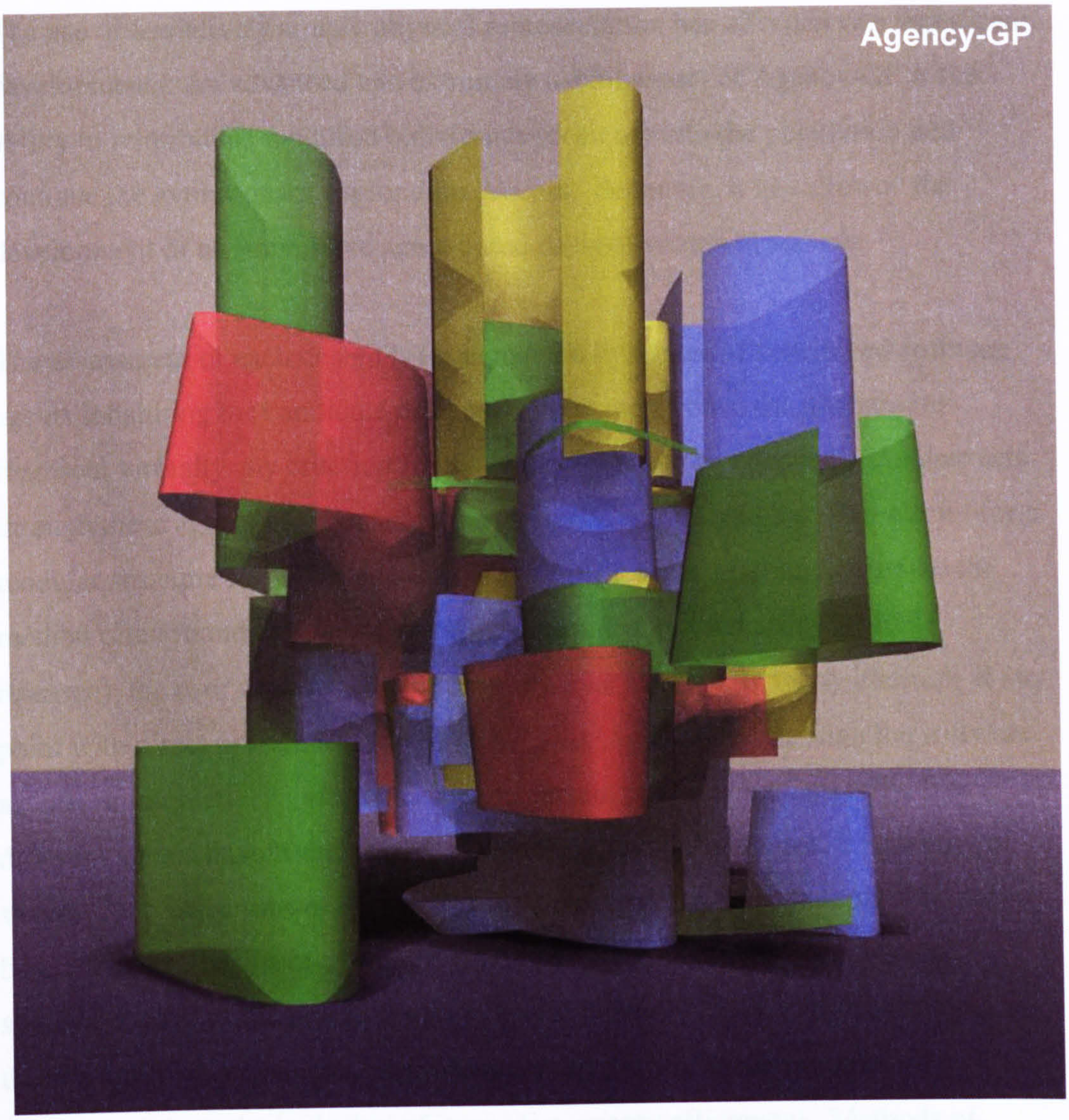
Agency-GP

Two software projects emanating from the Emergent Design Group [EDG] are prominent in the literature, and are of varying significance to this research: 'GENR8: Generative Form Modelling' [Hemberg], is a design tool that generates surfaces using 3D map L-systems and develops them using evolutionary search. Agency-GP [O'Reilly] is an architectural design exploration tool, using genetic programming and software agents. Output from both of these programs is shown in Figure 2.1.6.

Agency-GP is one of a few examples of genuinely creative evolutionary form design. Although the context and output styles are unique in each case, there are prominent similarities between Agency-GP and the EFD research in their philosophy and representation construction process. As with this research, the Agency-GP software structure is noteworthy for its integration with a high-end three-dimensional modelling environment, which allows users to modify the evolved objects directly using the available modelling utilities.

Representation is through one or more closed NURBS (non-uniform rational b-spline) curves. These curves are extruded to form surfaces and enclosed 3D spaces. There is here an exact parallel with the EFD work, in that repeated genetic data structures (one structure defines one primitive in EFD, or one NURBS curve in Agency-GP) are used to form whole genotypes[1]. Variation is achieved by transformation of these NURBS surfaces (translate, rotate, scale, cut). The similarities with the EFD work continue in that Boolean operations are then used – intersect, subtract and unite – to create new and differently shaped spaces. The ultimate intention is that each space is assigned its own architectural function (what the space is used for); where surfaces intersect and form new spaces, new functions would be allocated.

[1] Theoretically, here would be a potential application of the EFD Teamforming technique, enabling the evolution of rules that defined the 'intelligent' grouping of spaces



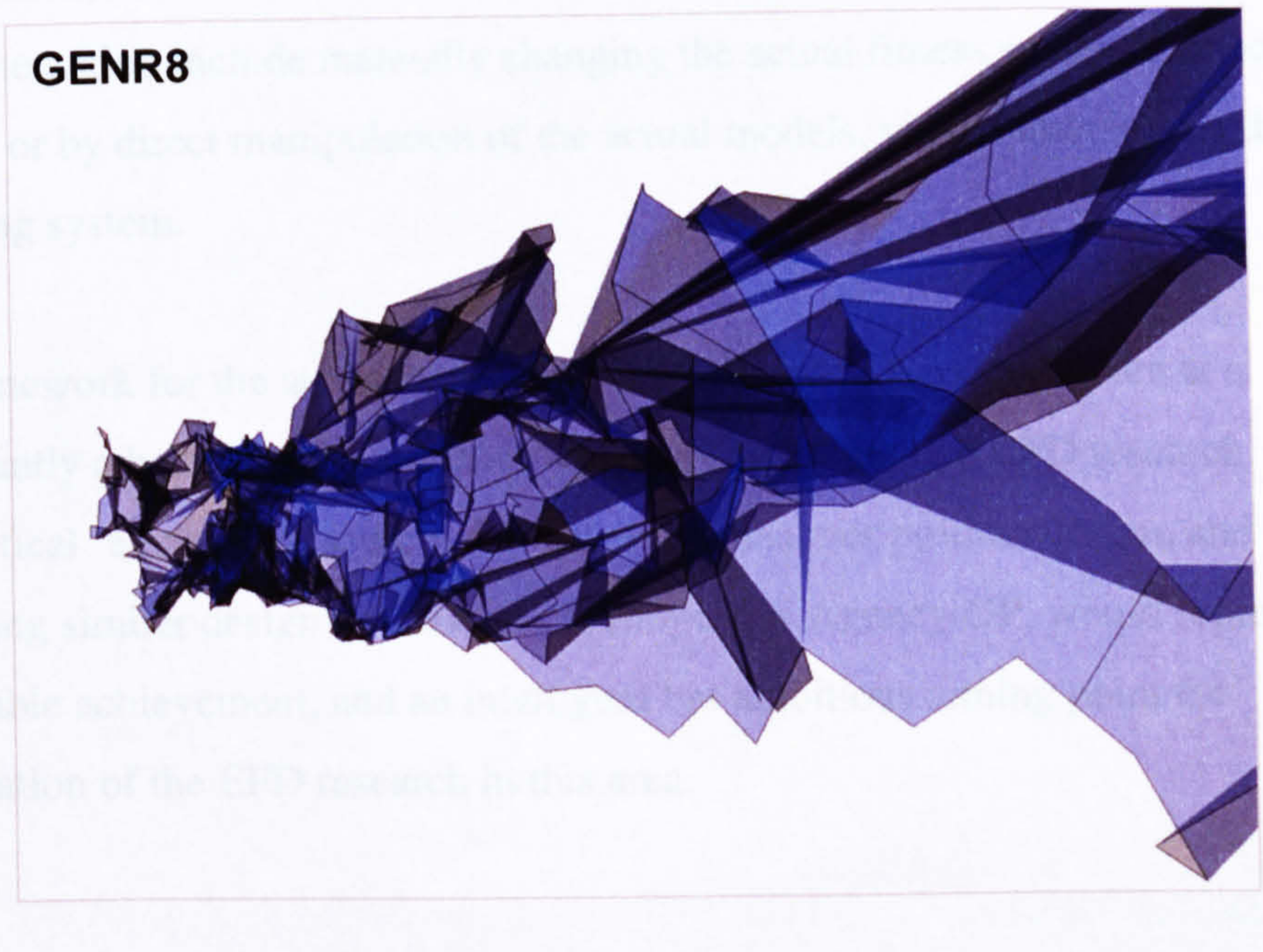


Figure 2.1.6 - Demonstration images from Agency-GP and GENR8

The use of a relatively simple physical representation has afforded two significant developments: An advanced and extremely useful aspect of Agency-GP is the ability to reintroduce modified individuals (designs) into the population and continue the evolutionary exploration process. Secondly, it has allowed the development of an impressive agent based design evaluation scheme.

The assessment of each design is implemented by means of distributed software agents inhabiting the candidate designs. These agents have the potential to represent virtually any criterion for evaluation that can be encoded – the user acts through these agents in influencing the evolution of the designs. This allows for a modular structure for the integration of multiple fitness criteria. So, unlike the method often found in interactive evolutionary design (including the EFD research), the user does not have to evaluate every design, but may interrupt at any point with direct or indirect intervention. This can either be through the software agents, or interactively, after viewing the current population. Currently, five different agents have been created, considering; 1) size of shapes, 2) quantity of shapes, 3) intersections of shapes, 4) compliance within a user-supplied bounding box, and 5) the height of the entire structure. More advanced concepts are suggested, such as the conveyance of general desires ('create a high structure over there'), specifying quotients, constraints or targets (i.e. space or light requirements), or the inclusion of issues (i.e. energy efficiency). Methods of direct interaction include manually changing the actual fitness scores of selected designs, or by direct manipulation of the actual models, via the interface of the modelling system.

The framework for the automated fitness capabilities in Agency-GP are at a significantly advanced stage of development compared to the EFD research. A hypothetical 'equivalent' system dedicated to consumer product design, and containing similar design evaluation capabilities as Agency-GP, would represent a remarkable achievement, and an intelligent but ambitious aiming point for continuation of the EFD research in this area.

Further examples of evolutionary design systems

Architecture

- Celestino Soddu's work has been widely publicised [Graham-Rowe], and exhibited internationally. Applications are predominately architectural (Figure 2.1.7), but have branched into design, producing families of chairs, lamps and coffee-pots [Soddu].
- John Frazer, over the past 30 years, has developing a theoretical basis for architecture using analogies with nature's processes of evolution and morphogenesis. A variety of evolutionary techniques have been used to explore architectural concepts (Figure 2.1.7) [Frazer].
- In contrast, Orestes Chouchoulas is only two years into research on architectural *Shape Evolution*, which combines Shape Grammars and GAs. An application that evolves apartment building designs (Figure 2.1.8) has been presented [Chouchoulas].

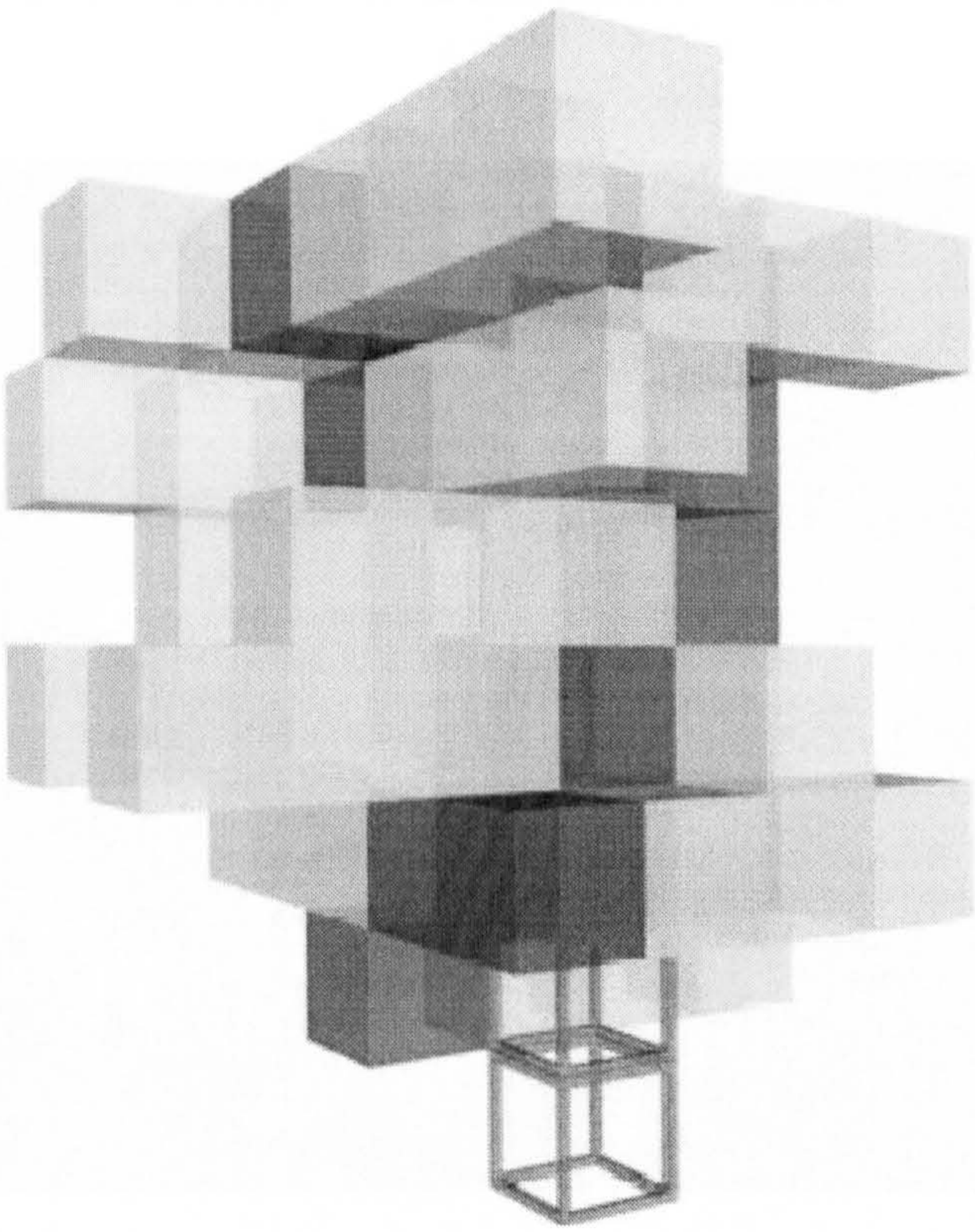
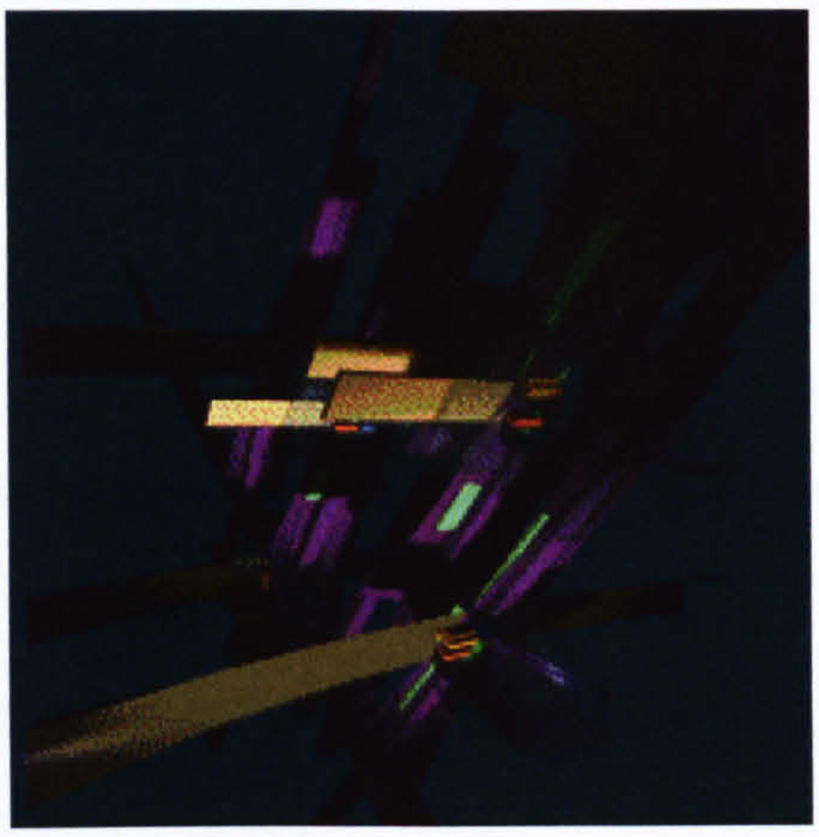
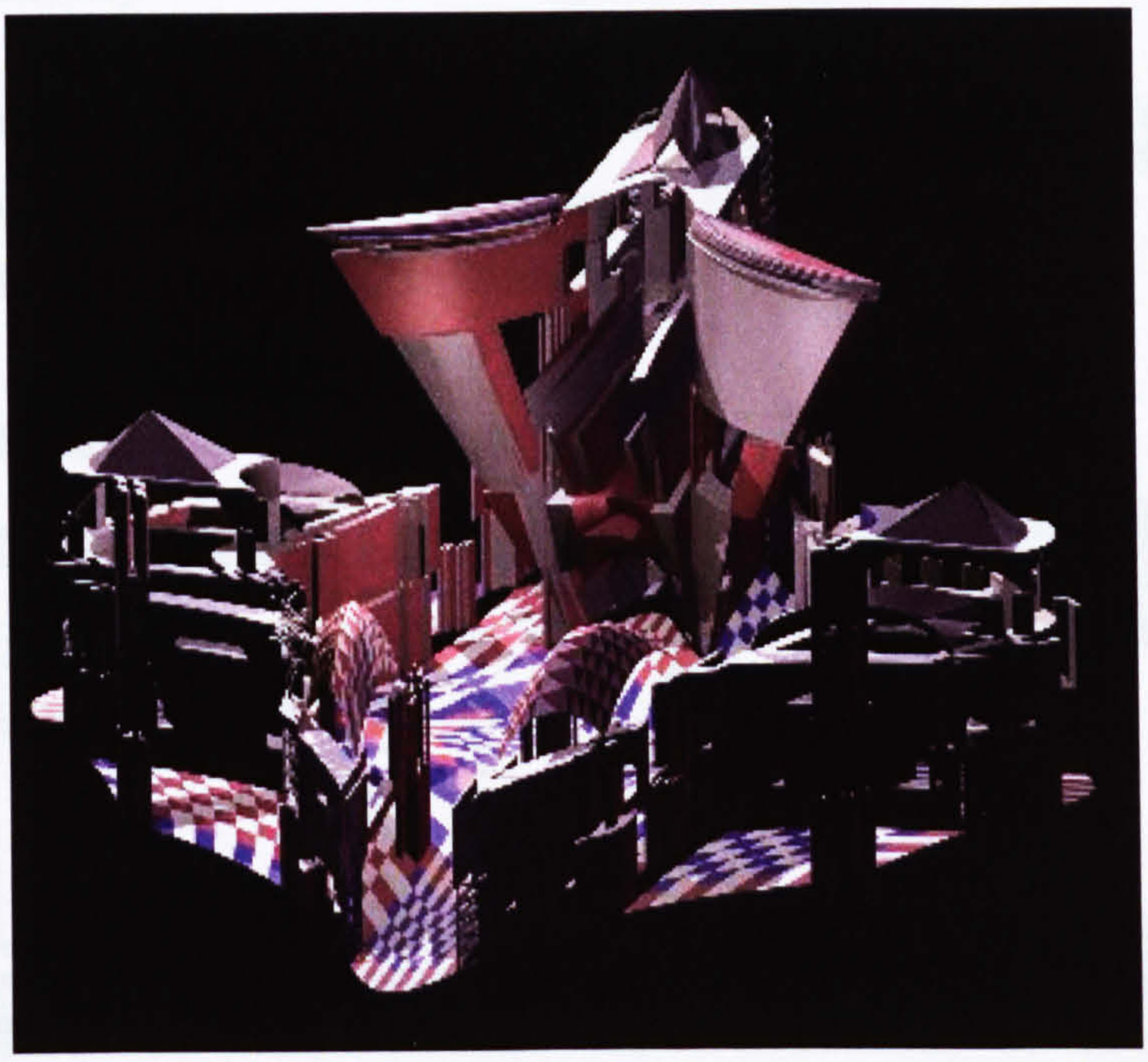


Figure 2.1.8 - Shape Evolution, Orestes Chouchoulas



Evolutionary Architecture, John Frazer



ARGENIA, Celestino Soddu

Figure 2.1.7 - Examples of Evolutionary Architecture

Industrial Design

- The design consultancy company *affinnova* uses interactive evolutionary techniques to enable clients to explore multiple product concepts. Variation is achieved by de-constructing designs (provided by clients) into features, expanding each feature into a range of alternatives, and then creating a population of new products by re-constructing combinations of these altered features. The concept is demonstrated through the example of bottled water packaging (Figure 2.1.9) on the company's website [affinnova].
- Michael Pontecorvo, chief technologist behind *Emergent Design,* presents three applications of generative design on the company's website: Chairs (Figure 2.1.9), packaging design, and virtual cityscapes [Gatarski].
- Mathew Lewis tackles the human form, virtual game environments and artistic textures (Figure 2.1.9) in demonstrating the versatility of utilising data flow networks for software, developed as part of research into aesthetic evolutionary design [Lewis].
- Claudia Eckert and colleagues have developed a generative system for garment design (Figure 2.1.10) that combines user interaction with an expert system [Eckert2].

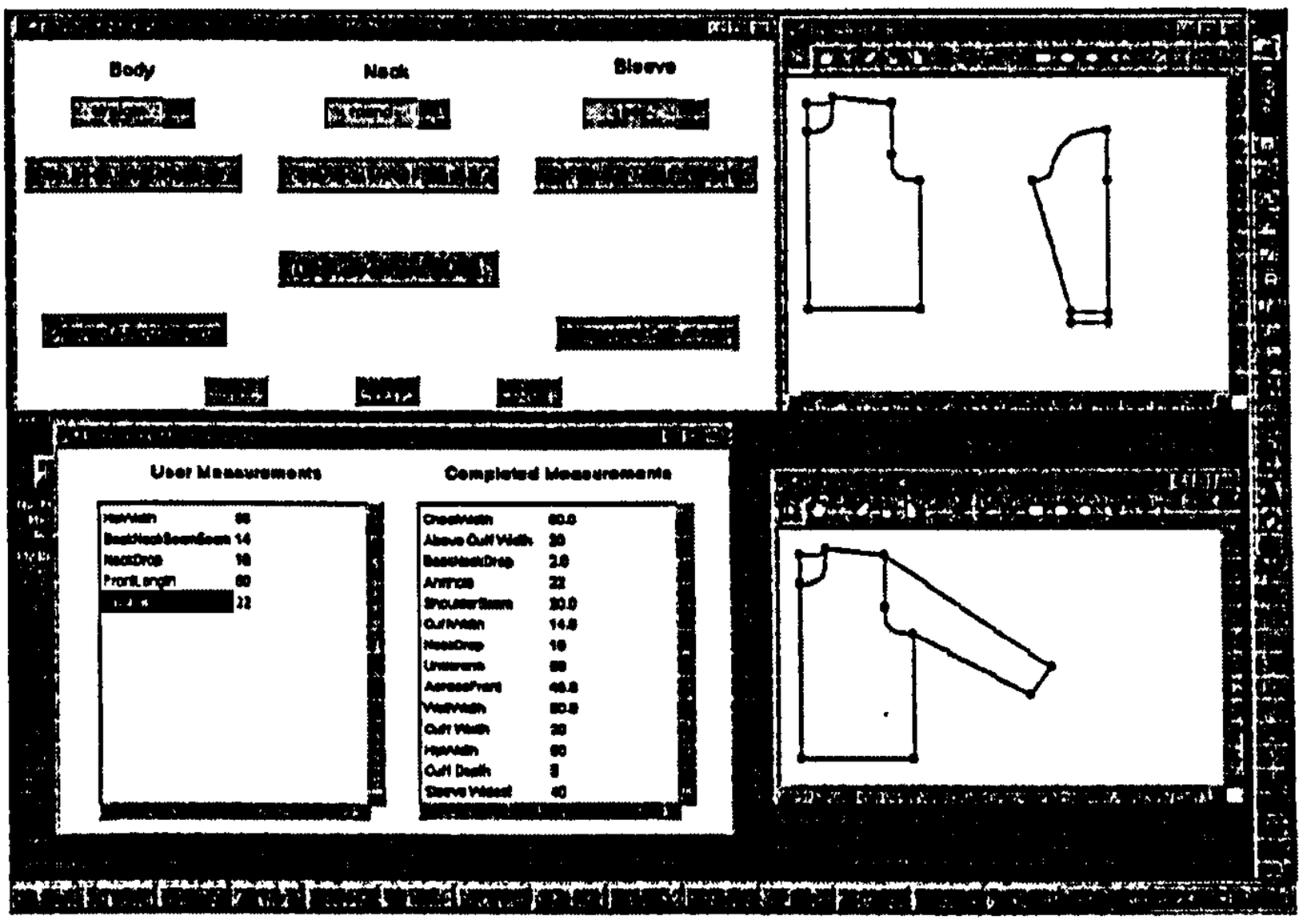
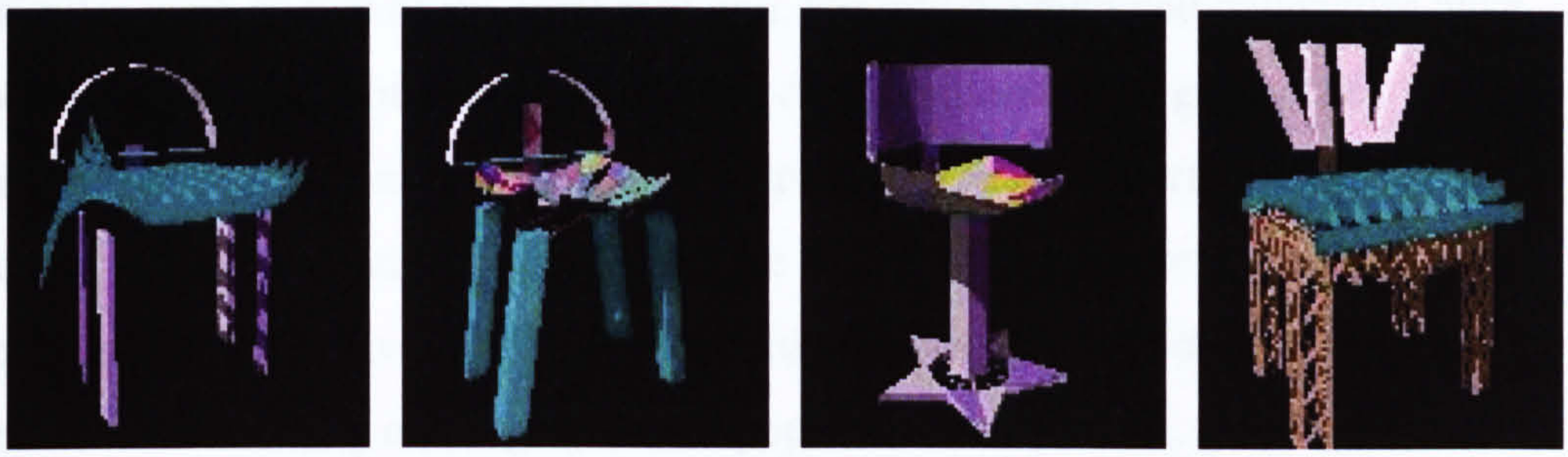
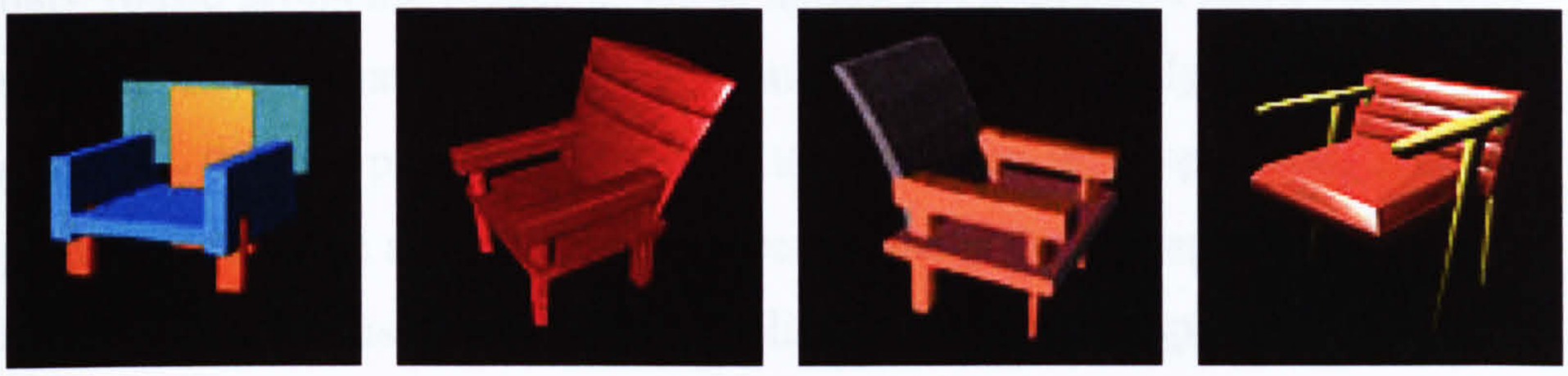


Figure 2.1.10 – Garment Design, Claudia Eckert



Argenic Design, Celestino Soddu



Emergent Design, Michael Pontecorvo

Metavolve, Matthew Lewis

Interactive Design by Evolutionary Algorithms (IDEA™)

Figure 2.1.9 - Examples of Evolutionary Industrial Design

In similar work to that of the EFD research, Hiroaki Nishino and colleagues have developed a digital prototyping system for designing novel 3D geometries [Nishino1]. The system representation employs the implicit surface method, allowing a global blend to be applied to the underlying structure of collections of superquadratic primitives with deformations. This representation allows the creation of interesting and highly varied populations of smooth, flowing forms, but is restrictive in applicability, given a tendency for abstract, free-form shapes. The problem has been addressed through the creation of 'roughly modelled initials' (basic geometric models). These initials are used (with slight variations) to seed the first population. Fitness allocation is provided solely through interaction – the user providing scores for the 20 shapes in a population. Experiments to create a green pepper shape compared the two approaches [Nishino2]. Results using randomly initialised populations displayed characteristics of the required shape between the 15^{th} and 25^{th} generations, and convergence to an approximation of the shape between the 30^{th} and 50^{th} generations. When the roughly modelled initial was used, not surprisingly, pepper-like shapes were visible from the outset, with convergence to a population of varied, but realistic pepper-shapes between the 7^{th} and 10^{th} generation (Figure 2.1.11). In concentrating on the optimisation of shape to a predetermined, preconceived form, the work has neglected the area of *creative* design, which it is clearly capable of addressing.

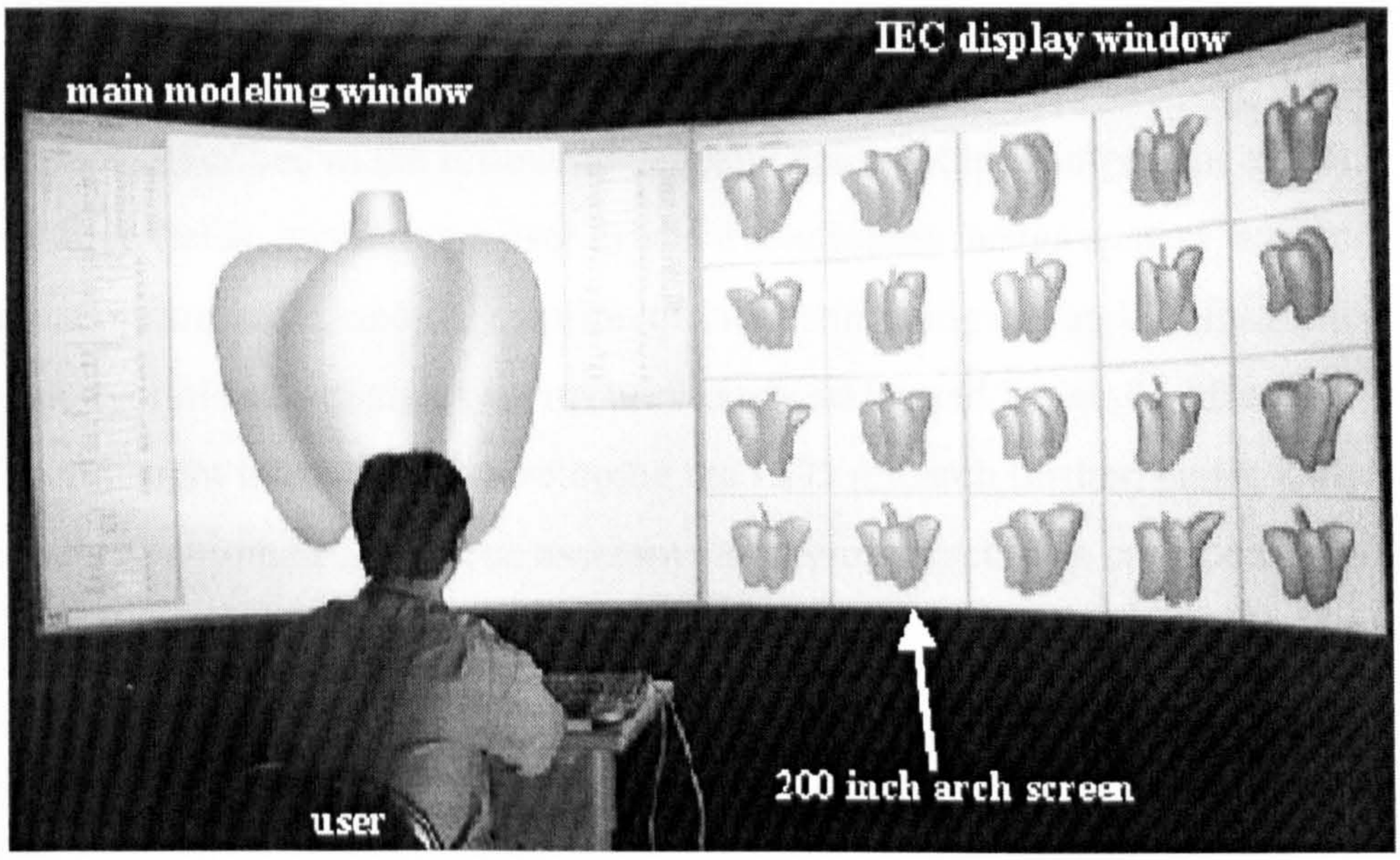


Figure 2.1.11 - Novel 3D Geometries, Hiroaki Nishino

2.1.4 Conclusions

1. The majority of relevant examples of evolutionary computation in engineering in the literature deal with optimisation of specific mechanical components against physical and measurable properties, such as stress, mass, heat flow, and aerodynamic resistance. The engineering systems studied are typically non-generic, being dedicated to particular components in each case, and non-interactive, relying on objective optimisation criteria. Examples addressing aesthetics are generally confined to civil engineering.

2. Evolutionary and genetic 3D art demonstrate a variety of ways of producing interesting shapes, steered by a user. The systems studied carry a certain 'signature' or style in their outputs, prohibiting their use outside the field of abstract art.

3. Bentley's work on generic evolutionary design is the closest to achieving the evolution of consumer products. The innovative 'clipped stretched cuboid' representation addresses the representation efficiency and genetic transfer issues associated with GA optimisation, but neglects aesthetic quality, perhaps restricting applicability in the field. The advantages of automatic optimisation of designs are offset by the requirements of the set-up process – new software modules are needed for any new parameters.

The research described in the following chapters has combined important aspects of these three fields, producing a user driven evolutionary design system, capable of internal optimisation, and the creation of interesting, original and useful forms. The other evolutionary design systems previously addressed in detail[1], offer significant insight into ways of developing the EFD research further, particularly in the areas of automated aesthetic assessment and manufacturing considerations.

[1] Wallace, O'Reilly, Nishino

2.2 Form and Aesthetics

Perception

Aesthetics is a more of a philosophical science, than a technical one, because perception depends on comparing images to previously stored images in memory. The brain, on receipt of information from the eye, compares the stimuli with its stored experience of similar stimuli patterns. Although there are many varied and often conflicting theories about aesthetic perception [Vihma], it seems important to note that the aesthetic properties of a product are not intrinsic, but are formed in the interpretation of the product when considering the product's function. In this light, it will be important to bring some functional consideration to the research in order to make rather less subjective and slightly more objective decisions about aesthetics.

Form

There are two separate, though strongly related manifestations of form. There is the internal form of the underlying basic structure, the anatomical embodiment of form, and there is the external form of the visual shape, the surface, the cosmetic embodiment of form [Ashford]. There are several well-established components of form; composition, surface, unity and proportion, these are discussed below.

Composition

The composition or arrangement of the parts of a visual pattern is, with engineering design, naturally very strongly influenced by the requirements of mechanical function and structure. However, people do have the capacity to make subjective comments about works of abstract sculpture, for example.

Surface

Surface describes the external form, or the visible shape. The quality of the external form is of no less importance than that of internal form, but relies upon the internal form being right.

Visual Unity

Unity and harmony are virtually synonymous and there is clearly some connection between unification and harmonic relationship, although the former is associated more with the reflection of visual characteristics and the latter with steps in size. The link is in the repetition of similar qualities, which can assist identification by making a properly unified visual arrangement easier to perceive. For example, in Figure 2.2.1, the left form is weak and visually disruptive, whereas the form on the right displays visual unity.

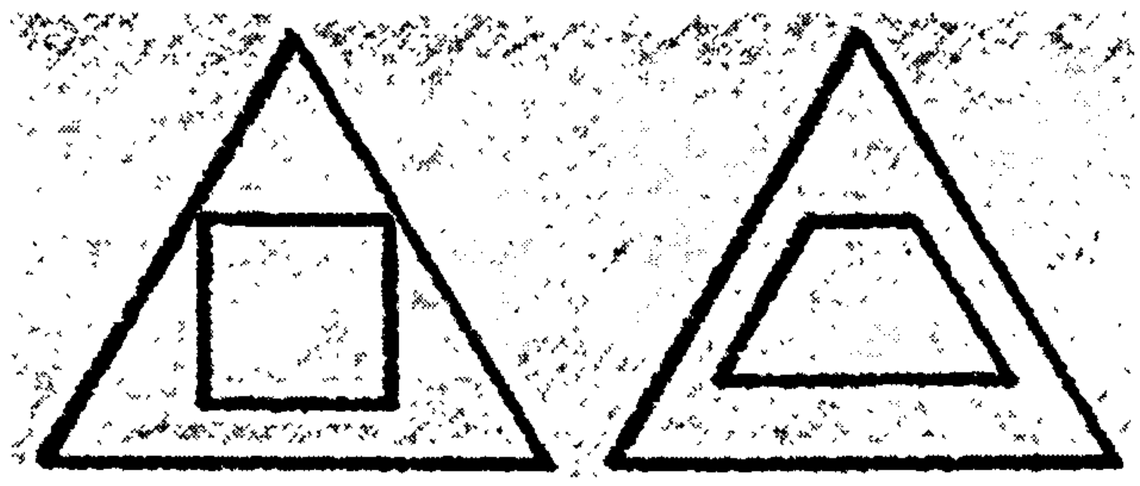


Figure 2.2.1 - A diagram representing visual unity

(Based on Fig 4.3 from 'The Aesthetics of Engineering Design' [Ashford])

Proportion

Proportion, the relationship of a part to the whole or of one value to another, can be concerned with what is a suitable ratio within a particular context and also with several formal systems having a more universal application. The Root Five Rectangle system is a commonly used guide to proportion, with others including the Golden Mean system, the Summation Series, compiled by Fibonacci, and the Archimedian or logarithmic spiral. These are introduced in the following section.

2.2.1 Formal Systems

Instead of trying to impose a theoretically ideal aesthetic quality upon a situation, it is better to integrate it with the situation [Ashford]. In other words, whilst there are many guidelines and 'rules' published and taught in the field of aesthetics and design, some of which are outlined below, they cannot all be applied to all products indiscriminately. It is for this reason that, early on in this research, it was decided to always include a human designer/user as part of the EFD system. Having said this, it would be interesting and possibly useful, to attempt to integrate artistic 'rules' of form into the software to some degree, and is the reason for reviewing this area.

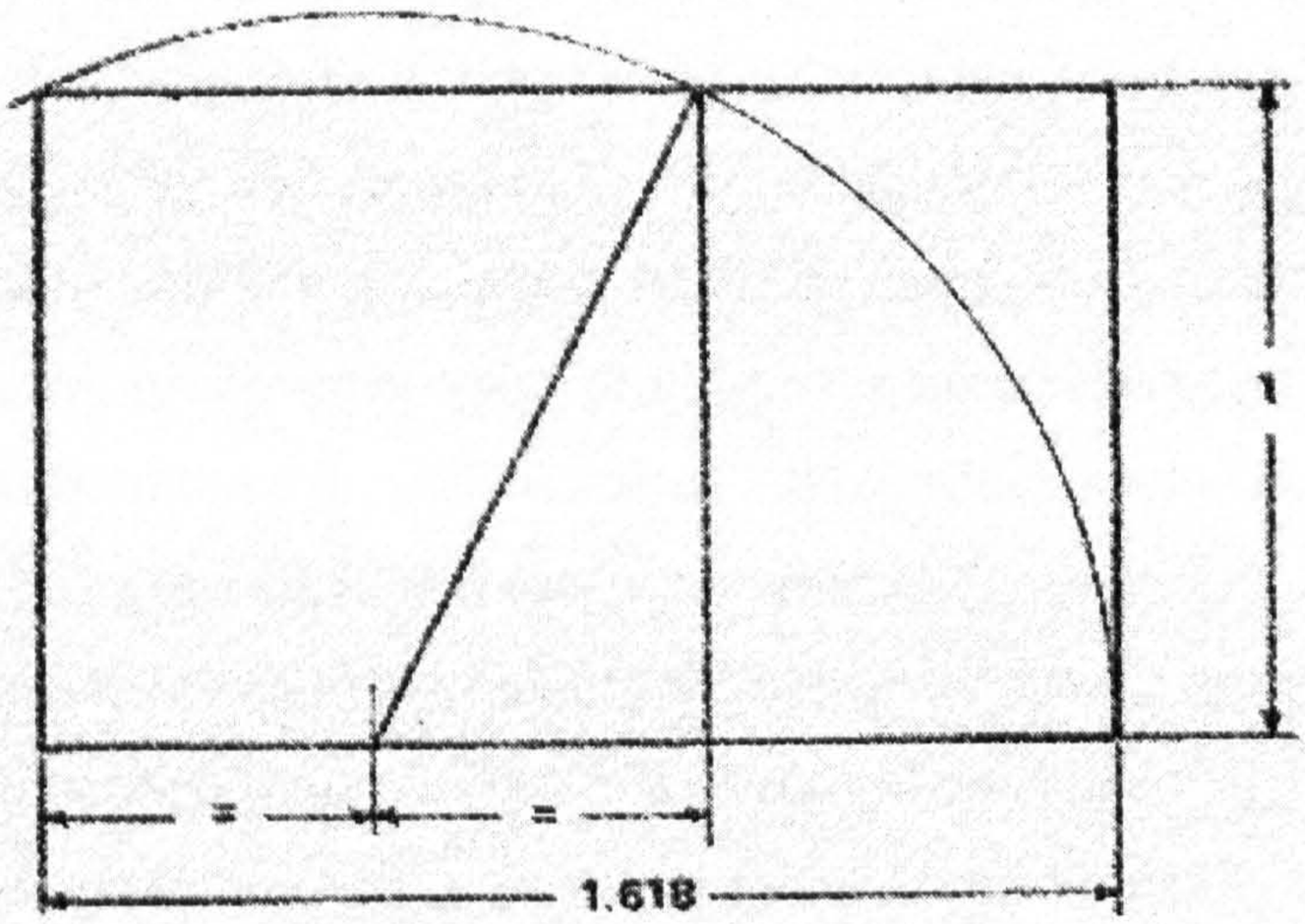
Why is proportioning important? Imagine a square with both dimensions equal. The onlooker will view it as square. If one dimension were to increase slightly, so that the ratio of the sides remains fairly close to 1, the outcome would be a rectangle that will consciously or subconsciously unsettle the viewer. The shape will seem *like* a square but not *exactly* square, as if a slight error has been made in constructing it. Forms with sides of ratios about 1:1¾, or more, will overcome this and it will seem pleasant and balanced. The Golden Mean (or Golden Ratio) has been preferred over any other ratio and has classically been thought of as giving a *perfect* balance. It has the value of phi (Φ):

$$\Phi = \tfrac{1}{2}\left(1+\sqrt{5}\right) = 1.618034...$$

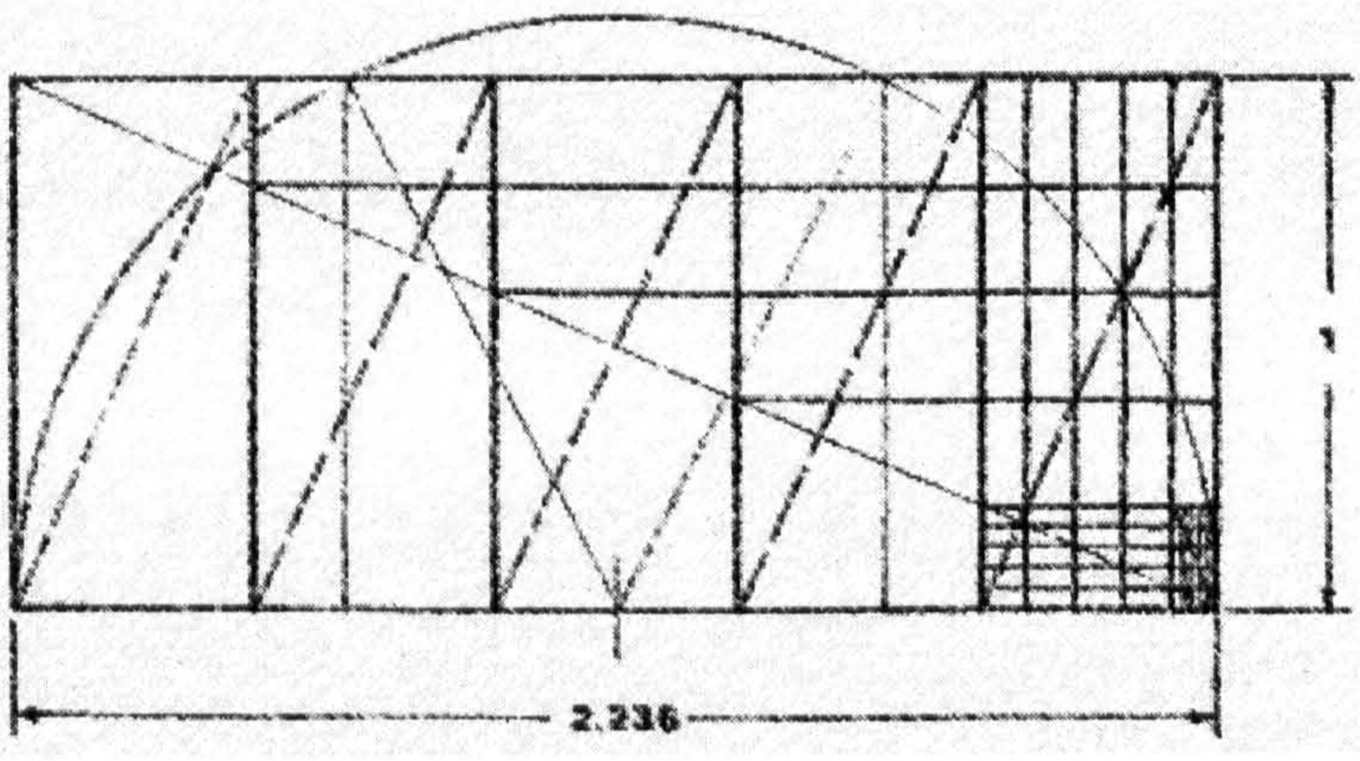
The Golden Mean System

A method for designing an object is often needed if the object is to be pleasing and balanced to the viewer. The method of proportioning objects using different incarnations of the Golden Mean has been known for many ages and was certainly known to the ancient Greeks, although it is still a matter of debate as to whether it was used formally in the construction of the Great Pyramid or the Parthenon. Figure 2.2.2a shows the construction of a Golden Rectangle by rule and compass methods, the same method as the ancient Greeks are known to have used. It is a method of proportioning that is most useful in two-dimensional works and has been used in many artworks, most famously in Salvador Dali's ' The Sacrament of the Last Supper.' The three related systems, which were all realised independently, are linked mathematically, and occurrences can even be found in nature.

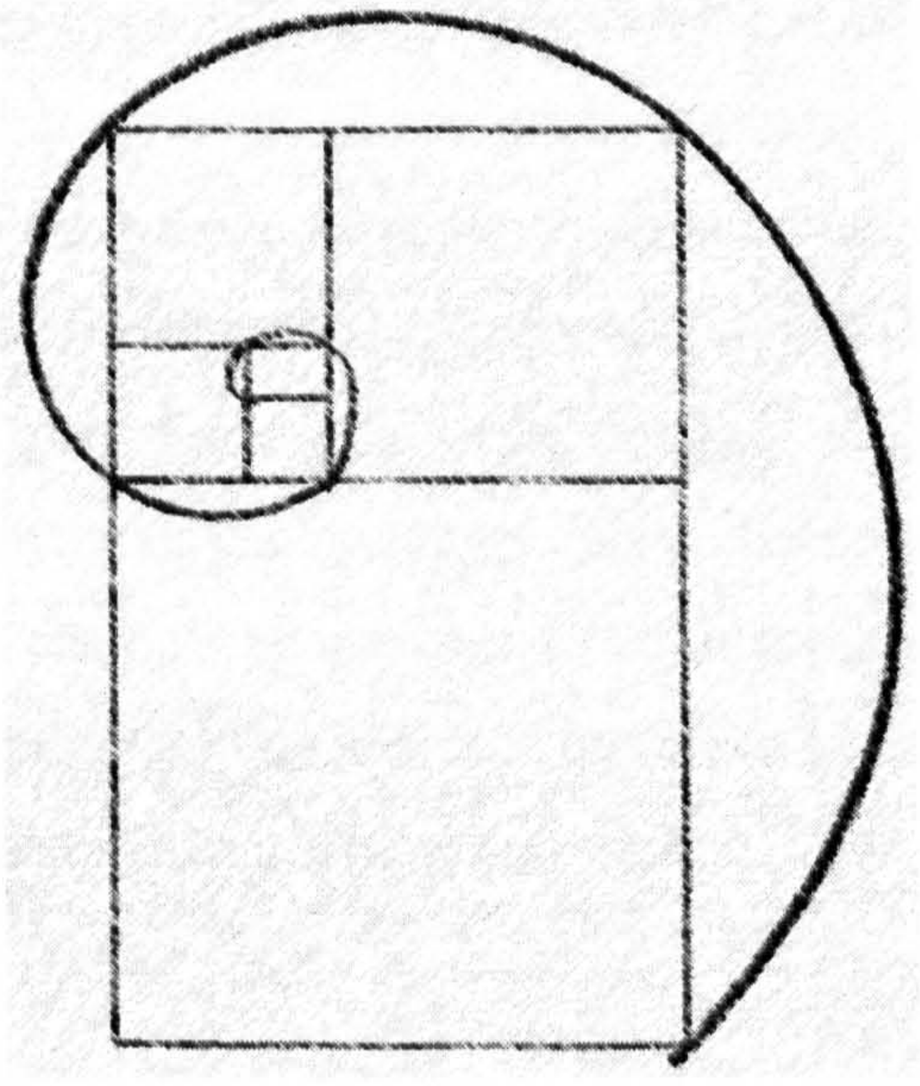
The Golden Mean system also gives rise to the Root-Five Rectangle system (Figure 2.2.2b), as its lengths (1 and $\sqrt{5}$) average to the Golden Mean, and shares the benefits of the Golden Mean system. Any Golden Rectangle is also infinitely divisible, always retaining its original proportion. Any smaller segments are harmonically related to the whole. It is for this reason that it remains such a useful tool. It enables a designer to proportion the bounding surface of an object and then create many subdivisions within that object whilst still maintaining the balance overall.



a) The Golden Mean



b) The Root Five Rectangle System



c) The Archimedian Spiral

Figure 2.2.2 - Mathematical proportion systems

A sequence of Golden Rectangles, as seen in Figure 2.2.2c, gives rise to a logarithmic spiral. Named the Archimedian spiral it is similar to spirals that can often be seen in nature. The seeds of a sunflower radiate from the centre in approximations to logarithmic spirals as do the segments around a pineapple, up from the base. An Archimedian spiral will have the same form however many iterations it has been through, such as a snail's shell, which always has the same form whether it is a tiny shell, of a couple of millimetres across, or a large one, several centimetres across. The numbers of segments in these natural spirals appear in the Fibonacci sequence, due to the ratios of corresponding Fibonacci numbers. Adding together consecutive terms to make the following term creates the Fibonacci sequence: 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34 etc.. Displayed below are the ratios of corresponding Fibonacci numbers, which converge to phi (Φ):

$$\frac{1}{1} \rightarrow \frac{2}{1} \rightarrow \frac{3}{2} \rightarrow \frac{5}{3} \rightarrow \frac{8}{5} \rightarrow \frac{13}{8} \rightarrow \frac{21}{13}$$

$$=$$

$$1.0 \rightarrow 2.0 \rightarrow 1.5 \rightarrow 1.666... \rightarrow 1.6 \rightarrow 1.625 \rightarrow 1.615...$$

Conclusions

Fundamentally, the matter of proportion is associated with an intuitive consideration of balance and composition, and with the achievement of a sense of direction in form. Formal methods, such as those discussed, are frequently used (Yee), but have limited use when creating three-dimensional objects. They may be used, in the first instances, to design one aspect of an object, however, as soon as the object is viewed in three dimensions, or the designer wishes to make small alterations, the formal methods can no longer apply. Therefore the Golden Mean and Root Five methods can only be used as a preliminary guide to a designer, but could be usefully incorporated in the research discussed here, during work on developing aspects of automated aesthetic assessment.

CHAPTER THREE

TECHNOLOGY REVIEW

Although there are many ways of tackling problems with large search spaces (Constraint Modelling, Hill Climbing, Simulated Annealing etc.), only Evolutionary Algorithms such as the Genetic Algorithm have been applied successfully in all areas of automated design (Bentley5, Khatib, Bäck, Forrest, Gen). Uniquely, EAs are *generative*, and deal with populations of solutions concurrently, allowing designers to explore numerous, *creative* solutions to widely varying problems. Evolutionary Algorithms are thus most likely to benefit from an interactive approach. In addition:

- EAs work effectively with complex, ill-bounded, unspecific problems[1], and are good, general-purpose problem solvers.
- GAs resemble natural evolution more closely than other evolutionary methods, and share many similar characteristics with the human design process.
- Genetic Algorithms remain the most widely used evolutionary technique.

Since the application outlined presently, is in a relatively new and unexplored area of research, it is appropriate to focus on a well documented technique that has been shown, through extensive trails, to be reusable and robust (Goldberg).

---

[1] Where the functional relationships between parameters and objective function values are of unknown, arbitrary mathematical character

3.1 Genetic Algorithms

The Genetic Algorithm is a search procedure closely based upon the mechanics of natural selection, and as such, various terms from biology are adopted to describe aspects of the technique. GAs work with a *population* of solutions, with each individual (or member) within the population referred to as a *phenotype*. This arises from the fact that, uniquely to GAs, each solution is encoded and manipulated as an artificial chromosome (or chromosomes), referred to as its *genotype*. The space defined by the genetic representation is therefore the search space, with the solution space being defined by the phenotype representation. The genetic representation is usually made up of strings of 1s and 0s, but can be real coded using lists of parameters, or consist of sequences of instructions.

So for each consecutive *generation* of solutions, the genotypes are mapped to the phenotypes to evaluate them according to the task in hand. The evaluation of phenotypes distinguishes good solutions from bad ones, and can involve complex computer simulation or modelling of each individual, or simply require a human operator to intuitively rate each or some of the designs. In either case, it is this allocation of *fitness* that guides the GA to evolve improved generations.

The operation of a GA is made up of three phases: Initialisation, fitness determination, and reproduction. After the initialisation phase, where an initial random population is created (or sometimes the initial population is seeded with variations of known good solutions), the fitness determination and reproduction phases repeat cyclically, the intention being for a solution or pattern to progressively emerge. During the reproduction stage, *selection* and *genetic operators* act on the population. Selection involves choosing good solutions to be *parents* of new solutions. Parents are usually selected in pairs, using ranking, probabilistic, or tournament methods, based on fitness scores. Genetic operators are used to carry out *recombination* and *mutation* of the parents' genotype, creating these new and possibly improved offspring. Recombination is usually achieved through *crossover* of the parents' chromosomes, while mutation simply modifies an individual's genotype.

3.2 Representation

Phenotype

Product representation defines the subset of the shape space that the GA can search, and as such will characterise any work on evolutionary shape design, more than any other factor. The amount of data that needs to be processed by a GA should be kept to a minimum, to reduce the size of the search space and the behavioural complexity of the system. This is a well-recognised consideration, especially when dealing with 3-dimensional objects, and is acknowledged by most of the referenced examples in the literature.

3D representations vary in their directness to the genotype, ranging from; explicit definition of every point in a design, as in voxel representation; through partial definition, such as specifying nodes, from which FE meshes are generated; to embryogeny models, where designs are grown according to a set of rules, such as cell division models. The representation selected for the research discussed here – geometric primitives combined through Boolean operators – lies somewhere in the middle, in terms of the amount of decoding necessary during the mapping process.

Perhaps the simplest representation conceptually, voxel representation defines every point within a model explicitly. The grid of voxels can be represented by one long single-string binary chromosome, although this format of genotype requires a specific crossover operator to function meaningfully. A voxel-based shape representation offers a number of potential advantages for shape optimisation; topology need not be defined, geometric constraints are easily imposed and theoretically, with adequate resolution, any shape can be approximated [Baron]. In reality, various techniques have had to be invented to cope with the problems of voxel representation. For example, the lack of boundary smoothness and the inclusion of holes in the design can be alleviated with various smoothing mutation operators. It could be the case that this is an example of the point made in [Bentley5], that a poor representation will be disrupted by all standard operators and may require the creation of specially designed 'non-disruptive' operators. Additionally, it is often necessary to

initialise the GA with variations of an existing design if successful voxel-based optimisation is to be achieved. However, its conceptual simplicity, and the affinity with FE analysis may mean that voxel representation remains in use for component optimisation applications.

A cell division model, employed in the system for free-form shape representation introduced in the previous chapter [Taura], takes representation to the other extreme, in terms of mapping the genotype to the phenotype. Here, a small number of cells placed on the surface of a sphere, are divided (according to a set of evolving rules), and spread out on the surface of a sphere. The surface shape is generated by referring to the cell density: The density of cells at a point (A) on the surface is converted to a distance, and a point is created at this distance from the sphere's centre (O) in the direction OA. The points are converted to a tessellated surface, with resolution dependant on the number of surface points sampled. There are no specific problems identified with this technique, just the limitations of applicability mentioned before.

A good example of a compromise between the amount of defining data and representation potential is the technique employed by Bentley for his GADES system [Bentley2]. Here, early designs were made up of collections of aligned, regular cuboids, keeping the amount of data to a minimum, but restricting the range of useful product definition. Angled faces and edges were introduced by allowing each cuboid to be intersected by a plane, of any 3D orientation and relative position, allowing the creation of more complex shapes, but without dramatically increasing the amount of data required. This 'clipped stretched cubes' representation (Figure 3.2.1) was arrived at through the natural progression illustrated overleaf.

Representation	Parameters	Summary
1) Cuboid defined by origin and lengths	6 parameters: *x*, *y*, *z*, *width*, *depth*, *height*	Unable to define angled faces
2) 6-sided polyhedron defined by corner points	24 parameters: 8 × *x*, *y*, *z*	Too many parameters
3) 6-sided polyhedron defined by intersecting planes	18 parameters: 6 × *angle1*, *angle2*, *distance*	Still too many parameters, 'ambiguity'
4) Cuboid with movable side	9 parameters: *x*, *y*, *z*, *width*, *depth*, *height1*, *height2*, *height3*, *orientation*	Combines benefits of 1 and 3, but tessellation problems, and limited to rectangular base
'Clipped Stretched Cube'	9 parameters: *x*, *y*, *z*, *width*, *depth*, *height*, *angle1*, *angle2*, *distance*	Low number of parameters, allows angled faces and multiple sided polyhedra

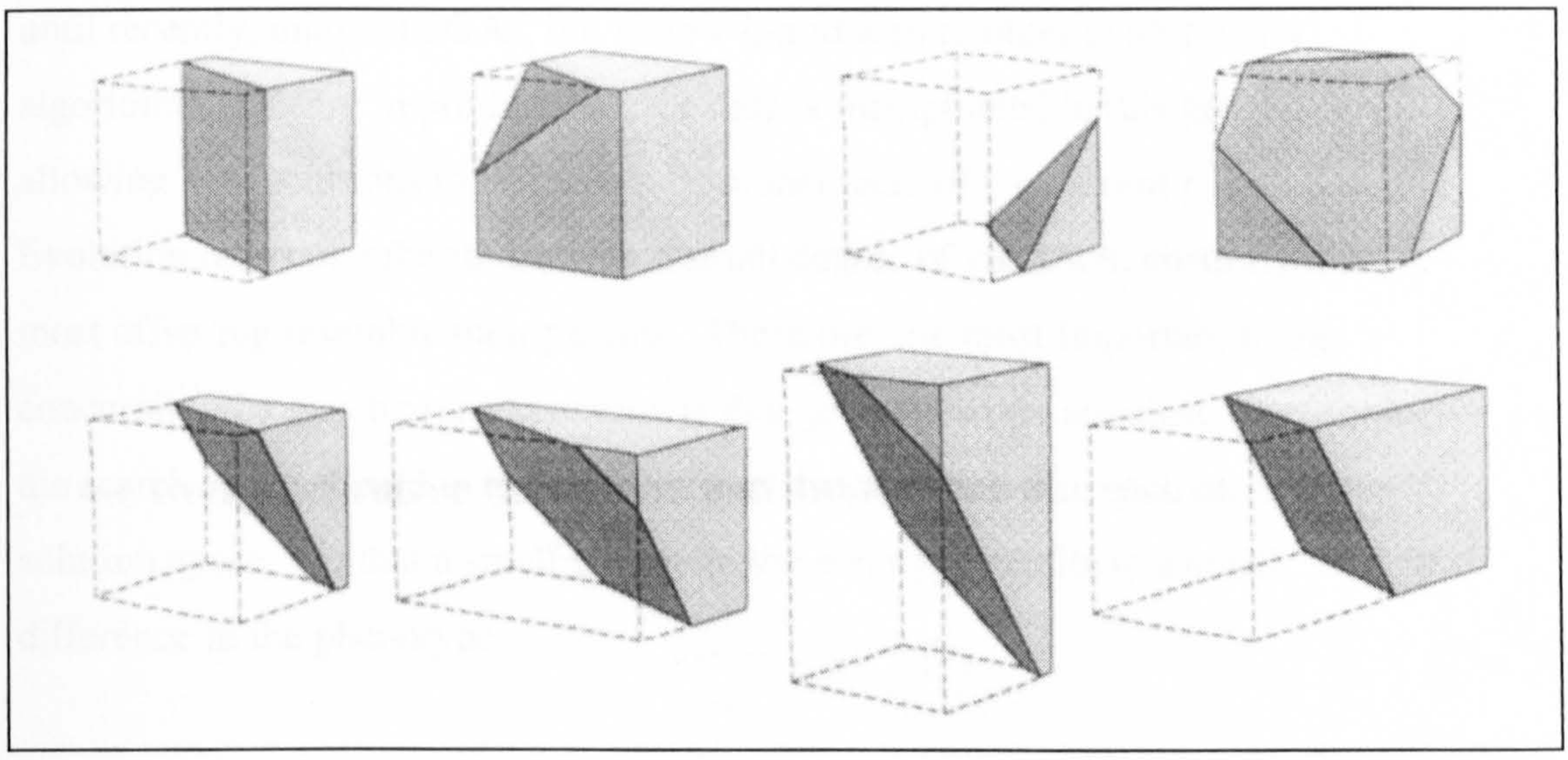


Figure 3.2.1 – Clipped Stretched Cubes

(Reproduced from Figure 4.4 in [Bentley2])

Genotype

One of the defining features of GAs is that populations of solutions are encoded in a genetic format, the genotype, which is then translated, during a mapping stage, into the solution, or phenotype. The way the data is stored in the genotype is analogous to biological DNA, although the strings of data usually employ '0, 1' binary representation rather than the four-letter 'A, T, G, C' alphabet of DNA. Bit strings are often divided into artificial 'chromosomes', or 'genes', according to their decoded purpose, and may be further subdivided into segments for each decoded value (or allele), as in Figure 3.2.2 below.

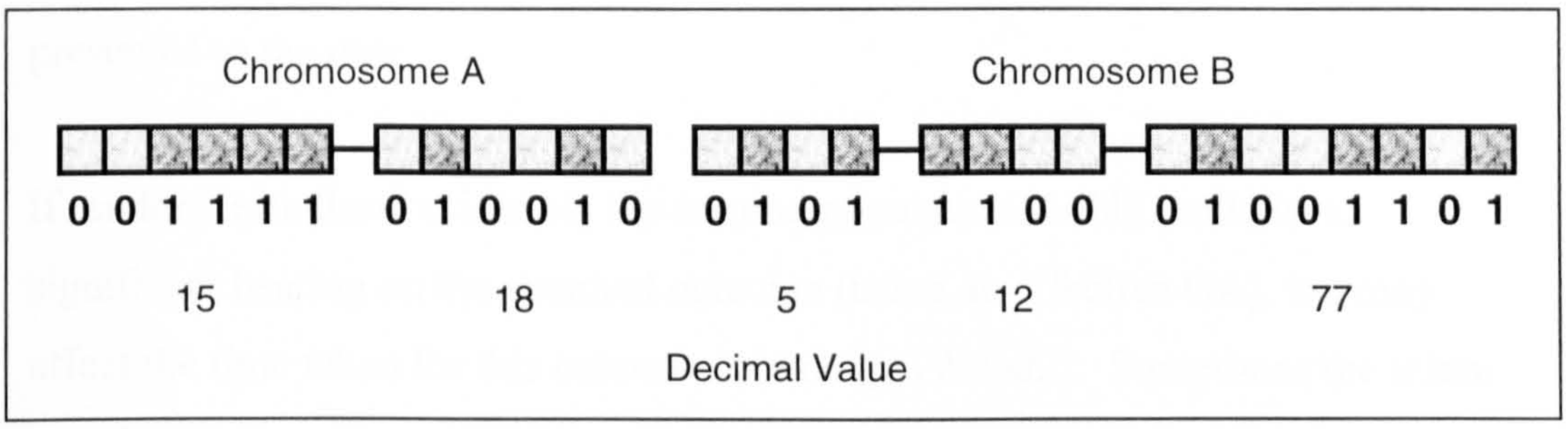


Figure 3.2.2 – Binary Genotype Example

This basic information is 'decoded' (in the fitness determination stage) in order to relate the information to the particular objectives. This 'mapping' process was, until recently, unique to GAs, but is now introduced to other evolutionary algorithms. During 'reproduction', the data is manipulated in this genetic form, allowing new solutions to be created from members of the current population. Evolution relies on inheritance with a small degree of variation, ensuring that most offspring resemble their parents. Therefore, the most important thing concerning the genetic representation is that genotypes that are close together in the search space should map onto solutions that are similar to each other in the solution space. So that a small change in the genotype results in a small difference in the phenotype.

3.3 Initialisation

The first population is created during an initialisation stage, usually by randomly selecting each bit of a binary chromosome(s) for each individual population member. So 'generation 0' is populated by solutions that have fixed structures and meaning, but random values. The aim is to create a diverse population of solutions. By maximising the initial distribution of members, the algorithm has the best chance of finding potentially fruitful areas of the search space, rather than homing in on sub-optimal solutions. This is equally important for interactive evolutionary design, so that a wide selection of different 'starting points' is presented to the user.

If random initialisation is used, the starting population should not have a significant bearing on the eventual outcome (using an effective GA), but may affect the time taken for this outcome to emerge [Wood2]. Sometimes the initial population is constructed from variations of a user-supplied solution, such as in the 3D Shape Optimisation work [Nishino] outlined in the previous chapter.

The initialisation section is also used to set the control parameters, such as number of members, number of generations etc., and to read input parameters. When this stage is complete, the GA moves in to the two-phase evaluation-reproduction cycle, which is repeated until either a satisfactory solution emerges, the population converges[1], or the GA has run for a pre-determined time, or number of generations.

[1] When the genotypes, phenotypes, or fitness values of all individuals are static for a number of generations.

3.4 Fitness Determination

Decoding

A genotype may, for example, contain 2 chromosomes, each split into 2 segments. If each segment contained 6 bits, then it has the capacity to represent a range of 64 integer values. The pair of such chromosomes could be decoded to describe an area in a two-dimensional 'search space', containing 64^4 variations or solution points. This example is illustrated in Figure 3.4.1 below, where the first chromosome is used to define the origin (lower left corner), and the second defines the length and height.

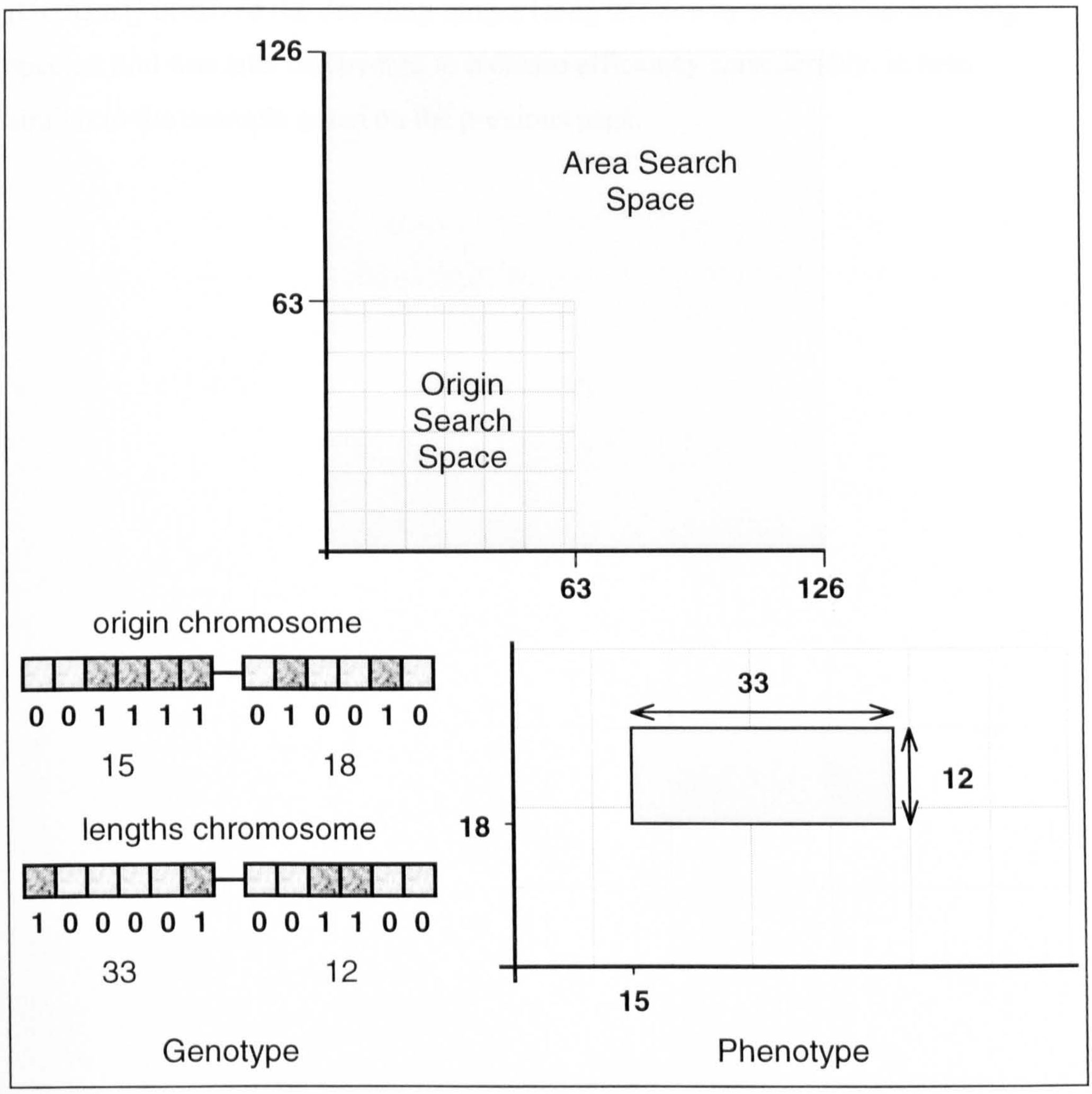


Figure 3.4.1 - Decoded values used to define an area

An alternative scheme would involve the second chromosome defining an opposite corner, this would mean that both points are independent, rather than the first example, where the location of the second corner is dependent on the first. The first method, where chromosomes define tangible characteristics without an increase in data, is generally preferable, allowing more progressive changes during evolution.

If these values are to be interpolated within ranges (rather than used directly) it is usually necessary to predetermine upper and lower limits, although there are methods of dynamically re-scaling the search space as the GA progresses, in order to increase accuracy and efficiency [Wood1]. Previous work by the author [Graham1] involved the decoding ranges being defined by a second co-evolving species, and was later discovered to increase efficiency considerably, in tests similar to the example given on the previous page.

Objective Function

Decoded chromosome data are effectively estimates of the required parameters. Solutions made up of these individual parameter values are evaluated (often by external software) through simulation, analysis or calculation, relating the estimates to ideal values, or the rest of the population. An objective function (S) is a method of calculating how well a solution fulfils the problem objectives, or more specifically, how close the individual estimates (A) are to target values (T). The following equation is generally used to increase the effect of large differences and convert any negative values to positive:

$$S = \sqrt{\sum (A - T)^2}$$

Treating a collection of difference values together in this way only works if the values are of the same order and significance. If this is not the case, then a multi-objective approach is often necessary, involving more than one fitness function. If the problem is subjective, relying on interaction with a human evaluator, or if comparisons with other solutions are the sole route to fitness, often only a fitness function is required.

Fitness Function

A fitness value (or fitness values, for multiobjective problems) describes the relative 'success' of individual members, compared to the whole population. Where fitness is derived from an objective function, a simple, but usually quite adequate method of defining the fitness function (f) is taking the reciprocal of the objective value. A scaling factor, k, (for multi-objective problems) can be introduced at this point:

$$f = \frac{k}{k + S}$$

Fitness values are therefore typically real, positive numbers, between 1 and 0, where 1 represents perfect fitness. Average fitness, and other derived statistical values, are necessary for subsequent operators, but are often also used to evaluate how effectively the GA is performing.

3.5 Reproduction

3.5.1 Parent Selection

Selecting fitter solutions to parent the next generation is the usual method of inducing a pressure towards the evolution of fitter populations. The selection of parents is intended to give fitter members a greater chance of reproducing than the less fit individuals.

Typically, one of three methods is utilised: Fitness ranking, tournament selection, or fitness proportionate selection. In fitness ranking methods, the likelihood of selection is proportional to the position in the ranked list of solutions. In tournament-based methods, selection is based on the comparison of pairs of (randomly or otherwise chosen) solutions. Fitness proportion methods, discussed below, use the solutions' actual fitness values to govern the probability of selection. The three techniques can range in the amount of randomness afforded, a balance has to be found which suits the application, maximising average fitness through the deterministic elements of the technique, while allowing random elements to keep the search open to potentially valuable solutions.

Roulette wheel selection (the most common fitness proportion method), is very much probabilistic in operation, resulting in higher variance, which can help to alleviate premature convergence to sub-optimum solutions [Wood2, Forrest]. Although robust, the high stochastic error associated with roulette wheel selection is often prohibitive. The basis of the technique is as follows: An analogous roulette wheel (though with unequal trap sizes) is formed from the fitness values of the whole population (i.e. a pie chart of all the fitness values). Each solution therefore has a numerical range associated with it, so that when a random number is generated, between 0 and the total fitness value, the solution whose range the value lies within is selected.

Thus each solution (e) has a chance of selection equal to that of its fitness (f) as a function of the total fitness:

$$P(e_i) = \frac{f_i}{\Sigma f}$$

More deterministic techniques are often used, especially for complex problems. Stochastic remainder selection is a common technique, which uses only a small probabilistic element. With this method, the number of times a solution becomes a parent is allocated according to the whole part of its normalised fitness value (the solution's fitness value is divided by the population mean fitness). A small random element is introduced by using the fractional parts of solutions' normalised fitness as a probabilistic weighting to select the remaining parents.

Fertility

The fertility of a parent solution is subtly different from its fitness. Where fitness defines the likelihood of reproduction, fertility defines how many children that parent can have. Fertility, although inherently incorporated into fitness proportionate selection, can therefore be treated as a separate method of exerting selection pressure. EAs will usually have a parameter that limits the number of children one solution can parent, thus limiting its dominance on the next generation and alleviating premature convergence [Yu].

Replacement

There is also the option as to whether to replace the entire population with the next generation of members, or, as is sometimes beneficial, to transfer some of the fitter members from the previous generation directly into the current population (elite replacement), simulating the overlapping that usually occurs in nature. A departure from the sequential approach, used in non-continuous EP, is to create child solutions until the population size has doubled, and then remove the most unfit half. More involved variations on this theme, such as life-span *(death)* operators, used to prevent immortality if elite replacement is used; and *kill* operators, mostly used to enforce constraints, can also be employed [Bentley5].

3.5.2 Recombination

The reproduction stage is responsible for the creation of new, possibly better solutions and is the foundation of all evolutionary algorithms. It is the ability of new solutions to inherit properties from multiple parent solutions, with some small but significant changes that enables the evolutionary properties.

At this stage in the GA, genetic information from elected parents is combined in some way, producing a new generation of members. The recombination of chromosomes is normally carried out using a variety of crossover operators, illustrated in the following diagrams. There may be one (Figure 3.5.1), or several (Figure 3.5.2) crossover points, which can be predetermined, or randomly selected for individual matings, or for the whole population. It is often desirable, to restrict crossover to whole segments (Figure 3.5.3), or even whole chromosomes (Figure 3.5.4), in order that differences between parents and offspring remain small. Intra-segment crossover (Figure 3.5.5) produces greater variation, enabling the creation of many more new decoded parameters through recombination.

There is also the option of whether to use complementary or non-complimentary crossover. With complementary crossover, pairs of corresponding chromosomes are produced using the same crossover point, where the 'right half' of one parent's genes are matched with the 'left half' of the other parent's, and vice versa, thus producing two offspring from each recombination (Figures 3.5.1 – 3.5.5). An advantage of this technique is that all genetic data from both parents is passed on. Alternatively, only 'half' of each chromosome(s) is (are) used to produce a single offspring, and the remaining portion(s) discarded. The process is then repeated with another crossover point, with either the same (Figure 3.5.6), or a newly selected pair of parents. Non-complementary crossover is the method used in nature, and tends to produce greater diversity [Wood2], by allowing a greater opportunity for variation than with complementary crossover techniques.

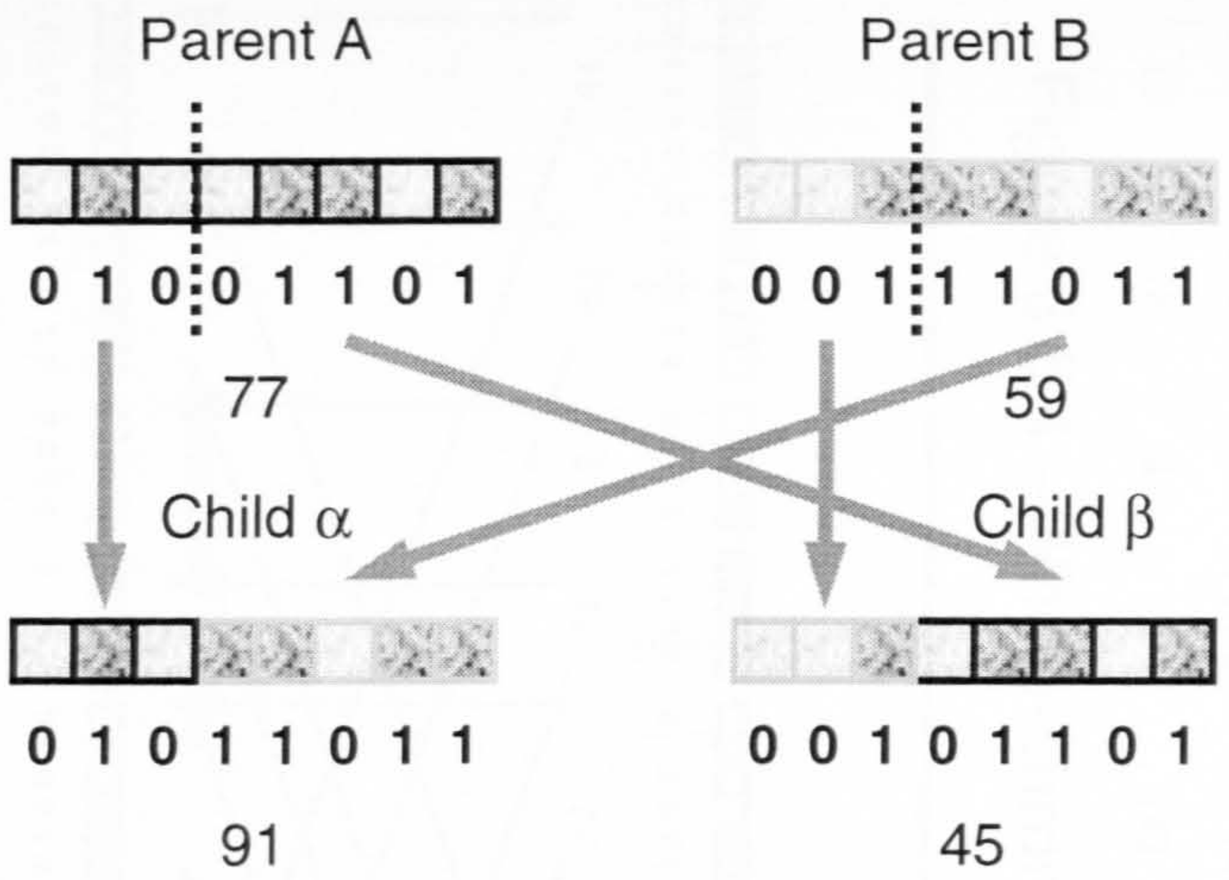


Figure 3.5.1 - Single-point complementary crossover

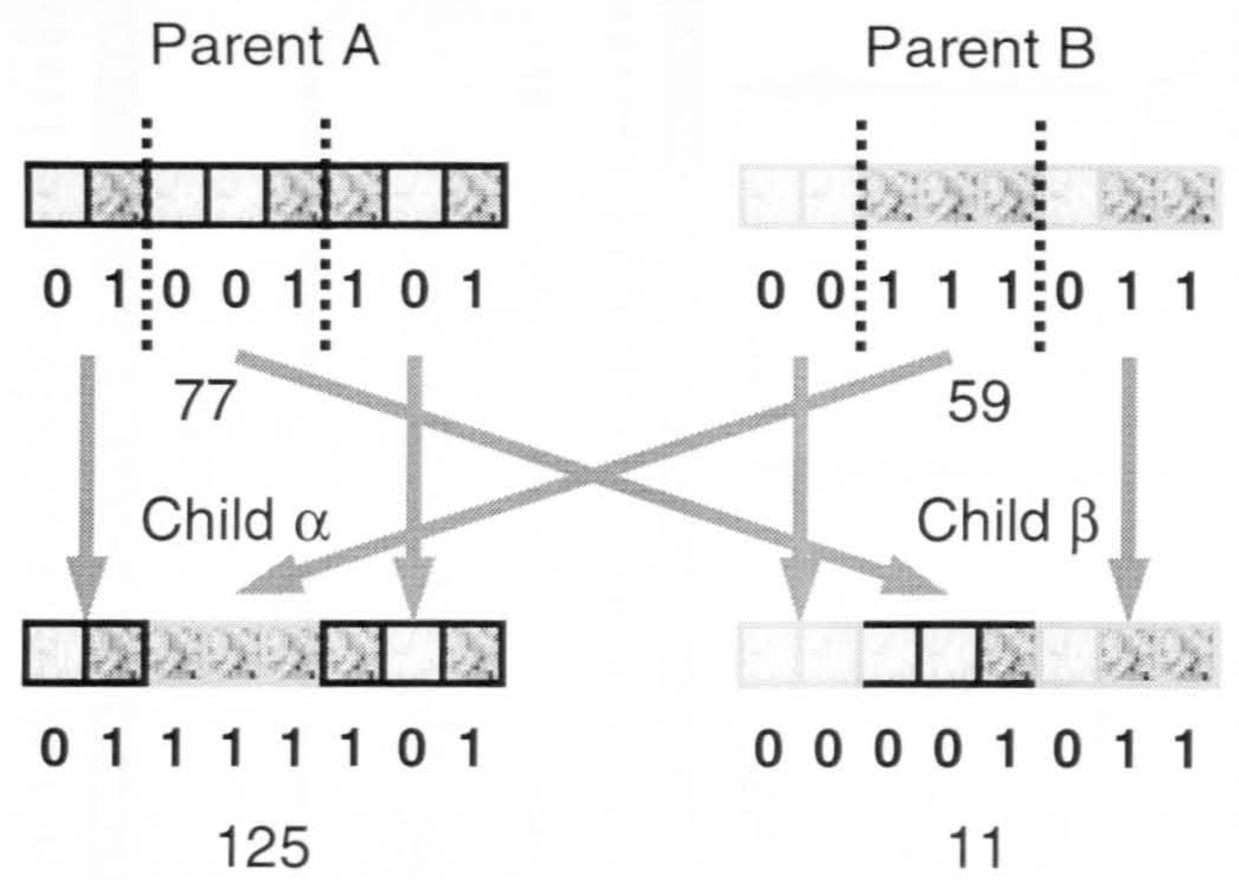


Figure 3.5.2 - Multi-point complementary crossover

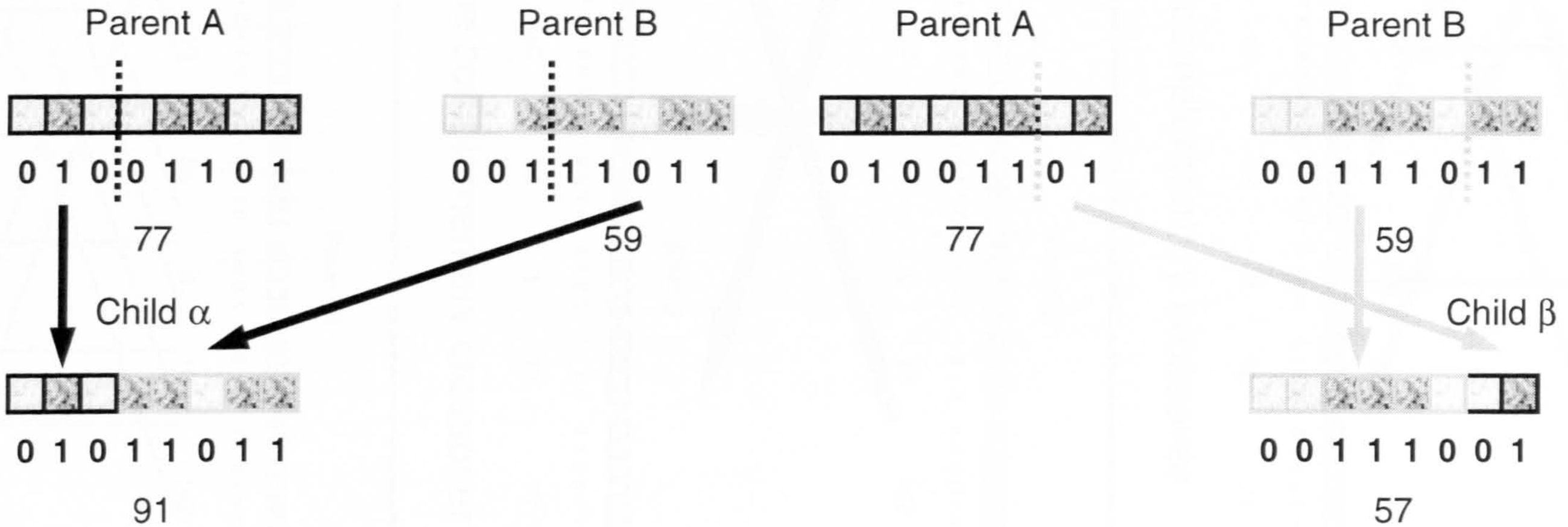


Figure 3.5.6 - Single-point non-complementary crossover

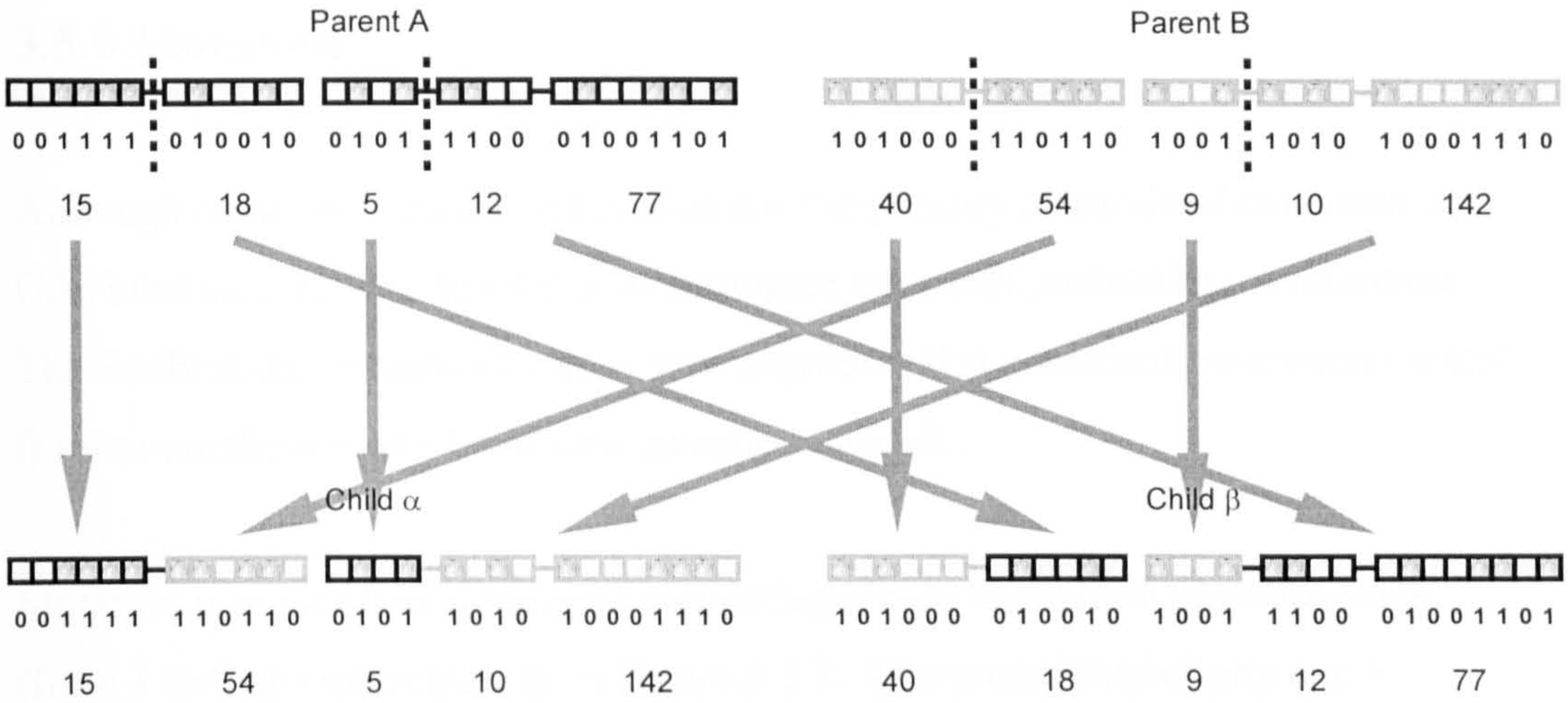


Figure 3.5.3 - Whole-segment complementary crossover

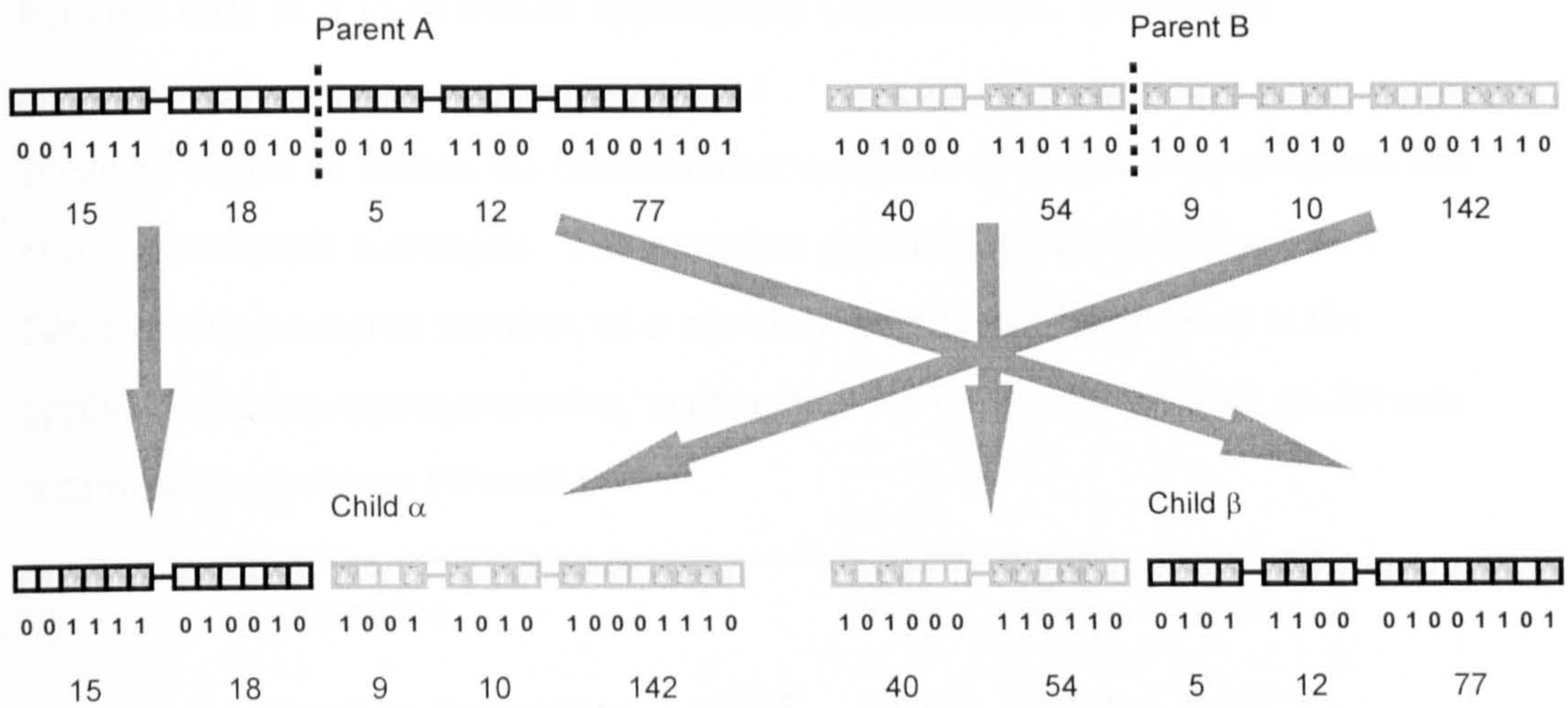


Figure 3.5.4 - Whole-chromosome complementary crossover

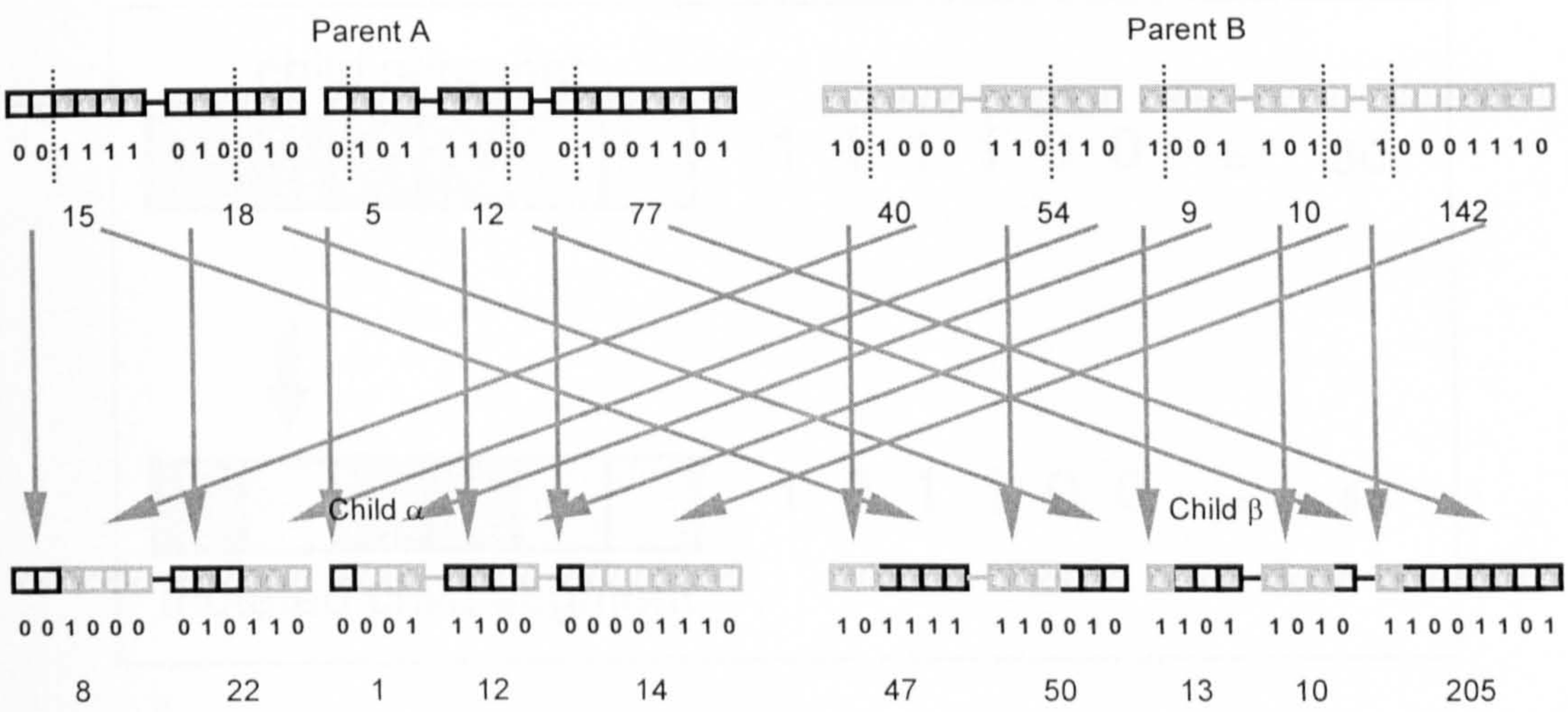


Figure 3.5.5 - Intra-segment complementary crossover

3.5.3 Mutation

Although crossover and recombination are the primary methods of evolution in GAs (and nature), it is beneficial to introduce the small probability of mutation. This enables the creation of totally new segments that could not have been created from re-combinations of available genetic material.

Mutation is most often achieved by inverting single bits within chromosomes (from 1 to 0 or vice versa), as in Figure 3.5.7. The probability of any one bit being affected are typically quite small, ranging from 0.001 to 0.075 in optimisation problems reviewed [Bentley5, Wood2], but generally rising to between 0.01 to 0.11 in artistic applications [Rowbottom, Witbrock].

It can be useful to reduce the likelihood of mutation as generations progress and converge towards a solution. The mutation probability can be defined as a function of generation number, or a mutation profile can be tailored to the application after experimentation, such as the one below, taken from an inverse optimisation problem [Wood2].

Generation		>25	>50	>100
Mutation Probability	0.075	0.050	0.025	0.010

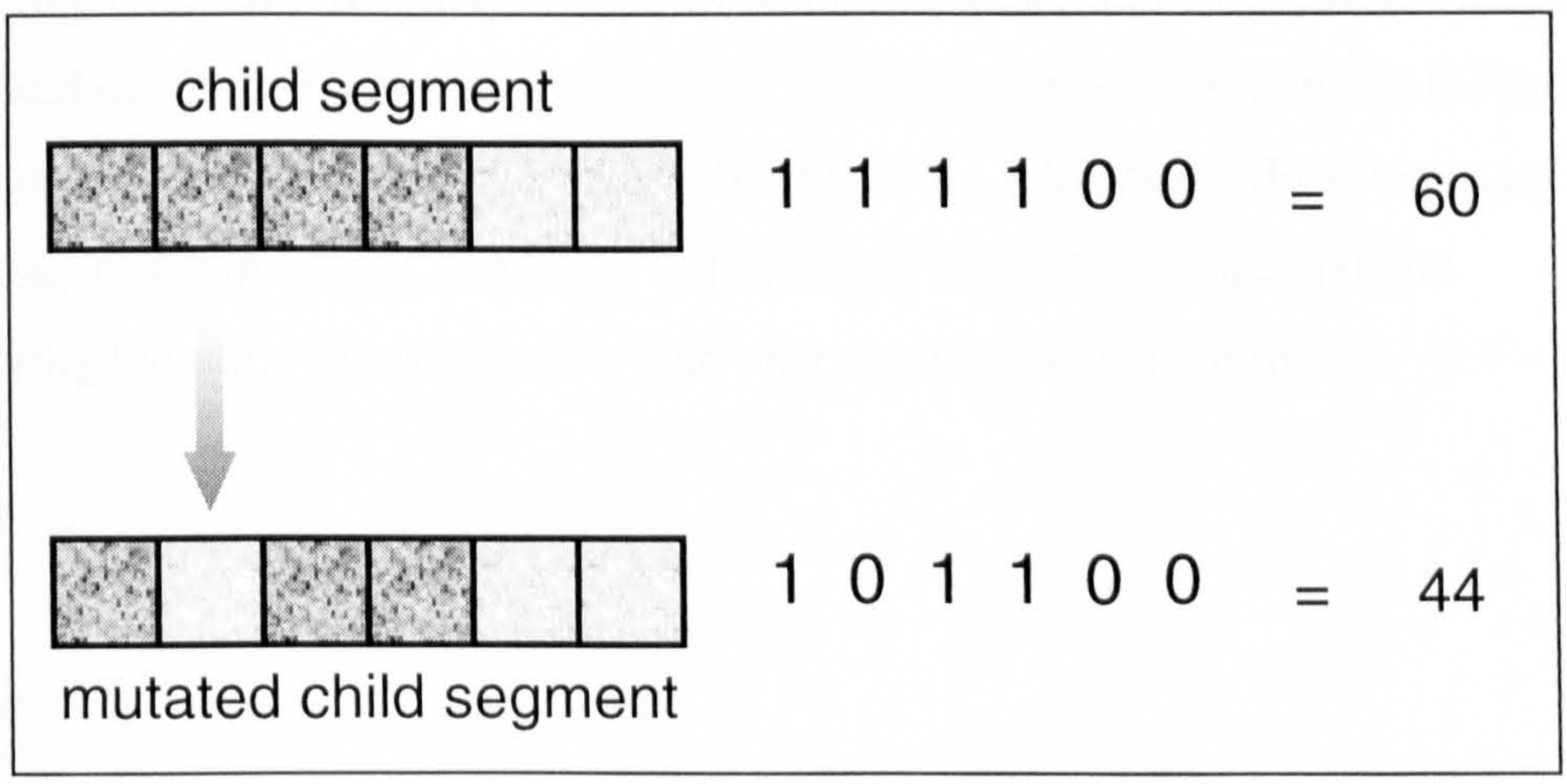


Figure 3.5.7 - Mutation

3.6 Specialised GAs

The most common problem to arise when applying GAs is that of premature convergence, where the population converges on a local, sub-optimal solution, too soon. This is not such a problem for interactive systems, where the user can detect the problem and take avoiding action[1], but may be a sign that the system is running with less than ideal parameters. A second widely acknowledged limitation of the simple GA (such as has been described here) is difficulty in dealing with optimisation of multiple criteria, and is briefly discussed below. The steady improvement of Genetic Algorithms has been carried out since their first introduction, to address these and other problems. Consequently, distinct types of advanced GAs have developed, for example: Distributed GAs and GAs with niching and speciation have been developed to increase efficiency [Eby]; and GAs with diploidy and dominance that improve variation and diversity, especially when tackling functions that vary with time [Smith]. GAs are often combined with other techniques (Hybrid GAs), or heavily modified to suit specific problems, such as grouping GAs [Falkenauer].

Multiobjective GAs

Simple GAs can (and are usually left to) work with multiobjective problems by combining weighted objective values into one fitness value [Horn]. Frequently, however, this does not produce the best results, and it is necessary to employ a Multiobjective GA (MOGA). MOGAs work with more than one fitness value at a time and use Pareto optimality to define the better solutions. A solution is Pareto optimal if it is not dominated by any other solutions [Goldberg] (In some cases this may result in a large number of solutions all being Pareto optimal and receiving the same fitness score, which may causes problems in itself).

---

[1] This circumstance should not be relied upon to relieve the problem while developing an efficient system – ideally the problem should be addressed through the usual means, i.e. careful selection of genetic operators and the 'tuning' of the operating parameters

3.7 Conclusions

Clearly a CAD tool where any number of random forms are presented to the designer would be of limited use. Firstly these forms should be useful, and secondly, there needs to be an intuitive method for a designer to guide the process of form generation according to the task in hand. This method should adapt, according to the changing thought processes that occur during conceptual design, which are also triggered by the interaction with form on the screen. These complex, changing criteria are best addressed through evolutionary computational techniques, with Genetic Algorithms standing out as the most successful and appropriate tool in this field.

By imparting some of the qualities of natural evolution (a well known and reasonably well-understood process) GAs can naturally be adapted for use in evolutionary design systems. This chapter has introduced the fundamental aspects of GAs, with some common (and some novel) techniques for their implementation. For applications such as the research discussed here, the following points should be considered.

Representation

The 3D representation (phenotype) should be readily described using a genetic data structure, and data efficient, but not so much so that it is incapable of effectively defining a suitably wide variety of useful solutions. The genetic representation should be so defined, that small changes in the genotype (through genetic operators) result in small differences in the phenotype, allowing inheritance with a small degree of variation. Given these two conditions, initialisation should produce a diverse population by assigning random values to each chromosome.

Selection

It is preferable to use some selection operator, rather than the user having the inconvenience of explicitly selecting every pairing of parents (i.e. 14 in the case of the EFD research). The main requirement of a selection technique then, is that it reflects a user's intentions. This suggests a fitness proportionate, deterministic biased scheme, such as stochastic remainder selection, though programming practicalities must be considered. The selection technique employed will depend on the demands of the user at different stages in the process. During the early stages, where a relatively large number of individuals are usually involved, a deterministic approach, such as stochastic remainder selection will work well. However, if the user wants to select just 1 or 2 parents, there are often conflicts within the selection routine[1], and the GA may experience difficulties. It is suggested that the probability-based and consequently more robust roulette wheel selection technique be used in these circumstances. If fitness-proportional parent selection is not used, controlling fertility is an effective method of restricting the dominance of a single individual over following generations

Replacement and Recombination

Whole population replacement would seem the most suitable technique for interactive systems (which inherently use small populations), offering an entirely new set of solutions each cycle, although some of the advantages of elite replacement type operators would be beneficial. Several alternative crossover techniques exist, offering varying degrees of continuity. Although mutation is generally the preferred route to producing entirely new values, intra-segment crossover has the capacity to produce a large number of *new* decoded values in offspring (rather than just 'shuffling' existing values). It is suggested that intra-segment crossover is particularly well suited to interactive evolutionary systems, producing high variation and rapid evolution.

---

[1] A problem is often caused by the conflict between the 'maximum population fraction' parameter and the 'remove identical parents' function – both useful aspects that should not be bypassed. Theoretically, these problems could be detected as they happen, and only at this stage are these functions temporarily ignored

CHAPTER FOUR

DEVELOPMENT OF THE EFD SYSTEM

System Background

The Unigraphics (UG) CAD suite includes an Advanced Programming Interface (API) called UG/Open, providing access to routines within the UG graphics terminal, file manager, and database. The prototype EFD software, written in the C programming language, is an internal UG/Open API program and is run as a user function, from inside a UG session[1]. A screen shot is shown in Figure 4.0.1.

The EFD software contains a Genetic Algorithm at its core, and as such, produces evolving populations of solutions. In this case, populations containing 10, 12, or 14 product representations, or solid geometric models (herein referred to as *objects*) are produced, created from a small number of interacting geometric primitives. During an interactive EFD session, the user is required to rate, from 0 to 10, each object in the population, providing (or contributing to) each object's fitness value. Another population is then generated; the fitness scores of the last generation influencing which objects were selected as parents to create this new population. A user-interface is provided for this scoring system and the other user-selectable options, described below – the first four of these options correspond to sections within this chapter:

Create mode: *Normal* or *Cohesive*, the create mode determines at which stage the geometric primitives interact, during object creation.

GA controls: The maximum number of generations, maximum population fraction, parent selection technique and mutation profile can be changed from default values.

Optimisation: If cohesive objects are used, the options available for internal geometric optimisation (volume, surface area, bounding box size and dimensions) are listed for selection, in addition to user scoring.

---

[1] 'New Session' or 'Teamforming' is selected from the EFD menu

Blend mode: 4 options for edge blending exist: Random, simple, whole object or none. The method selected greatly influences the style of objects.

Session ID: This number allows different starting populations by incrementing the random number generator 1000 × the 'ID' value entered.

Run mode: The software can be run in 'User Mode', where some prompts are given, 'Optimisation Mode', where no prompts are given and part-names are automatically generated (allowing rapid progress for internal optimising), or 1 of 3 testing and development modes.

Filename: Each population is treated as a new part and therefore requires a filename (a default filename with population number is generated).

Prompt: Before the population is rated by the user, a 'save/continue/quit' prompt is provided, at which point the CAD menu system can be accessed without disrupting the EFD session. This enables (e.g.) individual objects to be rotated and studied in detail by blanking all other objects.

Populations of objects are displayed in figures throughout the thesis, so a brief description of some adopted standards may be useful. Each object is created in a different colour, this is for clarity only – the colour is based on its position on the screen, and is not an inherited property of the object:

1 Blue	**2** Green	**3** Cyan	**4** Red	**5** Magenta
6 Yellow	**7** White	**8** Olive	**9** Pink	**10** Grey[1]
11 Orange	**12** Purple	**13** Maroon	**14** Aquamarine	

The objects are created from left to right, in rows of 4 or 5, with second and third rows beneath, and are usually displayed in isometric or trigonometric views. The numbering and colour schemes form the basis of the object naming convention:

g2p4-red	=	generation 2	phenotype 4
g6t8-olive	=	generation 6	team 8

[1] Object 10 is coloured aquamarine, not grey, in populations of less than 14

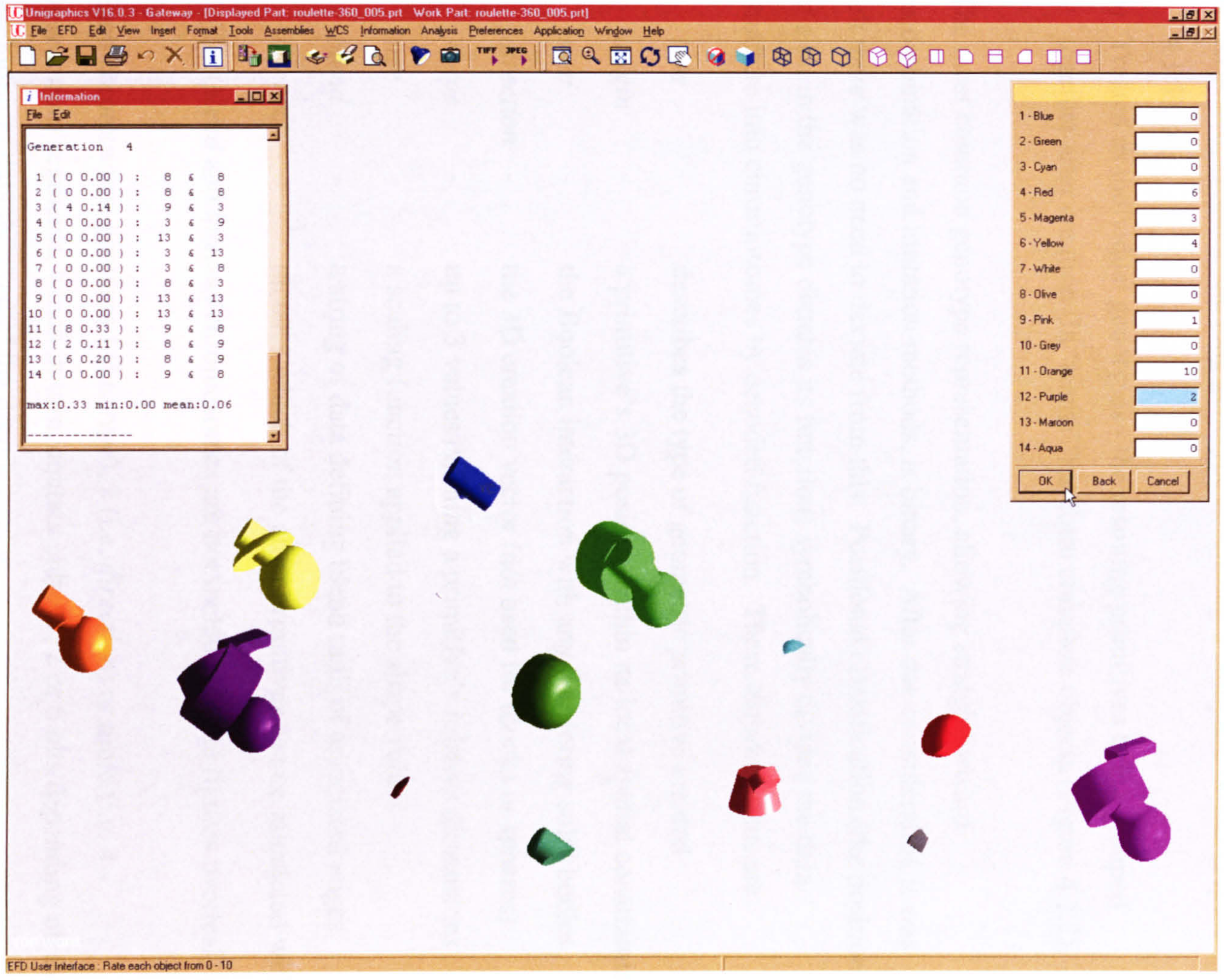


Figure 4.0.1 - EFD System screenshot showing information window and scoring interface

4.1 Genotype

A genetic data structure capable of efficiently describing individual geometric primitives has been established. Several of these structures are either:

a) Repeated, creating one long genotype describing a complete object (Figure 4.1.1), or,
b) Treated as individual genotypes - the resulting primitives being grouped together after creation (in 'teams'[1]) to form complete objects (Figure 4.1.2).

The most common genotype representation, allowing straightforward recombination and mutation methods, is binary. After due consideration it was felt there was no need to deviate from this. Positional classification (the position of a bit in the genotype dictates its function) symbolically divides the data structure into chromosomes by decoded function. These chromosomes are:

- *type* - describes the type of geometric primitive created
- *origin* - a primitive's 3D position within its local spatial constraints
- *sign* - the Boolean interaction with any interfering solid bodies
- *direction* - the 3D creation vector (not used for blocks or spheres)
- *shape* - up to 3 values defining a primitive's relative dimensions
- *size* - a scaling function applied to the shape values
- *blend* - a string of data defining blend radii of associated edges
- *interact* - dictates which of the other primitives can be interacted with

(The *blend* and *interact* chromosomes are not included in the figures overleaf)

Each chromosome contains 1 (i.e. *type*), 3 (i.e. *direction*) or more (i.e. 4 - *interaction*) segments, which in turn, contain either 1, 2 or 6 bits depending on the resolution of data required of the decoded value.

[1]The 'Teamforming' technique is covered in more detail in a later section

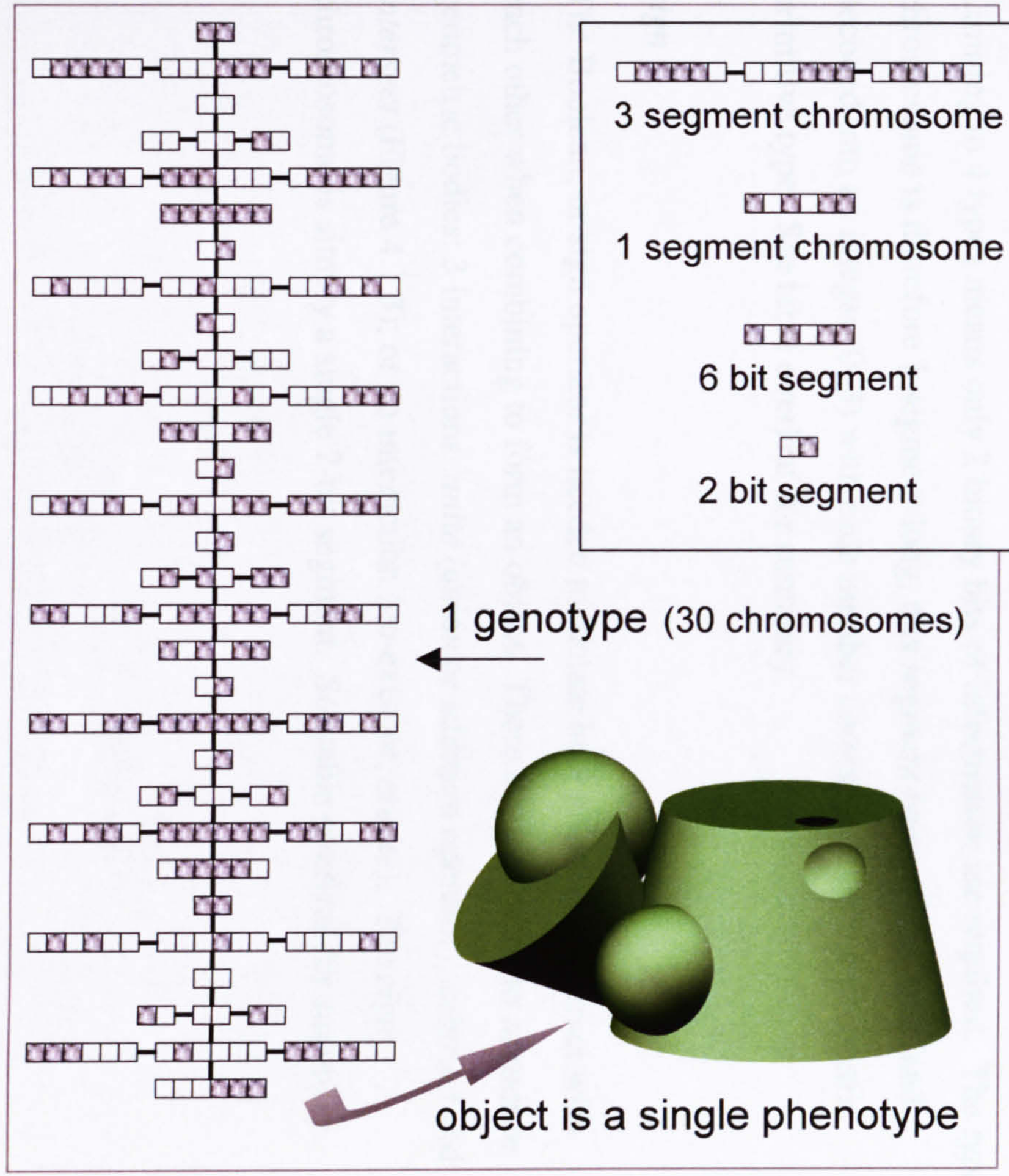


Figure 4.1.1 - Object formed from 1 genotype

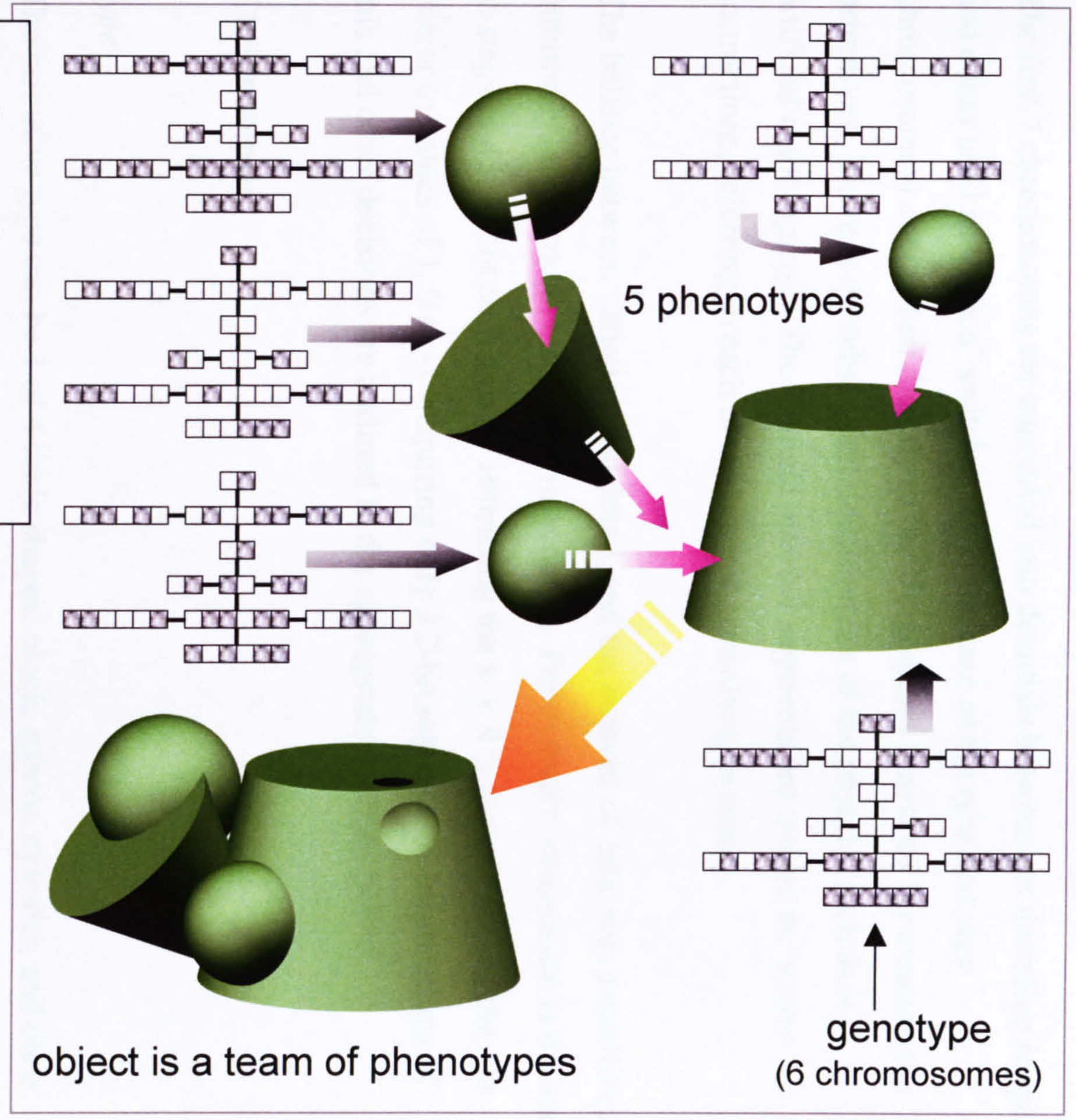


Figure 4.1.2 - Object formed from a team of 5 phenotypes

The first 7 chromosomes are converted into decimals between set decoding ranges and either used as 'if/then' switches (as in the case of the *type* and *sign* chromosomes) or supplied directly to the Unigraphics functions for creating the primitives. There is no subsequent development of the objects (such as in artificial embryogeny). The 4 single *interact* segments are treated as 'yes/no' instructions, referring to each of the other 4 primitives, in order.

The balance between variation and detail, and the amount of data, was established intuitively, and through brief experimentation. For example, orientation is limited to steps of 45° about each axis by restricting the x, y & z components of the 3D vector to values of 1, 0 or -1 (requiring only 3 2-bit segments). Implications of this and other decisions are outlined in the appropriate sections below.

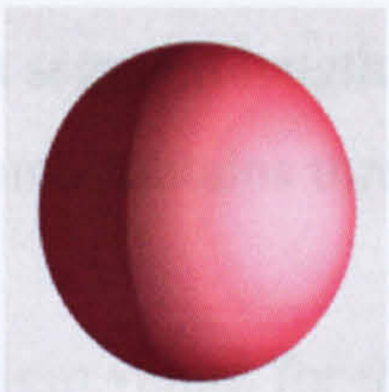
Chromosomes

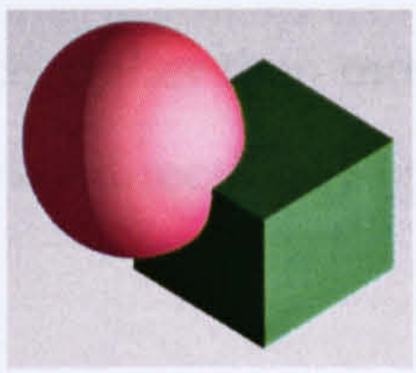
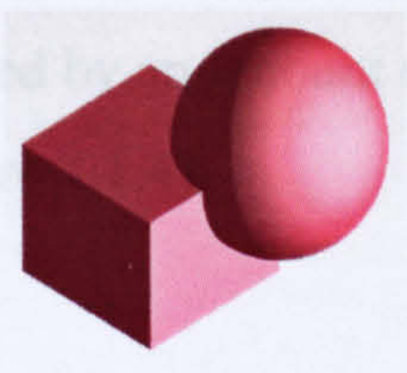
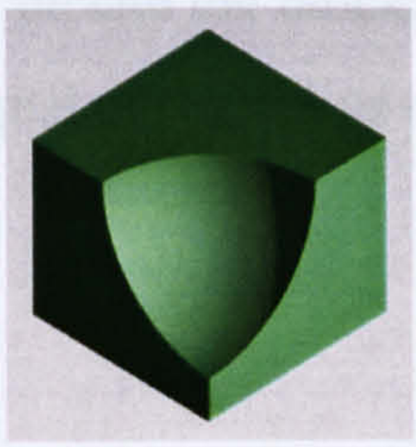
type

The primitive type can be 1 of 4 basic shapes: block, sphere, cylinder, and cone. Limiting to 4 types means only 2 binary bits of information are required. The *type* chromosome is therefore 1 segment long, this segment being 2 bits long, and decoded into an integer (0–3) with each number corresponding to a geometric primitive type. See table overleaf for summary.

sign

The Boolean, or sign operator is needed to dictate how primitives interact with each other when combining to form an object. There are four ways to introduce geometric bodies: 3 interactions: *unite* (union or addition operator), *subtract* and *intersect* (Figure 4.1.3); or no interaction (co-exist or, *create*). The *sign* chromosome is simply a single 2-bit segment. See table overleaf for summary.

Primitive	*block*	*sphere*	*cylinder*	*cone*
				
Segment				
Binary	0 0	0 1	1 0	1 1
Integer	0	1	2	3

Sign	*create*	*unite*	*subtract*	*intersect*
				
Segment				
Binary	0 0	0 1	1 0	1 1
Integer	0	1	2	3

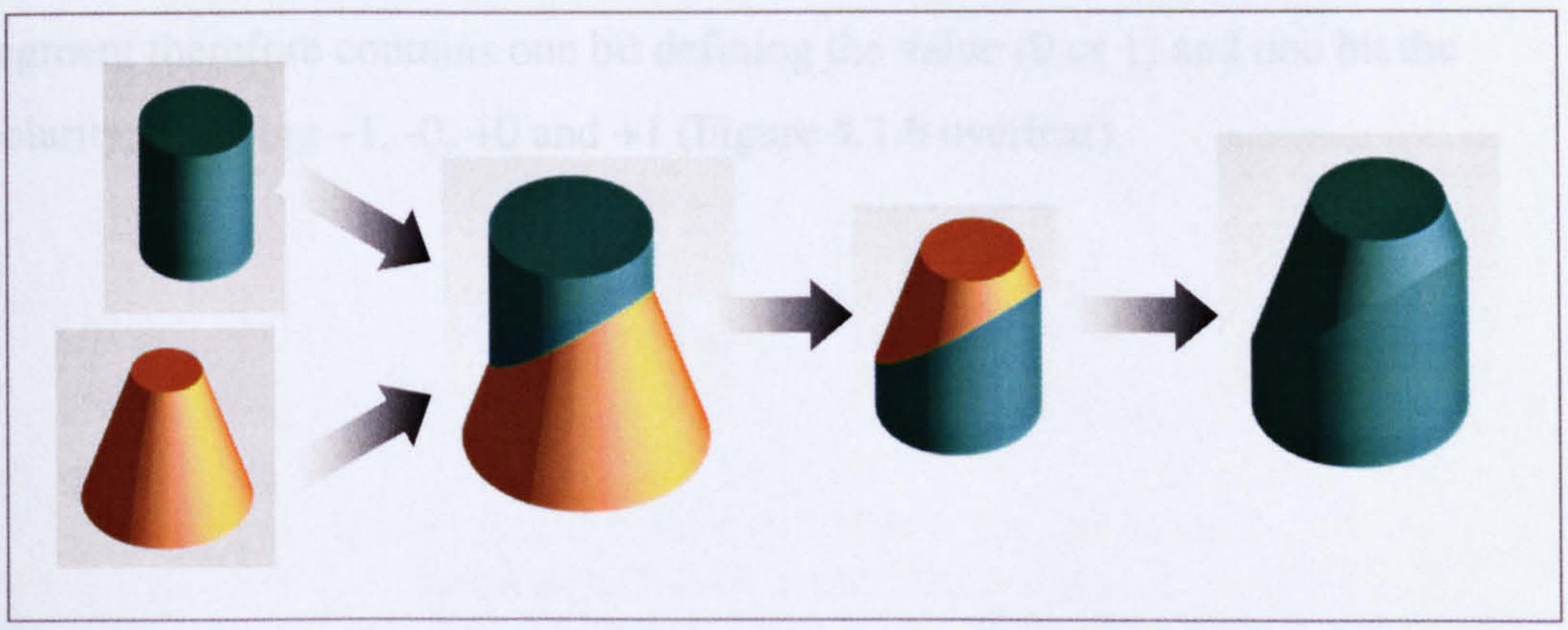


Figure 4.1.3 – The *intersect* Boolean operator

origin

The primitive's origin, relative to its local co-ordinate system, requires three values. Given that an approximate resolution of 1, within a range of between 0-50 and 0-100 was desired, a 6-bit segment length was decided upon, giving a range of 0-63. So the *origin* chromosome contains three 6-bit long segments.

It is necessary to adjust the origin values for the formation of blocks, cylinders and cones. Instinctively, a solid's origin should constitutes its centre (of gravity), as with spheres, and not the CAD system's default requirement for the bottom left front corner for blocks and the centre of the base for cylinders and cones. This is simply for consistency, enhancing the inheritance properties.

For example, if a sphere of a certain size mutated into a block in a later generation, the expectation of continuity would dictate the block should be in the same place, rather than moved by an amount equal to the radius of the sphere in the +x +y +z direction (Figure 4.1.4).

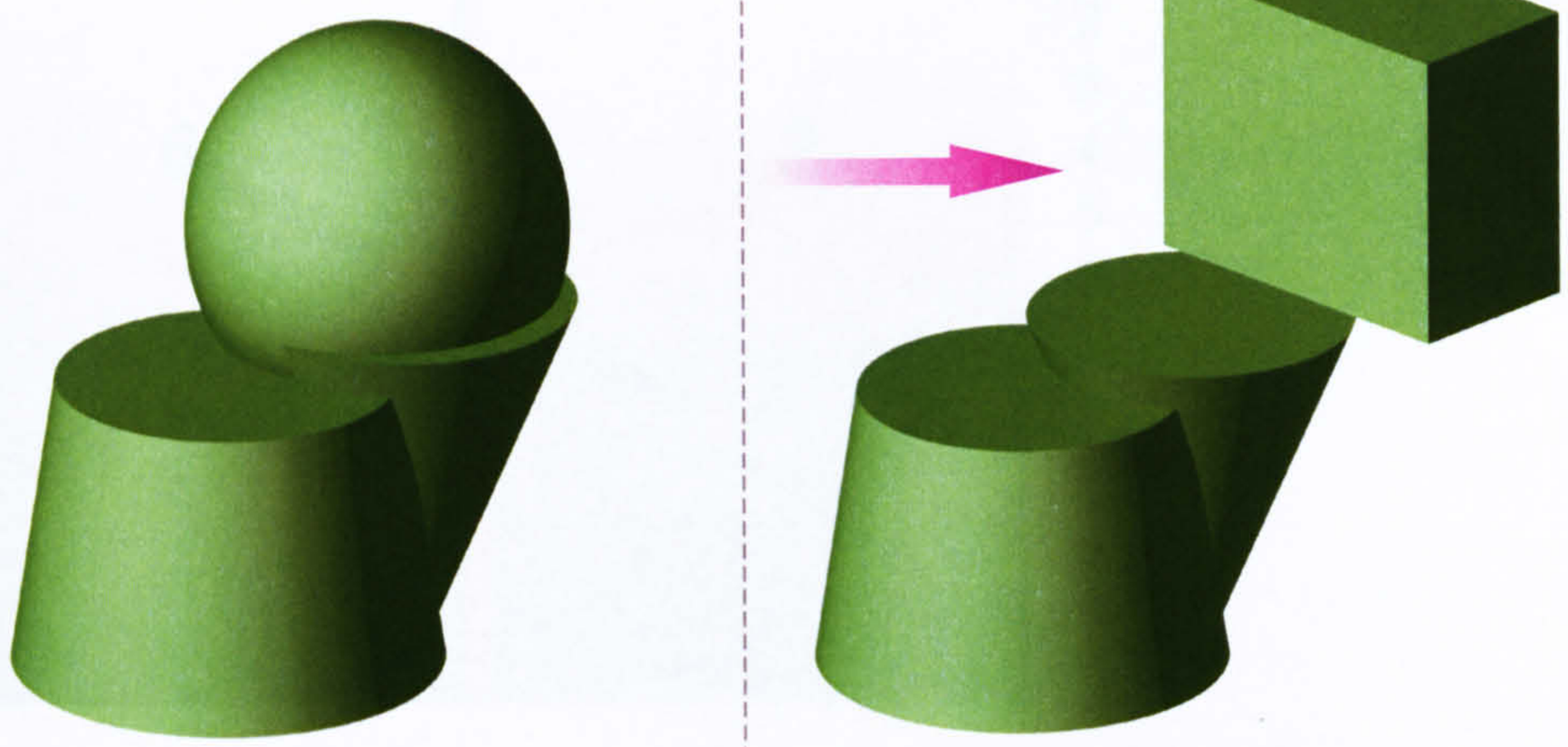
direction

Cylinders and cones are created along 3D vectors[1]. Limiting the angular resolution to steps of 45° permits a simple vector made up of the values of –1, 0, and 1 (Figure 4.1.5) and provides 26 possible permutations (the null vector 0,0,0 is converted to 0,0,1). 2-bit segments produce 4 output values so it has been decided to bias the outcome with a double chance of the zero value - a tendency to align shapes with the individual axis further encouraging visual unity. A *direction* segment therefore contains one bit defining the value (0 or 1) and one bit the polarity, allowing –1, -0, +0 and +1 (Figure 4.1.6 overleaf).

[1] It is noted that ideally, for perfect consistency, the creation of blocks should be able to use the direction function.

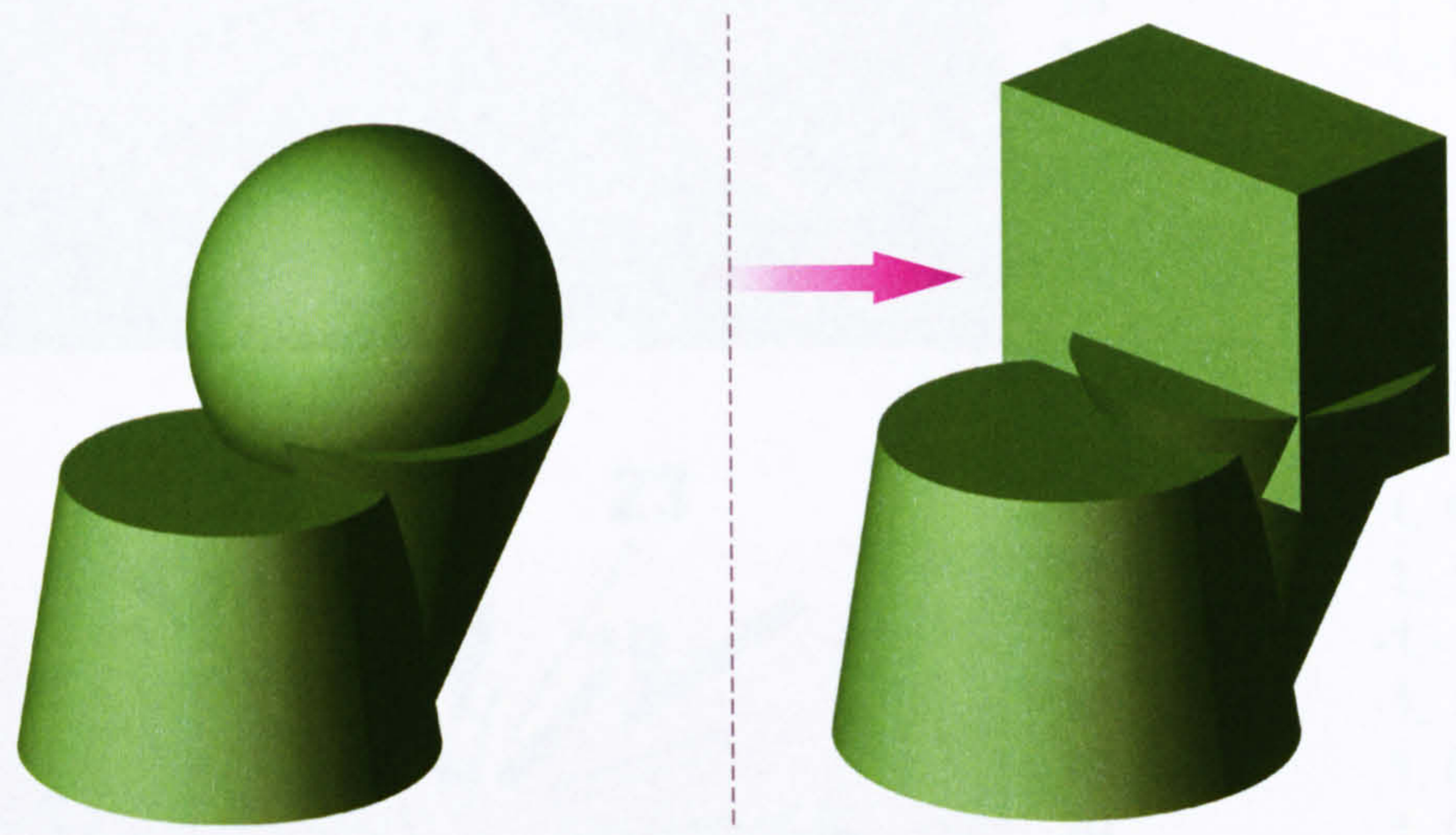
Parent Object

Child Object
sphere mutated to block



No Origin Adjustment

***type* change also causes positional change due to differing origin standards in CAD system**

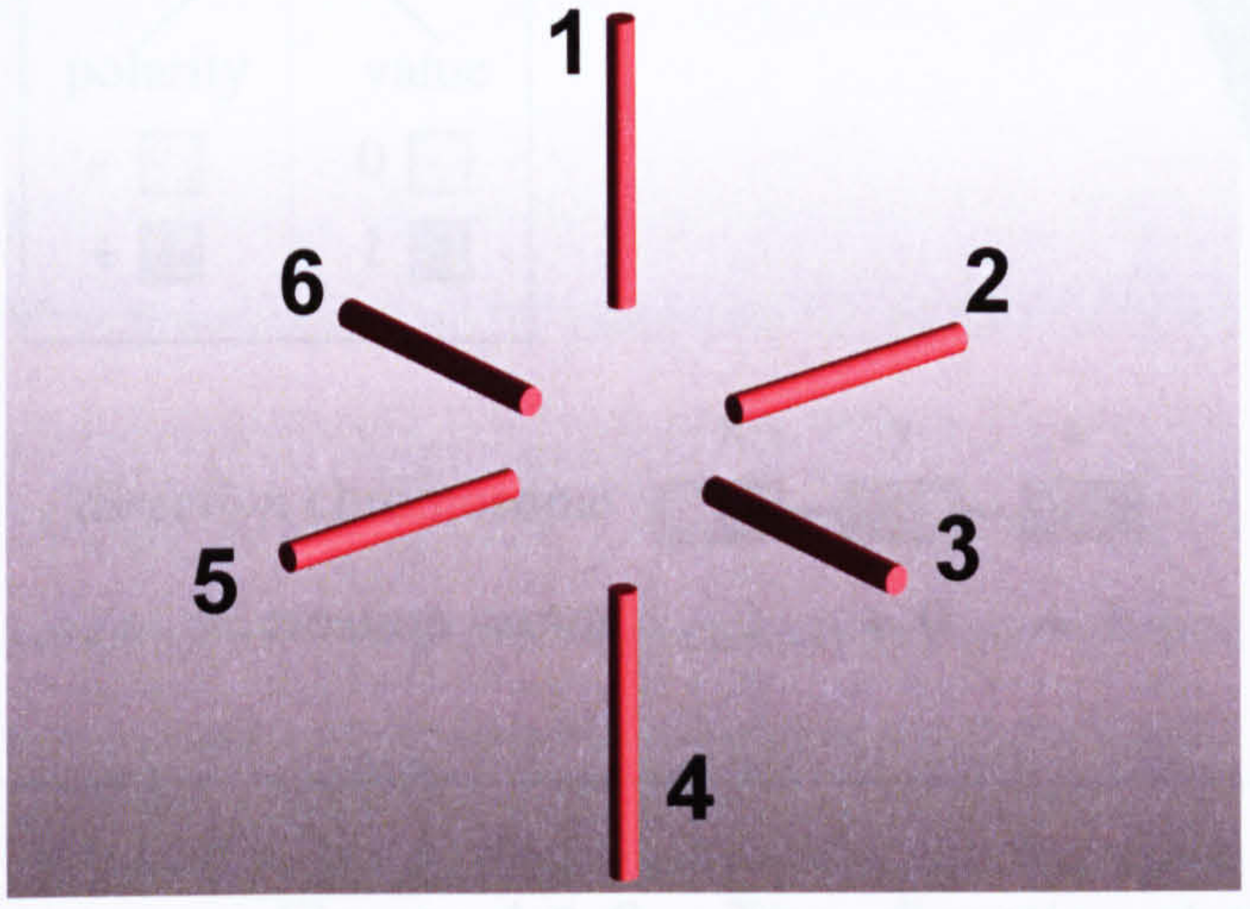


With Origin Adjustment

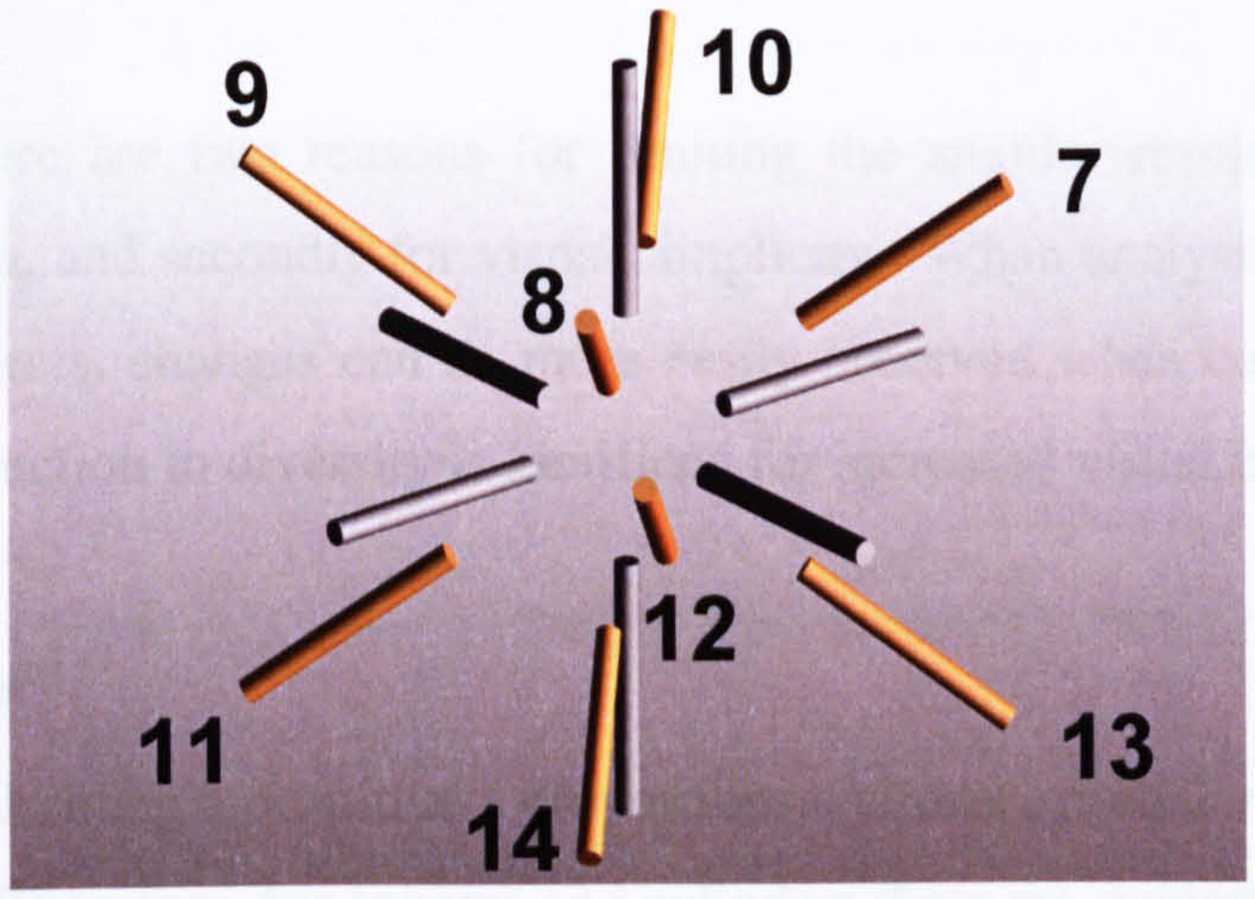
adjusted origins = decoded origins - ½ length values

origin adjustment creates a better result, consistent with user expectations

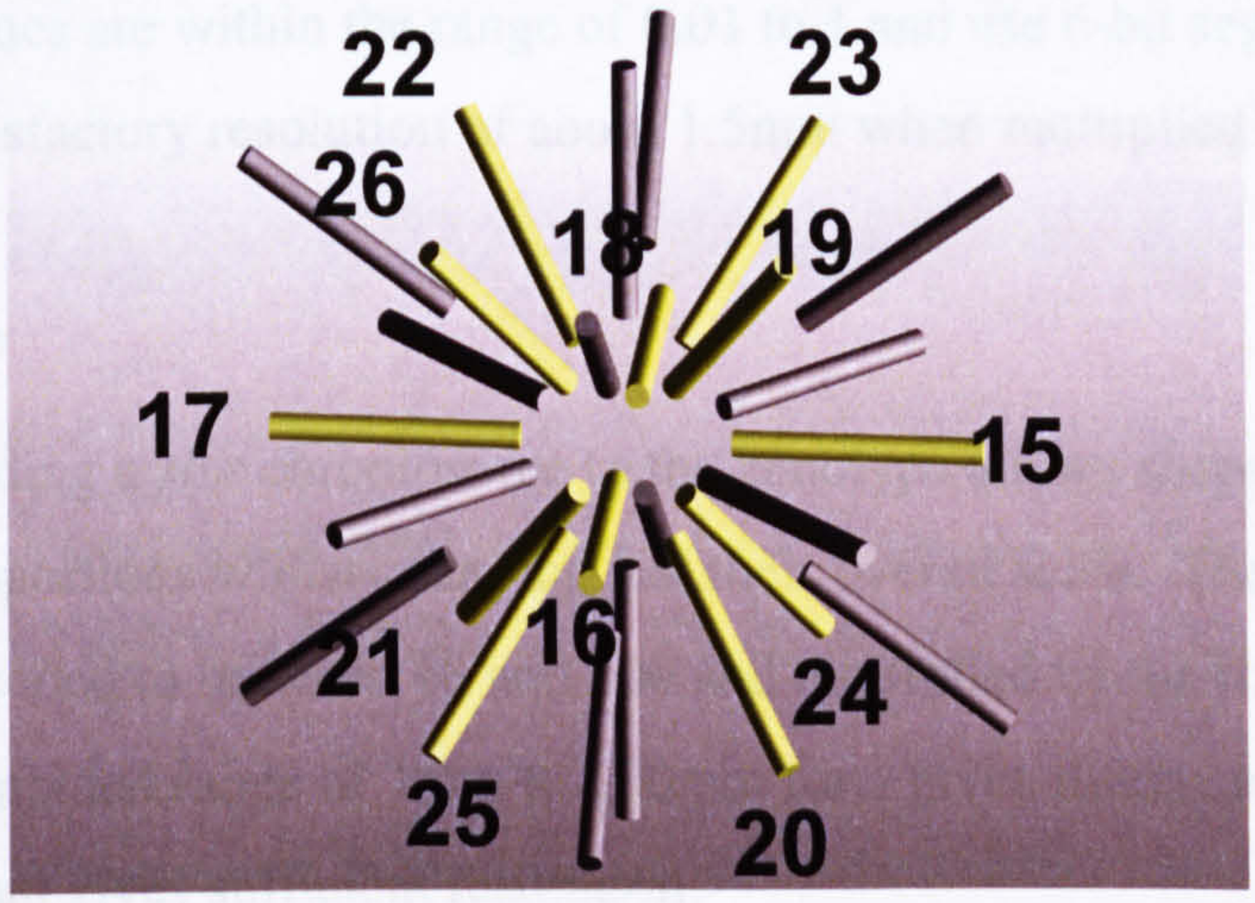
Figure 4.1.4 - Origin adjustment



1	0, 0, 1
2	0, 1, 0
3	1, 0, 0
4	0, 0,-1
5	0,-1, 0
6	-1, 0, 0



7	1, 1, 1
8	1,-1, 1
9	-1,-1, 1
10	1,-1, 1
11	-1,-1,-1
12	-1, 1,-1
13	1, 1,-1
14	1,-1,-1



15	1, 1, 0
16	1,-1, 0
17	-1,-1, 0
18	-1, 1, 0
19	1, 0, 1
20	1, 0,-1
21	-1, 0,-1
22	-1, 0, 1
23	0, 1, 1
24	0, 1,-1
25	0,-1,-1
26	0,-1, 1

Figure 4.1.5 - Angular resolution of creation vector

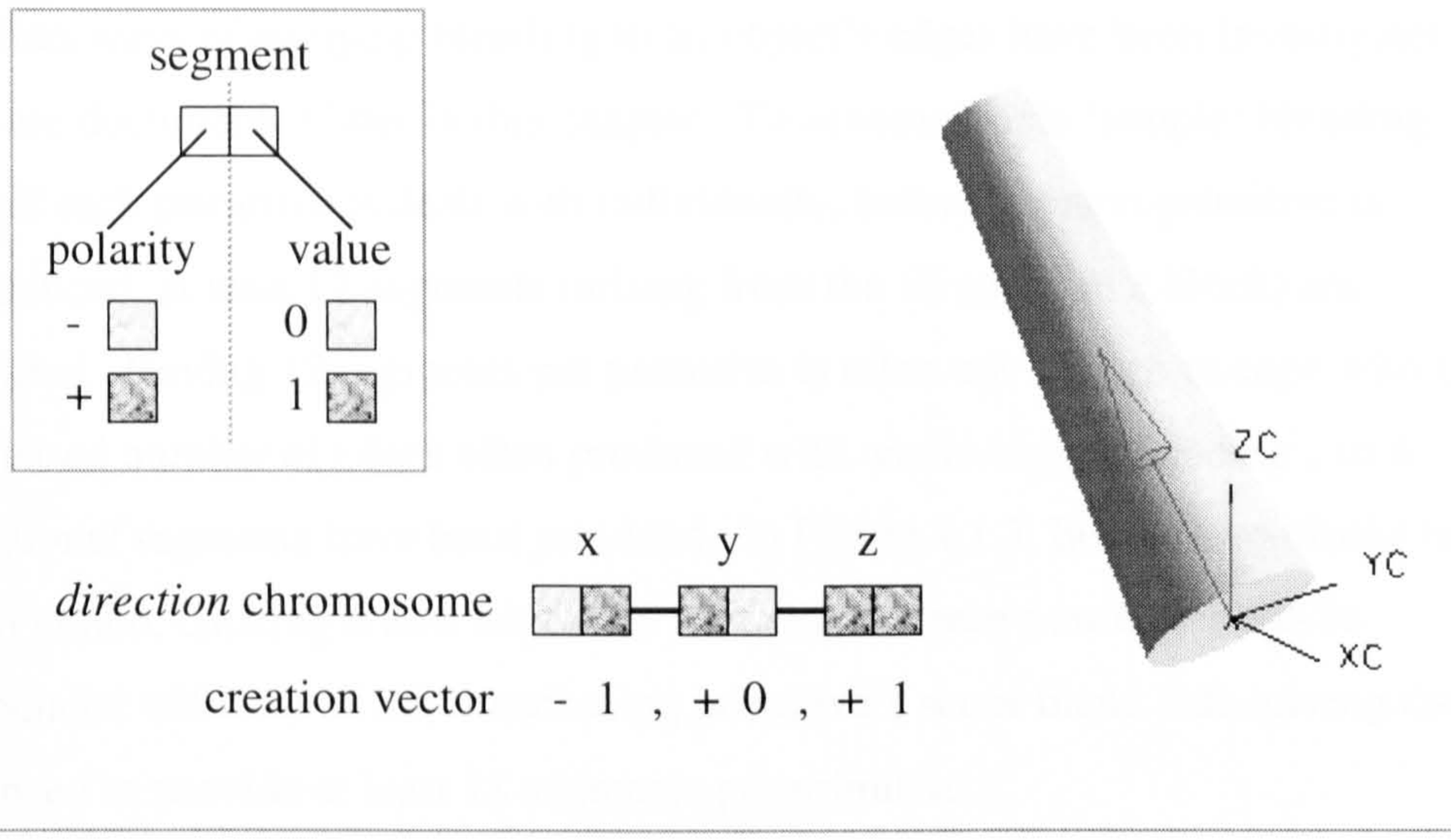


Figure 4.1.6 – The *direction* chromosome

There are two reasons for limiting the angular resolution: Firstly, economy of data, and secondly for visual simplicity. When analysing the inherited features of objects, changes can be more easily observed when comparing generations. The reduction in diversity is sacrificed for increased visual unity.

shape

In defining a primitive's proportions, cylinders need 2 values, cones 3, and blocks 3. Therefore, for spheres and cylinders there are redundant genes. The decoded values are within the range of 0.01 to 1 and use 6-bit segments to give a satisfactory resolution of about 1.5mm when multiplied by the *size* value.

size

Adding a *size* chromosome to the genotype allows shapes to retain their proportions whilst changing just their overall scale. The *size* chromosome is decoded to between 10 and 100 and multiplied by the shape data. This gives a combined range of 1mm to 100mm for a given dimension. A six-bit segment again gives sufficient resolution.

blend

Various ways of applying blending to an object's edges have been investigated, and are documented later in this chapter. To accommodate 'simple' blending, where each primitive is dealt with individually, before the next primitive is introduced, at least 12 segments (arising from the 12 edges in a block) are required. Having 12 segments per primitive is often not enough to cope with the increased number of edges often produced with whole object blending[1], so 6 additional segments have been provided. In Figure 4.1.7, below, two blocks have been united, creating 6 new edges. In some cases, these new edges can be associated with one of the contributing primitive's set of blend data, giving rise to the need to provide at least 18 segments per primitive.

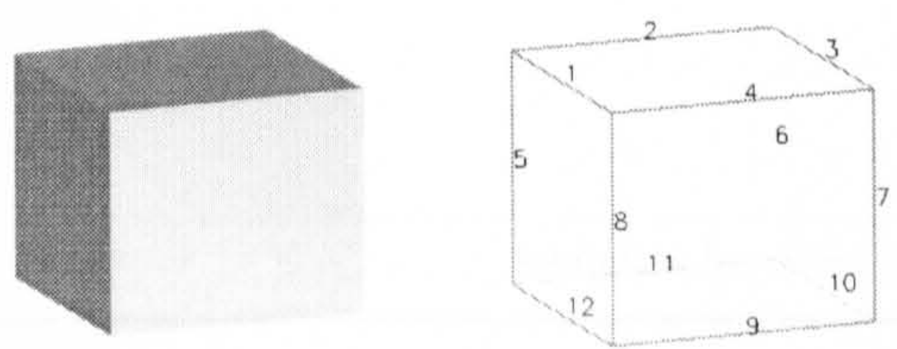
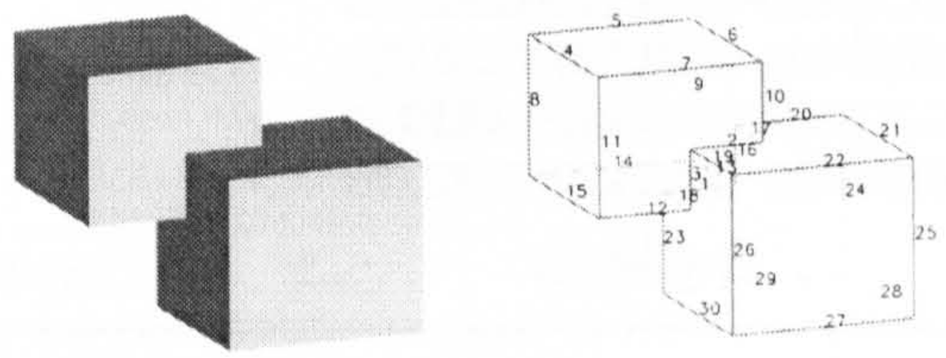
Single block	Two united blocks
	
12 edges	30 edges

Figure 4.1.7 – Provision for additional edges

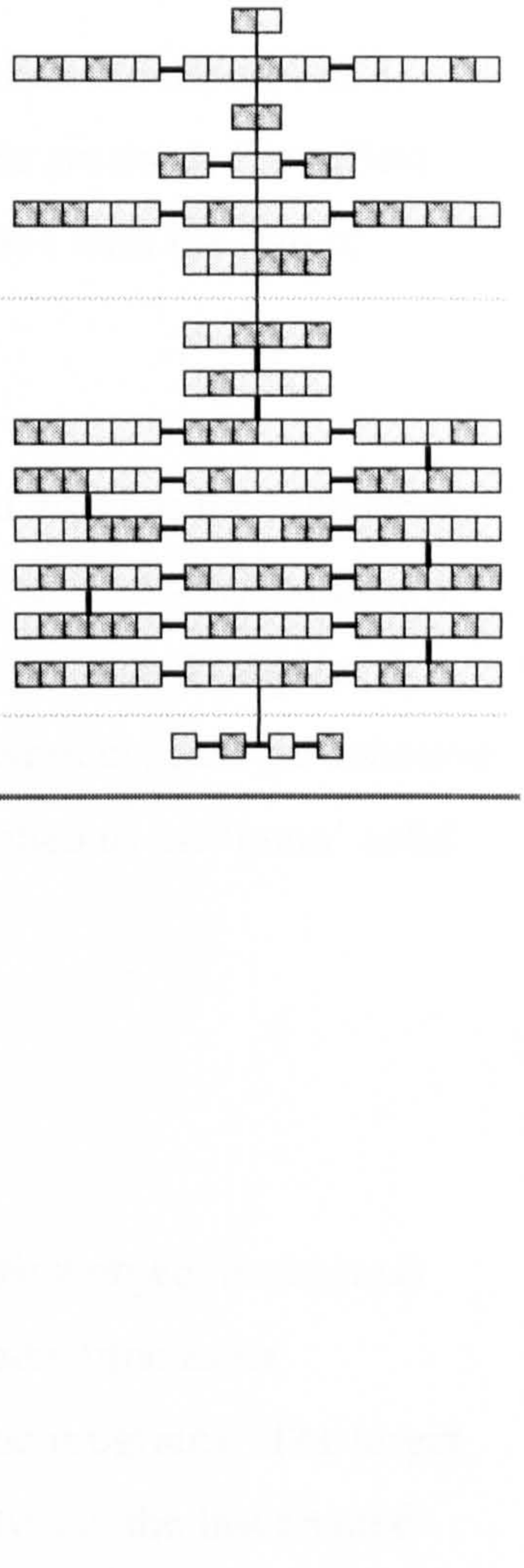
As with the *size* and *shape* co-operation, a multiplier segment is included, dictating the 'blendedness' of an object. Another segment is also provided, dictating the frequency of blends (or how many edges in an object are blended).

interaction

As described in the following section, a chromosome has been included to dictate which adjoining primitives can be interacted with. Each 1-bit segment corresponds to one of the four other primitives contained within an object.

[1] The objects are formed (all primitives are introduced, and all Boolean operations are completed) before blending is applied – further details are given later in the chapter

Summary of the genetic data structure

Chromosome number	Chromosome name	Range	Translated as :	Number of segments	Segment size	Chromosome Diagram
1	*type*	0 –3	block, cylinder, cone, sphere	1	2	
2	*origin*	0 – 63	local x, y, z co-ordinates	3	6	
3	*sign*	0 – 3	create, add, subtract, intersect	1	2	
4	*direction*	0–1, 0–1	3D vector	3	2	
5	*shape*	0.01 – 1	x, y, z lengths	3	6	
6	*size*	1 – 100	multiplier value, × shape	1	6	
7	*blend*	1 – 100	multiplier range, × radii	1	6	
		1 – 5	frequency	1	6	
		1 – 32	radii	18	6	
8	*interact*	0 – 1	switches for other primitives	4	1	

4.2 Object Creation

Objects are constructed from up to five geometric primitives, their ultimate form depending on how and when the Boolean operations are carried out. The number of primitives per object can be changed if desired, though using five primitives per object has been shown to consistently produce good results. The idea of variable numbers of primitives per object is explored in the next section.

The more attractive and usable objects generally occur when the most (4 or 5) primitives contribute and a variety of Boolean interactions are used. To reflect this, several construction techniques have been investigated with the aim of achieving the most from each set of primitives.

Objects can be 'Normal', where all primitives are created whether they interfere with existing bodies or not (some objects are made up of several solid bodies) or 'Cohesive' where potentially-unattached primitives are not created (objects consist of just one solid body). 'Normal' objects will always contain the cohesive part, but also any of the other primitives that are not attached to the 'main' solid body.

4.2.1 Sequential Object Creation

The simplest approach is the method whereby (after the first object is *created*) each primitive's Boolean operation is attempted at the same time as its introduction (calling on just one modelling function in the program). The target body (where necessary) is, by default, the last active body; i.e. the last created primitive or last updated body. Although code efficient, the disadvantage of this method is that primitives are often wasted: Primitives with interactive (*unite*, *subtract* or *intersect*) operators are not created if they do not interfere with the active body. Figure 4.2.1, overleaf, shows the creation of an object using the sequential creation technique, alongside its genotype, and is described on the subsequent page.

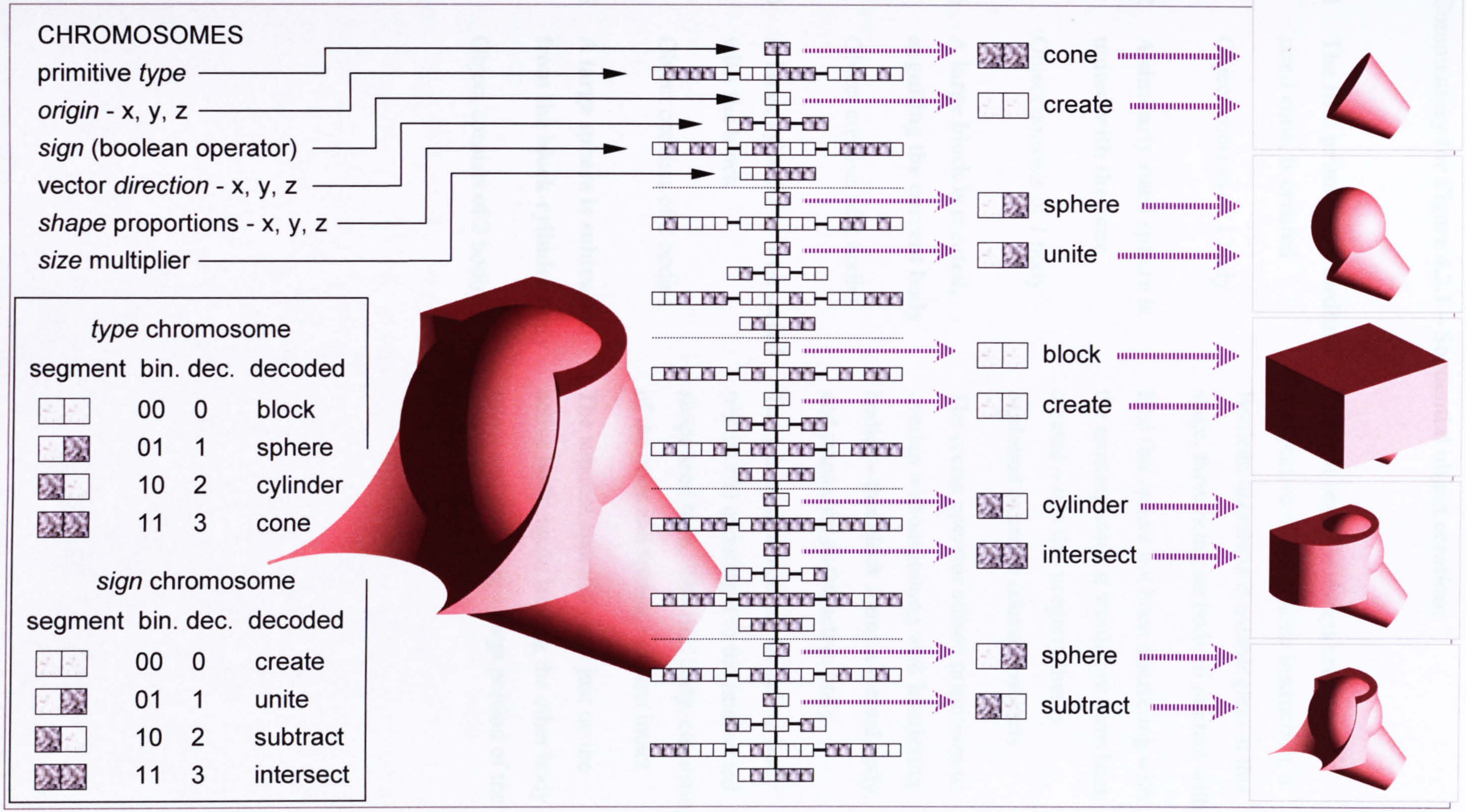


Figure 4.2.1 - Sequential object creation

Commentary for Figure 4.2.1 – Sequential object creation:

1	**The first primitive, a medium sized cone, is created** Object consists of 1 body	The first object is always created, irrespective of the genetic instruction, a Boolean operation is meaningless at this stage, there being no body to interact with
2	**A similarly sized sphere is united with the cone** Object consists of 1 body	Had this sphere not been interfering with the existing cone, it would not have been created – it is this property that is exploited to create cohesive objects
3	**A large block is created, engulfing the current body** Object consists of 2 bodies	The create operator allows primitives to overlap without joining with interfering bodies – this block forms a second body, and takes over as the active body
4	**A large cylinder is intersected with the block** Object consists of 2 bodies	The intersection acts on the active body (the block) updating it to the intersected shape seen here. The first body, consisting of the cone and sphere, remains intact
5	**A large sphere is subtracted from the block-cylinder body** Object consists of 2 bodies	The subtract operation acts just on the active body, again leaving the other body intact and revealing a large portion of the original sphere

4.2.2 Cohesive Objects

Users of the early prototype system expressed the desire for exclusively cohesive objects. A suggestion was that the existence of fragmented objects significantly detracted from the overall 'look' of the system, giving an 'amateurish' appearance by emphasising the objects' simplistic 'collection of primitives' basis. The main counter to this point was that if the user selected only cohesive objects from early populations, then subsequent generations would contain predominantly cohesive objects. Also, that overly constraining early populations stifles diversity and reduces the potential quality of later objects. As it happens, it has been necessary to develop a robust method for creating cohesive objects to enable geometric analysis to perform effectively.

The sequential method described previously is adapted as a simple (but not particularly efficient) method of creating cohesive objects. Any *create* operators are changed to *unite*, meaning that a new primitive that would not interfere with the main body (if created) are not introduced. This is a heavy handed but robust way of ensuring cohesive objects, but results in a lot of unused primitives.

Leaving out a large number of primitives can exclude potentially valuable parts (Figure 4.2.2), and reduce the visual inheritance of related objects. With hindsight, a more effective technique would involve an adaptation of the post-creation Boolean process described presently - isolated primitives being *deleted*, *after* the whole object had been created. The problem of deciding what to do if an object consists of two separate bodies, each made up of two interacting primitives could, no doubt, be easily overcome.

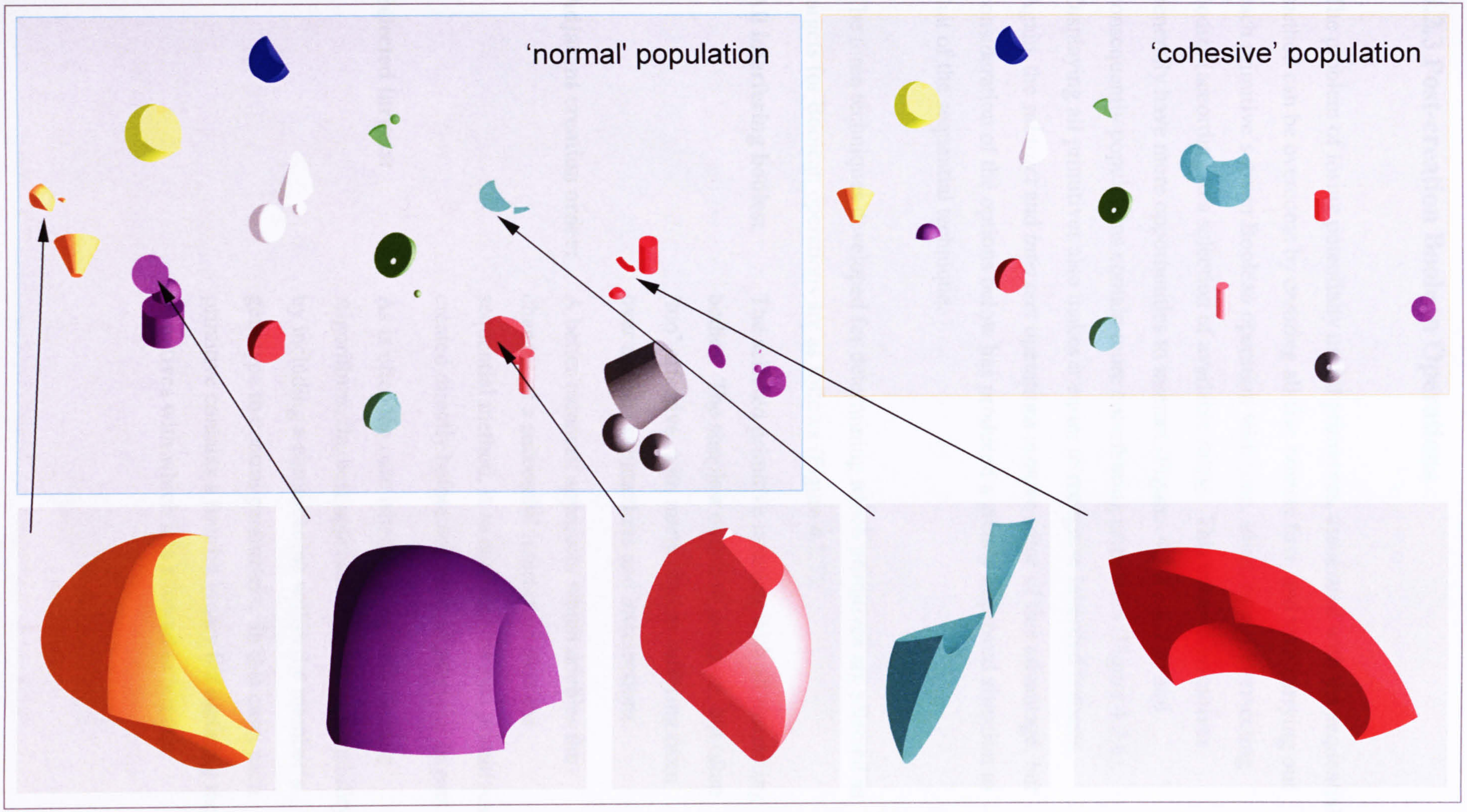


Figure 4.2.2 - Parts missing from equivalent 'cohesive' population

4.2.3 Post-creation Boolean Operations

The problem of losing potentially useful primitives, associated with the sequential method can be overcome by *creating* all five objects first, and then carrying out each primitive's given Boolean operation with (one, some, or all) intersecting bodies (according to a selection of available rules). This ensures primitives generally have more opportunities to interact (Figures 4.2.3 – 4.2.5) and consequently populations contain more contributing primitives (Figure 4.2.6). Displaying all primitives also makes it easier to recognise inherited features. Again, the *subtract* and *intersect* operations remove some of this advantage, but consideration of the options below has produced a greatly improved situation to that of the sequential technique.

The three techniques developed for determining which primitives are selected as targets for Boolean operations are as follows (Figure 4.2.7):

All interfering bodies: The selected primitive interacts with all interfering bodies. The simplest option to program, but often 'too' effective, with many objects suffering from over zealous subtractions and intersections.

Adjacent creation order: A better balanced approach, which doubles the chances of a successful interaction over the sequential method, is to consider the two primitives created directly before and after as potential targets.

Selected targets: As is often the case when dealing with Genetic Algorithms, the best approach is to allow variability by including a chromosome within the member's genotype to control parameters. In this case, each primitive contains a four bit binary list dictating the primitives with which it is allowed to interact.

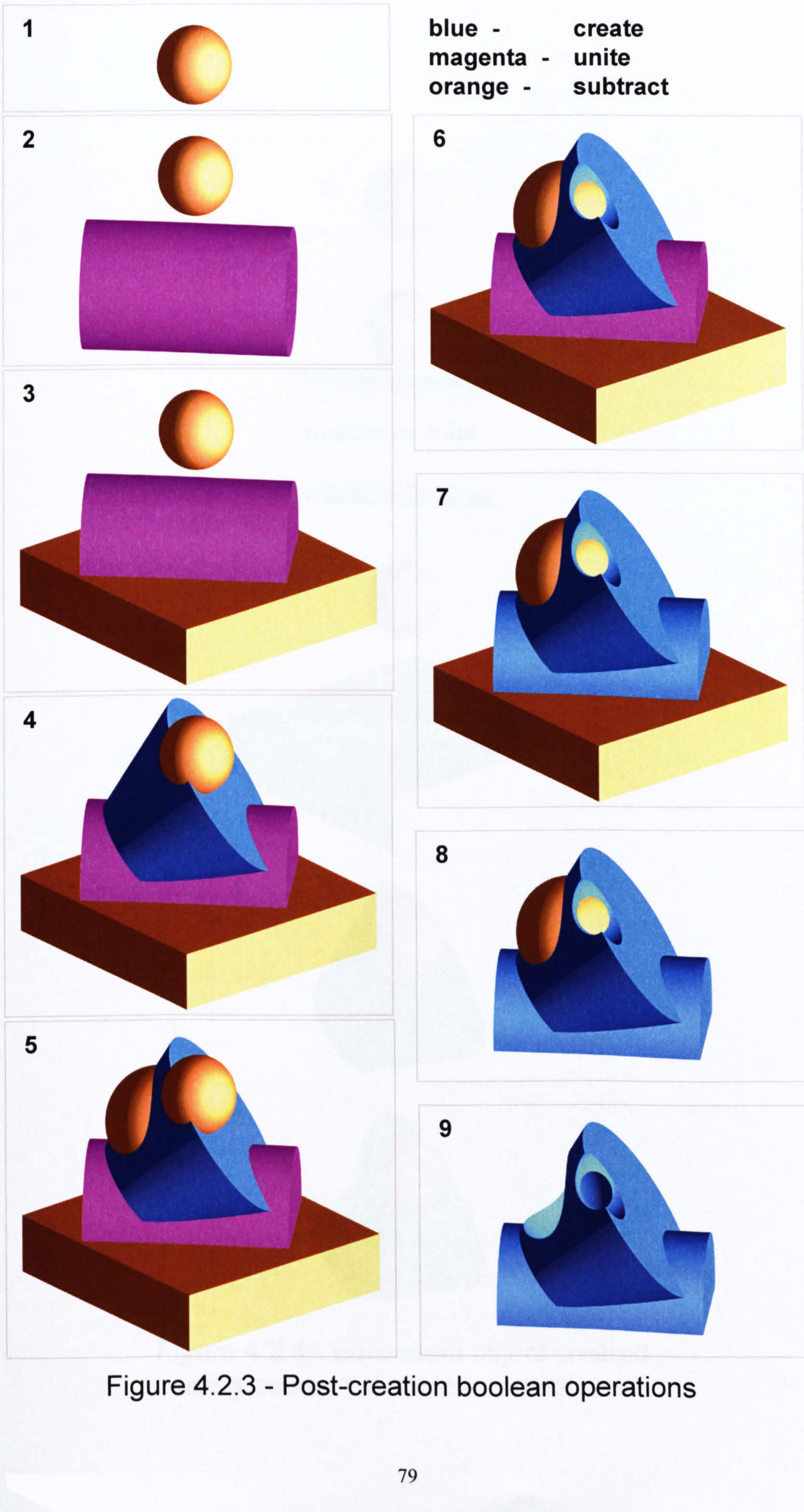


Figure 4.2.3 - Post-creation boolean operations

Figure 4.2.4 - Equivalent object created using sequential boolean operations

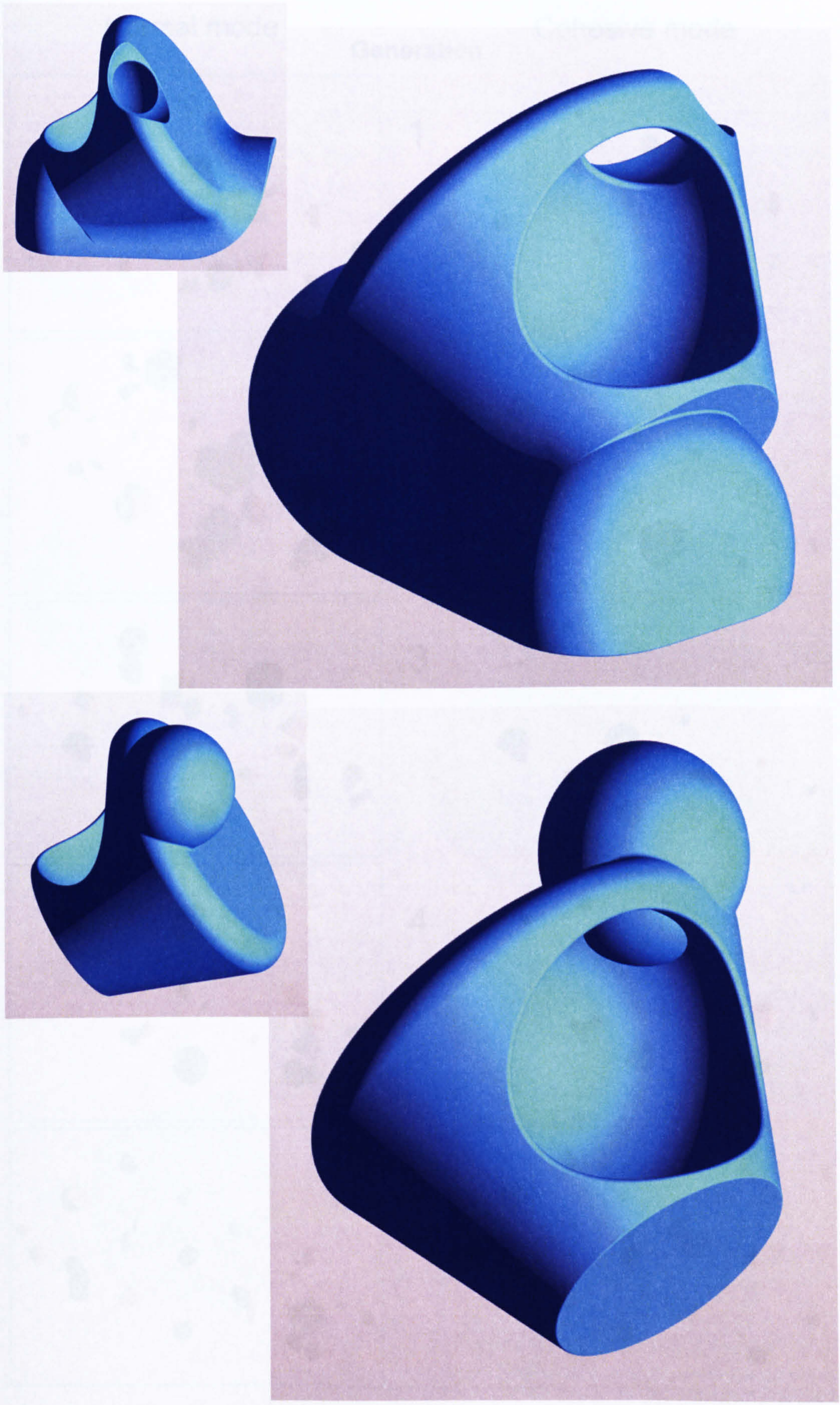


Figure 4.2.5 - Comparison of the two creation techniques

Figure 4.2.6 - Comparison of equivalent normal and cohesive populations

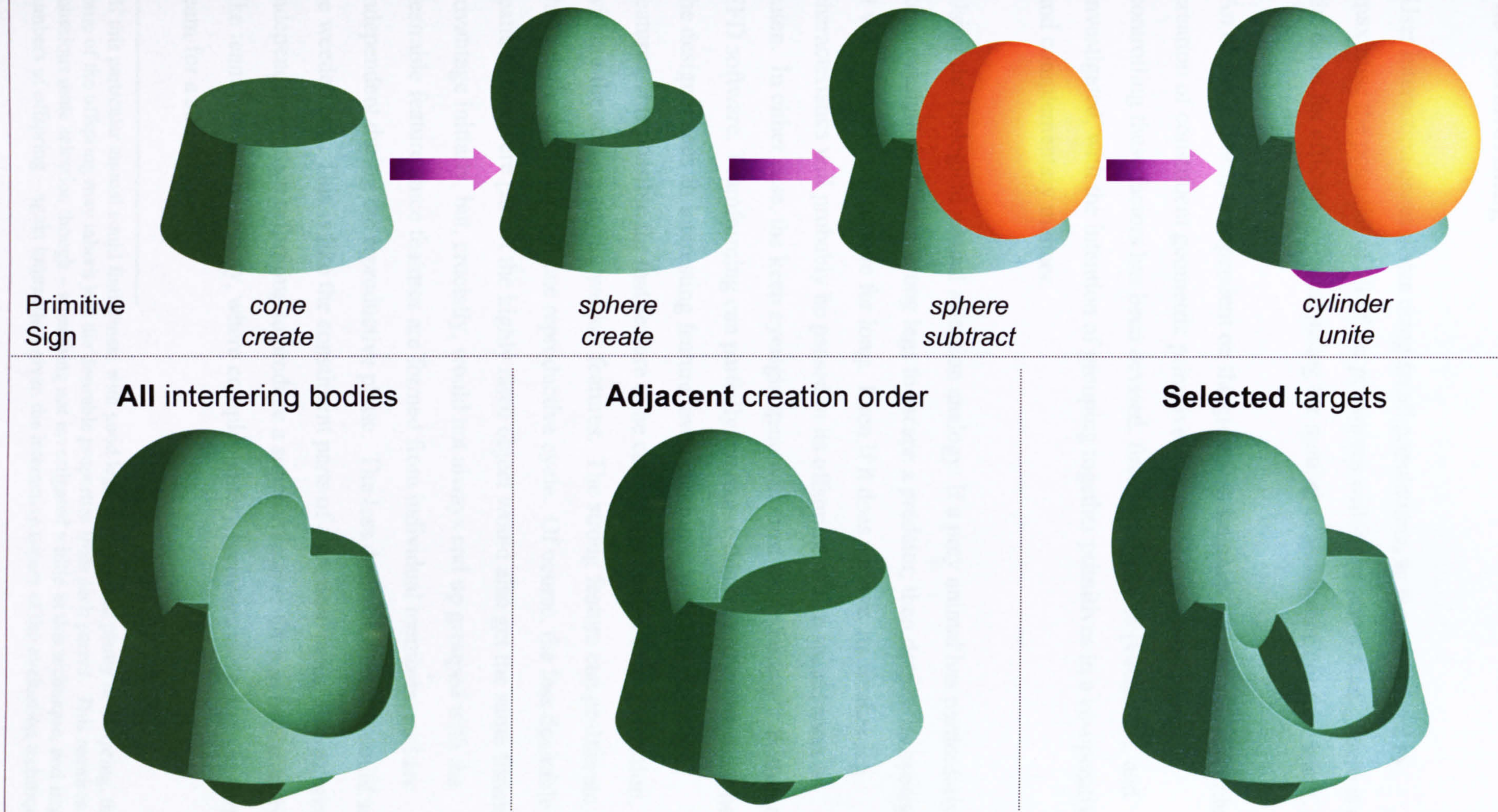


Figure 4.2.7 - The three methods of post-creation boolean target selection

4.3 Teamforming

User interaction necessitates using small populations, so there is a need to maximise the potential of all of the primitives within the population. There is also the drive for efficiency – minimising the number of generations or time spent.

An object's fitness is dependent on the grouping, interaction, method, and order of creation of constituent geometric primitives. A method of more closely controlling these factors has been devised, first introduced in [Graham2], and investigated, with the intention of grouping together primitives in a co-operative and complementary manner.

Using the biological natural selection analogy: If a prey animal has particularly keen eyesight, but lacks strong legs to escape a predator, then despite its eyesight it will probably not survive for long. Even if it does survive, its weaker leg characteristics will probably be passed to its offspring, which would soon be eaten. In either case, the keen eyesight gene will most likely be lost[1]. Within the EFD software, Teamforming can partially overcome this type of problem: When the designer sees an interesting feature, even combined with less desirable features, it is possible for that feature to be carried over to the next generation, without the accompanying weaker features. The strong feature can proliferate through the population in one reproductive cycle. Of course, the less desirable features that were part of the highly rated object would also get the same fitness advantage initially, but, crucially, would not always end up grouped with the desirable feature since features are formed from individual members, and are independent during the reproductive phase. The less desirable features could soon be weeded out. This is like the constituent parts of a creature being able to breed independently, then reforming to produce a super-creature. Or less implausibly, like team creation in society, where complementary members are sought to form a team for a suitable task.

---

[1] If this particular animal could find a mate with good legs, and they had plenty of offspring, then some of the offspring may inherit just the desirable properties from each parent. This requires conscious mate selection though – something not investigated within in this technique, and large numbers of offspring – again impractical given the interactive nature of the evaluation technique.

Variable size teams

The Teamforming method also offers the potential for variable numbers of primitives per object within a single population. The perceived benefit of this is that users could quickly 'discover' the optimum number of primitives per object for their particular application, rather than using the current default value of 5, or experimenting with a different value at the start of each session.

4.3.1 Reproduction within Teamforming

In the lower half of Figure 4.3.1 is the third-generation object 'g3t10-grey. This object is made up from a team of 5 members (phenotypes) – these team-members are individual primitives, and are shown above the object. Each member-primitive has a pair of parents, from generation 2. These are shown in the upper half of the figure, below the objects that they contributed to. Figures 4.3.2 to 4.3.4 show the 3 objects from generation 2 in the above example, and their contributing team-members. Some of these members are selected as parents for object 'g3t10-grey', shown in Figure 4.2.5. The creation order is dictated by the order in which the primitives are selected for their team, which is in turn dictated by the Teamforming tactic employed[1].

This technique allows objects to inherit their features from several sources (rather than the two parents for single genotype objects) – in this example, genetic material from three objects is combined (through Teamforming) to form the featured object. Interestingly, it also permits a smaller number of primitives to parent the contributing members – in this case 8 (whereas usually 10, five primitives from each parent, contribute). This added flexibility is accessed through the scoring method – the more objects that are rated, the larger the available gene-pool is for reproduction, and the more diverse the source of parents will be for a given object. If just 1 or 2 objects are given relatively high scores, their features will appear throughout the subsequent population.

---

[1] This example was produced using the 'no-tactic' option, where team-members are simply taken from the primitive list sequentially

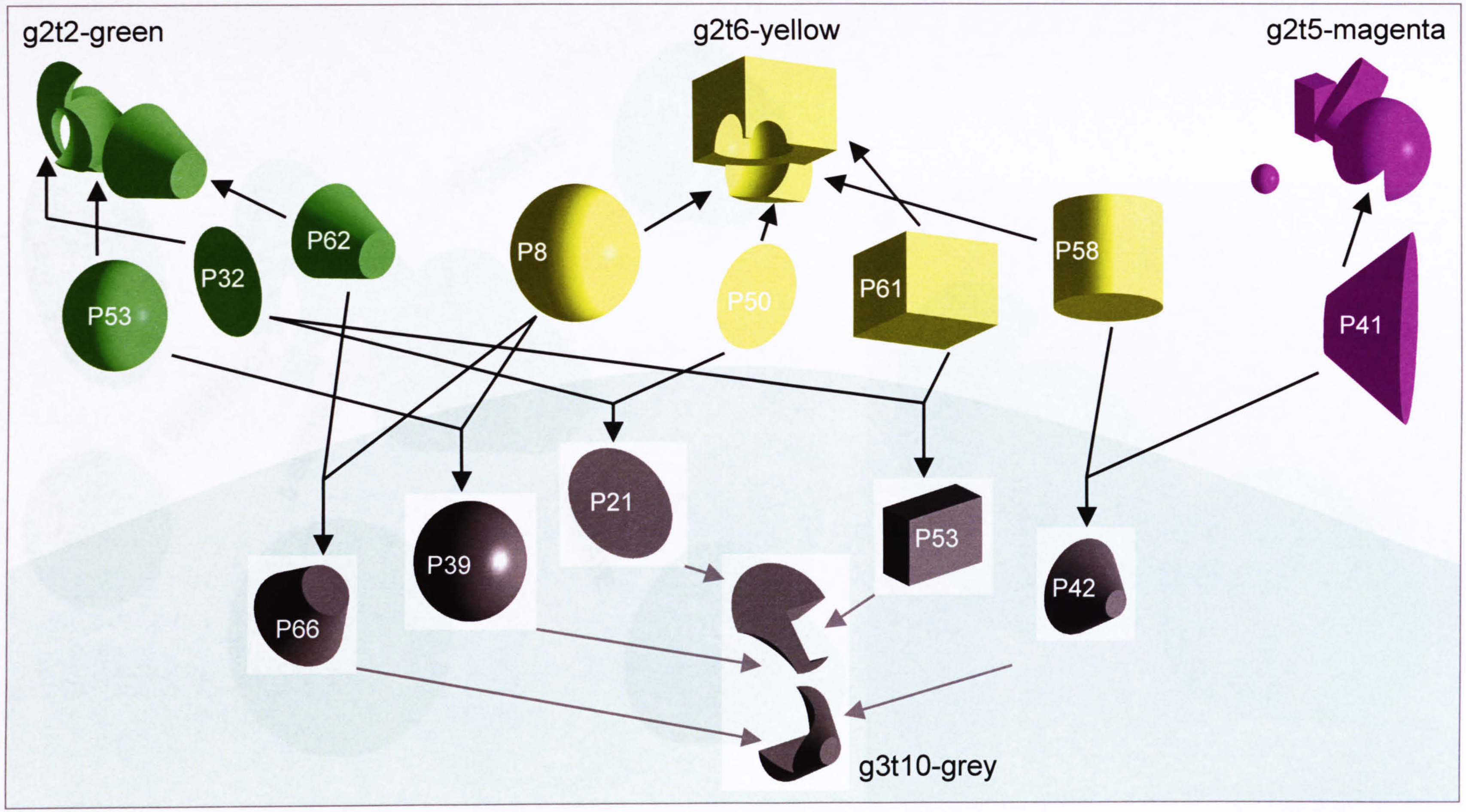


Figure 4.3.1 - Parents of ‘g3t10-grey’ team-members with associated objects

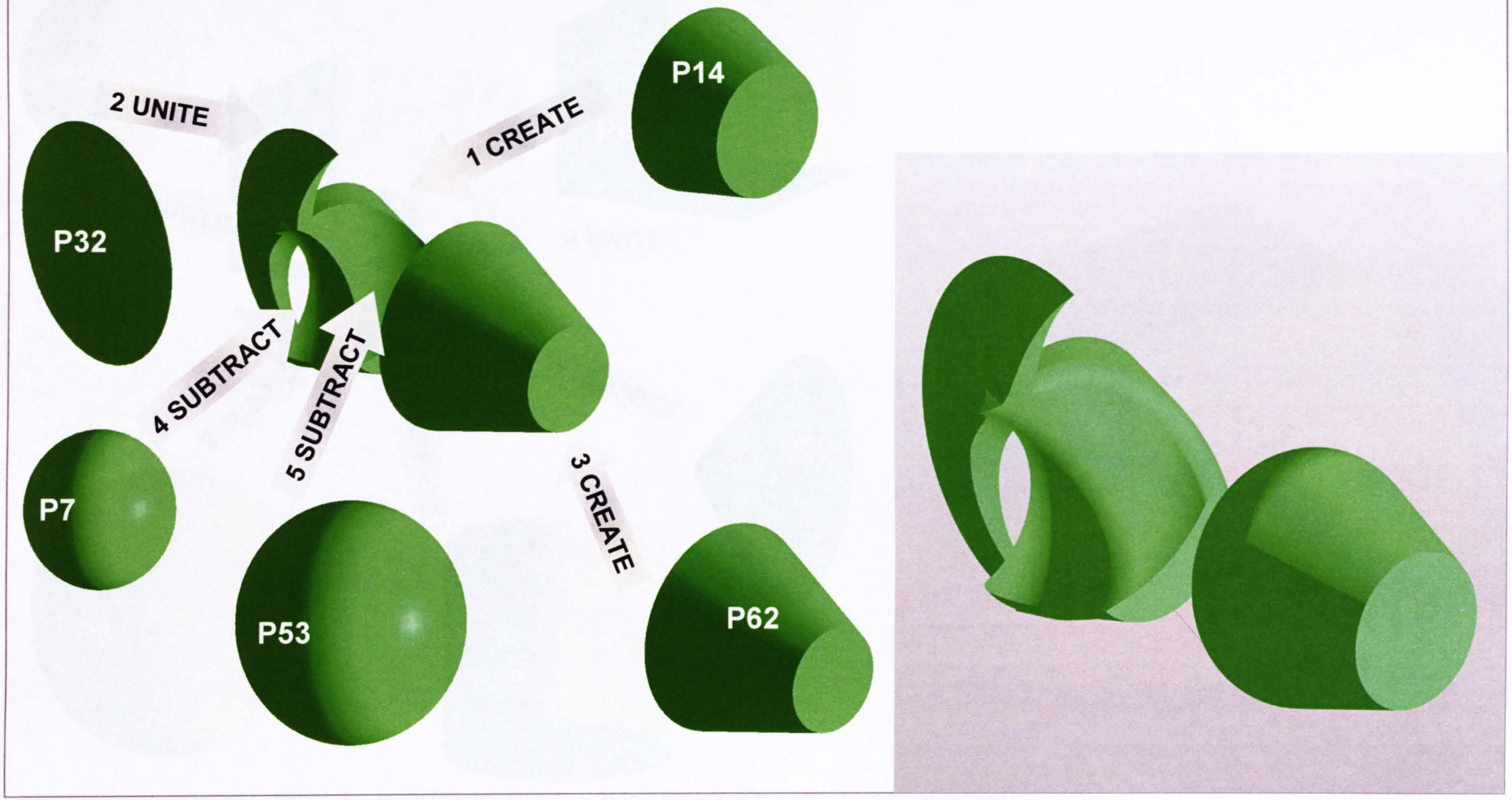


Figure 4.3.2 - 2nd generation object ‘g2t2-green’ containing parents of ‘g3t10-grey’ team-members

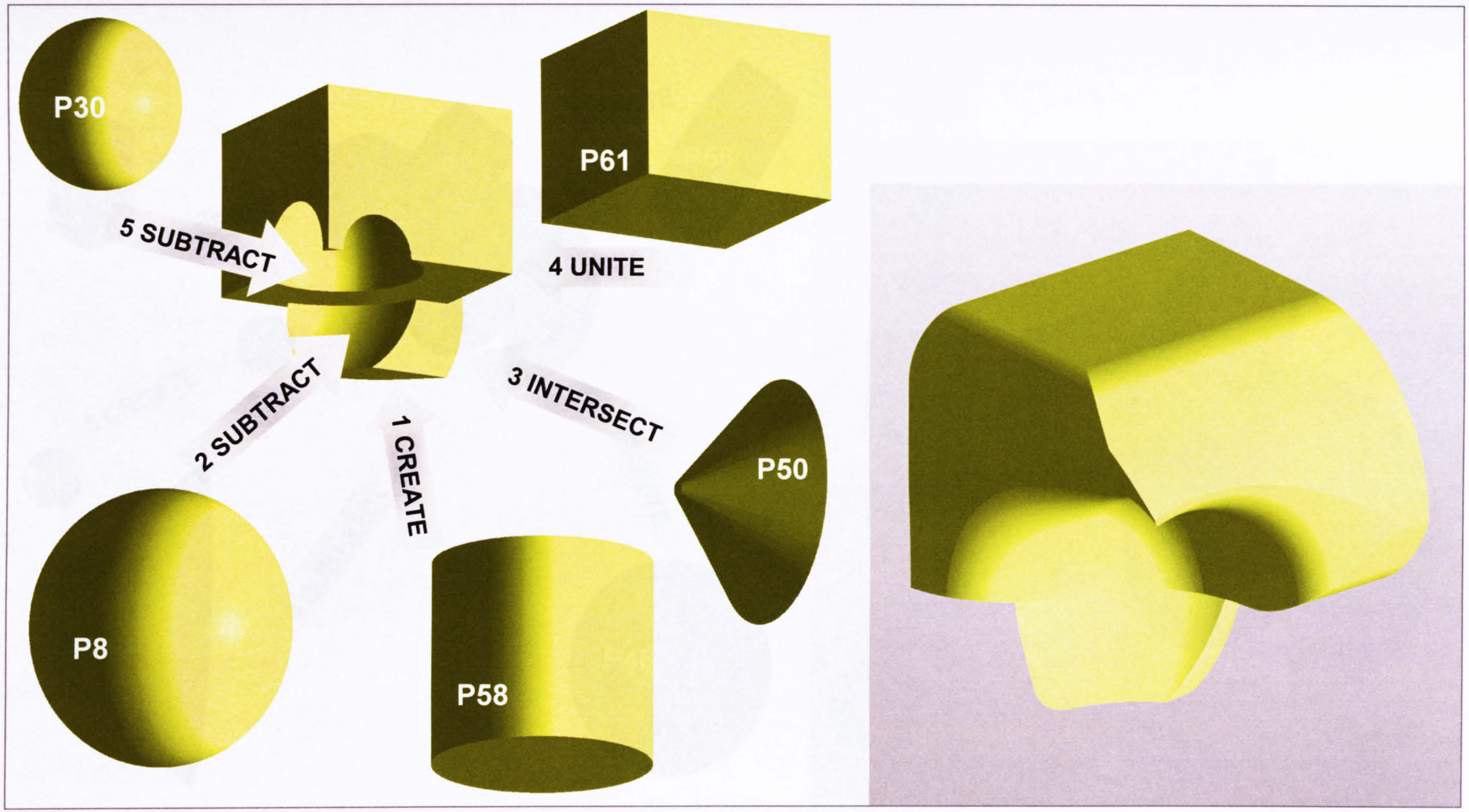


Figure 4.3.3 - 2nd generation object ‘g2t6-yellow’ containing parents of ‘g3t10-grey’ team-members

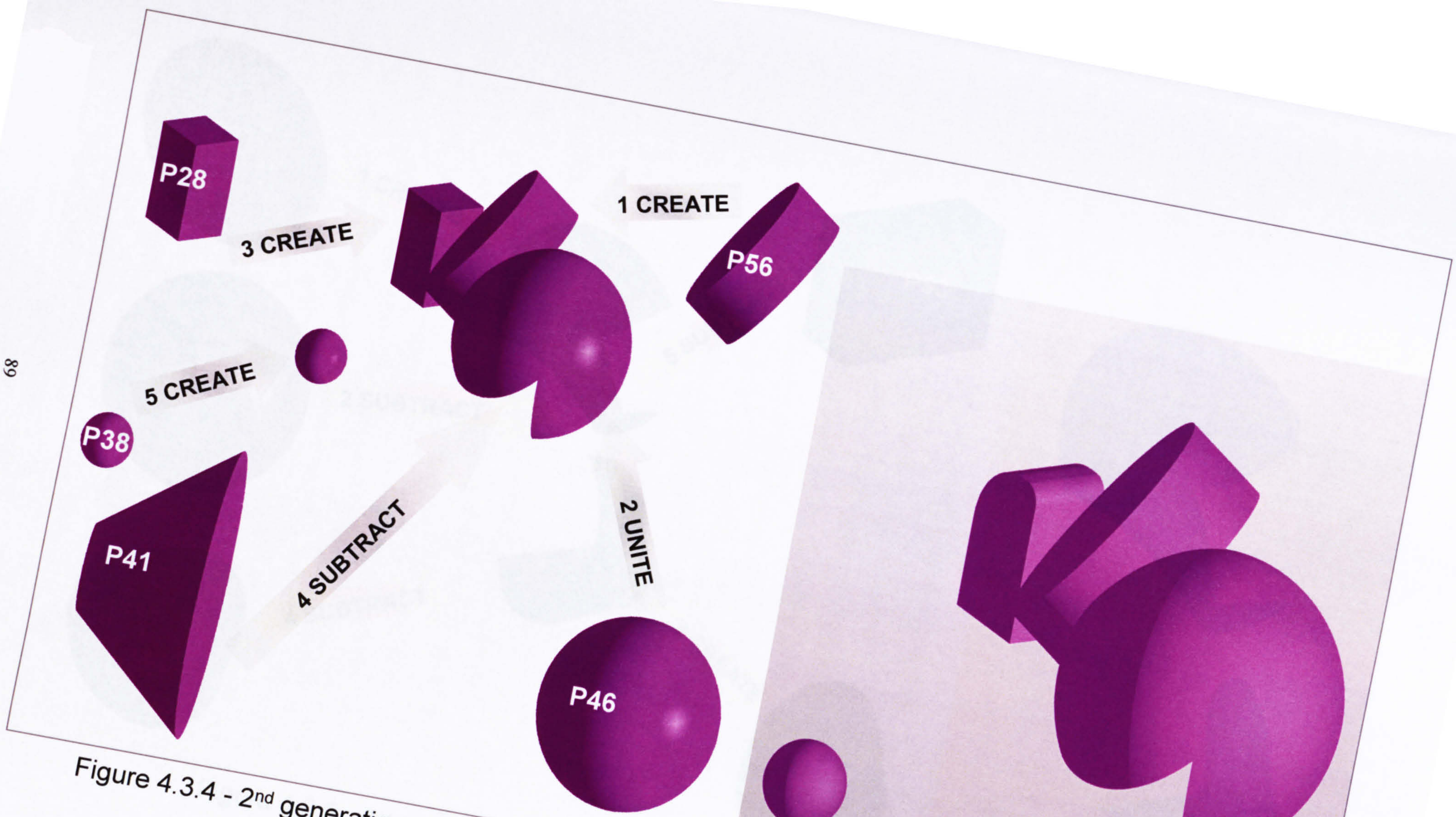


Figure 4.3.4 - 2nd generation object ‘g2t5-magenta’ containing parents of ‘g3t10-grey’ team-members

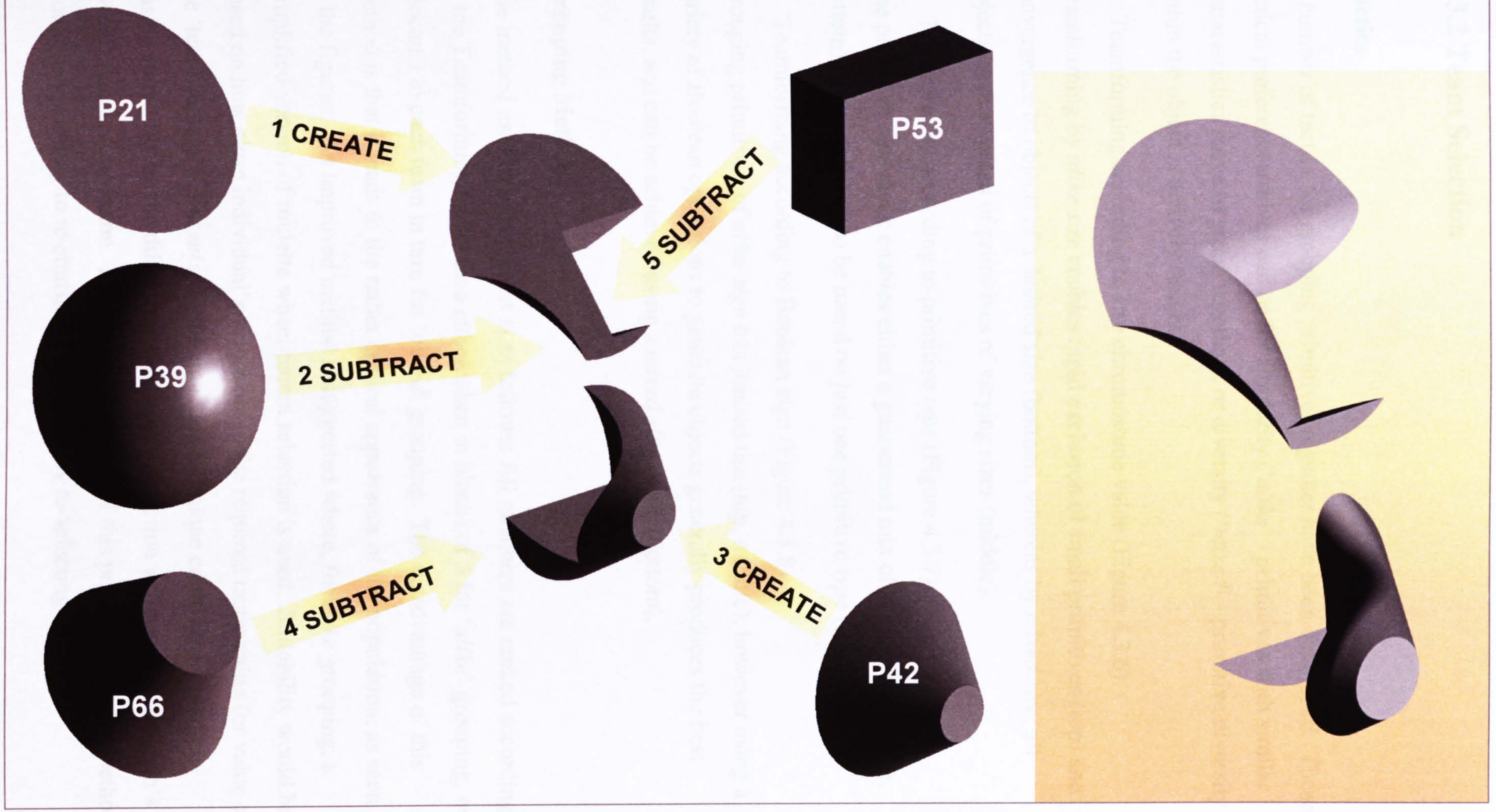


Figure 4.3.5 - 3rd generation object 'g3t10-grey' with contributing team-members

4.3.2 Team Selection

Tactics

A number of tactics for grouping primitives together have been developed. These tactical preferences/drives can be for similarity ('*alike*' - primitives with similar characteristics strive to group together), or diversity ('*mixed*' - primitive diversity within the object is actively sought).

1. Teamforming according to *size* chromosome value (Figure 4.3.6)

Teamforming by *alike size* enables rapid exclusion of small primitives (top) and convergence to objects of a desired size (bottom), whereas by *mixed size*, all objects have a spread of primitives of varying sizes (middle).

2. Teamforming according to primitive *type* (Figure 4.3.7)

The primitive *type* tactic enables either a guaranteed mix of primitives (middle, bottom) or most objects to be based on just one primitive *type* (top).

3. Teamforming according to Boolean *sign* (Figure 4.3.8)

Grouping primitives of *alike sign* has limited use (top, middle), however using a variety of Boolean operations to generate objects generally produces the best results, and can be achieved using a *mixed sign* tactic (bottom).

Grouping Method

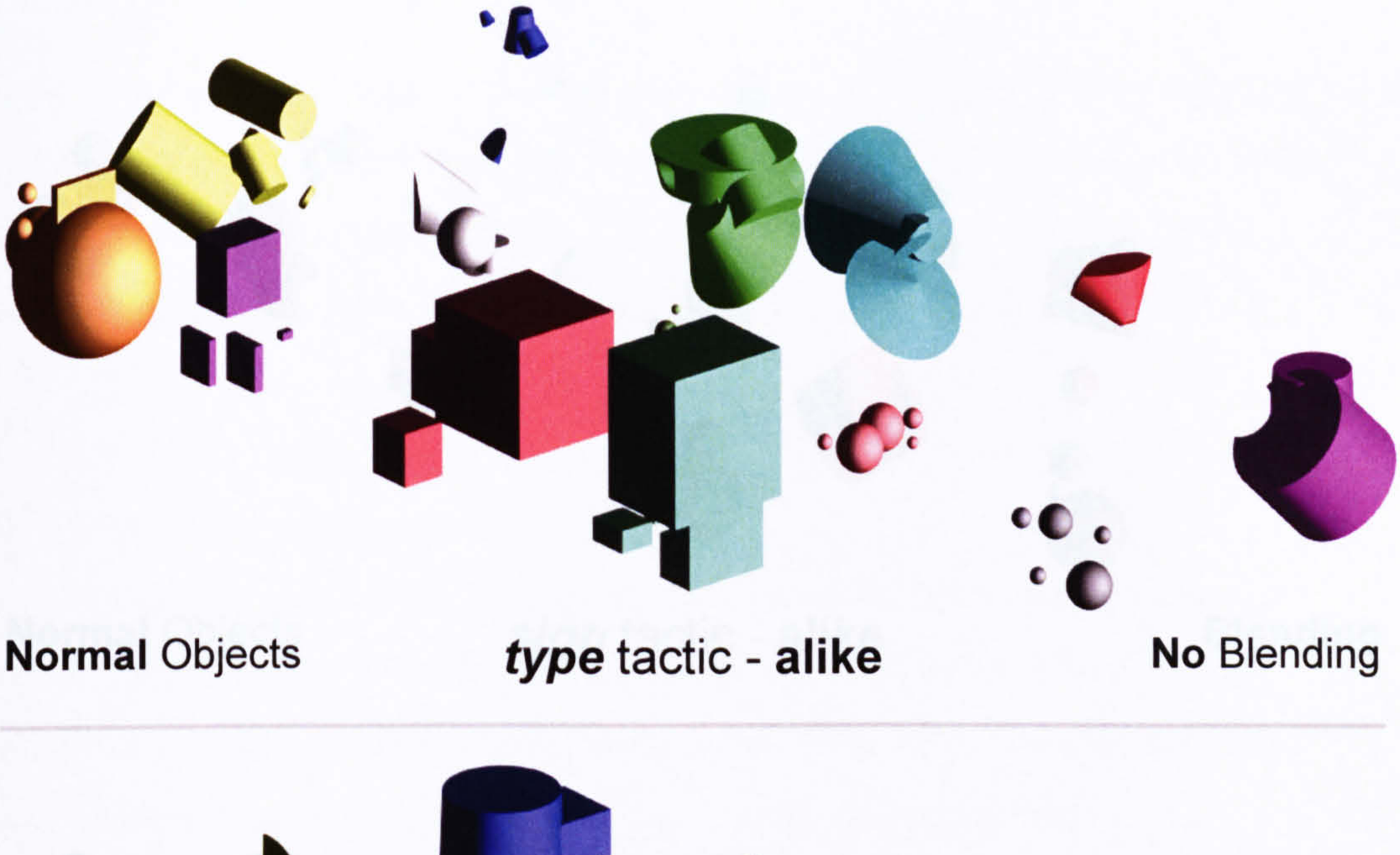
The method used to select teams is as follows: All members are ranked according to the Teamforming tactic, then either taken in blocks of 5 for '*alike*' grouping, or allocated to each team in turn for '*mixed*' grouping. The disadvantage of this method is that it leads to the rather clinical appearance of the population, as seen in the figures. An improved method is suggested where, for *alike* grouping, a simplified version of roulette wheel parent selection is used. Suitability would be based on how close individual's values are to the required tactic value (or value of the 'team captain'). For *mixed* grouping, this technique could be adapted by basing suitability on the difference a member's selection will have on the team's overall standard deviation. An additional benefit of this probability-based method would be the ability to re-shuffle a population by re-selecting teams.

Normal Objects | ***size*** tactic - **alike** | **No** Blending

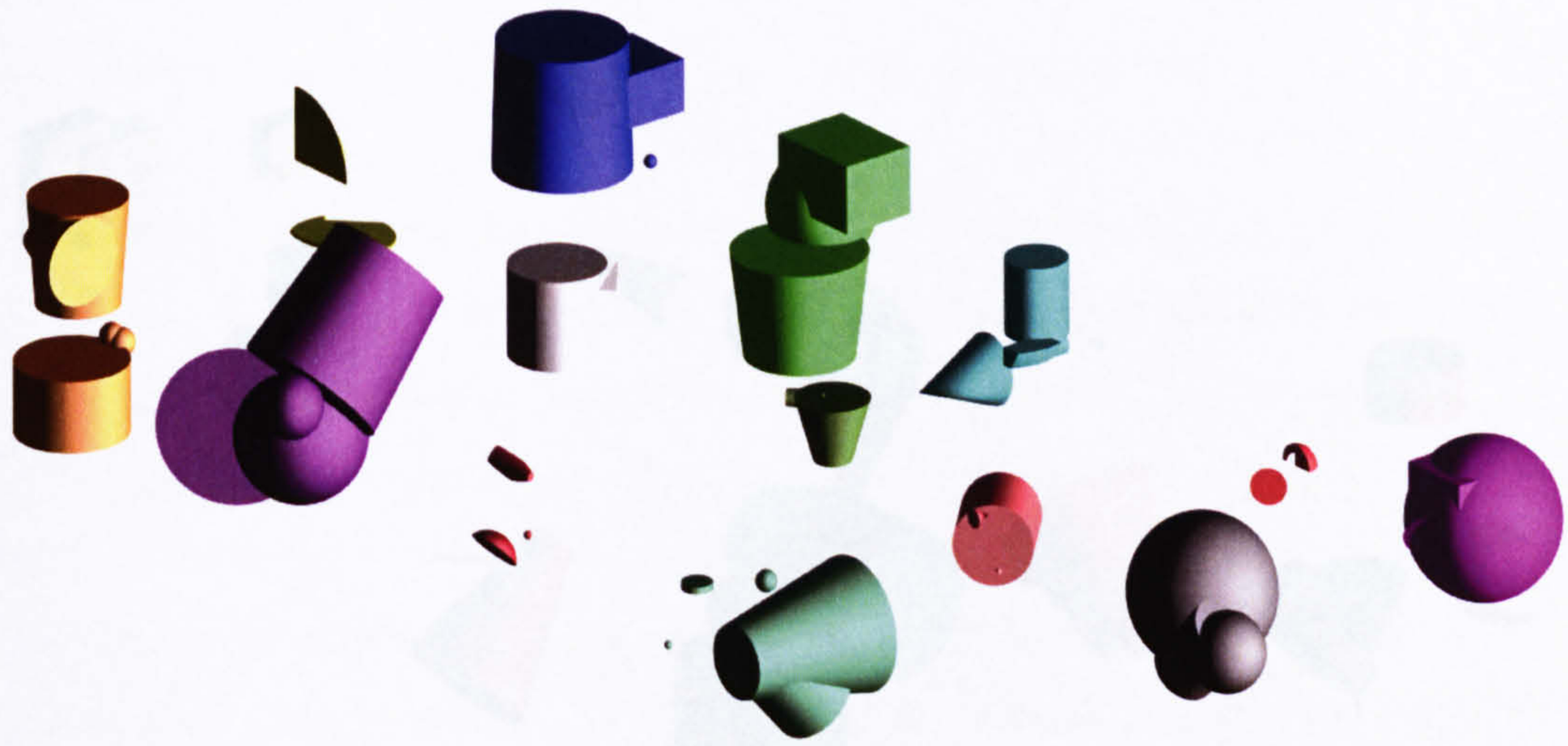
Normal Objects | ***size*** tactic - **mixed** | **No** Blending

Cohesive Objects | ***size*** tactic - **alike** | **Blending**

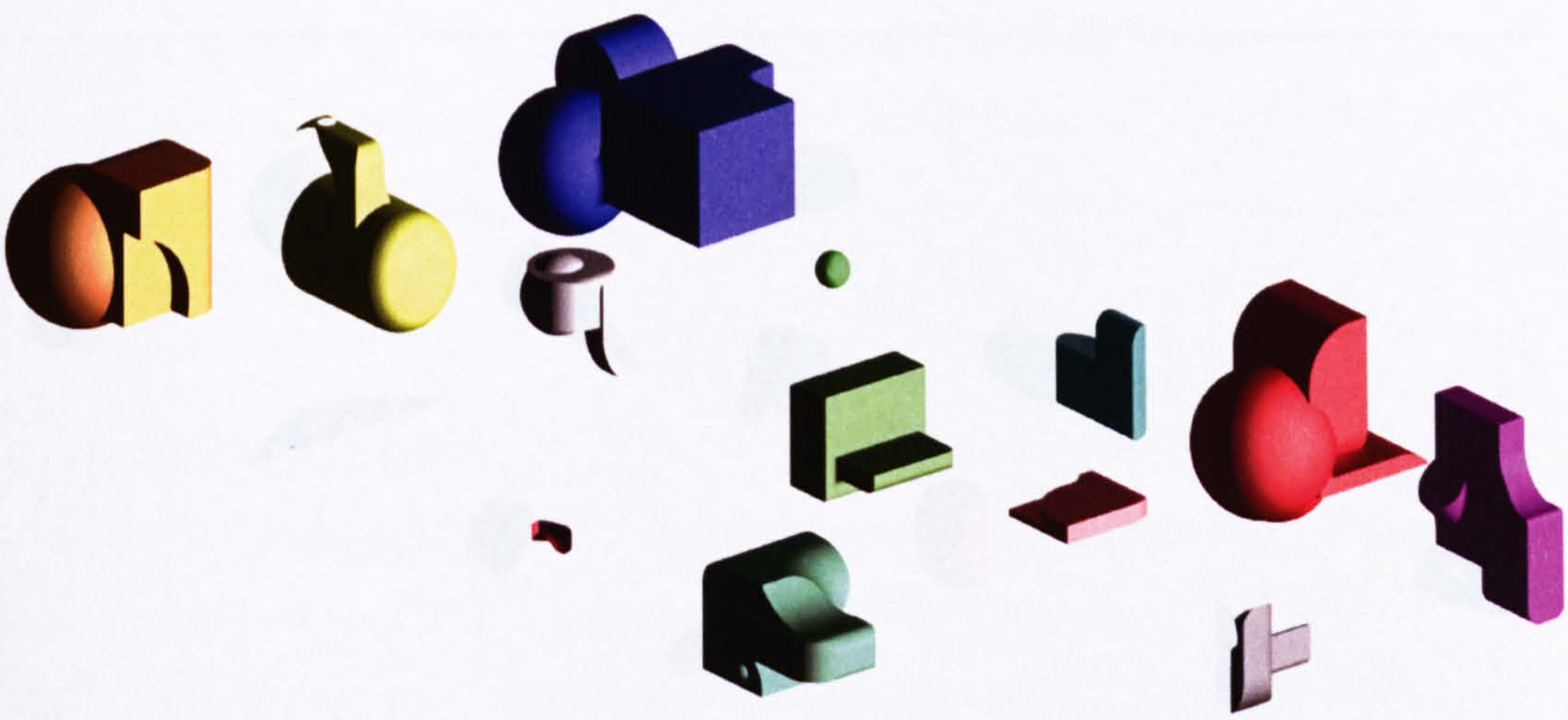
Figure 4.3.6 - Teams grouped by *size* chromosome value



Normal Objects | ***type*** tactic - **alike** | **No** Blending



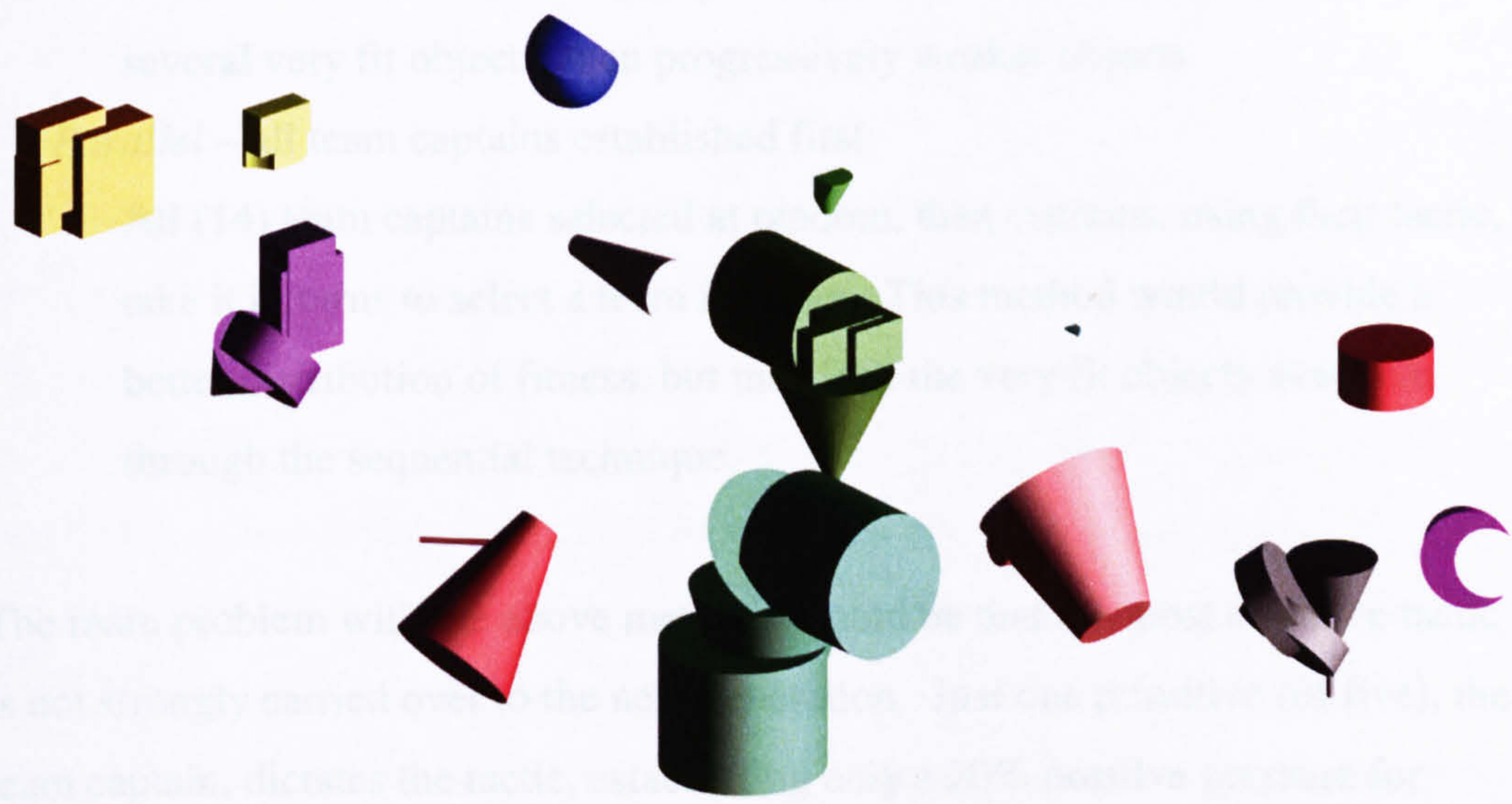
Normal Objects | ***type*** tactic - **mixed** | **No** Blending



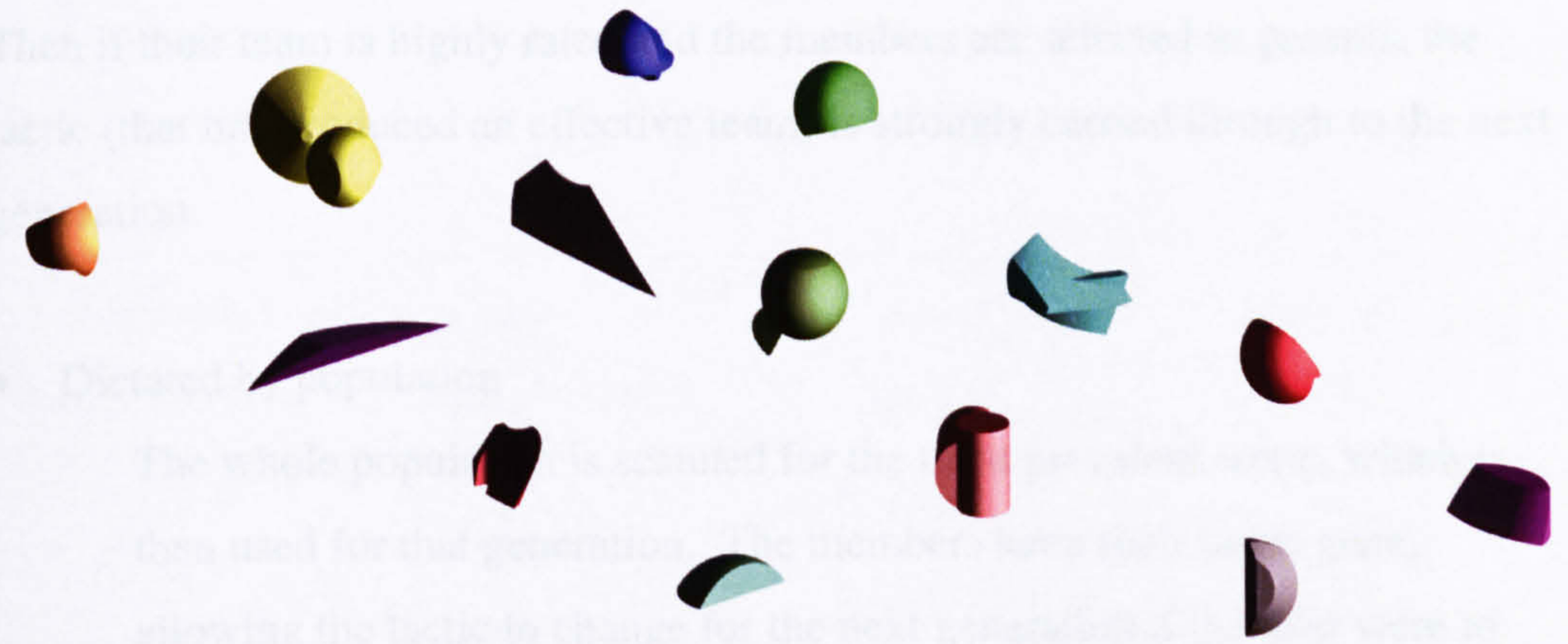
Cohesive Objects | ***type*** tactic - **mixed** | **Blending**

Figure 4.3.7 - Teams grouped by primitive *type*

Normal Objects | ***sign*** tactic - **alike** | **Blending**



Cohesive Objects | ***sign*** tactic - **alike** | **No** Blending



Cohesive Objects | ***sign*** tactic - **mixed** | No **Blending**

Figure 4.3.8 - Teams grouped by boolean *sign*

Tactic Selection

As well as the simple method whereby the tactic is established at the beginning of a session, which has been used here to demonstrate the technique, it is suggested that it would be interesting and beneficial for the tactic to be dictated by the population in some way. The following approaches are recommended:

- Dictated by members

  Sequential – one team selected at a time

  One member selected at random, as the team captain, and it's preferred tactic used to select four further primitives. The next team captain is selected from the remaining population. This method would produce several very fit objects, then progressively weaker objects.

  Parallel – all team captains established first

  All (14) team captains selected at random, then captains, using their tactic, take it in turns to select a team member. This method would provide a better distribution of fitness, but may lack the very fit objects available through the sequential technique.

The main problem with the above methods would be that the most effective tactic is not strongly carried over to the next generation. Just one primitive (of five), the team captain, dictates the tactic, establishing only a 20% positive pressure for tactic type. There seems to be a good case here for gene modification during evolution, whereby all team members inherit the tactic from the team captain. Then if their team is highly rated and the members are selected as parents, the tactic (that has produced an effective team) is strongly carried through to the next generation.

- Dictated by population

  The whole population is scanned for the most prevalent tactic, which is then used for that generation. The members keep their tactic gene, allowing the tactic to change for the next generation if the user were to select a majority of objects with an alternative certain tactic.

4.3.3 The Teamforming Genotype

As described in the earlier section on the genetic data structure, 2-bit segments (decoded to integer 1, 2, 3 or 4) control *type* and *sign* characteristics. These are perfectly suitable when using the non-Teamforming, 'single phenotype objects' method. However, when using this genetic structure for team selection, using these properties (type or sign), the bond between team groups is very weak, accounting for a total lack of object continuity between generations. This is because, whatever the specific requirement during team member selection (a sphere may be sought for example) there are many examples throughout the population to choose from. It then comes down to a random or arbitrary decision as to which member is selected.

Continuity is much improved when other Teamforming tactics are employed, grouping by *size* for example. Here, it is possible to sort all members according to this criteria, the data being continuous (real numbers are used). Therefore, when the instruction to select a body of specific size (or largest/smallest) is given, it is usually possible to select 1 specific member from the ranked population. The favourable results achieved, while using tactics based on properties defined by variable data, led to a change of the *type* and *sign* chromosomes: When the Teamforming technique is utilised, longer 6-bit chromosomes for type and sign (Boolean operator) are used instead of the two bit segments. These are decoded to a real number, between 1 and 64. This change has resulted in an improvement in visual continuity, by restricting the amount of movement between teams of related members, during the parent selection and reproduction stage.

Equivalent decoded 1-bit segment value	Decoded 6-bit segment range	Resulting *type* operator	Resulting *sign* operator
0	1-16	block	create
1	17-32	sphere	unite
2	33-48	cylinder	subtract
3	49-64	cone	intersect

4.3.4 Conclusions

With evolution-by-objects, where each object is one member of the population (e.g. of 14), the designer is evaluating the whole object and it is one half of the whole object's genotype that is carried over to each child in the next generation. Continuity (the visual links between the generations) is usually displayed between two parents and their two children. This method of evolution is necessary if a specific object is being designed. A disadvantage of this method is that it takes a while for weak objects to be weeded out of the gene pool, meaning that there are normally more unfit objects than fit ones, at least in early generations. In addition, weak features of good objects are generally carried through with the desirable features.

With evolution-by-features, where each object is a team of members (e.g. 1 object is made up of 5 team members from the population of 70), whilst still giving each object a rating, the designer is evaluating the features of the objects. This quickly creates a population of objects with desirable features, but the continuity between related objects is often lost (Figure 4.3.9).

One of the main problems is that the order of primitive creation is not firmly maintained within a team. With object based reproduction, the five primitives that make up each object, being defined by one long genotype, remain in the same order when reproduction occurs. With Teamforming, or evolution by features, the team selection process dictates the order of primitive introduction, which is not as rigidly maintained.

One aspect of Teamforming that has not been developed is the potential for formation of variable sized teams – this would enable the user to influence, and thus perhaps discover the optimum number of primitives per object, rather than relying on the current presumption that five primitives per object is best. This technique would most suitably be implemented with changing population sizes, but this is not a particularly difficult development.

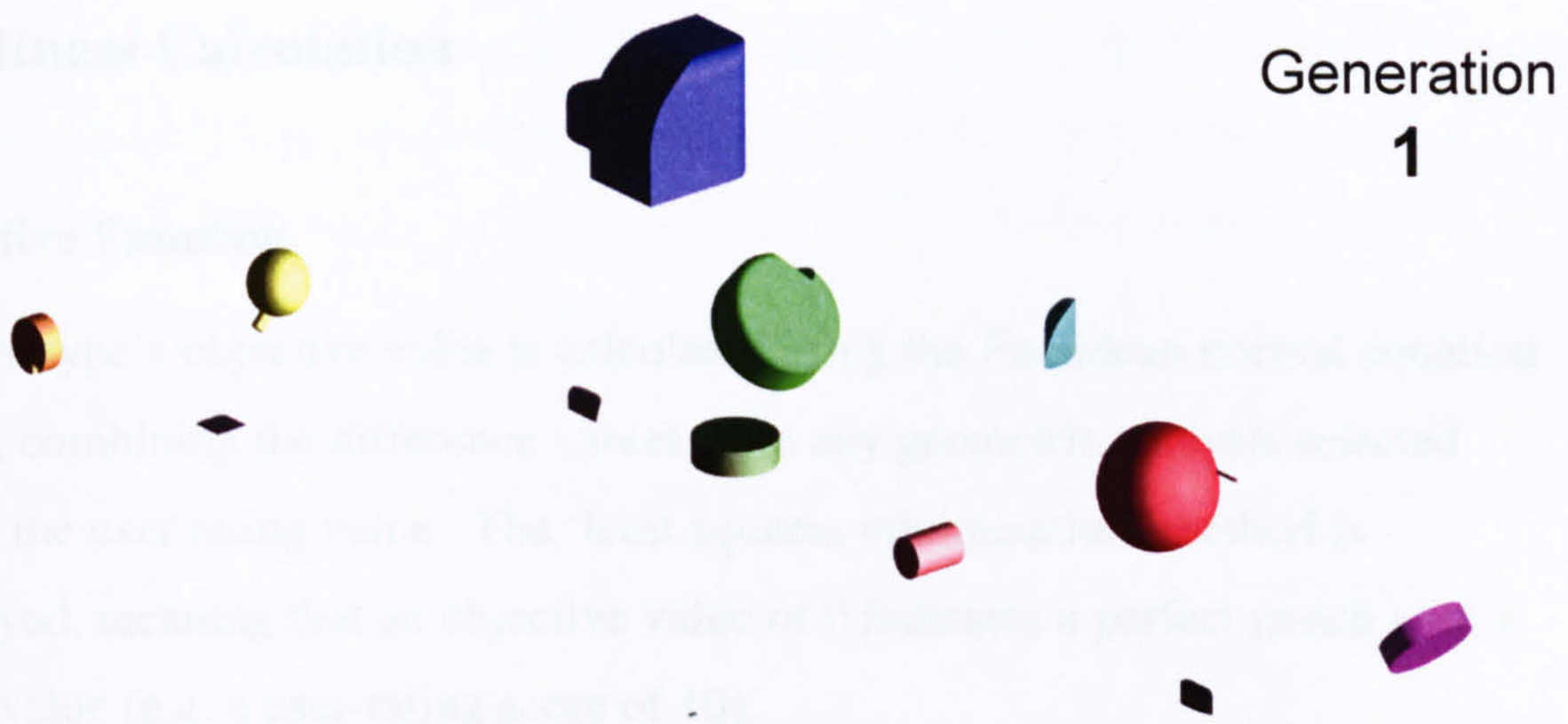


Figure 4.3.9 - Evolution in Teams grouped by *size* value

4.4 Fitness Calculation

Objective Function

A phenotype's objective value is calculated using the Euclidean normal equation below, combining the difference values from any geometric analysis selected and/or the user rating value. The 'least squares minimisation' method is employed, meaning that an objective value of 0 indicates a perfect match with a target value (e.g. a user-rating score of 10).

$$S = \sqrt{\sum_{j=1}^{M} (A_j - T_j)^2}$$

A member's objective value, S, provides a single number that indicates the difference between it's own geometric and rating parameters, A, and pre-set target values, T, where M is the number of evaluation parameters employed.

Fitness Function

A member's fitness value is calculated, to between 0 and 1, using the following equation:

$$f = \frac{1}{1+S}$$

The inclusion of the '1' values is to allow for a scaling factor, useful for multi-objective problems, but, as discussed below, has been found to be unnecessary at this stage.

4.4.1 Multiobjective Balance

In this case, sensible handling of the units of geometric analysis negates the need to set up multi-objective balancing. Using cm, dm^2 and dm^3, the properties of length, surface area and volume have similar influence. When geometric analysis is combined with user-ratings, and realistic target values are specified (generally values between 5 and 95), a good balance is usually achieved between user-ratings

and geometric analysis by generations 3-5. The following table illustrates this – the example geometric optimisation fitness values were taken from a session where the geometric target was height = 35.0cm. The table compares the automatically generated fitness values for a first generation population with the fifth generation population. To provide a comparative scale, listed to the right of these fitness values are the user-supplied score values that would have produced similar fitness.

Member	**Generation 1**		**Generation 5**	
	Fitness	Equivalent rating value	Fitness	Equivalent rating value
1	0.05	0	0.25	7
2	0.13	3	0.52	9
3	0.02	0	0.96	10
4	0.06	0	0.04	0
5	0.05	0	0.53	9
6	0.50	9	0.64	9
7	0.03	0	0.97	10
8	0.05	0	0.54	9
9	0.03	0	0.05	0
10	0.03	0	0.05	0
11	0.08	0	0.03	0
12	0.08	0	0.64	9
13	0.04	0	0.20	6
14	0.07	0	0.05	0
max.	0.50		0.97	
mean	0.09		0.39	

Reference user-rating values:

rating	0	1	2	3	4	5	6	7	8	9	10
fitness	0.091	0.100	0.111	0.125	0.143	0.167	0.200	0.250	0.333	0.500	1.000

This comparison gives an indication of the relative influence of the two evaluation methods at different stages in the evolution process had geometric analysis and user-rating been combined.

Generation 1: The mean fitness is low, with only 2 members having fitness values that compare with typical user scoring values. So at this stage, a typical spread of user-supplied scores would *dominate*. This gives users control at the early stages of evolution – enabling the types of objects desired to be defined.

Generation 5: A much higher average fitness has been achieved. 9 members have fitness scores comparable with user scoring values. A generous spread of user-ratings would have a *similar* influence to that of the geometric analysis.

After generation 5, geometric optimisation enables the objects to converge to an optimal solution rapidly. At this stage, the user could allow the software to take over, as objects fitness scores tended to 1, ultimately presenting a solution to the geometric target. Alternatively, the user could continue to influence the direction of the population, by allocating scores of 10 (equating to a fitness of 1, thus doubling the combined fitness to 2).

4.4.2 User-supplied Rating

When user supplied rating is the sole evaluation method, two conditions are exaggerated to increase the usability of the system (Figure 4.4.1): Firstly, a fitness value of 0.0909… ($^1/_{11}$, produced from a rating of 0) is changed to 0, so that by giving zero as a rating, the user ensures the object cannot be selected as a parent, thus entirely removing the object's genotype from the gene pool. Secondly, fitness values of 1, resulting from a user-supplied score of 10, are doubled to 2. The full set of values is given in the table overleaf. This simple process was found to be almost as effective in use and much more reliable than a complex technique developed to guarantee the selection of an object scoring 10.

Rating value	Objective Value	Fitness Value
0	10	*0.000
1	9	0.100
2	8	0.111
3	7	0.125
4	6	0.143
5	5	0.167
6	4	0.200
7	3	0.250
8	2	0.333
9	1	0.500
10	0	**2.000

* Adjusted from 0.091

** Adjusted from 1.000

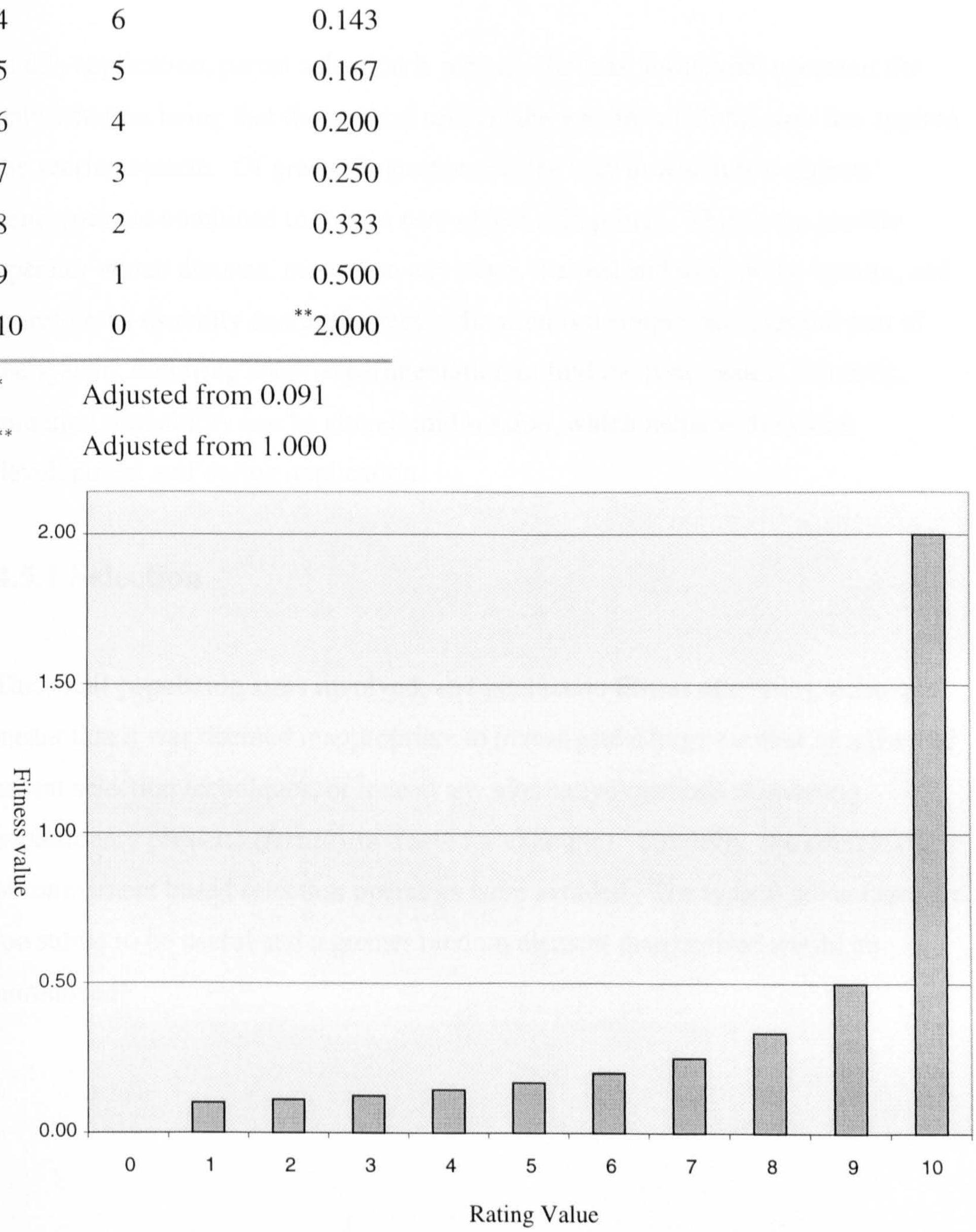


Figure 4.4.1 – Fitness values calculated from user ratings

4.5 Selection and Genetic Operators

A number of established methods exist to carry out the three main processes of:

- Selection
- Recombination
- Mutation

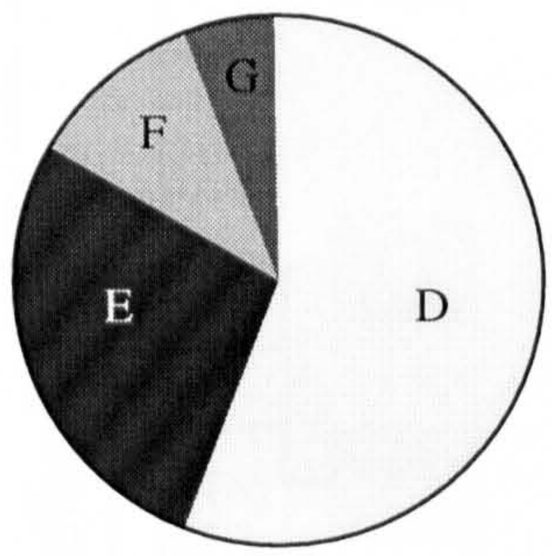
In this application, parent selection is perhaps the least influential operator; the only criterion being that the method reflects the way in which the user has applied the scoring system. Of greater importance is the way in which two objects' genotypes are combined to form a new object (offspring). This is the genetic operator which dictates, more than any other, the feel and look of the system, and therefore its usability and usefulness. Mutation is a simple but essential part of the system, requiring some experimentation to find the best values. Usefully, mutation probability can be altered mid session, which helps with system development and during application.

4.5.1 Selection

The small population sizes involved, and interactive fitness allocating technique meant that it was deemed inappropriate to investigate a large number of advanced parent selection techniques, or indeed any alternative methods of exerting evolutionary pressure (*fertility* or *death* for example). Similarly, the complexities of tournament based selection operators were avoided. The typical advantages are too subtle to be useful and a greater random element than desired would be introduced.

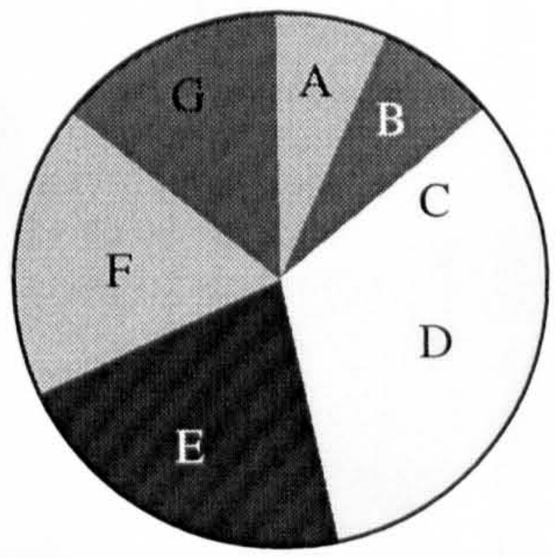
A 'fitness ranking' style operator would have been inappropriate since, with this technique, the actual fitness scores, once the objects have been sorted into fitness order, are ignored. So given following scores:

Object	Score
A	0
B	0
C	0
D	10
E	5
F	2
G	1



The detail of the scoring would be lost:

Rank order	Rank score	Probability
D	7	0.25
E	6	0.21
F	5	0.18
G	4	0.14
A	2	0.07
B	2	0.07
C	2	0.07



Of the two common fitness proportion techniques, stochastic remainder selection was the clear choice over roulette wheel selection. It works well with the scoring technique, giving the user a direct control over the selection, as with the first pie chart above. A further advantage using a fitness-proportionate method is that it incorporates the fertility selection method. This allows relatively fit individuals to parent a controlled but potentially unlimited number of offspring (Figure 4.5.1), thus dominating the next generations if the user so desires (indicated through scoring).

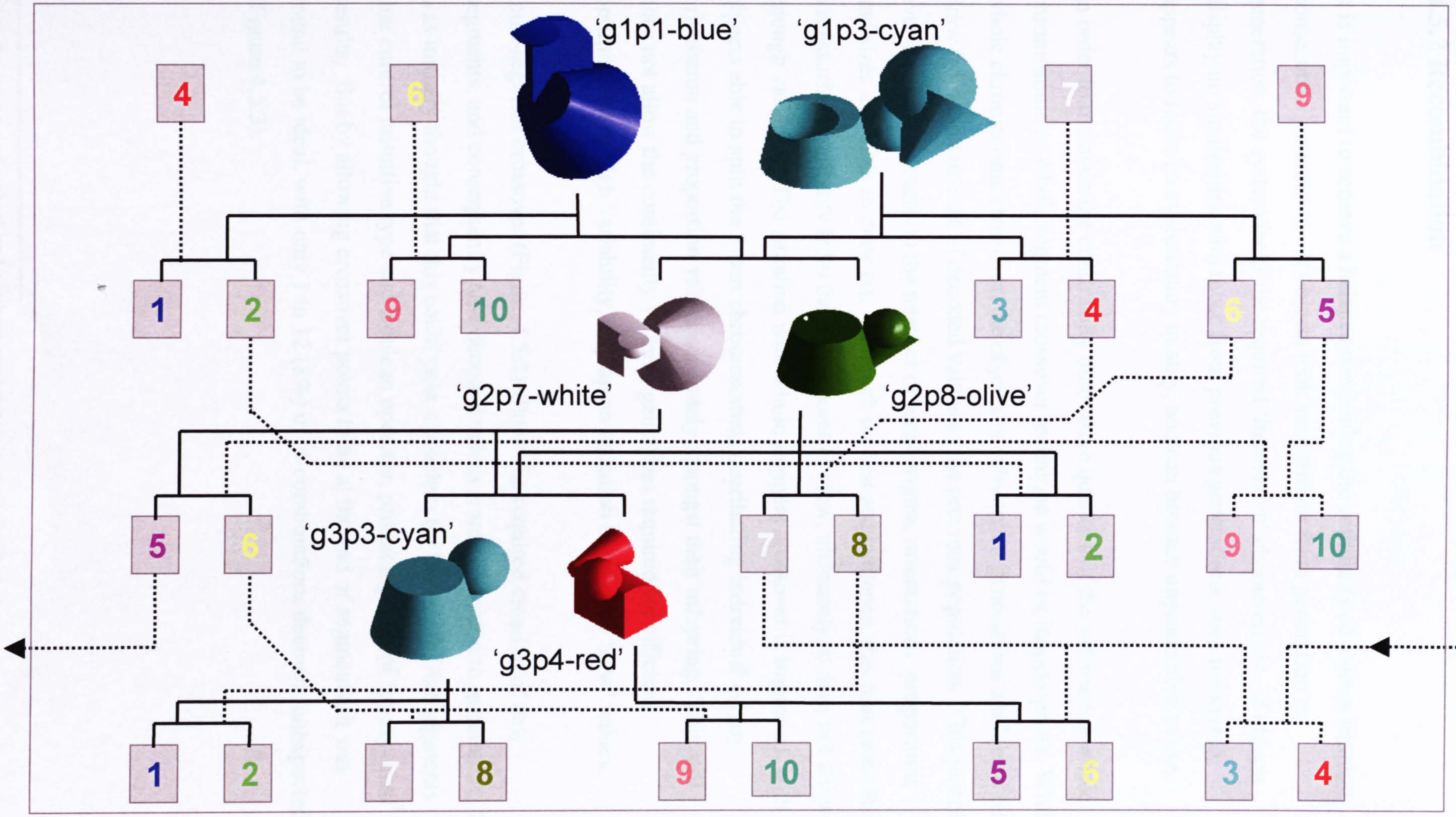


Figure 4.5.1 - Family tree showing the dominance of the 2 members 'g1p1-blue' and 'g1p3-cyan'

4.5.2 Recombination

It is important to achieve a balance concerning the continuity of objects through consecutive generations. If objects look very similar from generation to generation, the system lacks the required 'innovative' characteristic. If objects display no similarities with those from previous generations, then the system appears to loose its evolutionary quality, and can become unproductive to use.

In order to maintain the continually innovative qualities of the software, a whole chromosome or whole segment crossover technique would be inappropriate. With whole chromosome crossover, the objects in subsequent generations are forced to draw from the same set of decoded values as the previous population. This places ever-decreasing limits on the number of local origins, orientations, proportions and sizes available to draw on. Although on first acquaintance, this trait provides a reassuring continuity from the user's point of view, ultimately, it does not allow enough variation. The situation with whole segment crossover is improved, with objects able to split the 3-part chromosomes, distributing individual origin, orientation and proportion values separately amongst their offspring. This still does not allow the continually variable genotypes required for efficient optimisation or high variability, relying on mutation to produce 'new' values.

Intra-segment crossover (Figure 4.5.2) allows the required creation of new segments, and consequently new decoded values from generation to generation. It was initially thought that this could pose a problem for the small 2-bit segments that control primitive type and Boolean operator, producing a lot of 'unexpected[1]' results. But by allowing crossover points to be at the end of segments, it was found to be ideal, with only 1 in 12 (8%) of re-combinations deemed 'unexpected' (Figure 4.5.3).

---

[1] i.e. the crossing of a block and a sphere producing a cone and cylinder

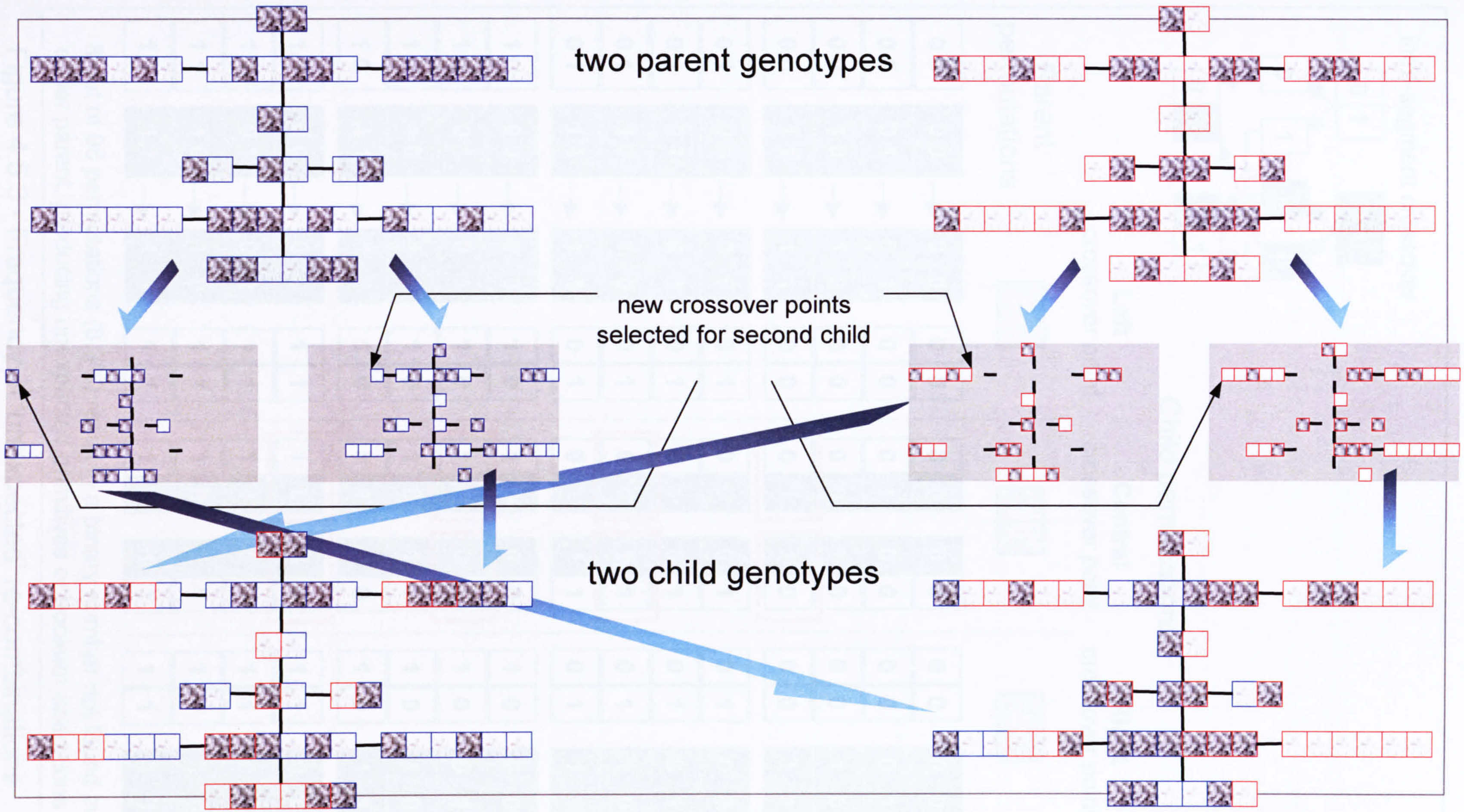


Figure 4.5.2 - Intra-segment non-complimentary crossover

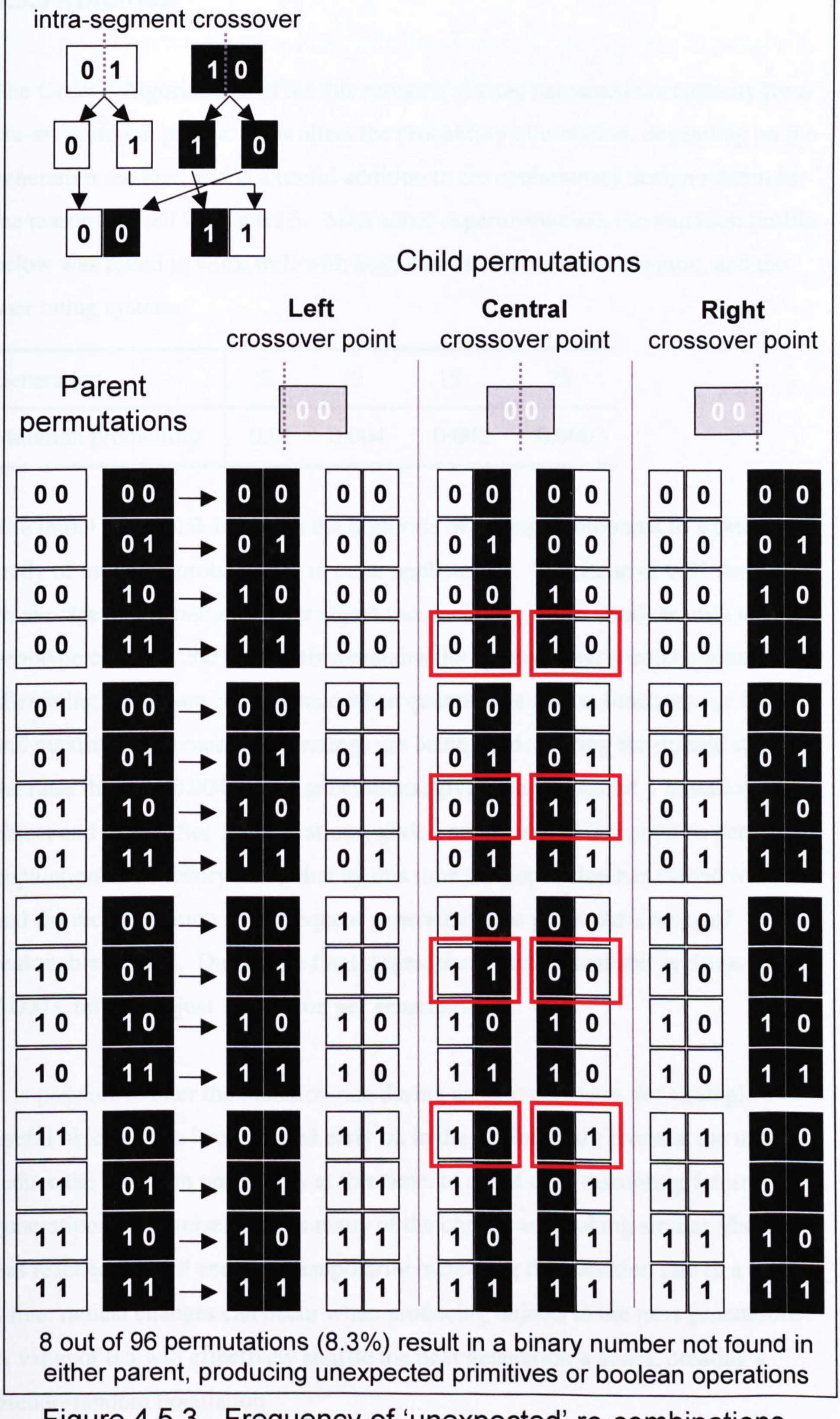


Figure 4.5.3 - Frequency of 'unexpected' re-combinations

4.5.3 Mutation

The Genetic Algorithm used for this research already contained the capacity for a pre-set mutation profile. This alters the probability of mutation, depending on the generation number, and is a useful addition to the evolutionary design system for the reasons set out in chapter 3. After some experimentation, the mutation profile below was found to work well with both internal fitness determination, and the user rating system:

Generation	5	10	15	25
Mutation probability	0.01	0.004	0.002	0.0003

The initial rate of 1% is just on the high side of average, compared to a general study of mutation probabilities in other applications. The value of 0.01 produces an average of 2-3 mutations per object (not including blend data), as each object's genotype contains 260 bits. This maintains the variety in early populations, alleviating premature convergence when quantitative fitness functions are used and maximising choice if user ratings are being used. During the middle stages, the rates drops to 0.004 after 5 generations, giving an average of 1 mutation per object and 0.002 after 10 generations, giving an average of 7 mutations per population. The theory being that by this time the population has started to settle and the recombination of subsequent generations has produced a range of reasonable objects. During the final stages, the mutation probability drops to just 0.0003, reflecting just 1 mutation per generation.

It is possible to alter the mutation rate during an active session, for example if useful objects have been created early on in the evolutionary process, the user can reduce the mutation probability at this time, to avoid over-disrupting future generations. Conversely, if too many of the objects are looking similar (the user has reached a 'dead end'), by temporarily increasing the mutation rate to a high value, radical changes can occur when producing objects in the next generation. A value of 0.5 will effectively shuffle the next generation's genes, creating a pseudo-random population.

There are two ways of altering the mutation probability mid session: By entering the actual segment probability (as a decimal fraction) or, a more user-friendly approach, by selecting from a menu:

Mutation level (from menu)	Mutation probability (each bit)	*Mutations* per object (mean)	*Mutations* per population (mean)
Randomise	0.5	130	1820
Very high	0.02	5	70
High	0.01	2.5	35
Medium	0.004	1	14
Low	0.002	0.5	7
Very Low	0.0003	0.07	1
Negligible	0.0001	0.036	0.5
No mutation	0	0	0

4.6 Internal Optimisation

This research's primary focus is on user-supplied scoring as the criteria by which objects evolve. As discussed previously, research in the engineering and product design fields tends to use automated objective functions provided by simulation or analytical software, to optimise components, such as flywheels, jet turbine blades, and tables. Measurable properties, such as specific energy density, air resistance & heat flow, and centre of mass are used evaluate objects, and enable a user to sit back and watch the objects evolve to fit these specific, quantitative, requirements.

It was felt that this aspect of evolutionary design should be investigated during this research, its inclusion demonstrating the principle of evolutionary optimisation and the wider scope of the research. Geometric analysis has been selected for several reasons: Firstly demonstrating that the software can be applied to problems like those mentioned above, and secondly to assist the designer if the form specification includes simple concepts such as volume, size, and stability. A further intention was to investigate aesthetic concepts (proportion, unity etc.) analytically, by geometrically evaluating high and low scoring objects and finding patterns using artificial intelligence techniques.

Gaining values for the geometric properties of objects is a relatively straightforward process: The CAD system provides a wide range of mass based geometric analysis including surface area, volume, bounding box, mass, centre of mass, and moments of inertia. Accessing the information is a case of identifying the solid body concerned, providing the tolerance data, and calling the appropriate function. The target values for those properties selected are entered by the user at the beginning of the session; the evaluation criteria included with the software are:

- bounding box volume
- individual x, y and z dimensions
- volume
- surface area

In terms of practicality, the first two criteria are very much dependent on orientation. For example, a thin 85cm long cylinder inclined at 45° to all three axes (creation vector 1,1,1) has the same bounding box volume and dimensions as a 50cm cube. However, during operation, due to the adaptive nature of the search, the most reliable shape at fulfilling the criteria usually prevails (e.g. cuboids for bounding box and multiple dimension optimisation, spheres for volume, surface area and single dimension optimisation).

Fitness Calculation

As mentioned in section 4.4, fitness is calculated by summing the squared differences (comparing a solution's evolved values with the target values). An even weighting between properties (including user ratings) is achieved by normalising the differences before summation.

4.6.1 Geometric Optimisation Examples

The first example, shown in detail in Figures 4.6.1 to 4.6.3, and animated on the CD-ROM, demonstrates optimising to a bounding box of volume $0.1m^3$. The software achieves a solution within 10 generations, accurate to 0.5%. The key stages in this process are described below:

Generation 1 Initial random population

Generation 5 Suitable object type established

Generation 7 Population converging to single blocks

Generation 8 All objects are now blocks except occasional mutations

Generation 11 Very fit object created

Generation 13 Very fit object's genes have proliferated through population

Generation 16 Approximately one mutation per population

Generation 17 Population has converged

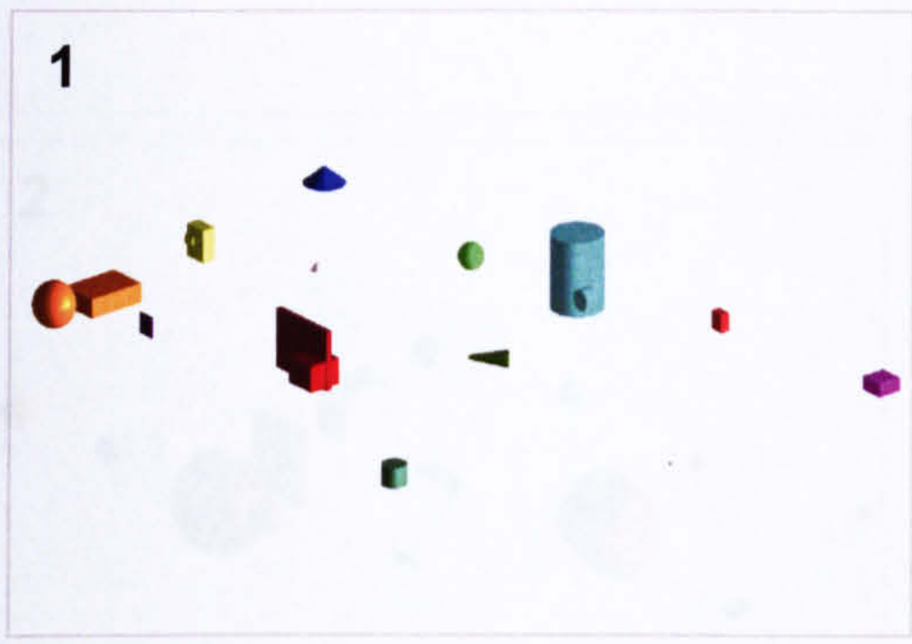
Generation 20 All blocks except one measure $31 \times 47 \times 69 = 0.1005m^3$

target: bounding box, volume of 0.1000m^3

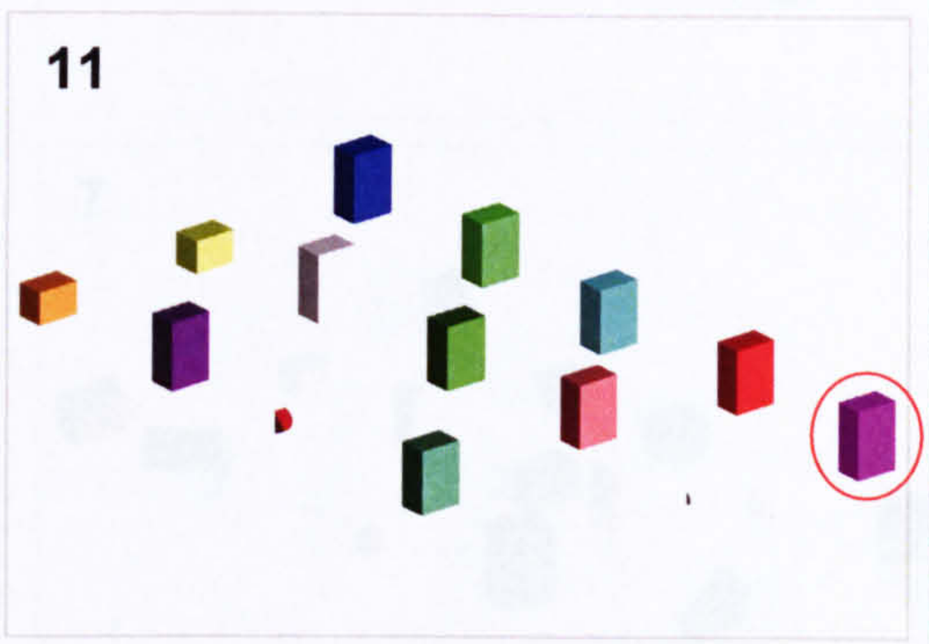
solution: block, volume of 0.1005m^3 (31x47x69cm)

accuracy: (0.1005 - 0.1) ÷ 0.1 = 0.005 = 0.5%

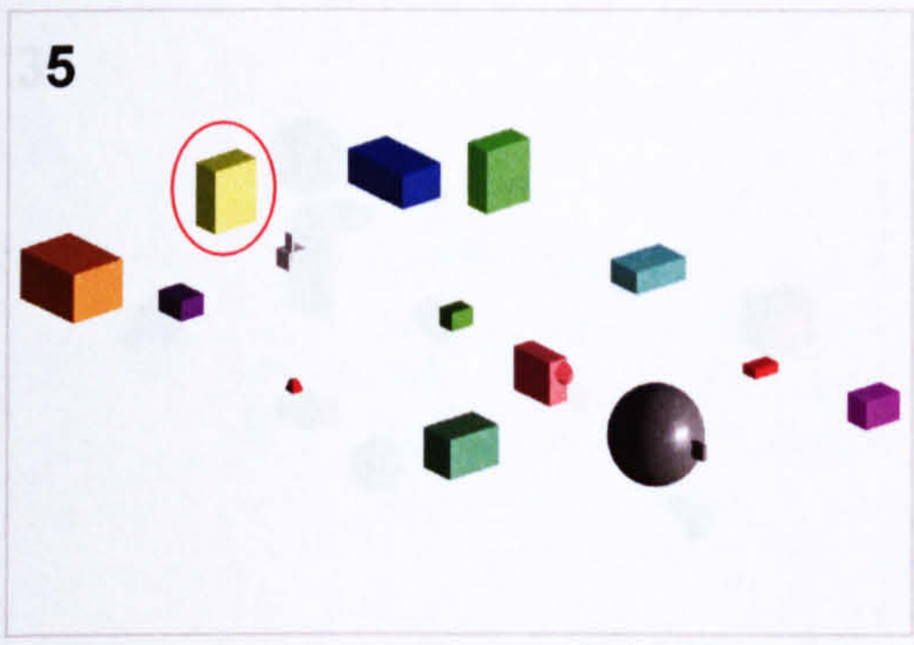
achieved in: 11 generations (population size: 14)



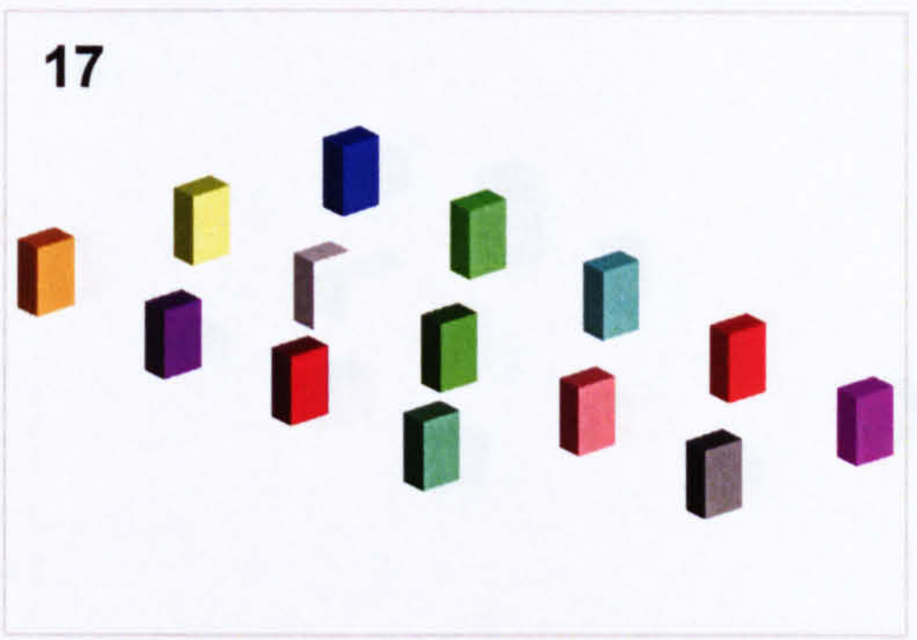
initial population



solution found



object type established



population converged

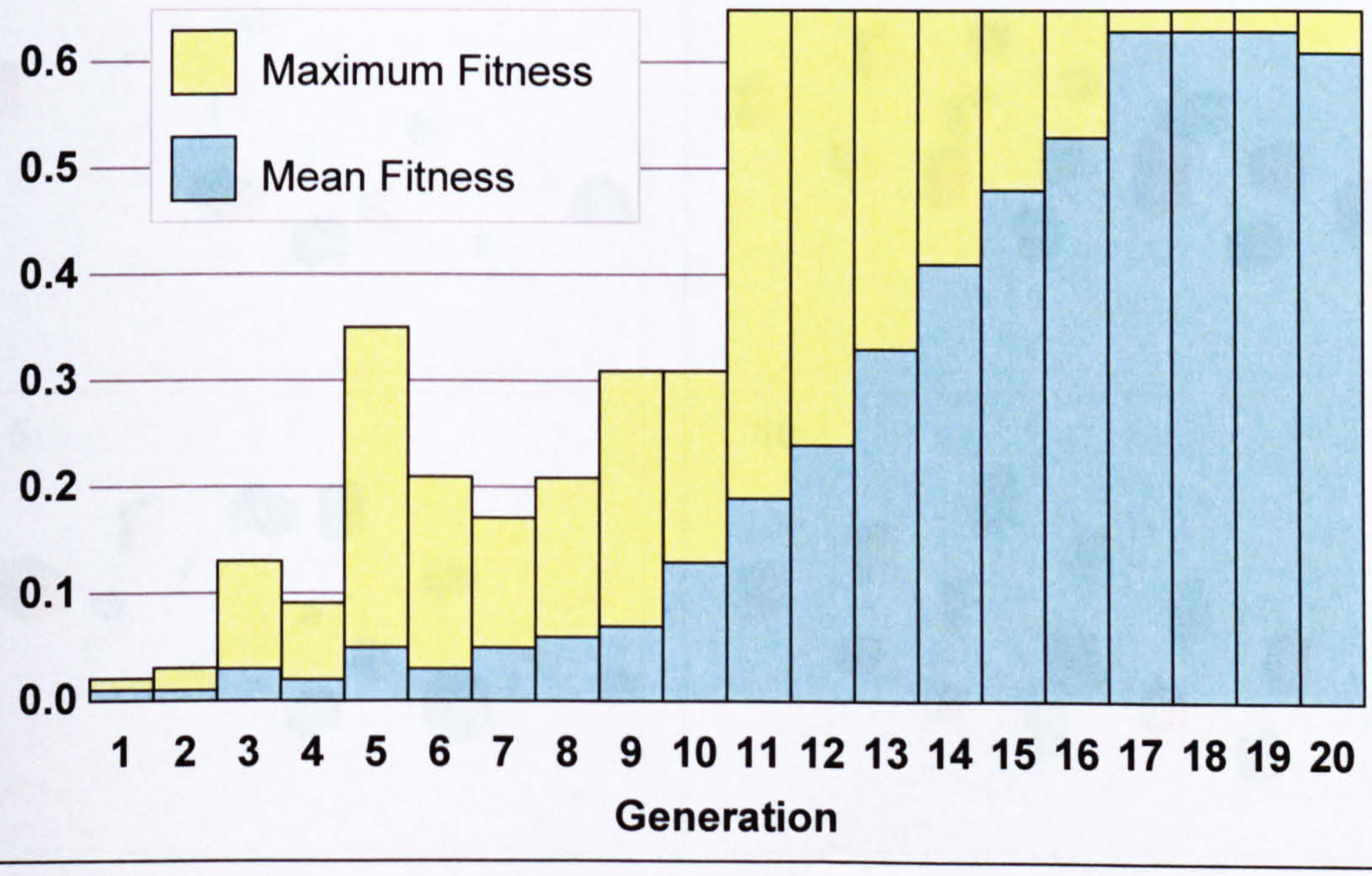
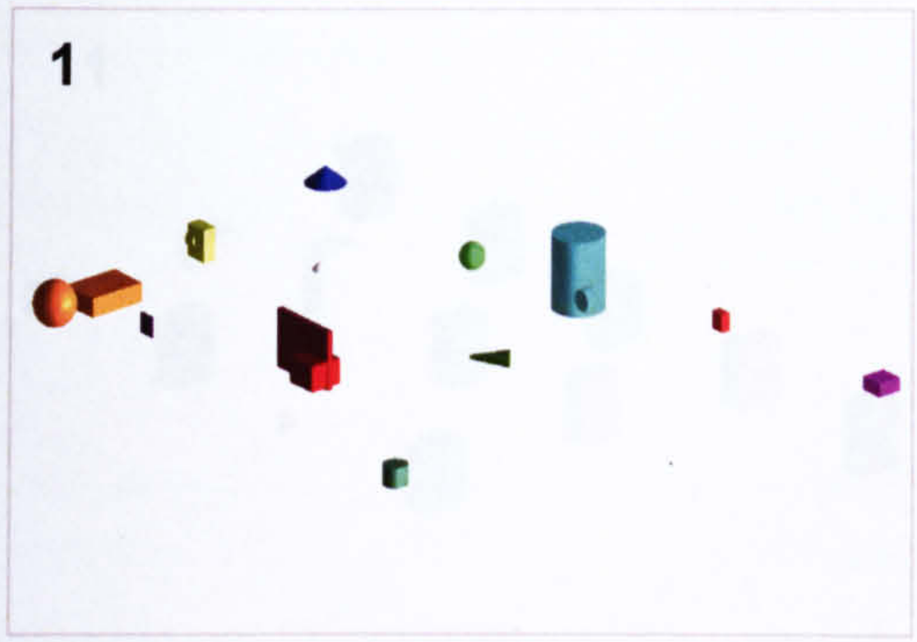
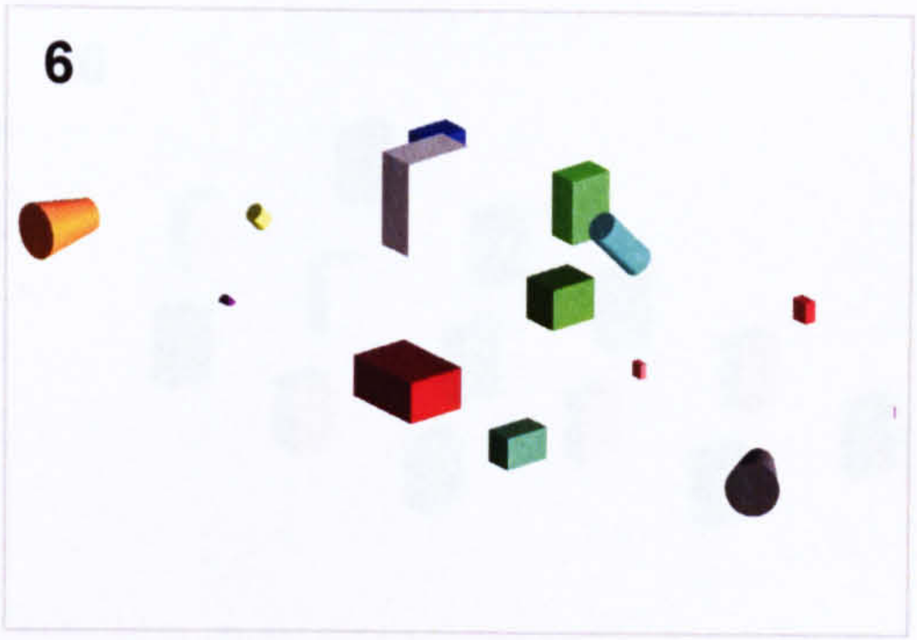
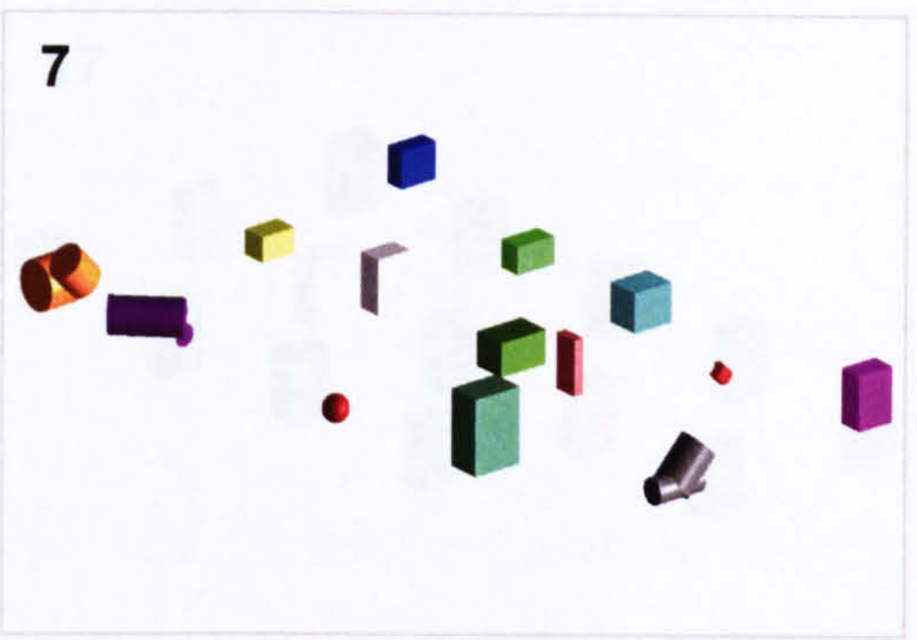
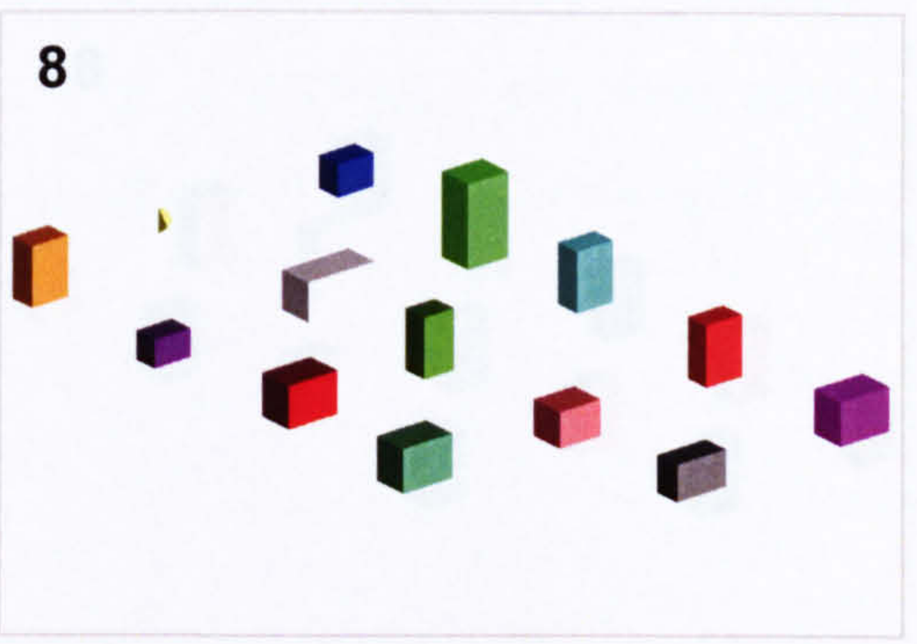


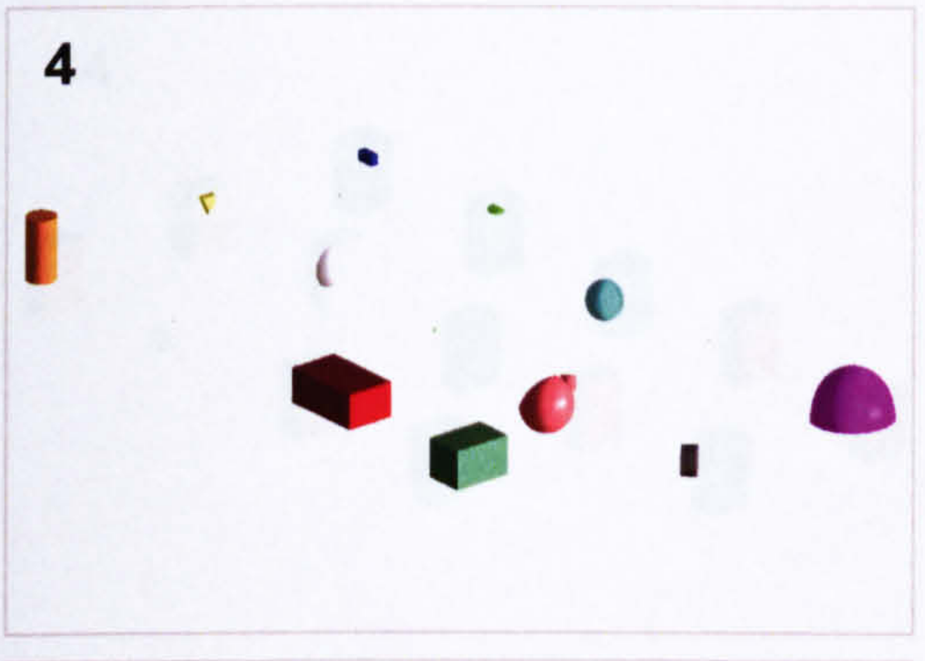
Figure 4.6.1 - Automatic Geometric Optimisation

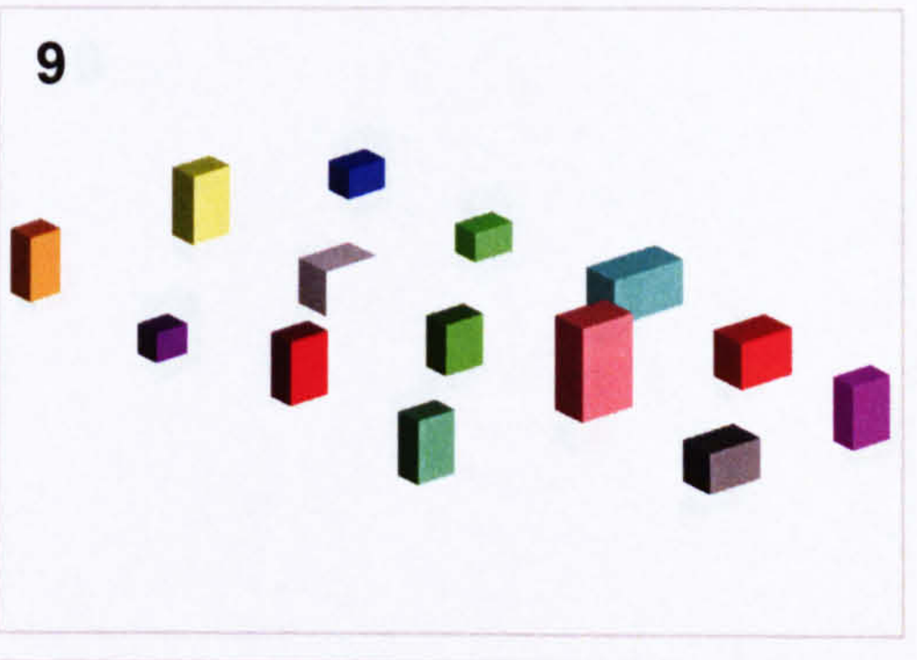


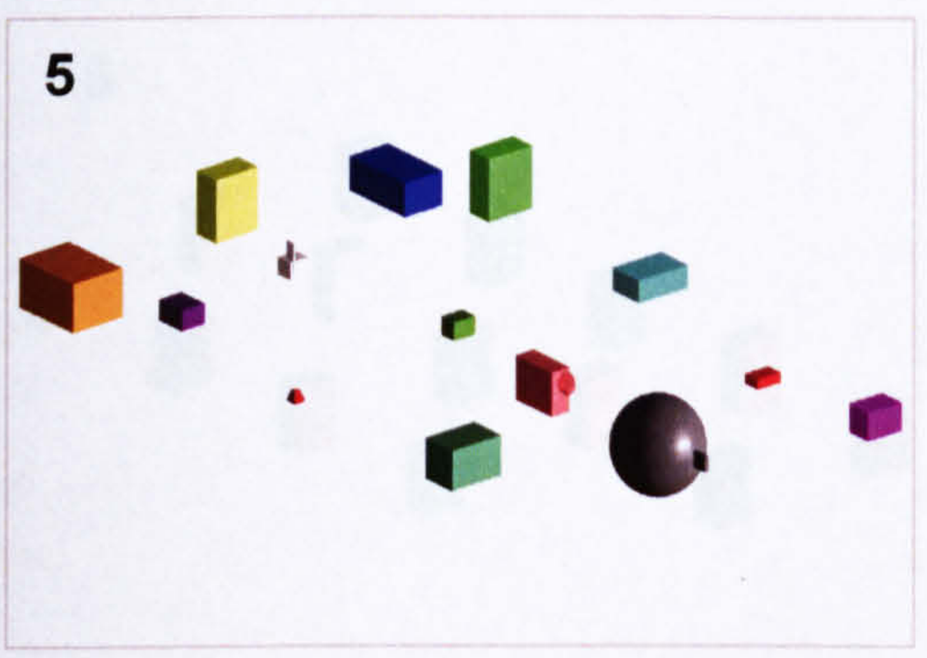












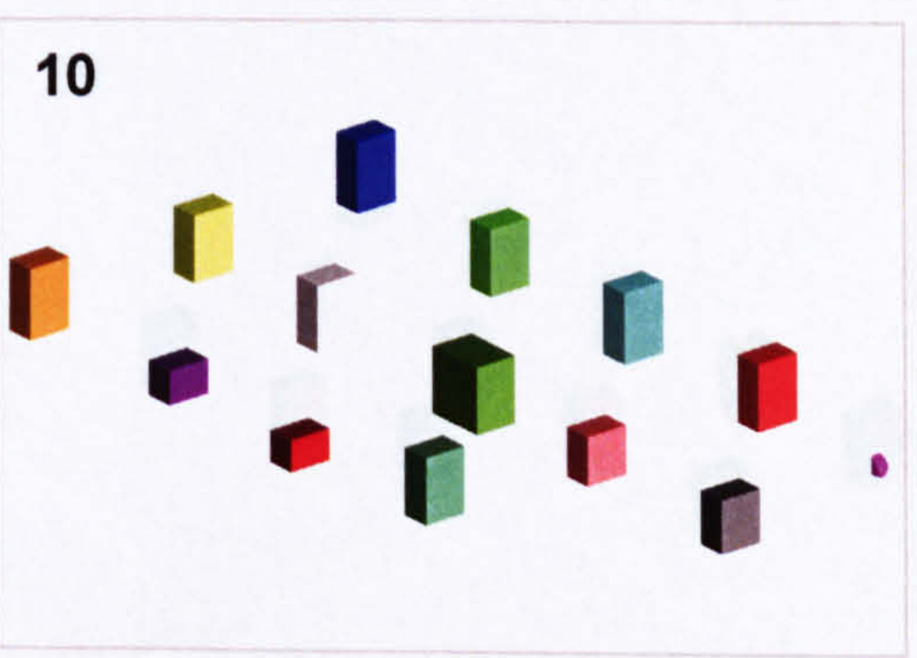
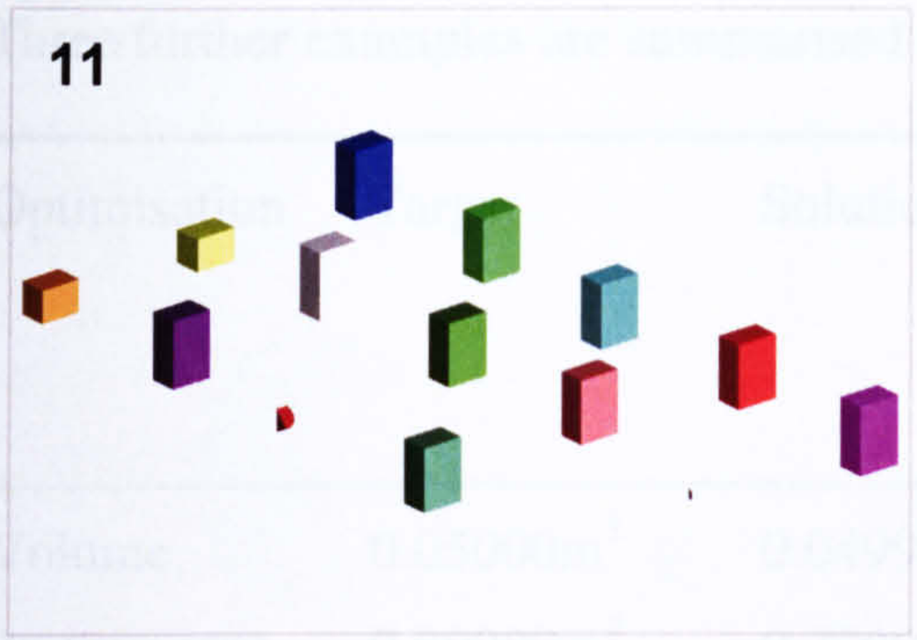
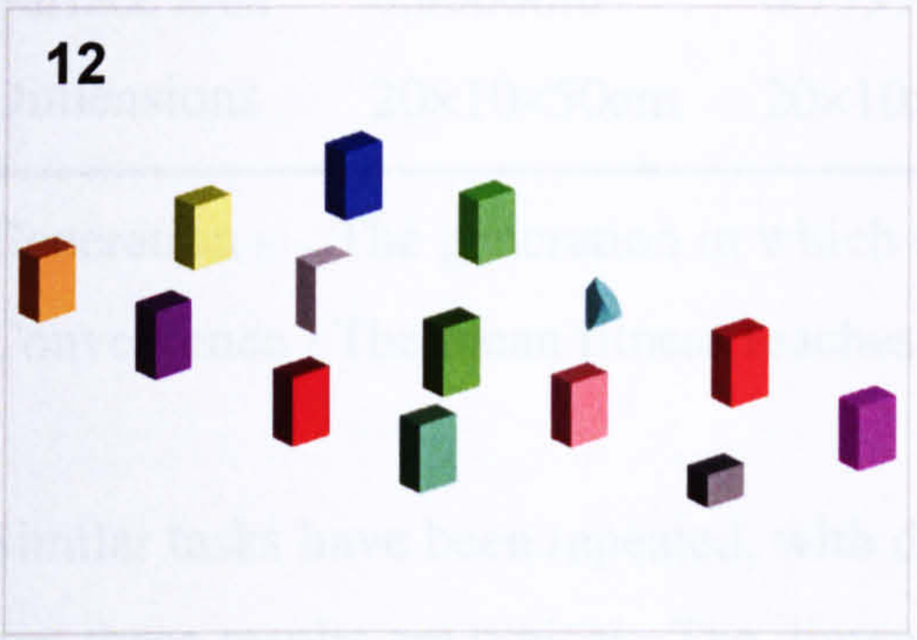
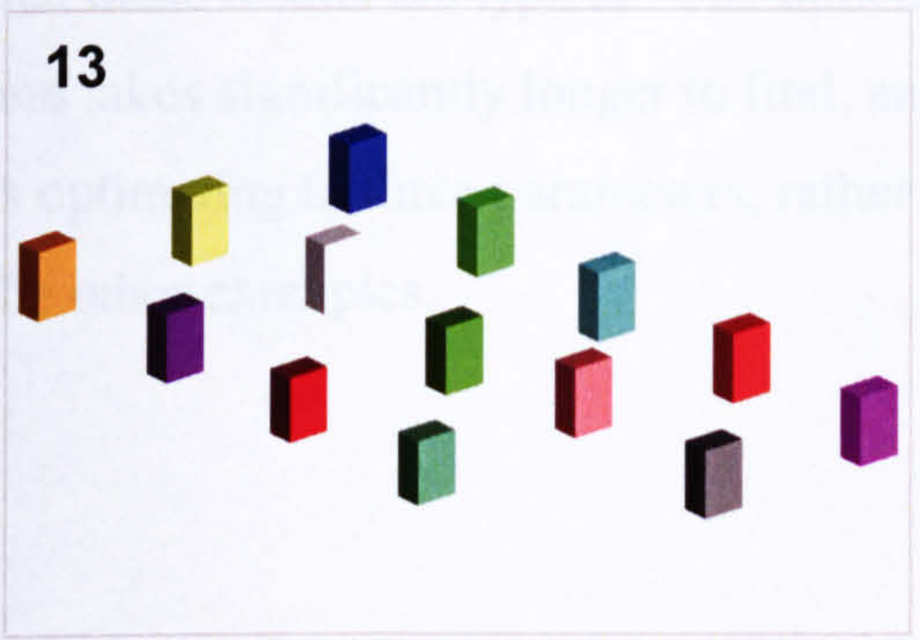
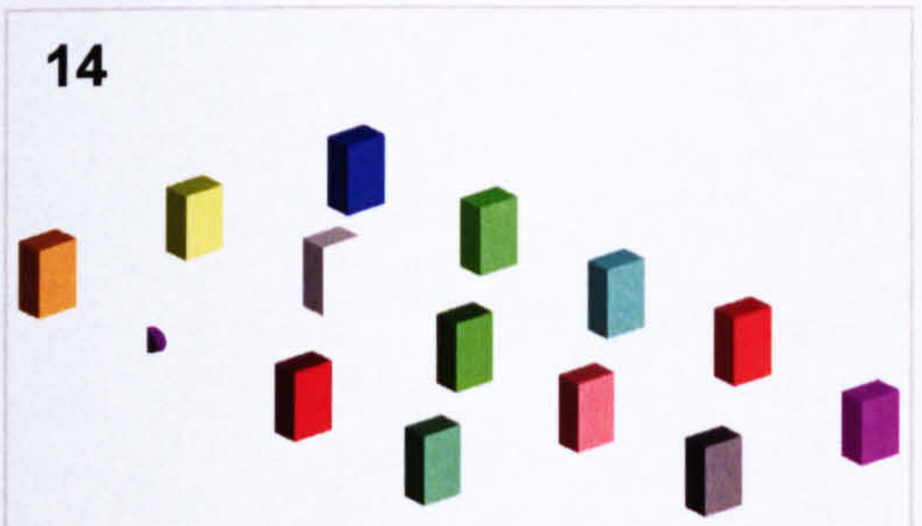


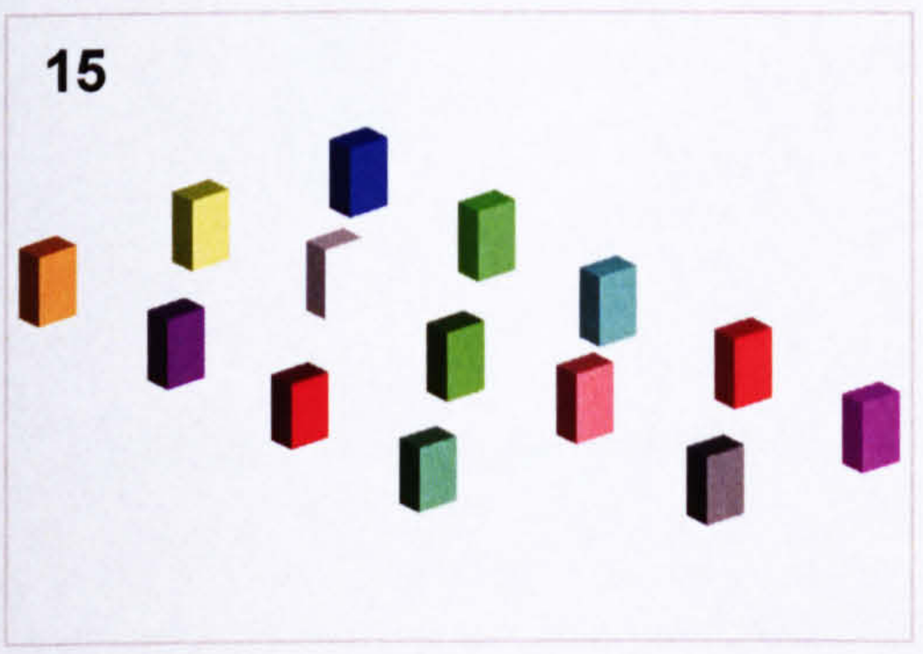
Figure 4.6.2 - Automatic optimisation: Generations 1-10

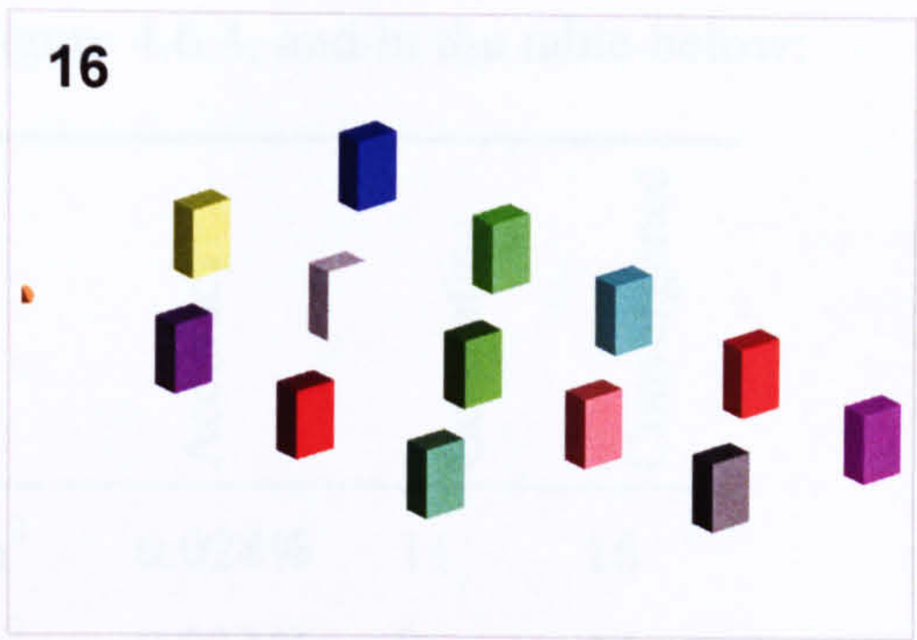


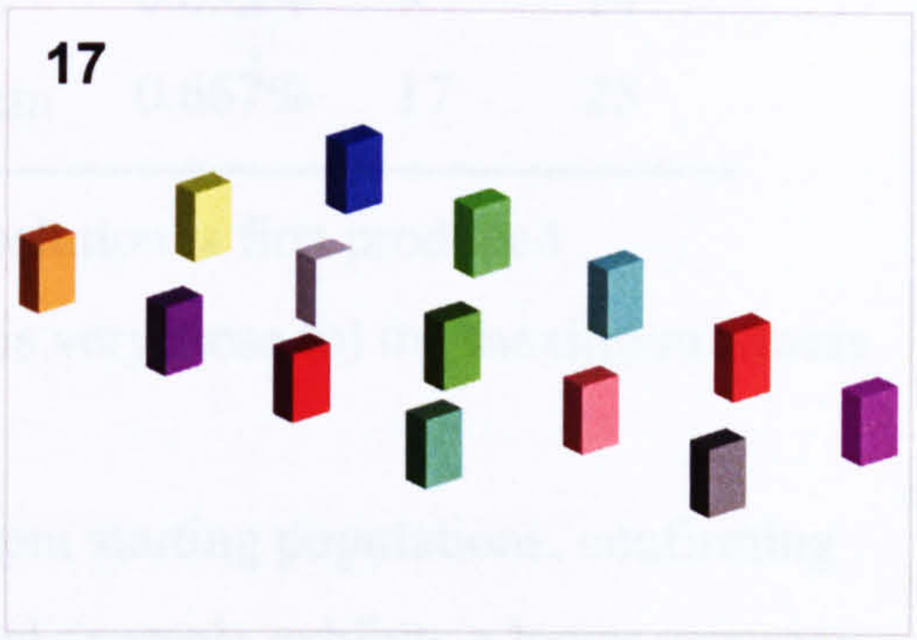


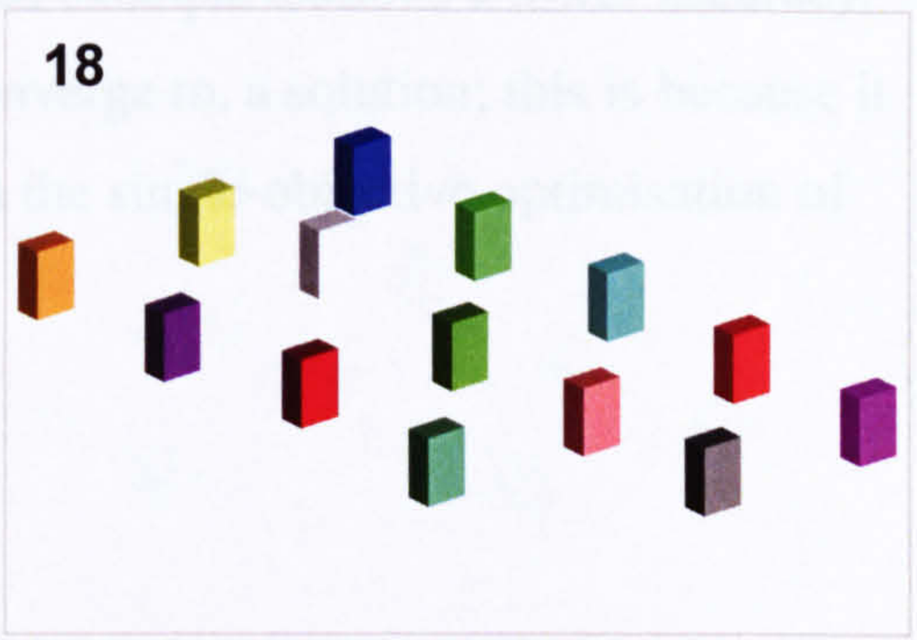












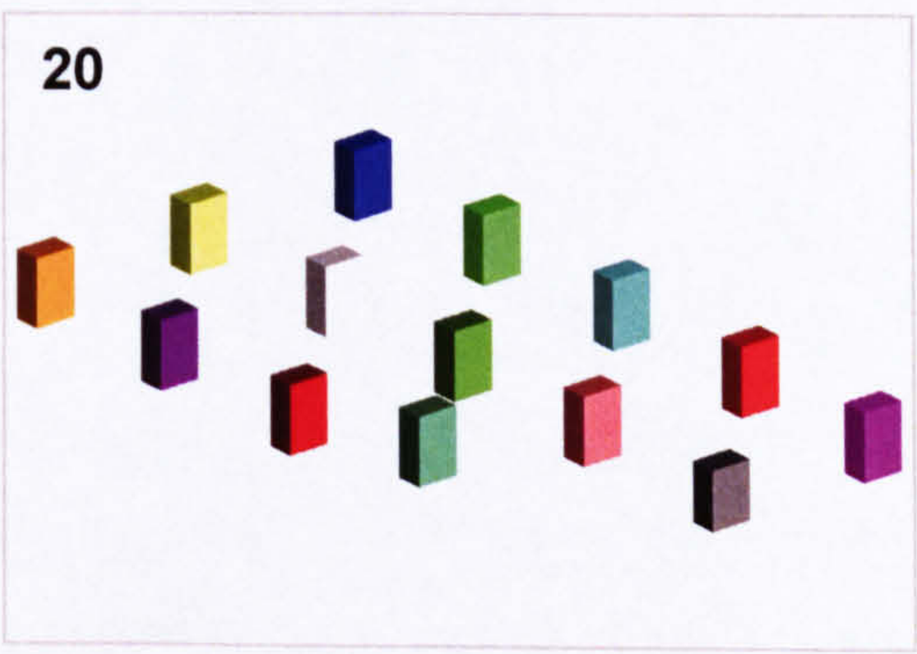


Figure 4.6.3 - Automatic optimisation: Generations 11-20

Three further examples are summarised in Figure 4.6.4, and in the table below:

Optimisation	Target	Solution	Accuracy	Generation	Convergence
Volume	0.05000m^3	0.049986m^3	0.028%	11	16
Surface area	0.80000m^2	0.779974m^2	0.033%	9	14
Dimensions	20×10×50cm	20×10×51cm	0.667%	17	25

Generation - The generation in which the solution is first produced

Convergence - The mean fitness reaches (or is very close to) the maximum fitness

Similar tasks have been repeated, with different starting populations, confirming that these results are typical. The dimensional example exhibits a lower accuracy, and takes significantly longer to find, and converge to, a solution; this is because it is optimising to three parameters, rather than the single-objective optimisation of the other examples.

Volume - Target : 0.05m³

generation	convergence	solution	accuracy
11	16	0.049986	0.028%

Surface Area - Target : 0.8m²

generation	convergence	solution	accuracy
9	14	0.79974	0.033%

Dimensions - Target : 20×10×50cm

generation	convergence	solution	accuracy
17	25	20×10×51	0.667%

Figure 4.6.4 - Three further optimisation examples

4.6.2 Conclusions

The inclusion of geometric analysis functionality has enabled the Genetic Algorithm to cycle through a series of generations without the intervention of the user, akin to more typical evolutionary optimisation. This has enabled observation of several common traits of genetic algorithms, including premature convergence. This can, however, tenuously be interpreted as a positive, in that identifying familiar and expected traits (the small population size of 14, needed for usability of the user-led rating method is the primary reason for the occasional premature convergence[1]) confirms predictable operation of the genetic algorithmic aspect of the programming. It also gives an indication that settings optimised for an interactive approach are not perfectly suited to automated optimisation against geometric properties.

[1] An automated optimisation of this type would typically use a population of 30 or so

4.7 Edge Blending

The union or subtraction of two geometric primitives can produce some aesthetically interesting results [Graham1]. Elliptical, parabolic and hyperbolic curved edges can be produced from these interactions [Anton]. However, it was felt that by enhancing the objects, through the addition of blending, the needs of product design could be better fulfilled. Applying random blend radii to edge lists of complete objects (collections of primitives) produces some highly pleasing results, showing what can be achieved using edge blending [Graham5]. The challenge has been in trying to produce blend radii lists from objects' genotypes and apply them to the objects' edge lists in a consistent and elegant manner, in order that continuity between generations is maintained.

4.7.1 Random Blending

Random blending of complete objects has been used to establish suitable values for blend frequency and radii (within the range of object sizes produced). A list of edges from each body in the population is created, and subjected to blending, with the probabilities and ranges given below.

Proportion	30%	5%	30%	5%	15%	15%
Radii range	1-2mm	2-5mm	5-10mm	10-25mm	25-50mm	50-100mm

Blending enhances the appearance of objects in three distinct ways (Figure 4.7.1): Small blend radii (1-2mm) smooth off sharp edges, increasing the realism of objects. Medium sized blends (5-10mm) round off edges and create fillets where objects join, creating more rounded and integrated looking objects. Finally, and most significantly, large blend radii (25-100mm) dramatically alter the shape of objects, creating more complex curves and producing innovative looking objects that belay their humble geometric primitive-based origins. Some of the objects shown in Chapter 5 (the 'Pelican' sculpture, and the bar seat and sofa designs) rely on large blend radii for their defining characteristics.

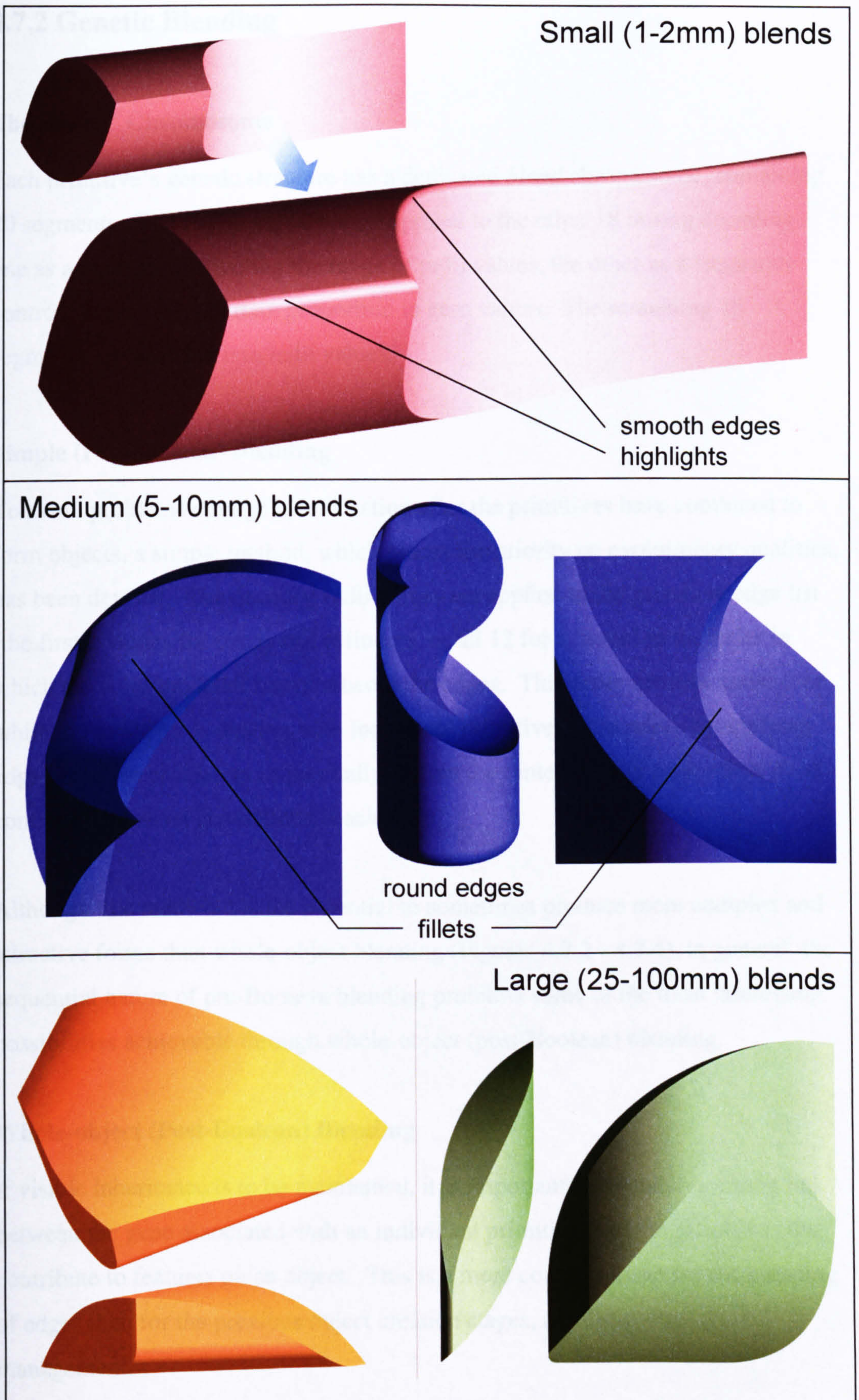


Figure 4.7.1 - Small, medium and large blend radii

4.7.2 Genetic Blending

The 'Blend' Chromosome

Each primitive's genetic structure has a dedicated *blend* chromosome, containing 20 segments. The first 2 segments are applied to the other 18 during decoding, one as a multiplier, affecting the range of radii values, the other as a frequency control, converting a certain proportion to zero values. The remaining 18 segments are decoded into radii values.

Simple (Pre-Boolean) Blending

To avoid problems of edge identification after the primitives have combined to form objects, a simple method, which places the priority on evolutionary qualities, has been devised. The decoded radii values are applied to the primitive edge list (the first 2 values for cones and cylinders, or all 12 for a block) in the order in which the solid modeller has numbered the edges. This order remains consistent while the modeller is dealing with individual primitives. Therefore by applying edge blends to primitives sequentially, before they interact with each other, good continuity between generations is achieved.

Although this method has the potential to sometimes produce more complex and attractive forms than whole-object blending (Figures 4.7.2 - 4.7.4), in general, the sequential nature of pre-Boolean blending prohibits some of the most interesting possibilities achievable through whole-object (post-Boolean) blending.

Whole-object (Post-Boolean) Blending

If visible inheritance is to be maintained, it is important to establish a strong link between the gene associated with an individual primitive, and the primitives that contribute to features on an object. This is a more complex issue for the blending of edges than for the previous object creation stages, requiring careful management.

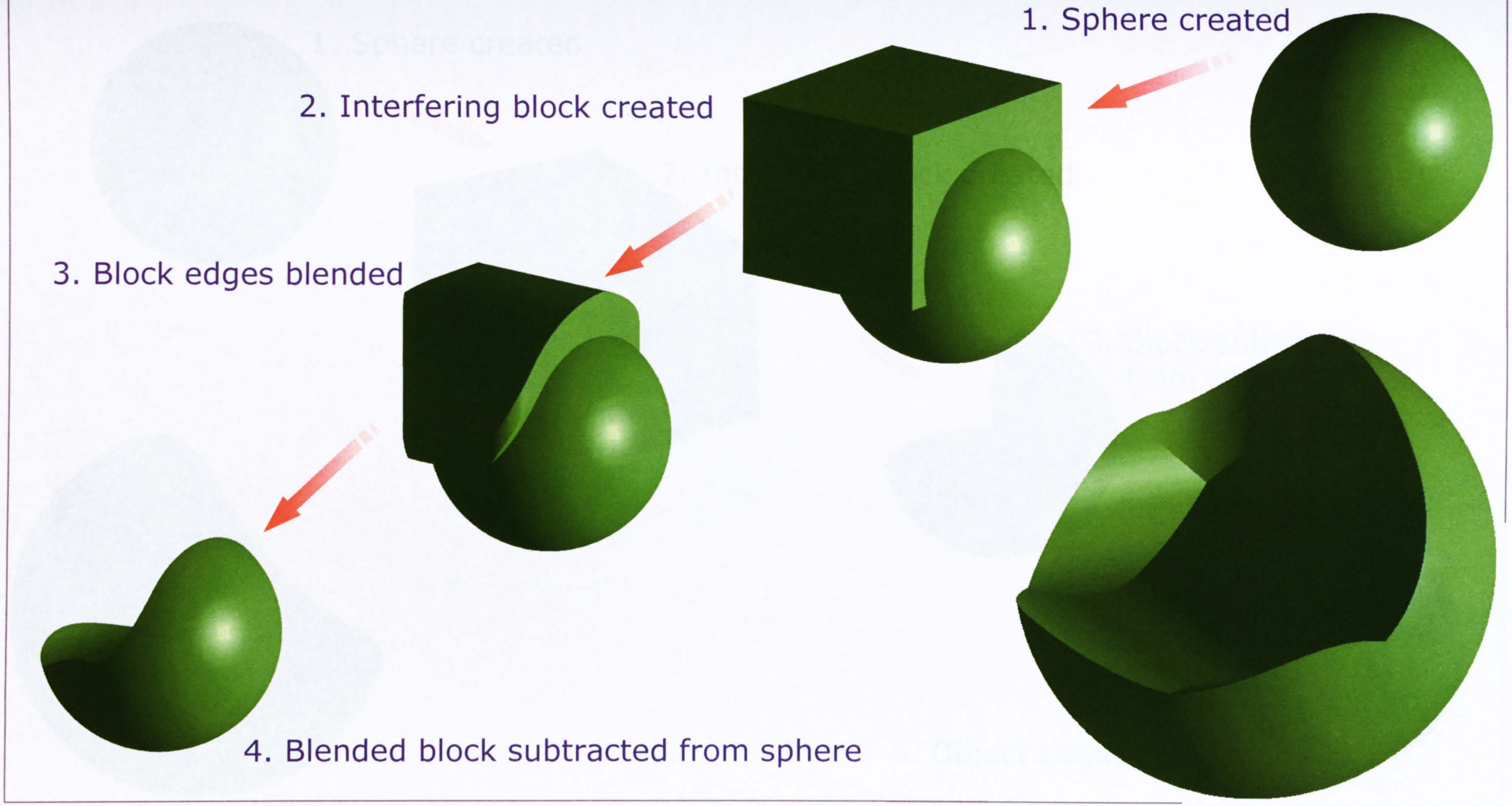


Figure 4.7.2 - Simple (pre-boolean) blending

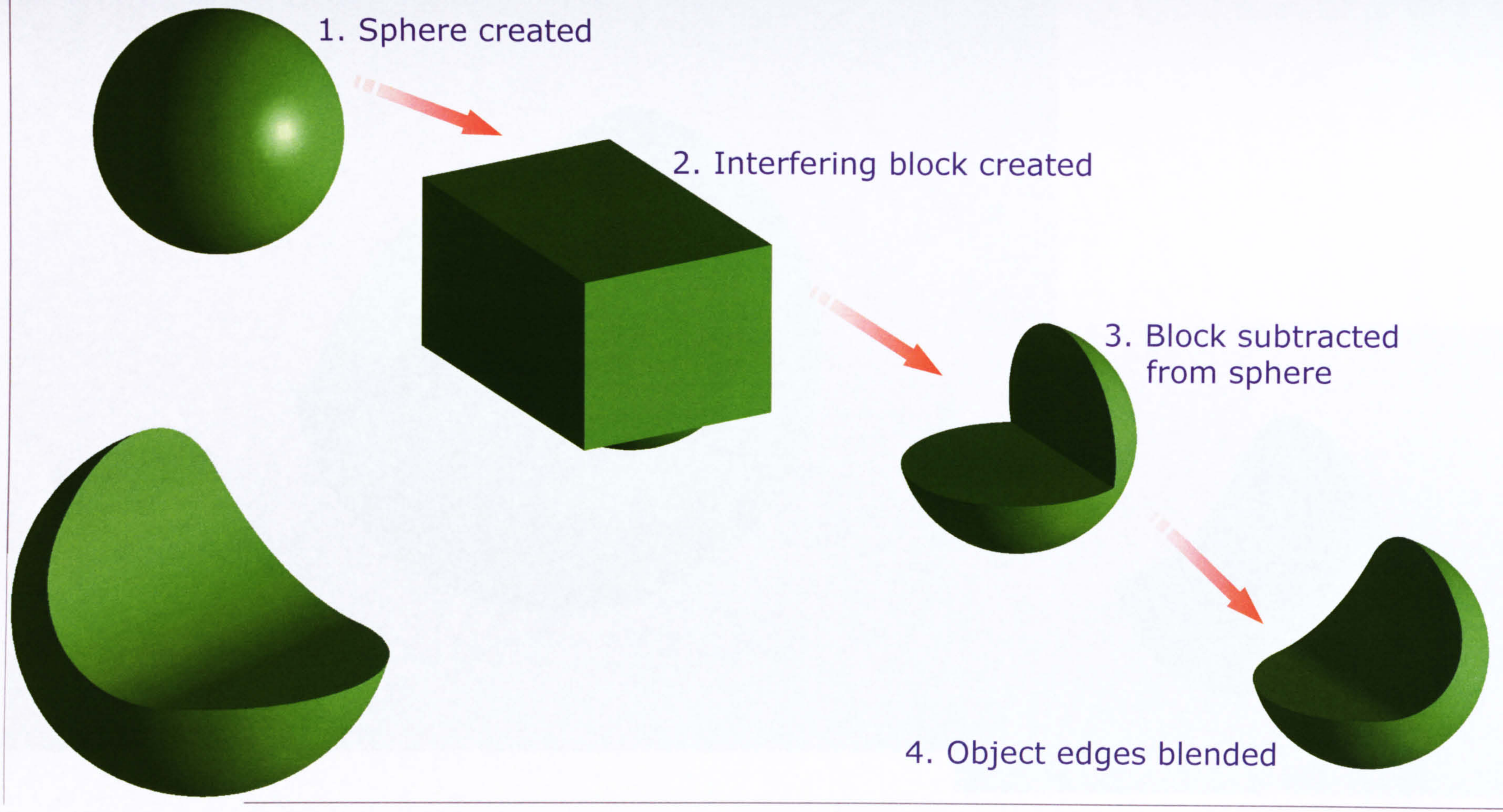


Figure 4.7.3 - Whole-object (post-boolean) blending

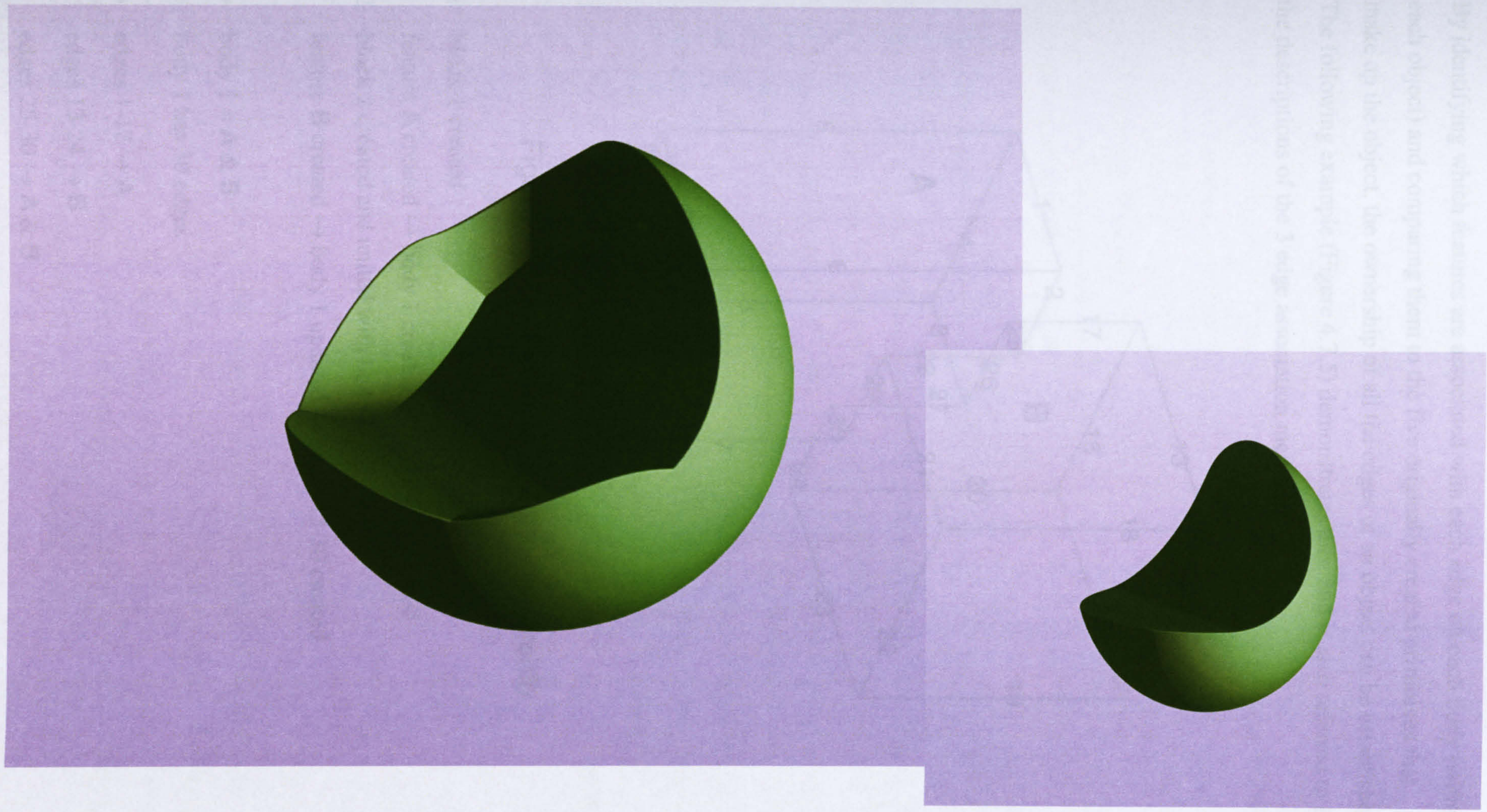


Figure 4.7.4 -Comparing equivalent objects with alternative blending methods

By identifying which features are associated with each edge on each body (within each object) and comparing them to the five originally created primitives that make up the object, the ownership of all the edges of an object can be established. The following example (Figure 4.7.5) demonstrates this, and is also referred to in the descriptions of the 3 edge association methods.

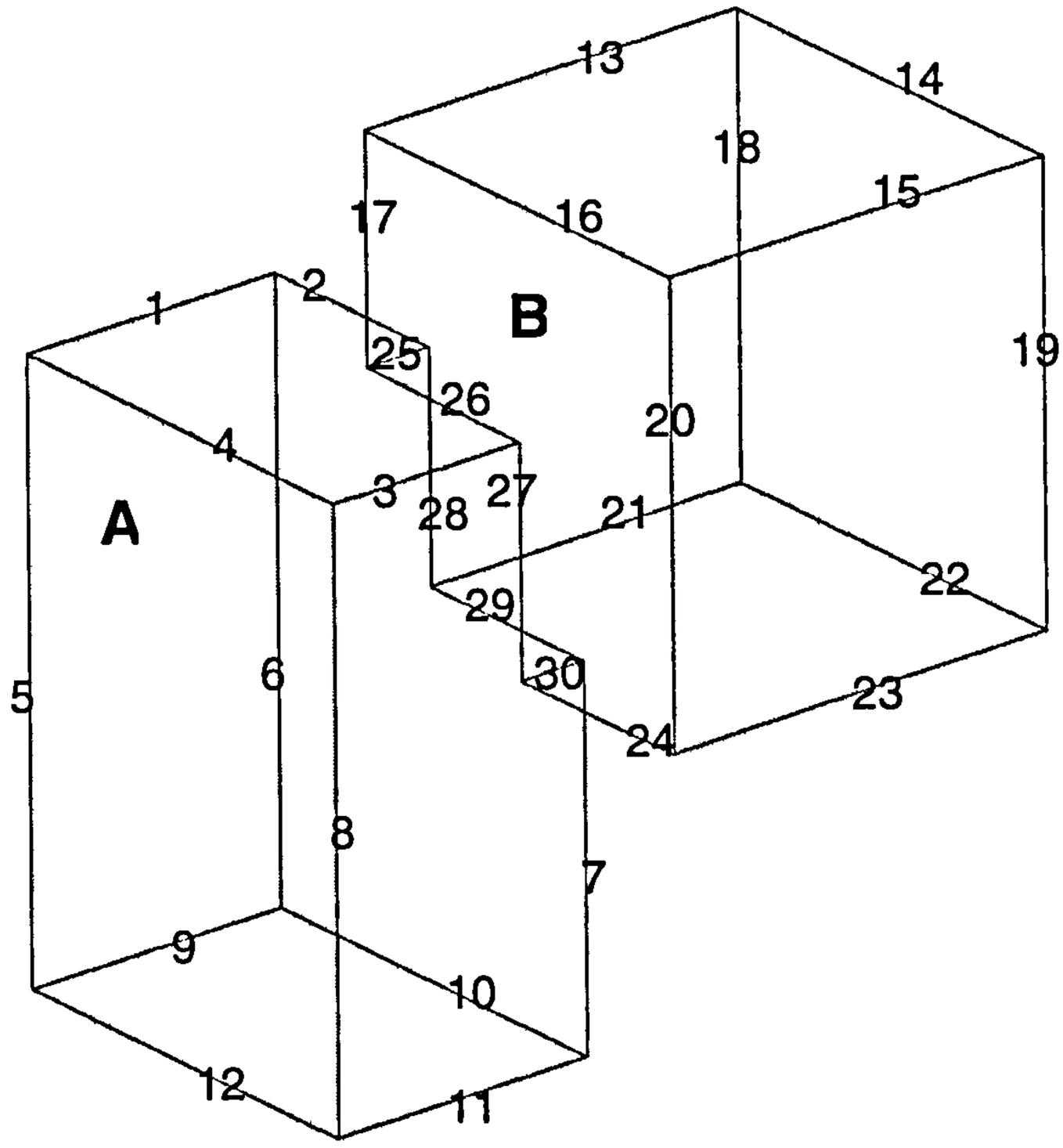


Figure 4.7.5 – Example of edge ownership

1. block 1 created

   feature **A** created → body 1 created → edges 1-12 created

2. block 2 created and united with body 1

   feature **B** created → body 1 updated → edges 13-30 created

- body 1 = **A** & **B**
- body 1 has 30 edges
- edges 1-12 → **A**

  edges 13-24 → **B**

  edges 25-30 → **A** & **B**

4.7.3 Shared Edges

When new bodies are formed through Boolean interaction, new edges are created that are associated with multiple primitives. Blend values for edges that are unaffected, that is to say, edges owned by just one primitive, are taken from the corresponding chromosome. A number of alternative methods have been devised to deal with shared edges (Figure 4.7.6).

Order Hierarchy Method

Contributing primitive	**A**	**B**
Edges	1 – 12 25 – 30	13 – 24

Data is allocated by primitive association in strict creation-order hierarchy, shared edges receiving no special treatment. So *all* edges associated with primitive **A** take blend radii from primitive **A**'s blend value list (irrespective of whether the edge is shared by another primitive). Any *remaining* edges associated with primitive **B** use primitive **B**'s values, and so on. This method is straightforward to implement, but values from early primitives' blend lists are used up quickly (and sometimes have to be re-used). Many later-created (3^{rd}, 4^{th} or 5^{th}) primitives' values are therefore often not used. Consequently the first primitive's influence becomes dominant, especially in cohesive (non-fragmented) objects.

Mean Value Method

Contributing primitive	**A**	**B**
Edges	1 – 12 25 – 30	13 – 24 25 – 30

Shared edges use a mean radius calculated from the 2 appropriate values from *each* of the two owning primitives' list. Both primitives share equal influence, but more blend values are used up in the process. More importantly though, large and small radii are normally lost in the averaging, significantly limiting the range of interesting shapes produced.

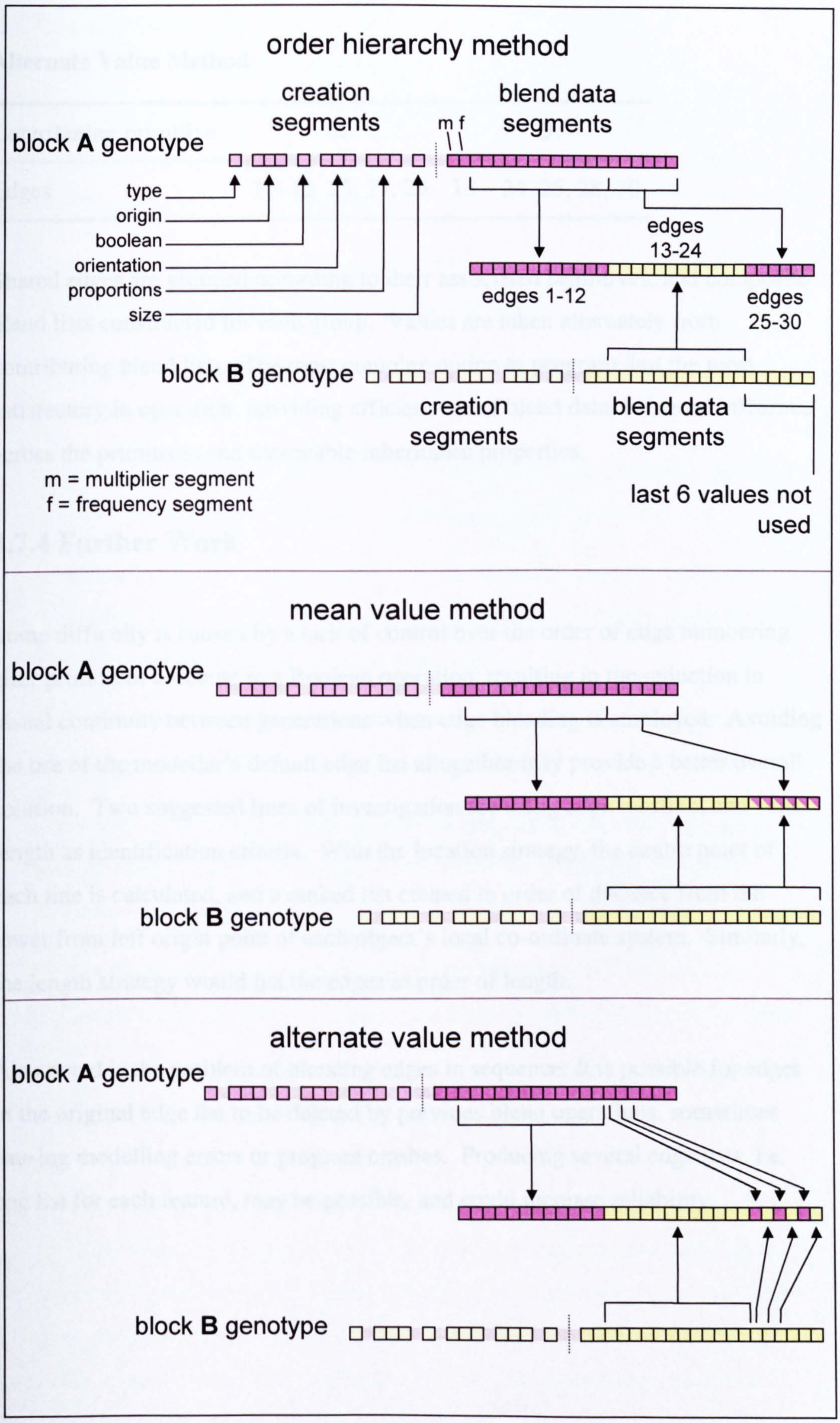


Figure 4.7.6 - Edge association methods

Alternate Value Method

Contributing primitive	A		B	
Edges	1 – 12	25, 27, 29	13 – 24	26, 28, 30

Shared edges are grouped according to their associated primitives, and composite blend lists constructed for each group. Values are taken alternately from contributing blend lists. The most complex option to program, but the most satisfactory in operation, providing efficient use of blend data, balanced influence across the primitives and reasonable inheritance properties.

4.7.4 Further Work

Some difficulty is caused by a lack of control over the order of edge numbering after primitives combine in a Boolean operation, resulting in the reduction in visual continuity between generations when edge blending is employed. Avoiding the use of the modeller's default edge list altogether may provide a better overall solution. Two suggested lines of investigation are using edge location, and edge length as identification criteria. With the location strategy, the centre point of each line is calculated, and a ranked list created in order of distance from the lower front left origin point of each object's local co-ordinate system. Similarly, the length strategy would list the edges in order of length.

Also noted is the problem of blending edges in sequence: It is possible for edges in the original edge list to be deleted by previous blend operations, sometimes causing modelling errors or program crashes. Producing several edge lists, i.e. one list for each feature, may be possible, and could increase reliability.

4.8 Conclusions

A prototype computer aided evolutionary design system has been developed, in order to explore and realise the aims and objectives of the research. This has involved the key aspects listed below:

- Establishing an efficient product representation, in terms of genotype (genetic data structure) and phenotype (3D geometric object).
- Developing effective methods of creating objects through Boolean interaction of geometric primitives.
- Devising a technique for grouping simple phenotypes (primitives) together to create complex solutions (objects), according to various tactics.
- Adapting standard fitness and objective functions for use with interactive evaluation, negating the need for specific multi-objective techniques through the intelligent treatment of measurement units.
- Preliminary research into automatic aesthetic assessment, through fitness penalties (reducing the fitness of objects with non-attached primitives).
- Investigating the effects of established genetic operators; adopting appropriate selection and crossover techniques, and establishing effective mutation profiles.
- Incorporating the capacity for automatic optimisation, using the volume, size and surface area geometric analysis tools from the underlying solid modeller.
- Using edge-blending techniques to increasing the aesthetic potential of evolved objects without loss of visual inheritance properties.

Figure 4.8.1 demonstrates the inheritance properties achieved early in the research (before edge blending was introduced). Evolution of objects with blending is demonstrated in the interactive ancestor diagram contained on the enclosed CD-ROM, in the detailed recombination example in Appendix B, and during the following chapter, where two applications of the EFD system are described.

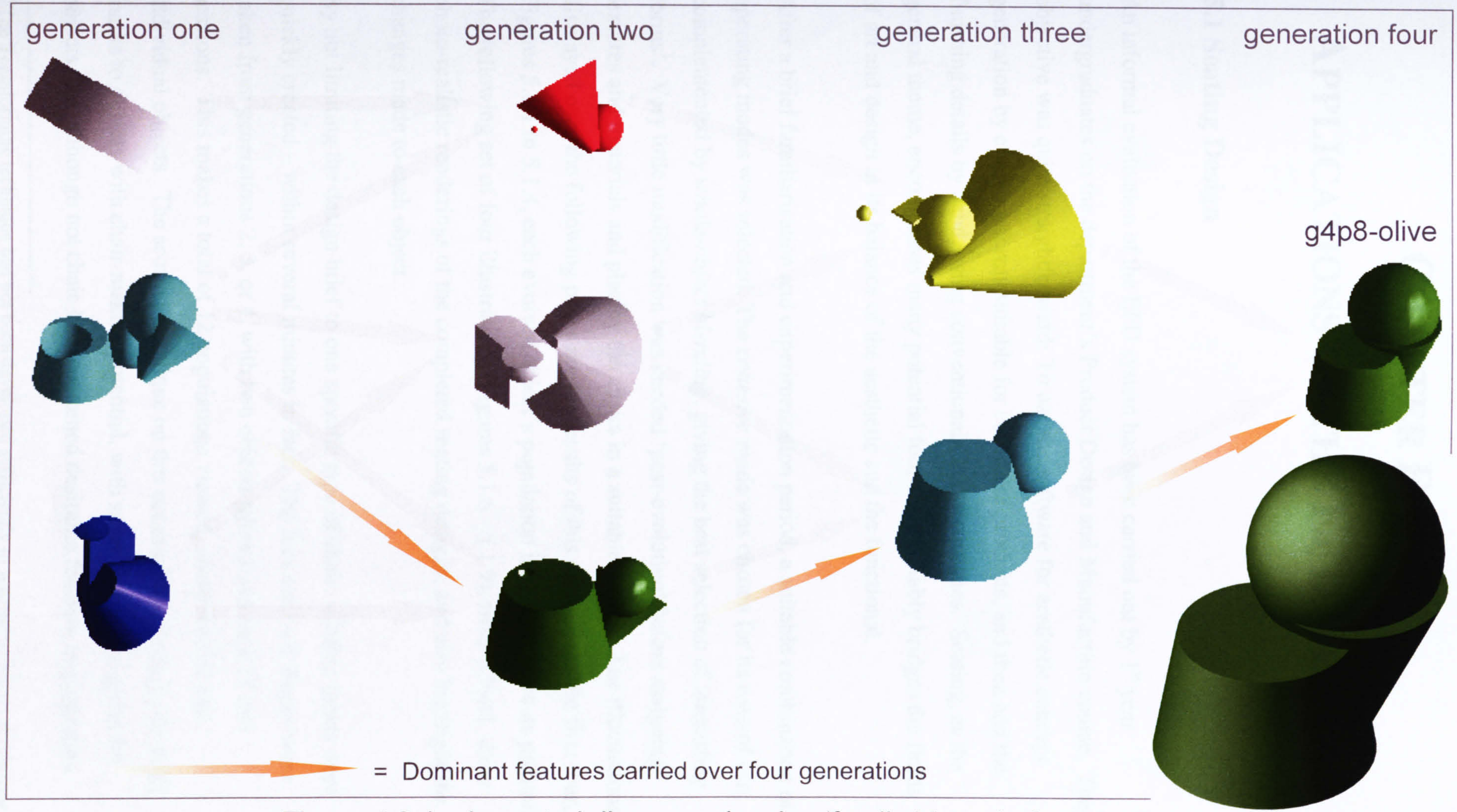


Figure 4.8.1 - Ancestral diagram, showing ‘family history’ of g4p8-olive

CHAPTER FIVE

APPLICATIONS OF THE EFD SYSTEM

5.1 Seating Design

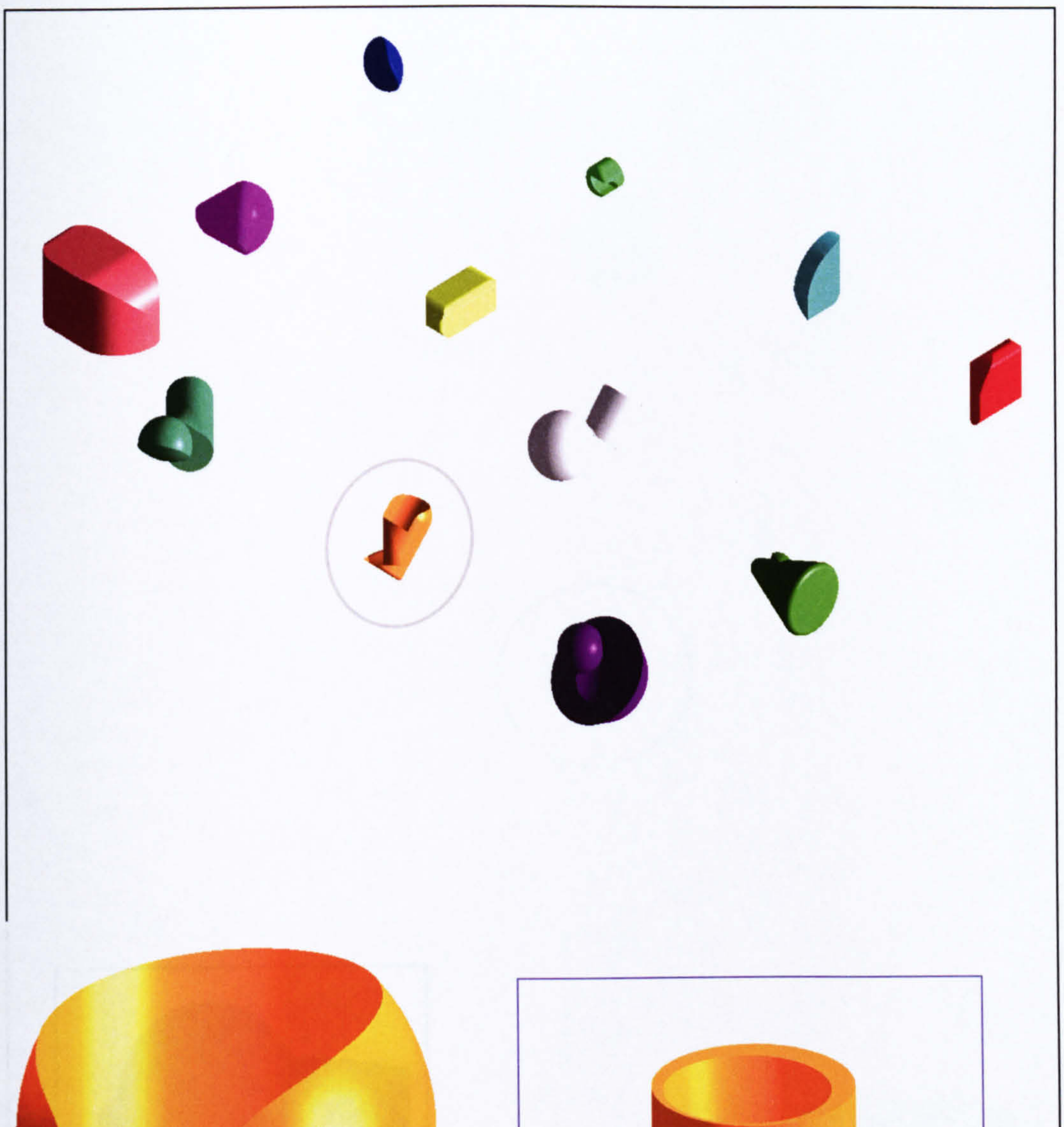
An informal evaluation of the EFD system has been carried out by 1st year undergraduates on the department's Product Design and Manufacture course. The objective was quite straightforward: To use the software for aesthetic concept generation by evolving forms suitable for consumer products, and then add the finishing details by hand, using conventional CAD techniques. Seating, as the general theme, encompasses many potential forms and suitably bridges the fields of art and design in its balance of the aesthetic and the functional.

After a brief familiarisation and experimentation period, a suitable combination of operating modes was selected: The *cohesive* mode was chosen for its ease of use, complemented by *whole-object blending*, giving the best selection of interesting forms[1]. Very little modification was needed 'post-evolution' before assigning textures and materials and placing the chairs in a suitable setting. The illustrations displayed over the following pages are the results of this exercise: In the first four, Figures 5.1.1 to 5.1.4, each evolved object's population is shown above its picture. The following set of four illustrations (Figures 5.1.6 - 5.1.9), from [Case], show photo-realistic renderings of the completed seating designs, and also highlight the changes made to each object.

By not limiting the design brief to one specific type of chair, suitable forms were quickly created – within several minutes in fact. The four evolved shapes were taken from generations 2, 3, or 4, with two objects taken from each of two sessions. This makes a total of 12 populations viewed, comprised of 144 individual objects. The scoring technique on this occasion involved giving high marks to objects with chair-making potential, with some marks also given for objects that, although not chair-like, contained desirable features or properties.

---

[1] The Teamforming technique had not been developed sufficiently to warrant inclusion at this time



Evolved object with all blends suppressed

Figure 5.1.1 - Evolved 'Bar Seat' object with associated 3rd generation population

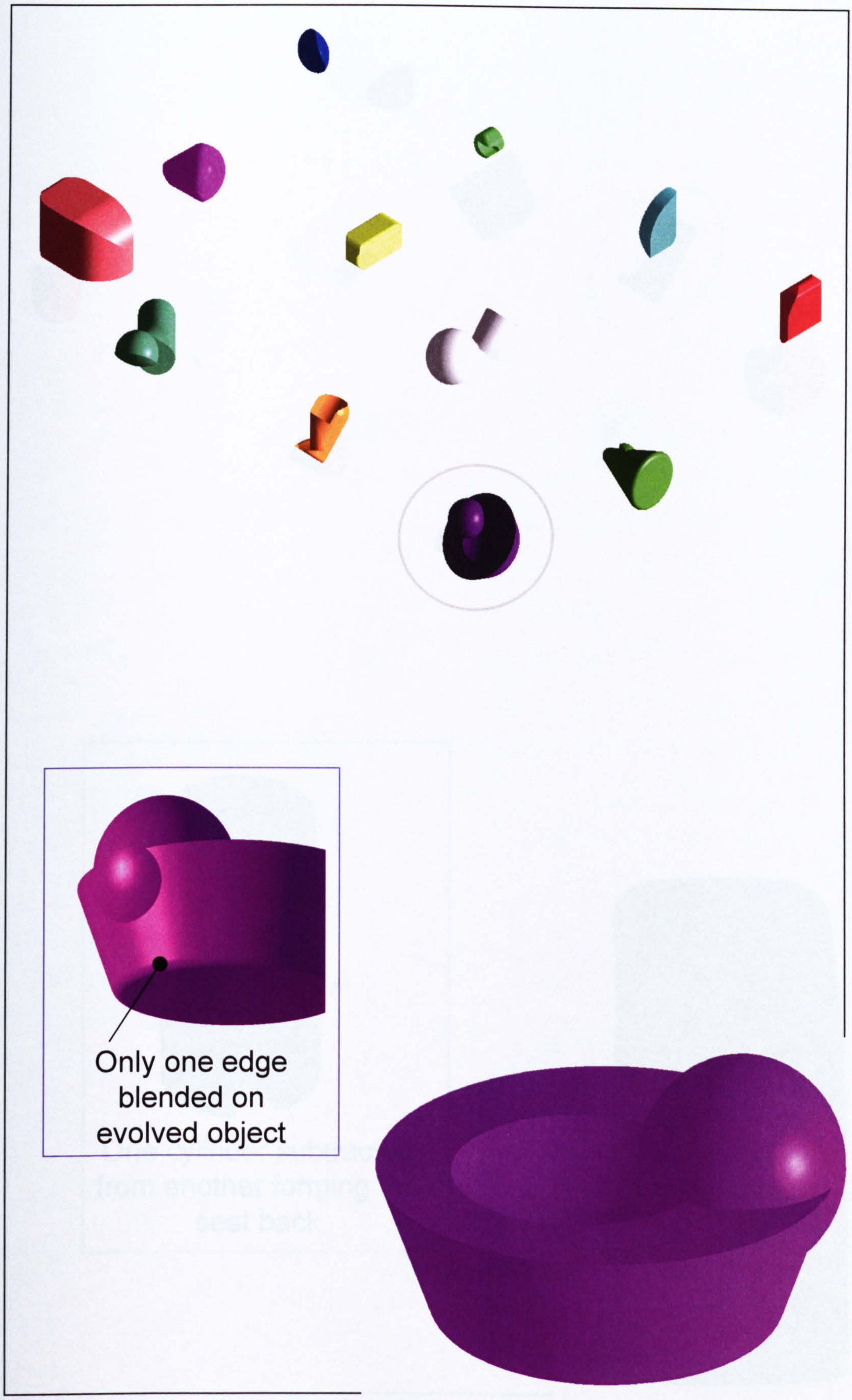
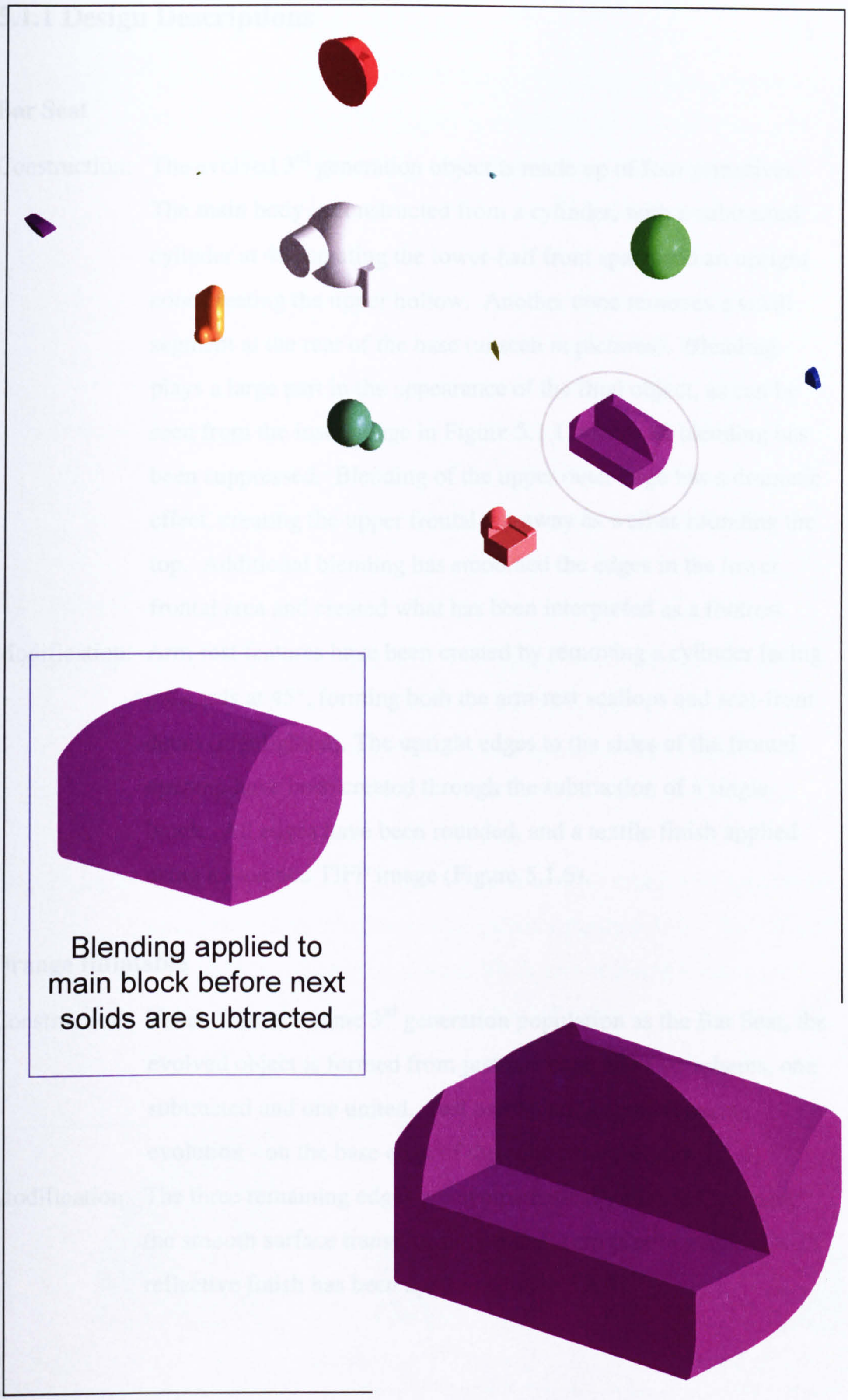


Figure 5.1.2 - Evolved ‘Orange Inflatable’ object with associated 3rd generation population

One cylinder subtracted from another forming the seat back

Figure 5.1.3 - Evolved 'Bond Villain Chair' object with associated 4th generation population



Blending applied to main block before next solids are subtracted

Figure 5.1.4 - Evolved 'Bad Taste Sofa' object with associated 2nd generation population

5.1.1 Design Descriptions

Bar Seat

Construction: The evolved 3rd generation object is made up of four primitives. The main body is constructed from a cylinder, with a subtracted cylinder at 45° creating the lower-half front space and an upright cone creating the upper hollow. Another cone removes a small segment at the rear of the base (unseen in pictures). Blending plays a large part in the appearance of the final object, as can be seen from the inset image in Figure 5.1.1, where all blending has been suppressed. Blending of the upper outer edge has a dramatic effect, creating the upper frontal cut-away as well as rounding the top. Additional blending has smoothed the edges in the lower frontal area and created what has been interpreted as a footrest.

Modification: Arm-rest features have been created by removing a cylinder facing outwards at 45°, forming both the arm-rest scallops and seat-front detail (highlighted). The upright edges to the sides of the frontal opening have been created through the subtraction of a single block. All edges have been rounded, and a textile finish applied using a wrapped TIFF image (Figure 5.1.6).

Orange Inflatable

Construction: Taken from the same 3rd generation population as the Bar Seat, the evolved object is formed from just one cone and two spheres, one subtracted and one united. Just one blend is applied during evolution - on the base edge of the cone (inset, Figure 5.1.2).

Modification: The three remaining edges are sympathetically blended, creating the smooth surface transitions. A translucent plastic material with reflective finish has been applied (Figure 5.1.7).

Bond Villain Chair

Construction: The original object was taken from a fourth generation population. The back section of the object is formed from two cylinders, one created, and one much larger cylinder subtracted (inset, Figure 5.1.3). A third cylinder is joined at the base, forming the seat, and finally a large block is subtracted from the lower portion of the object, creating the flush lower surface. The remaining, rather intrusive portion of the original created cylinder at the base of the object has been reduced significantly with blending.

Modification: The only post evolution addition is the chrome base. Routine edge smoothing and the selection of a black plastic material with a fine granular texture add to the realism of the chair (Figure 5.1.8).

Bad Taste Sofa

Construction: Of the four chairs, this is the only object created using 'simple' blending: The evolved object is taken from a 2nd generation population, and is dominated by a single block with large radius blends forming the main curves (Figure 5.1.4). A second block and a small cone are then subtracted.

Modification: The addition of two cushions, rounding of all edges and application of a wrapped textile image add realism (Figure 5.1.9). The left arm is created by subtracting a small block and applying a blend to the inner, upper edge, as shown from the sequence of images shown below in Figure 5.1.5.

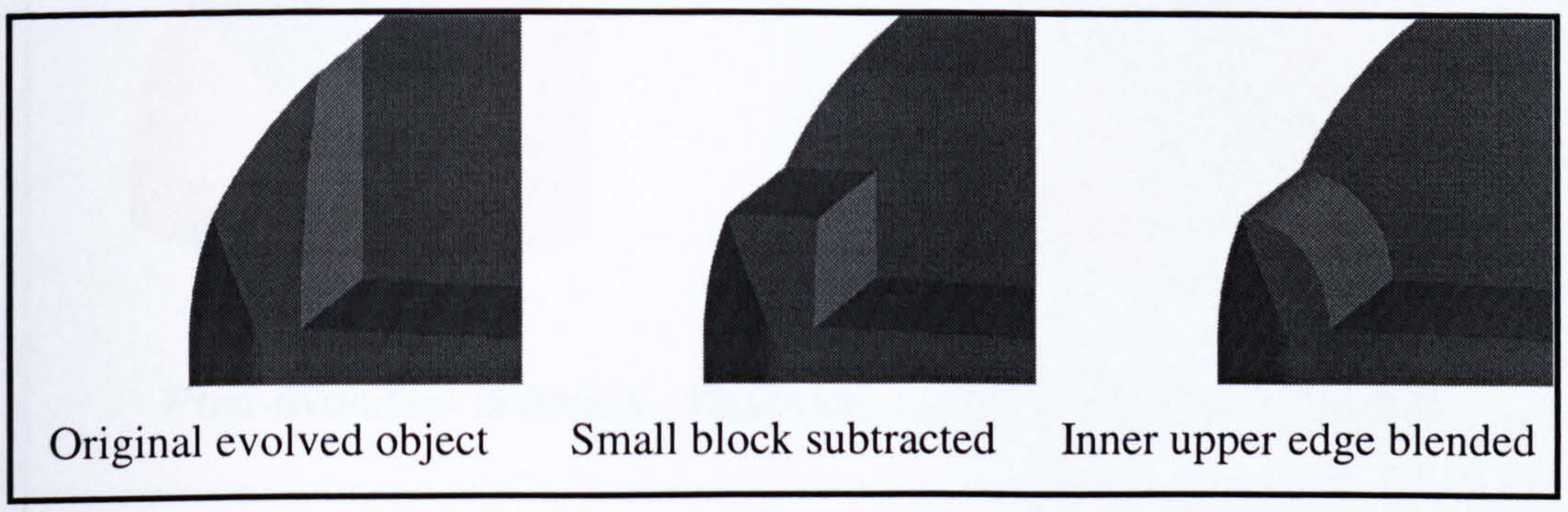
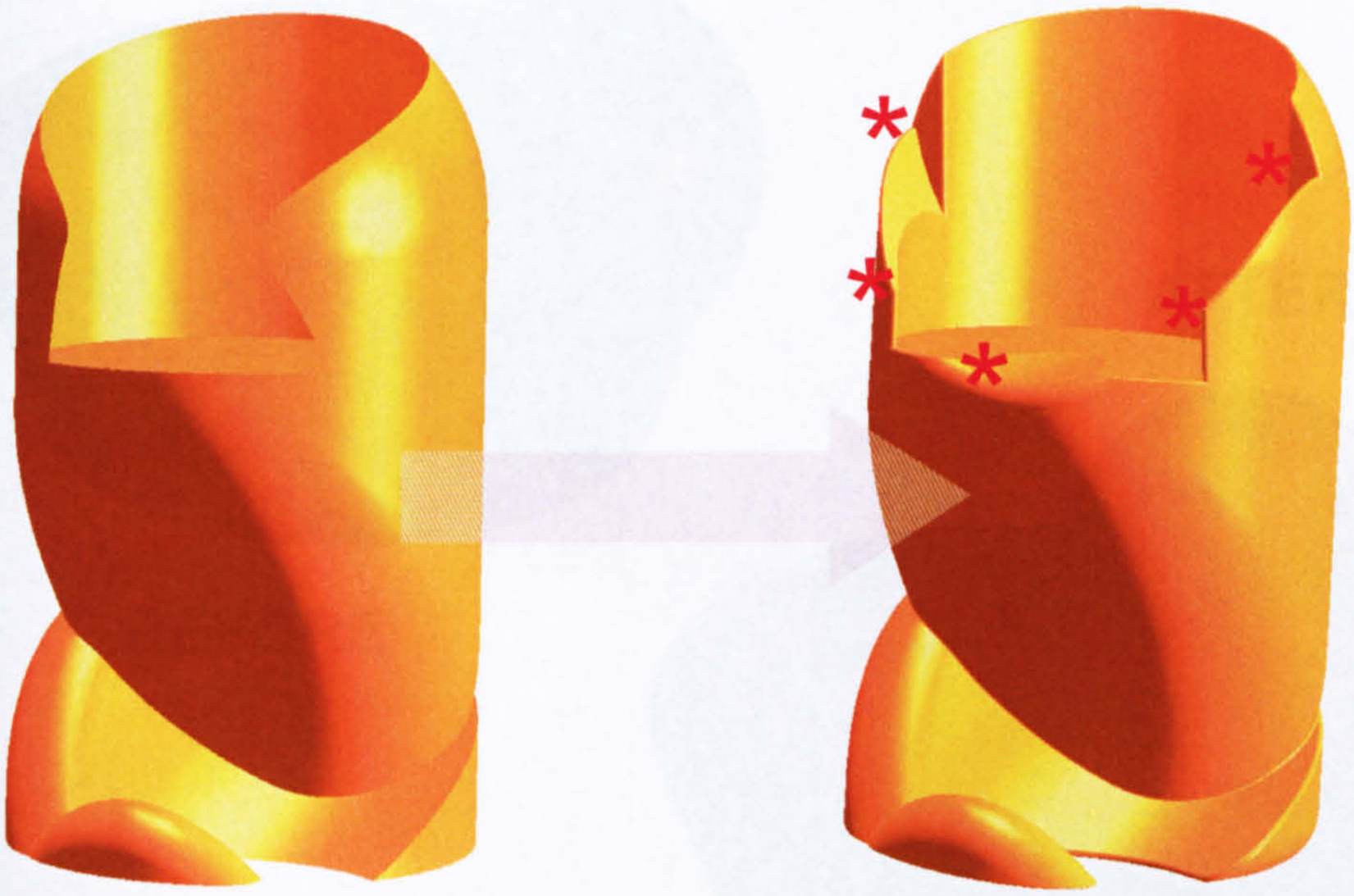


Figure 5.1.5 – Modifying evolved object to form sofa armrest

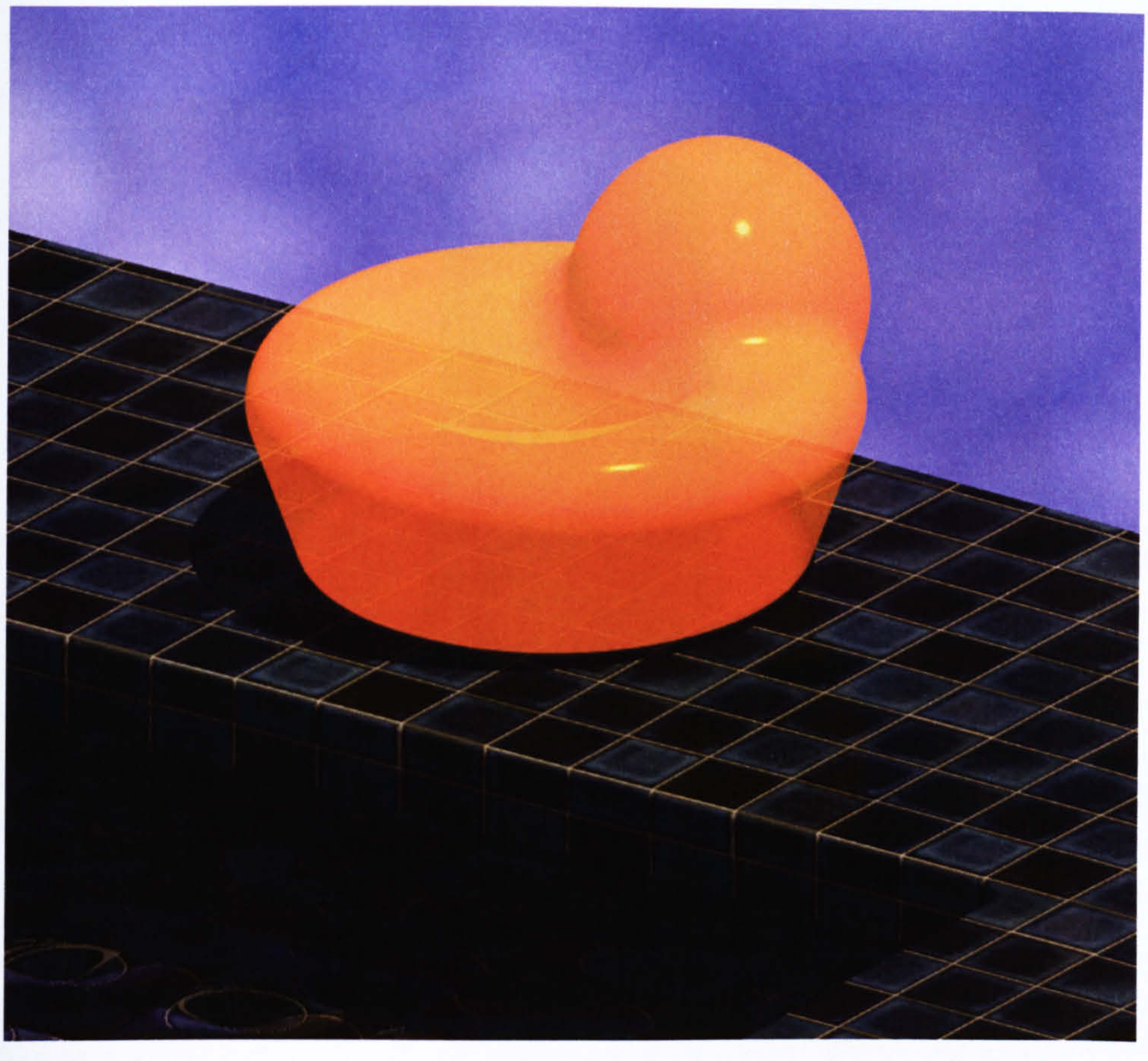
BEFORE

AFTER



Post-evolution changes : **Scallops, uprights and seat detail**

Figure 5.1.6 - **'Bar Seat'**

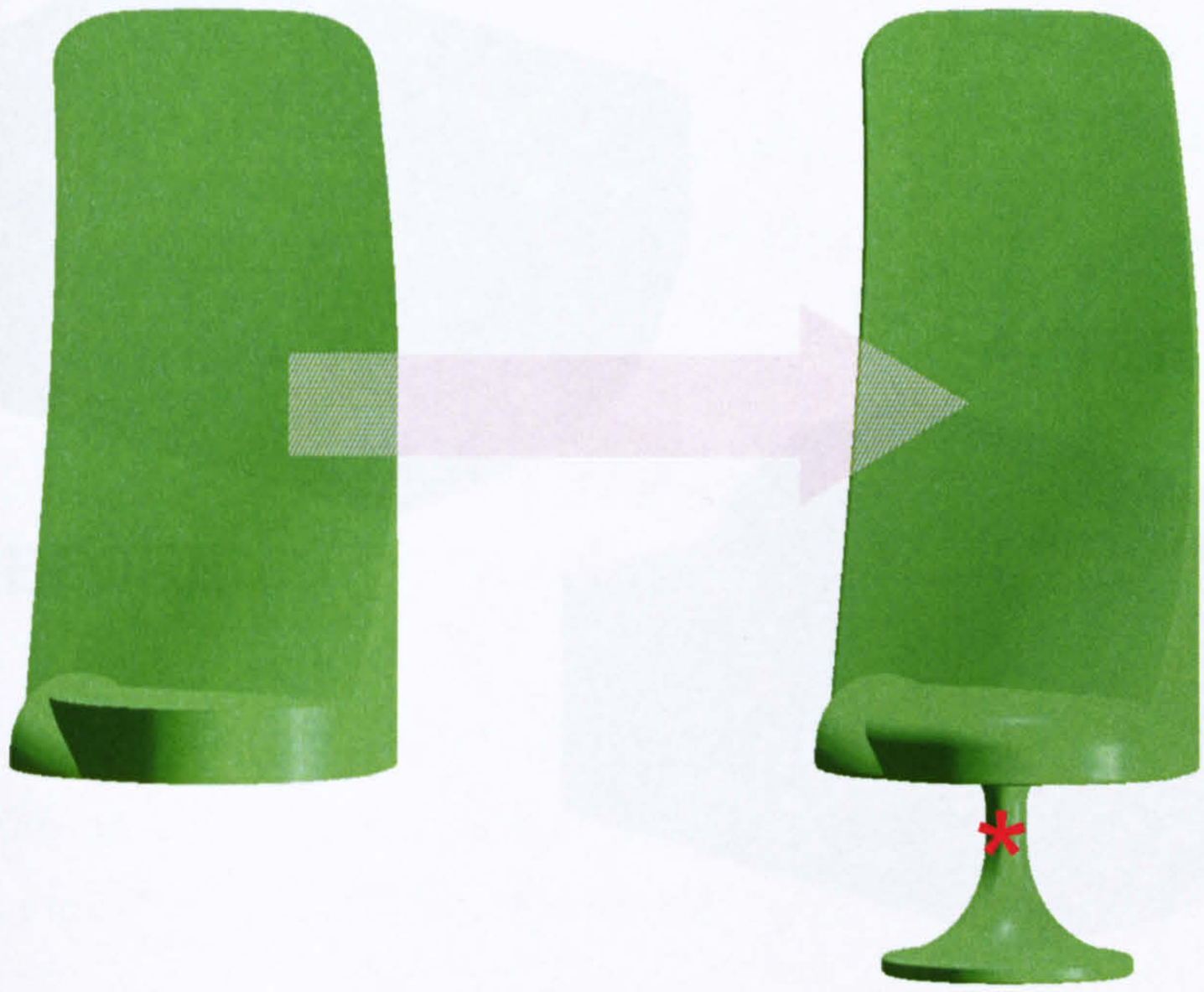


Post-evolution changes : **Edges blended**

Figure 5.1.7 - **'Orange Inflatable'**

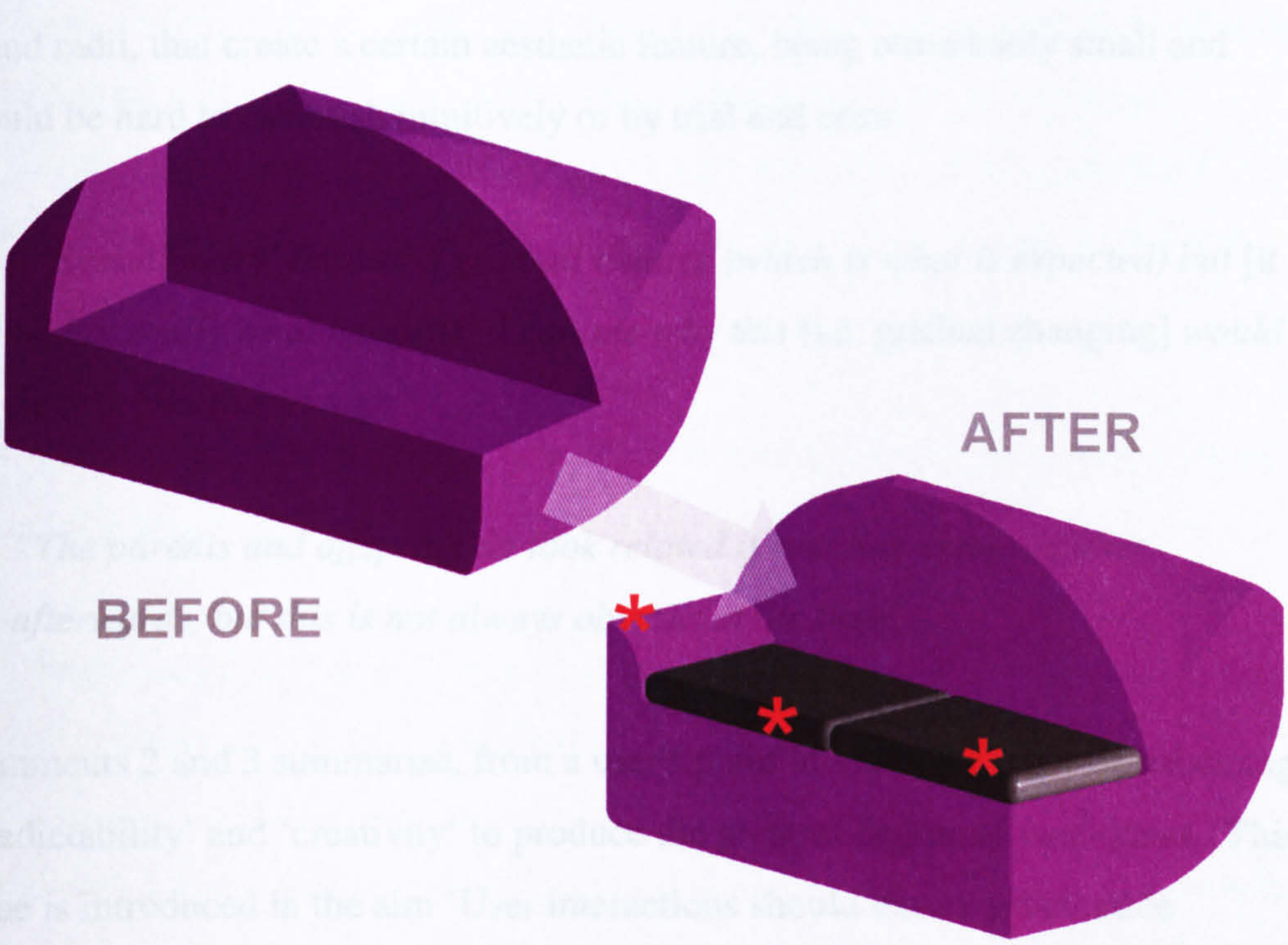
BEFORE

AFTER



Post-evolution changes : **Base added**

Figure 5.1.8 - **'Bond Villain Chair'**



Post-evolution changes : **Cushions added, arm modified**

Figure 5.1.9 - **'Bad Taste Sofa'**

5.1.2 System Assessment and Discussion

The following comments were made by the undergraduates after some prompting and discussion and are shown below more or less as transcribed:

1. *"The EFD system is good at producing attractive and usable shapes that I would not be able to create myself"*

This first comment relates to part of the initial aim, 'Forms on the screen should be aesthetically interesting…', and also to the fourth aim, 'The system has to actually assist the designer', in that forms can be created that many users would find difficult to produce using conventional conceptual design techniques.

It is particularly difficult to create the best of these kinds of objects using a conventional CAD interface, from just a few interacting solids and blends. The distinctive silhouettes, primary lines of form and the flow of surfaces and edges are often arrived at quite subtly: The envelopes of relative positioning, size and blend radii, that create a certain aesthetic feature, being remarkably small and would be hard to establish intuitively or by trial and error.

2. *"'Evolutionary' implies a gradual change (which is what is expected) but* [it is] *not really what happens. I can see why this* [i.e. gradual changing] *would be less creative though"*

3. *"The parents and offspring do look related if studying them together afterwards, but this is not always obvious at the time"*

Comments 2 and 3 summarise, from a user's point of view, the issue of balancing 'predictability' and 'creativity' to produce the greatest degree of usefulness. This issue is introduced in the aim 'User interactions should have a predictable outcome'.

When working with ten to fourteen objects per generation at one time, it is hard to visualise in detail the objects from previous generations, which contributes to the feeling of remoteness expressed in these comments. Some interactive evolutionary design systems overcome this problem by readily displaying the previous generations in a separate window. One could envisage a similar feature for this system that allowed the user to view freely the two parents of a selected object on the screen next to the object. The user could then see the similarities between parents and offspring – the consequence of their actions – and would thus feel more connected to and have more confidence in the system.

4. *"Objects definitely improve and have more similarities as the generations progress"*

5. *"The best results are obtained if you concentrate on just a few similar objects if designing something specifically"*

Re-establishing some confidence in the system, comment 4 confirms the steady reduction in frequency of ineffective objects, the gradual improvement of objects, and also that a degree of convergence takes place as generations progress. Comment 5 follows on from this, relating to the way object scoring methods usually develop, and that users can naturally develop effective scoring methods that suit the task or individual user.

6. *"Cohesive objects are easier to work with but this mode often creates lots of small useless objects and less inheritance is seen"*

7. *"Fragmented* [normal] *objects show better inheritance since all the shapes* [primitives] *are used in each object"*

Comments 6 and 7 support the decision to include two methods of object creation, each mode having its merits: Particularly suited to first-time users, the Cohesive method presents a clearer picture by not creating unattached primitives[1]. If

[1] and is also required for the geometric analysis function

improved visual inheritance is required the 'Normal' method can be employed. These objects will always contain the cohesive part, but also any of the other primitives that are not attached to the 'main' solid body.

8. *"Whole object blending produces the best objects, simple blending just rounds off the edges of basic shapes"*

Comment 8 reflects a preference in this case for the advanced 'whole-object' blending method, because of the generally more exciting objects produced. The fact that this method often masks inheritance is overlooked. The increased continuity afforded by 'simple' blending is complemented by a distinct character (belittled by the '*just rounds off the edges of basic shapes*' comment) which may be preferred by some users for certain product applications.

9. *"Objects are easy to take out of the evolution process and work on* [other versions of Unigraphics are opened] *but it would be good to carry on evolving them after changes have been made"*

Comment 9 reaffirms the benefits from a users point of view of a CAD environment, and solid modelling for the product representation, and adds the commonly documented (but extremely difficult to implement) desire for greater control over phenotypes during evolution

10. *"Geometric optimisation is quite clever but doesn't really help since the designer can see which objects are better anyway"*

Although geometric optimisation was presented to these users as a demonstration, comment 10 clearly indicates that much work is necessary if the inclusion of automated fitness functions is to be beneficial to the designer. While seemingly content in its current capability as an aesthetic concept generator, people who have used the system have expressed an interest in how this technology could be developed. These comments may influence the direction of further research and are discussed in the following chapter.

5.2 Animal Sculptures

During the later stages of the research, an invitation was received from the organisers of the International Congress on Evolutionary Computation (CEC2001, Seoul, South Korea) to produce a poster exhibit for the Evolutionary Art and Design competition session [Graham3]. From the literature survey carried out, it was evident that much of the computer generated evolutionary art being produced was 2-dimensional in nature (being analogous to painting / fine art in the 'real world'), often generic, abstract, and artificial in appearance. A particular advantage of the EFD software, being CAD based, is that, by using the materials and wrapped image textures available, realistic-looking renderings of objects can be produced. It was evident that there was a chance to produce something different – 3 dimensional in nature – differentiating this work from much of the evolutionary art that would be exhibited[1].

In producing the virtual sculptures, a similar approach was taken to that of the seating design task: Initial generations were rated for suitability as sculptures, and as sculpture-like objects proliferated through the populations, the best examples displaying animal-like properties, were favourably rated. The lack of any functional criteria made the task especially enjoyable, enabling the focus to remain purely on the aesthetic. More time was taken with this task, with 10 or so runs of up to 8 generations assessed before satisfactory objects were produced.

The materials, textures, backgrounds and plinths were added manually, but unlike the seating design task, no alteration was carried out to the objects' form, as this would have been an infringement of competition rules (which allowed interactive fitness rating but not direct manipulation of objects/images). The finished results, also illustrated in [Case], can be seen over the following pages (Figures 5.2.1 – 5.2.4).

---

[1] The reader may be interested to know the outcome of the competition: the congress delegates judged the entries, with over 250 votes being cast. Out of the 24 entries, the animal sculptures exhibit came second, missing out on winning by just 2 points.

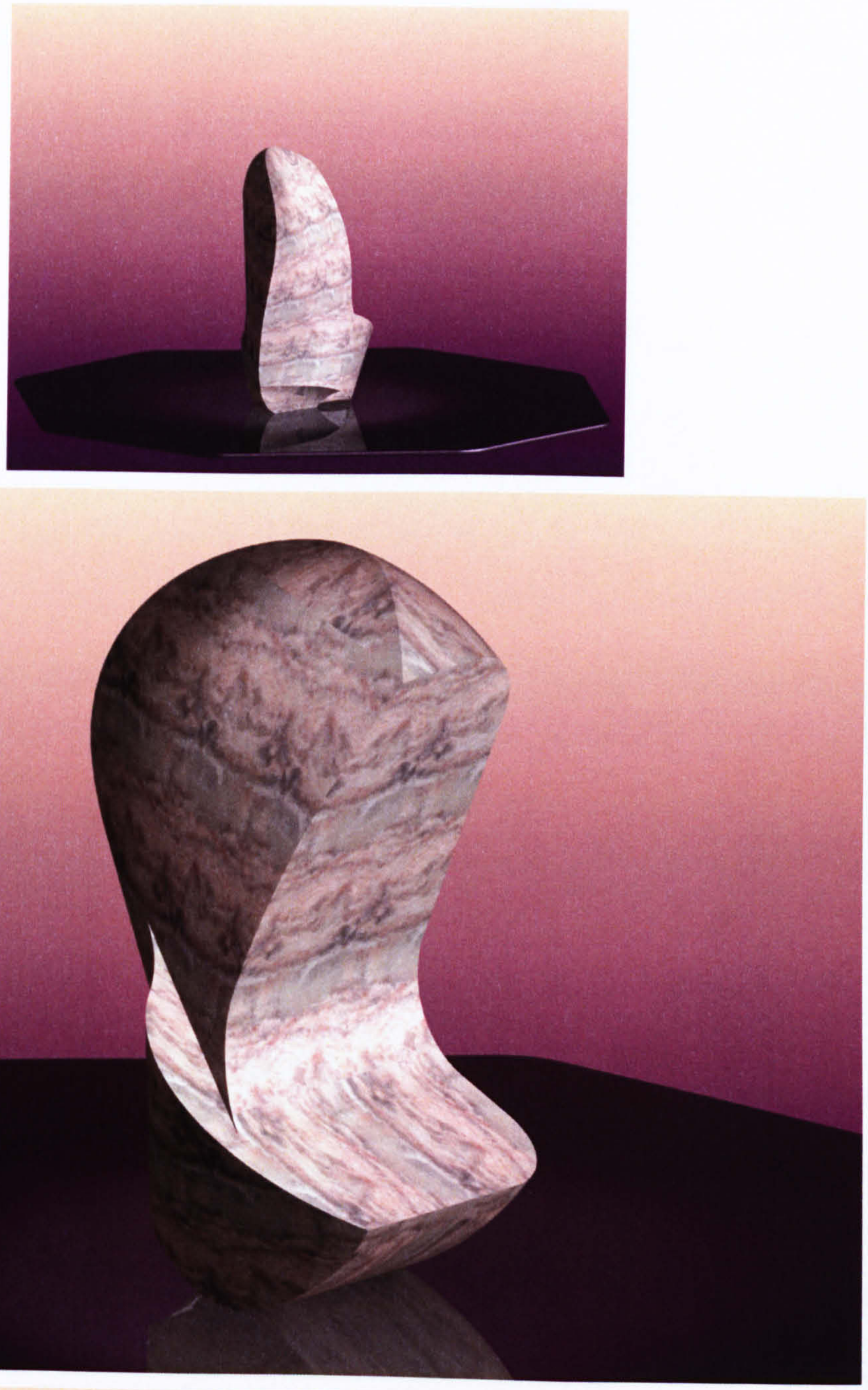


Figure 5.2.1 - **'Cobra'**

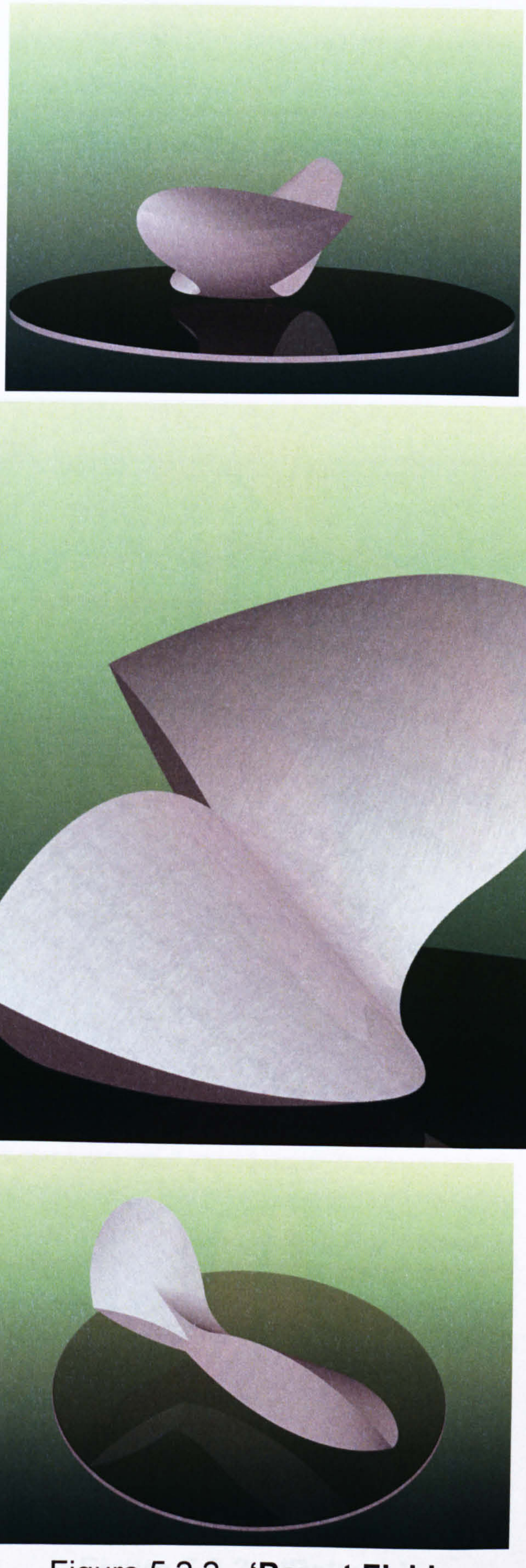
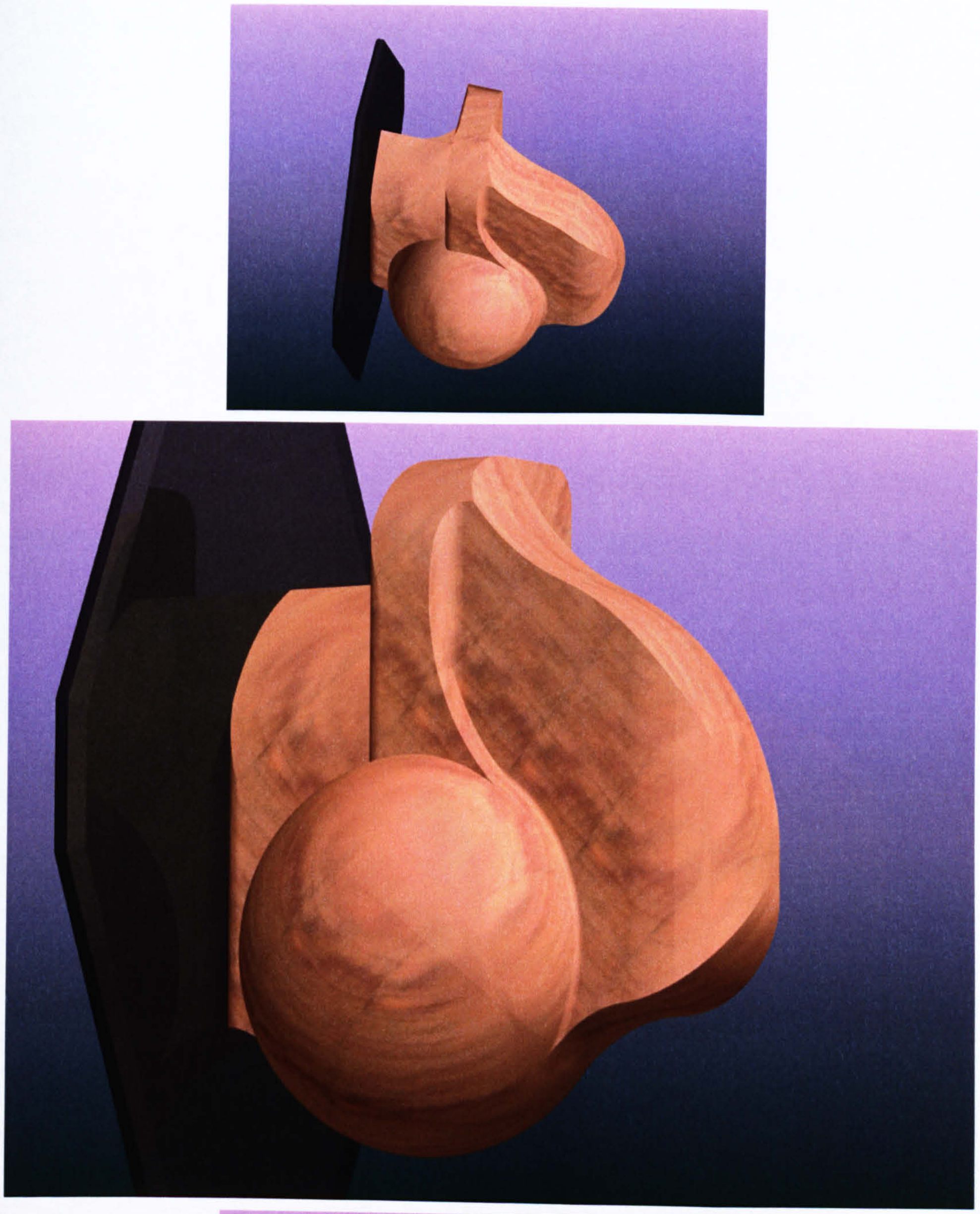


Figure 5.2.2 - **'Parrot Fish'**



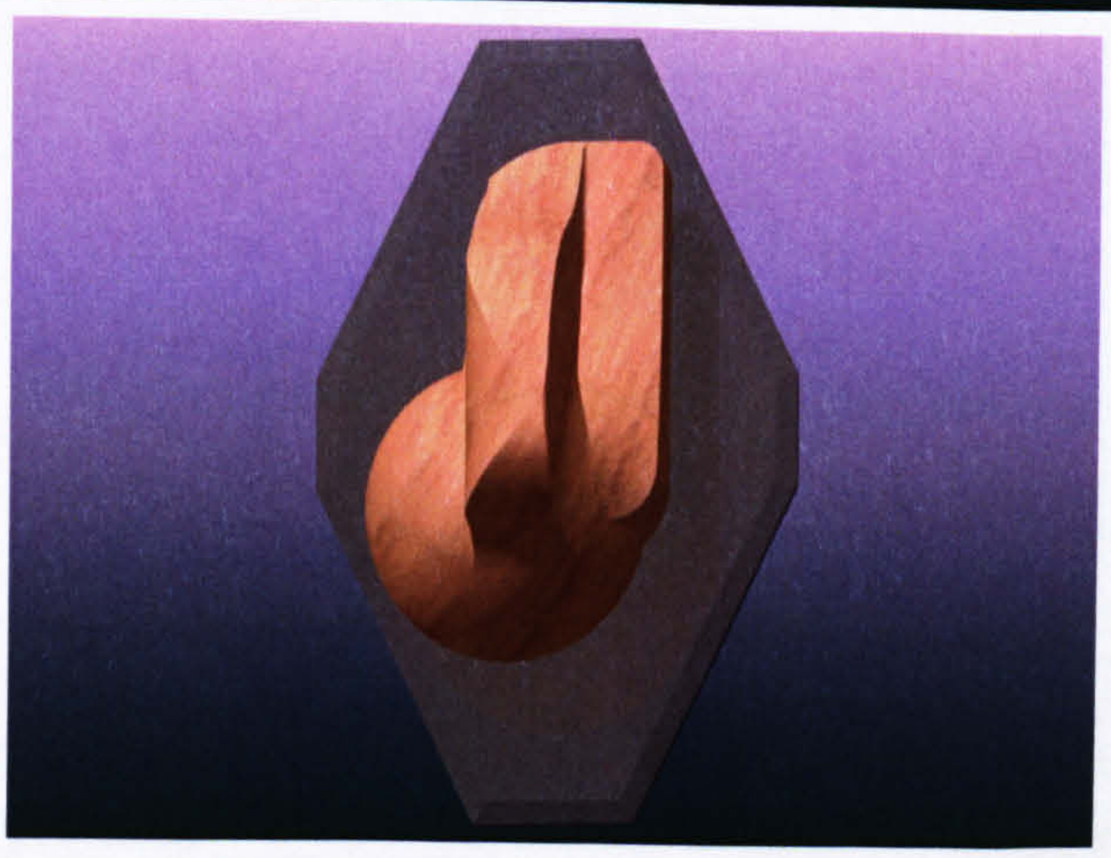
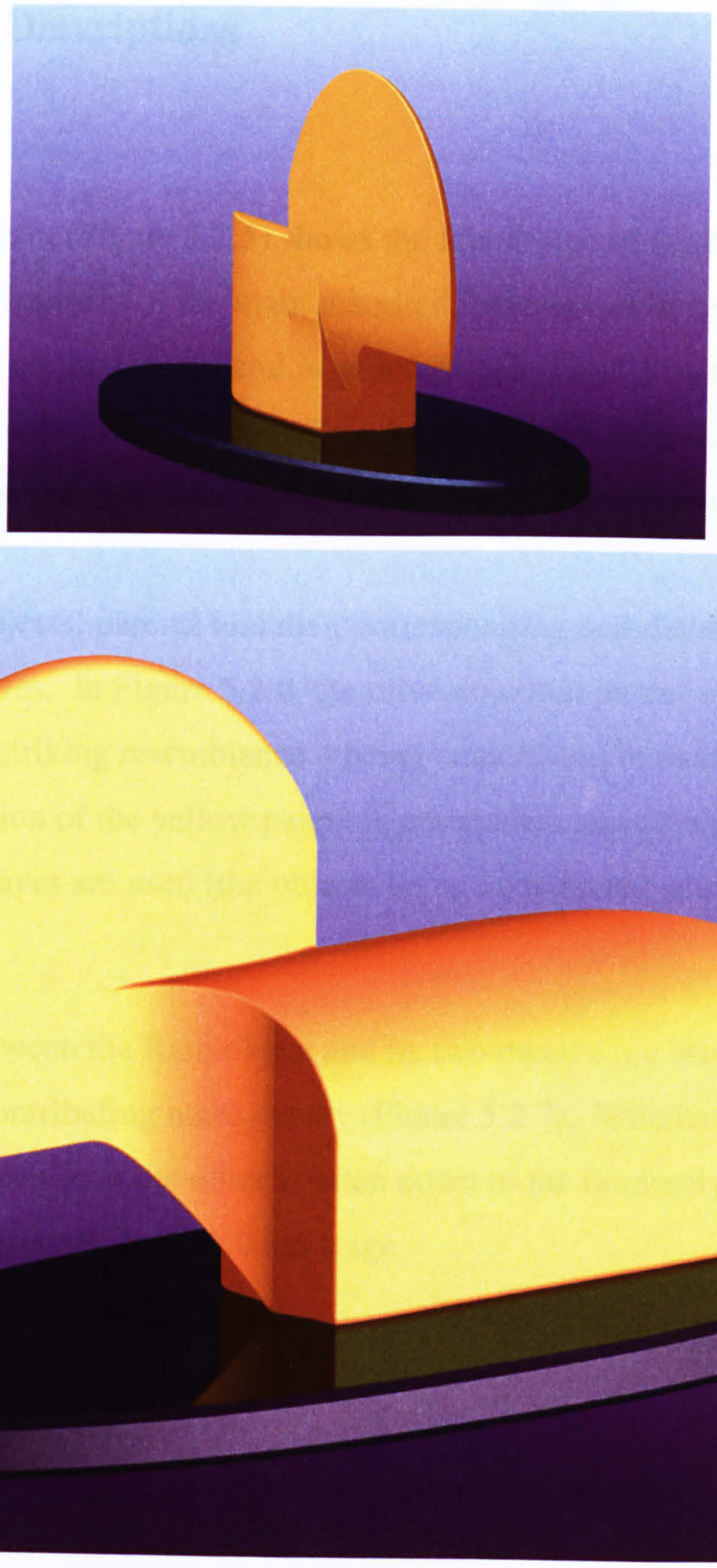


Figure 5.2.3 - **'Pelican'**



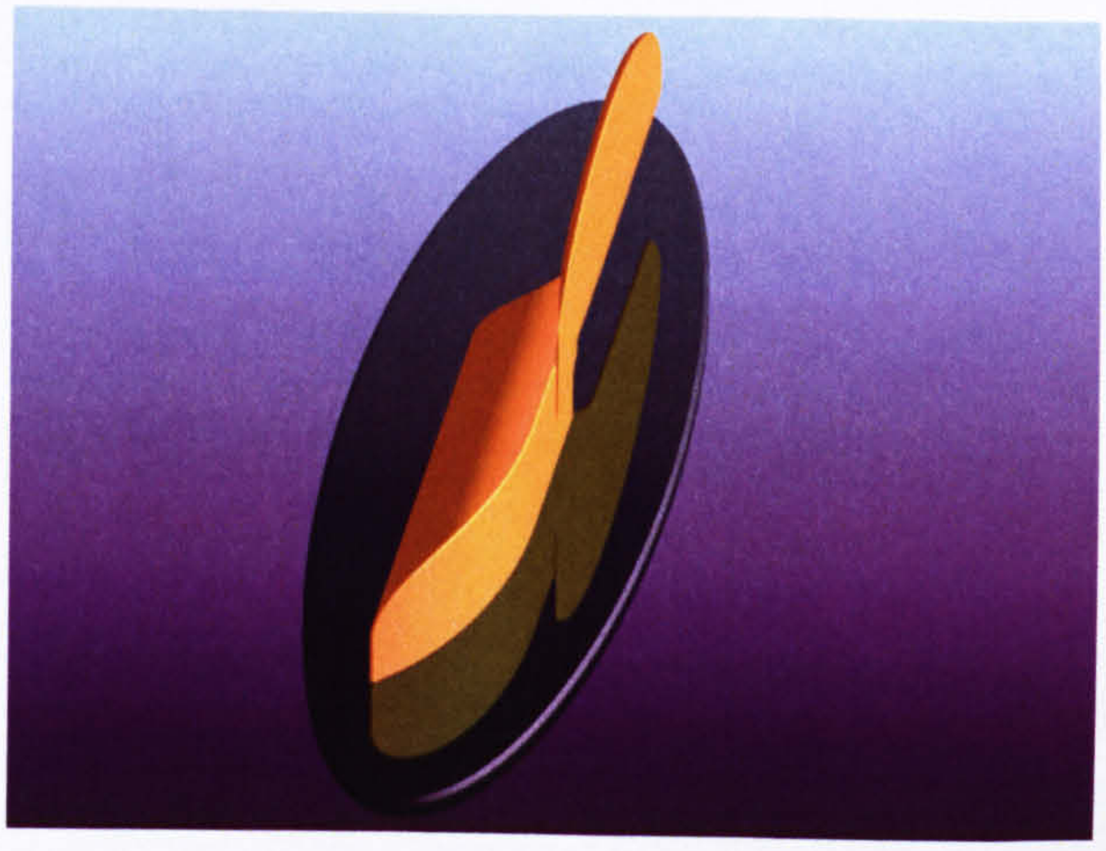


Figure 5.2.4 - **'Ram'**

5.2.1 Sculpture Descriptions

Evolution

The following diagram (Figure 5.2.5) shows the family tree of the Cobra sculpture: The development of the main inherited features can be easily seen in generations 2 and 3 on the far left, and in generations 1,2 and 3 on the far right. The left-hand side of the 'family' contribute the two intersecting cones configuration, whilst the right side contribute the intersection operator.

Two of the other objects' parents and their corresponding populations can be seen in the next two figures. In Figure 5.2.6, the olive-coloured parent of the Parrot Fish object bares a striking resemblance – being constructed in exactly the same way. The contribution of the yellow parent is not evident since only one of a possible four primitives are used (the objects being constructed using the *cohesive* method).

The visual links between the Ram object and its two parents are less apparent, with both parents contributing more evenly (Figure 5.2.7). Whether or not one parent seems dominant over the other is often down to the randomly selected crossover points during the reproduction stage.

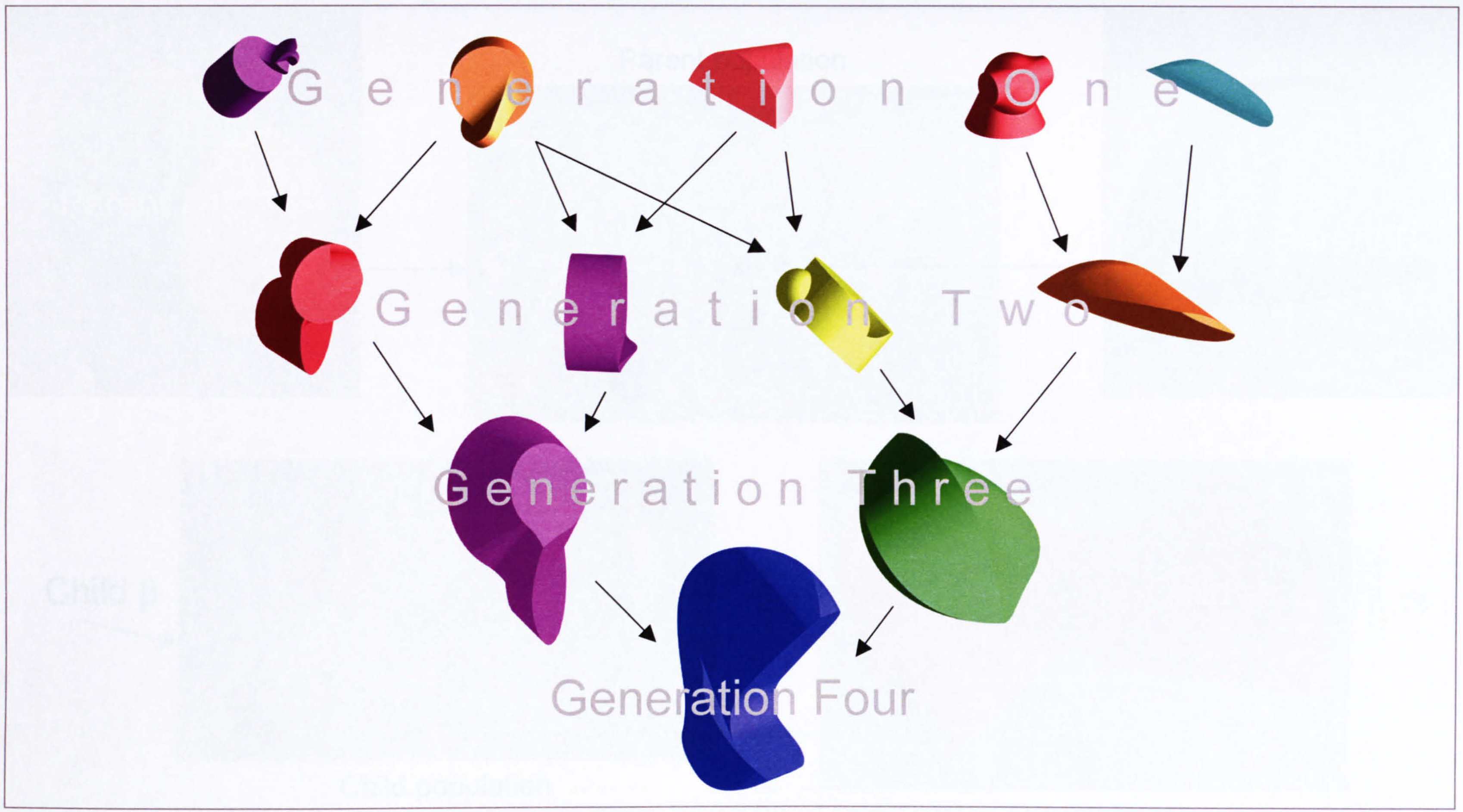


Figure 5.2.5 - Family tree diagram of ‘Cobra’ sculpture

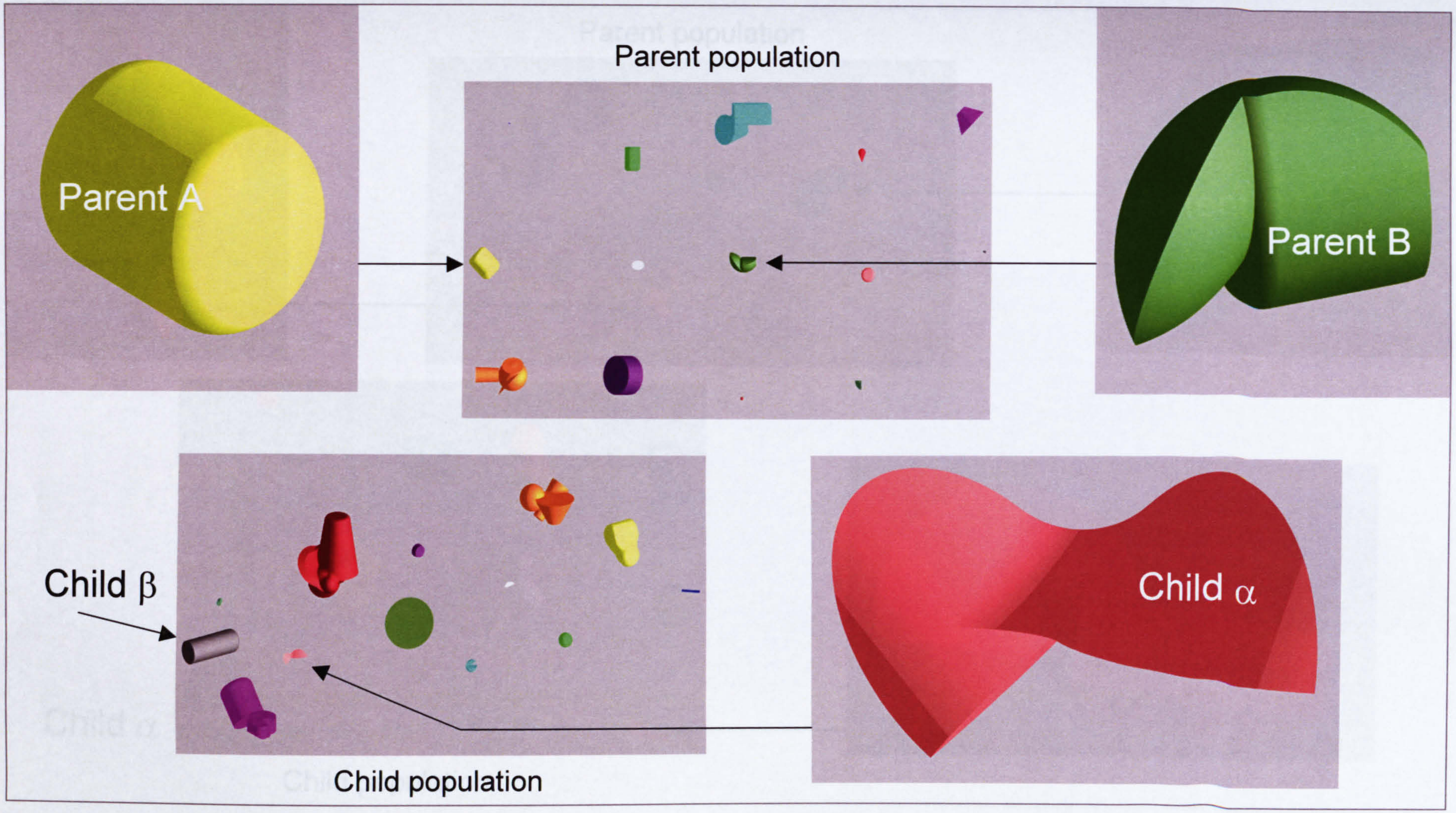


Figure 5.2.6 - Parents and associated populations of the 'Parrot Fish' sculpture

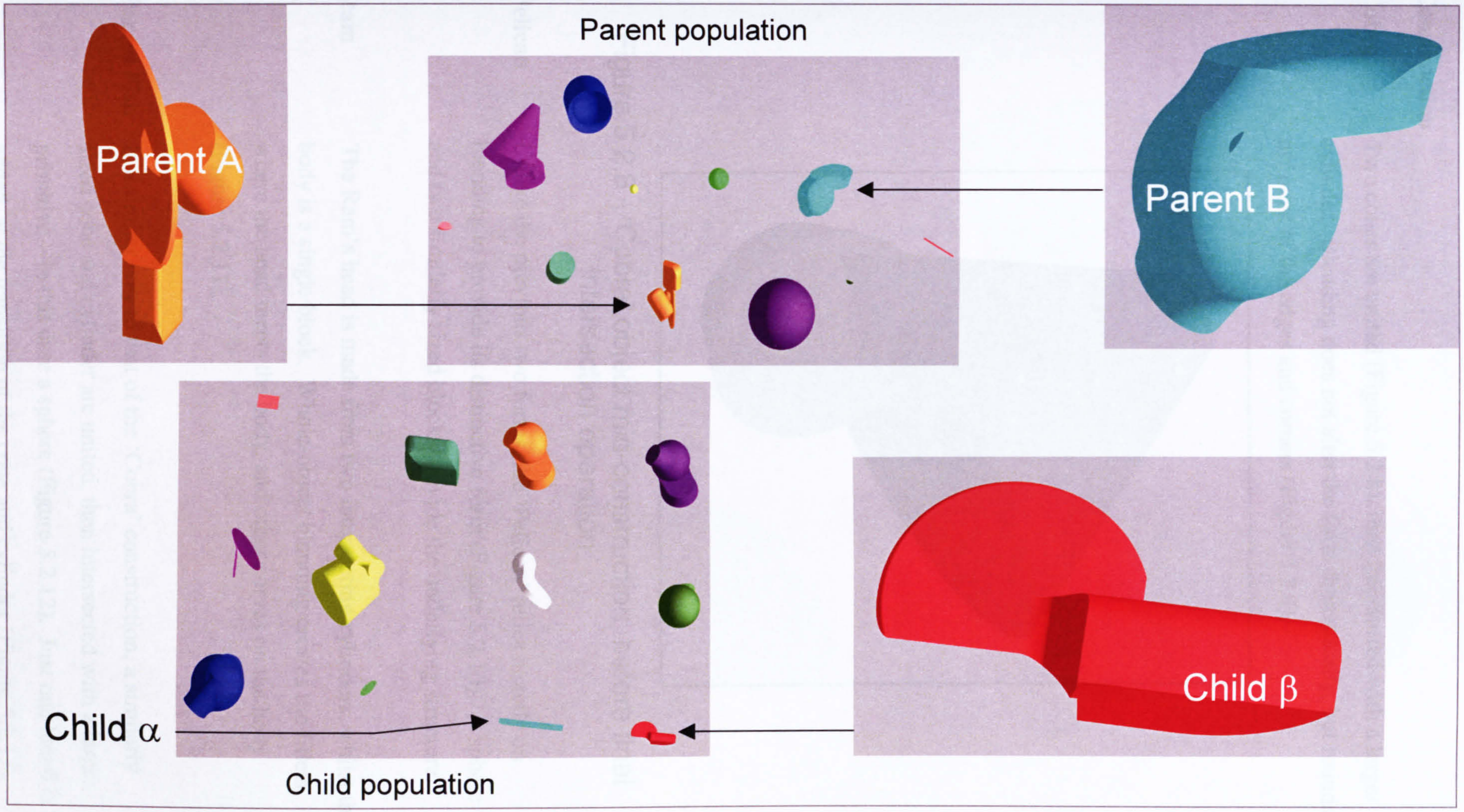


Figure 5.2.7 - Parents and associated populations of the 'Ram' sculpture

Construction

Cobra Two cones are united (Figure 5.2.8), then intersected with a larger cylinder. Blending does not alter the form dramatically, but rounds off most of the edges and corners (Figure 5.2.9).

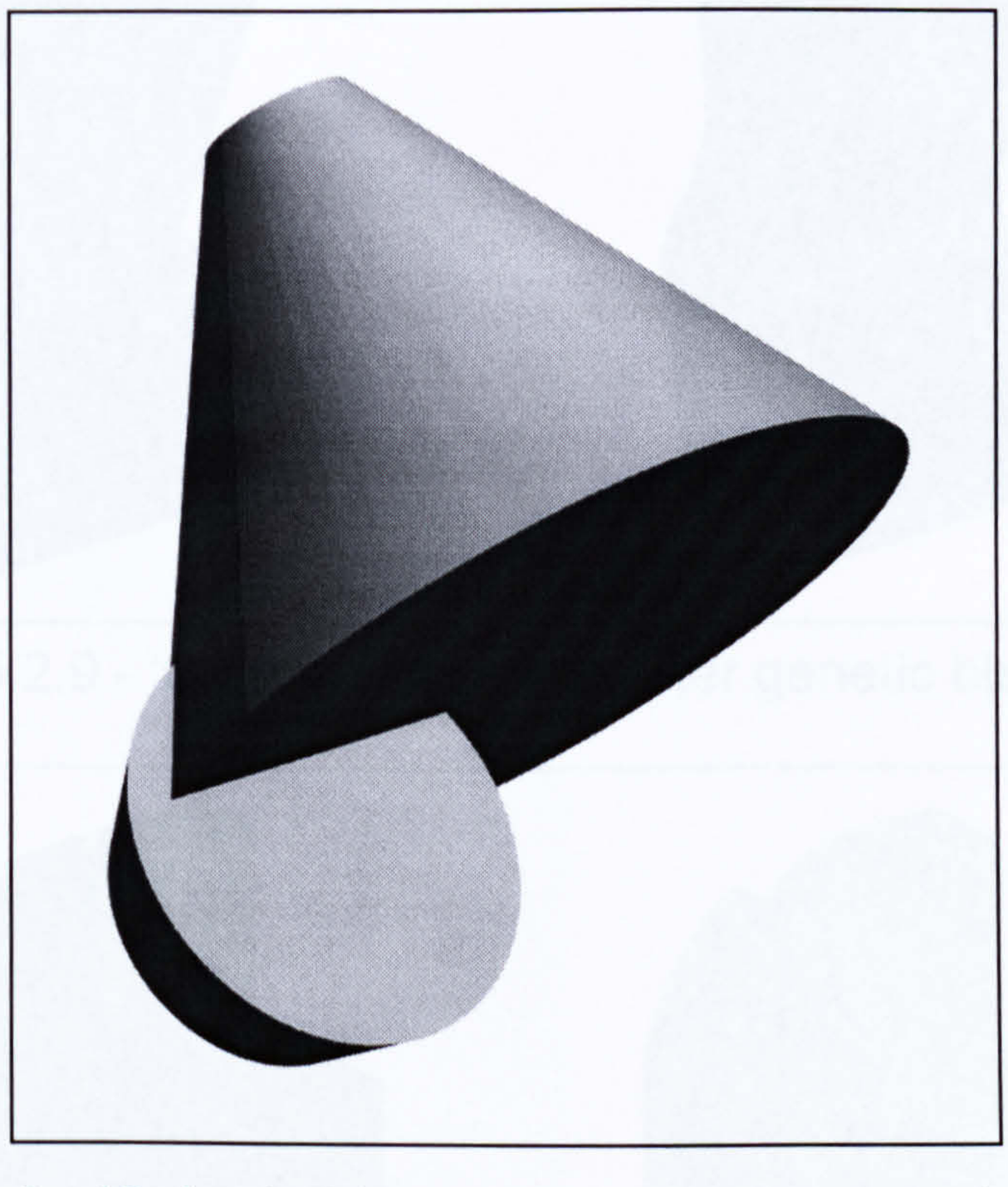


Figure 5.2.8 - 'Cobra' object mid-construction, before final intersection operation

Pelican Unlike the previous two forms, the 'Pelican' relies heavily on blending to provide its distinctive form (Figure 5.2.10). A sphere and two similarly sized blocks provide the underlying structure.

Ram The Ram's head is made from two intersecting cylinders, whilst the body is a single block. Whole-object blending creates the fillets where the head meets the body, and other curves on the body (Figure 5.2.11).

Parrot Fish In a similar way to that of the 'Cobra' construction, a similarly sized cone and cylinder are united, then intersected with a larger primitive – in this case a sphere (Figure 5.2.12). Just one blend is added, at the transition of the cone and cylinder (Figure 5.2.13).

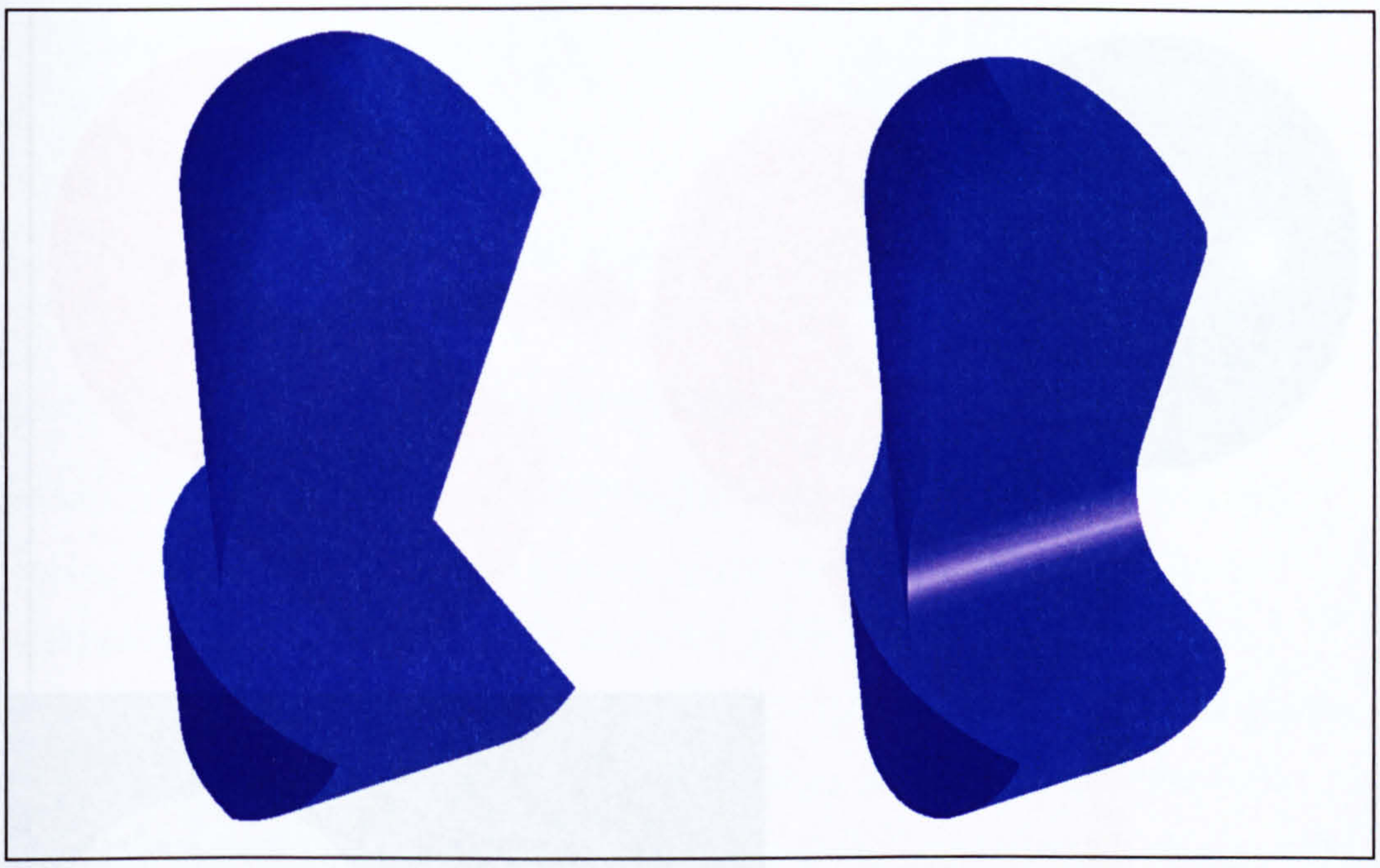


Figure 5.2.9 - ‘Cobra’ before and after genetic blending

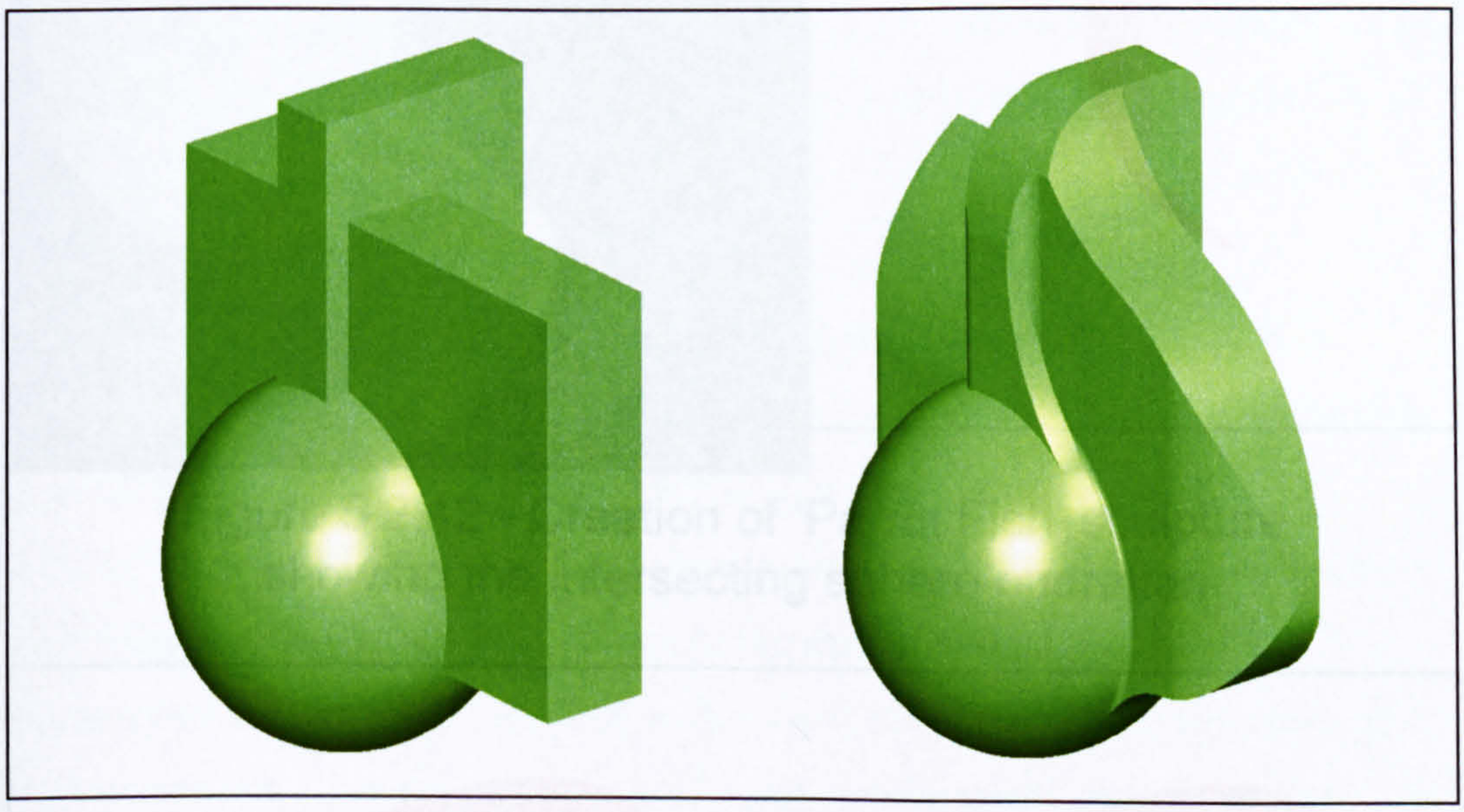


Figure 5.2.10 - ‘Pelican’ before and after genetic blending

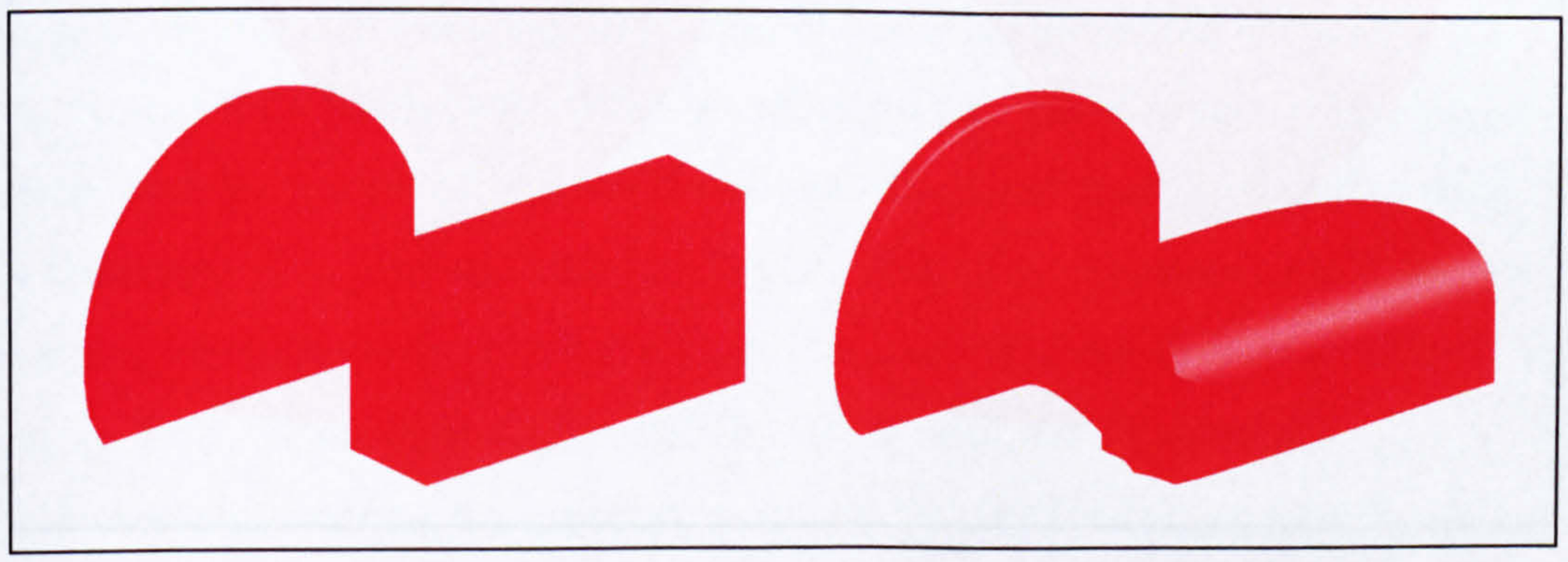


Figure 5.2.11 - ‘Ram’ before and after genetic blending

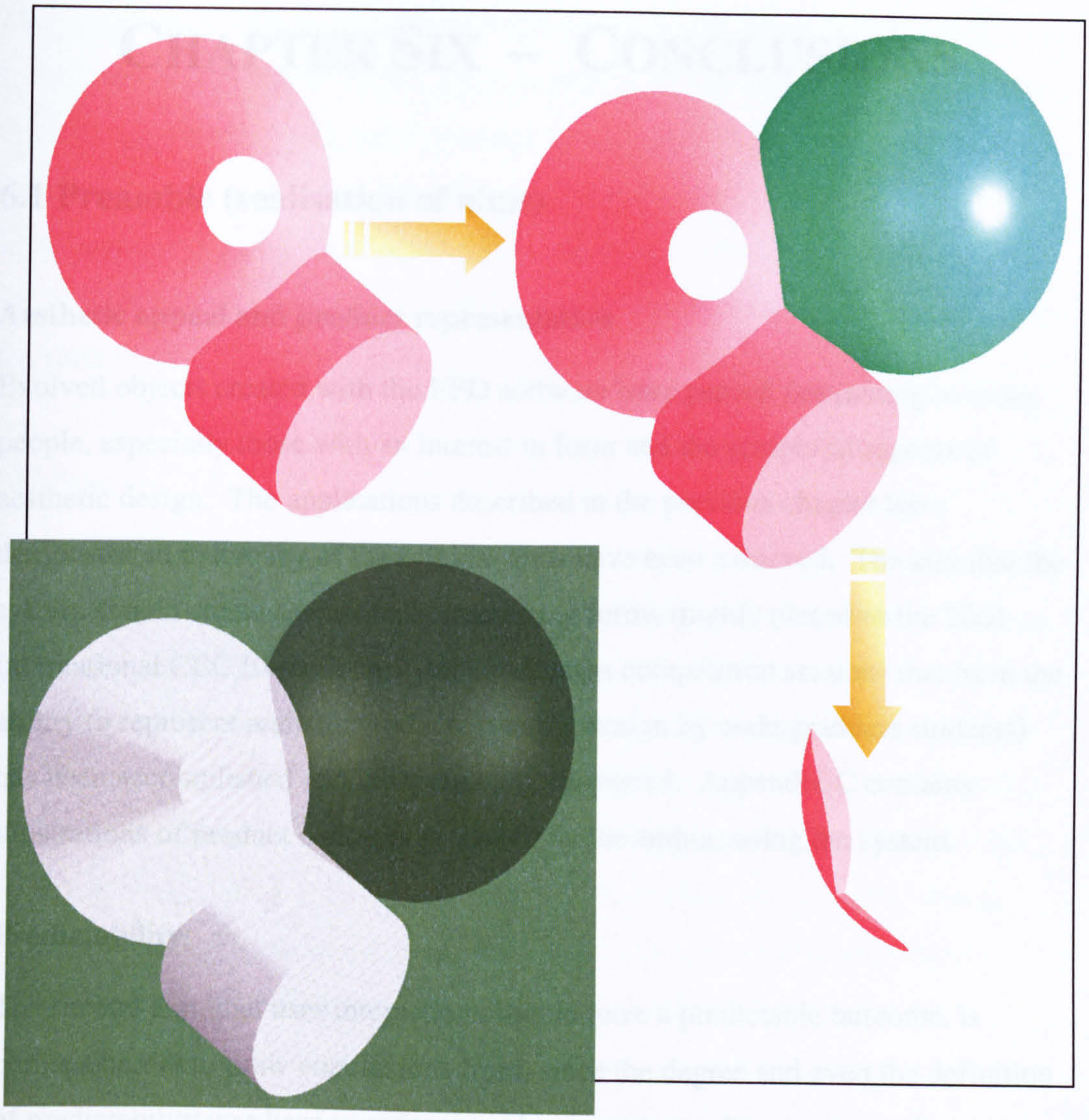


Figure 5.2.12 - Creation of 'Parrot Fish' sculpture showing the intersecting sphere operation

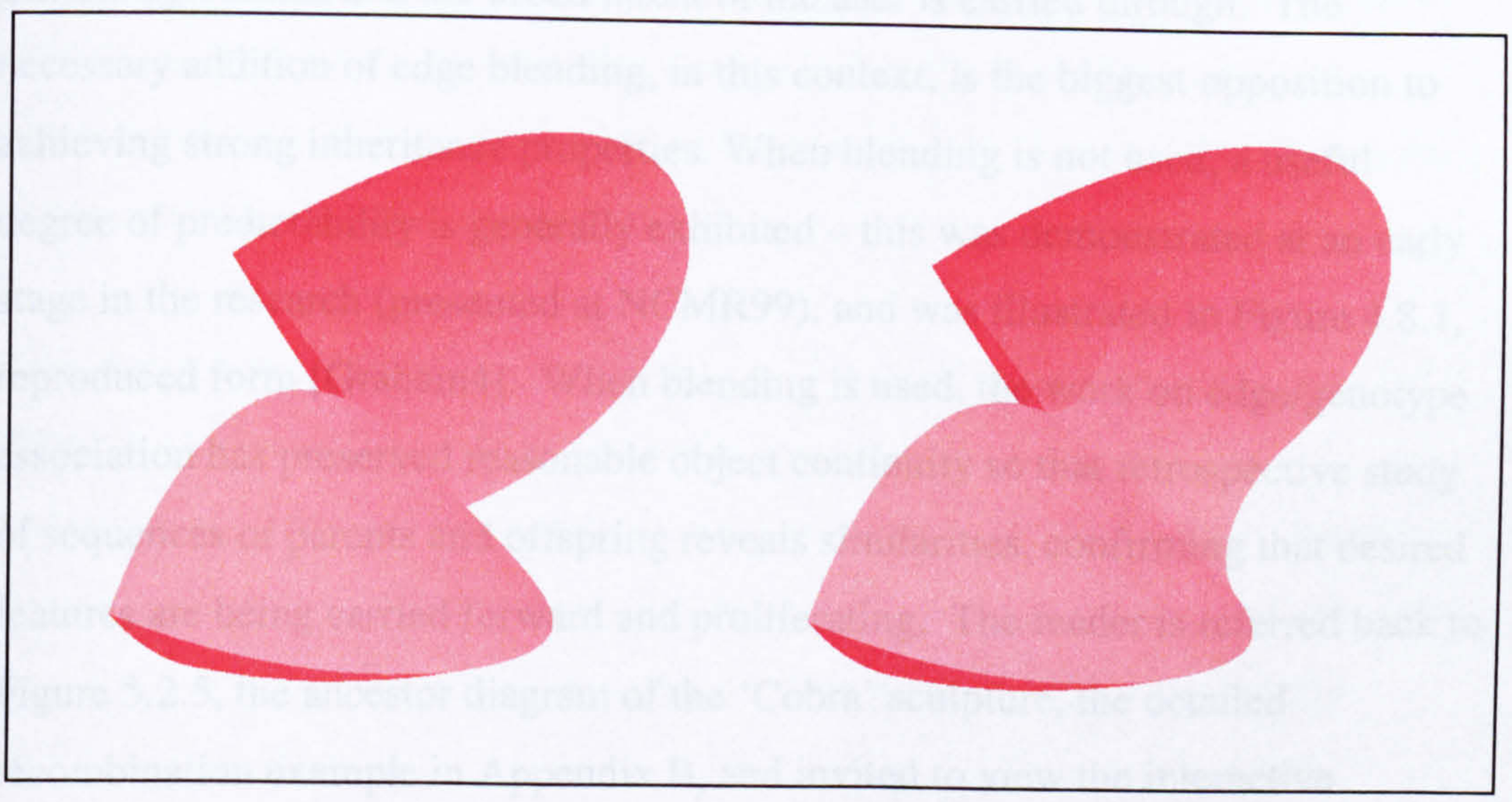


Figure 5.2.13 - 'Parrot Fish' before and after genetic blending

CHAPTER SIX – CONCLUSIONS

6.1 Preamble (realisation of aims)

Aesthetic appeal and product representation

Evolved objects created with the EFD software have proved fascinating to many people, especially those with an interest in form and the sculptural aspects of aesthetic design. The applications described in the previous chapter have demonstrated that many of the original aims have been achieved. The aim that the system should create aesthetically interesting forms (highly placed in the 2001 International CEC Evolutionary Art and Design competition session) that have the ability to represent realistic products (seating design by undergraduate students) has been accomplished and independently supported. Appendix C contains illustrations of product concepts produced by the author, using the system.

Predictability

The second aim, that user interactions should have a predictable outcome, is perhaps harder to draw conclusions from, since the degree and even the definition of predictability was hard to express in absolute terms. Comments on the seemingly haphazard progress of the populations have been tempered with the general agreement that the broad intent of the user is carried through. The necessary addition of edge blending, in this context, is the biggest opposition to achieving strong inheritance properties. When blending is not used, a useful degree of predictability is generally exhibited – this was demonstrated at an early stage in the research (presented at NCMR99), and was illustrated in Figure 4.8.1, reproduced form [Graham4]. When blending is used, the work on edge/genotype association has preserved reasonable object continuity so that retrospective study of sequences of parents and offspring reveals similarities, confirming that desired features are being carried forward and proliferating. The reader is referred back to Figure 5.2.5, the ancestor diagram of the 'Cobra' sculpture, the detailed recombination example in Appendix B, and invited to view the interactive ancestor diagram contained on the enclosed CD-ROM.

The evolutionary development of form strikes a balance, being neither totally random nor totally deterministic, and thus creates an innovative, but usable tool. Considering the need to preserve the innovative nature of the software (compared with an alternative scenario where changes between immediate generations are small and outcomes of all object recombination are geometrically apparent) the spirit of the original aim has been achieved, especially when considered alongside the further aims.

Efficiency

The aim to make the evolutionary design process efficient has led to the work on maximising the use of constituent primitives during the object creation stage, and the development of the Teamforming technique. The marked improvement of the current capabilities over those of early prototypes is clearly apparent. Although inheritance could be seen in the early versions of the software, the overall quality of populations throughout the evolution process was poor, and development was relatively slow, taking several generations (4 to 5) before many usable objects were generated. The increase in efficiency has been due, firstly, to the better overall quality of populations (with a greater proportion of fit objects), and secondly, to the rapid improvement of quality over a small number of generations, with usable objects often appearing from the second generation onwards. Referring to the seating design application outlined in the previous chapter, it only took between 2 and 4 generations for usable objects to be created. The 4 chairs were taken from a total of 7 populations, including the two initial random populations, comprising a total of 84 individual objects viewed (60 of these rated), giving a 'rating to useful' ratio of 15/1.

There are a few problem areas remaining, which, if addressed, could increase efficiency further. There are two specific areas requiring only modest development where one could expect to see significant improvements. Firstly some basic intervention to prevent detrimental Boolean operations, and secondly, the improvement of the cohesive object creation method. These are briefly outlined in section 6.3.

Efficiency ultimately depends on how specific the users' needs are – whether they already have a shape in mind, and the limitations the product type places on form variation. Still, whether utilising the convenience of 'cohesive' objects, or the completeness of 'normal' objects, it only takes a few generations (several minutes) before useful shapes emerge.

Sensitivity

The scoring system adopted is flexible and intuitive, allowing the user to employ several strategies for object assessment. An effective approach is to keep many objects involved (allocating similar scores to a large number of objects) early on in the process, then to concentrate on a few objects (employing more drastic scoring[1]). The ability to discard objects (score of 0), and the increased weighting of a 10/10 score have provided a considerable degree of control to the user. There is scope for work towards making the system more sensitive to the designer's input, by allowing explicit control of operators such as parent selection, as with methods often seen in evolutionary art applications.

Usefulness

The ability of the software to generate a multitude of novel forms provides a useful addition to existing methods designers draw on when looking for aesthetic inspiration. The fact that these forms are original, and influenced by the user through the evolutionary characteristics of the software, and not from a database of existing aesthetic concepts, further increases its usefulness. Other research in this field has provided tools for development of established aesthetic concepts, and there is no reason why this research should not expand along these lines.

[1] For example, 2 objects given 10 out of 10, 2 objects given 1 out of 10, and the rest 0

Currently, when a suitable object or objects are found, it is possible to change the parent selection technique from stochastic to roulette, which enables the selection of just two (or even one) objects for continuing exploration. Combining this with fine-tuning of the mutation rate creates a better environment for development of a single object. To increase the scope of this research, a development of this two-stage process can be envisaged: Firstly, a pleasing form is arrived at through the innovation-biased current software. And secondly, the form is developed from this single object using a dedicated technique involving subtle, controlled mutations and less disruptive genetic operators, such as in Evolution Strategies.

6.2 Methods (objectives, originality & contribution to knowledge)

Review

The review of the current state of research into evolutionary computation in engineering and design identifies the most widespread area of application of genetic algorithms as automated component optimisation (often using FEA techniques). Examples of interactive evolutionary design are mostly artistically based, often using repeating geometric patterns and mathematical series. The most comparable research in terms of philosophy and execution is the Emergent Design Group's Agency-GP architectural design exploration tool. Also comparable, in dealing primarily with the design of whole products, is Peter Bentley's work on the automatic generation and optimisation of generic products using a GA. The fundamental difference being that evolution is guided solely by computational analysis in Bentley's system, rather than predominantly utilising the users aesthetic judgements in this research.

This research gains its originality from the combination of user interaction and automated fitness determination to provide evolutionary characteristics. The research has also seen preliminary development and application of a new concept aimed at maximising the potential of each population: The Teamforming technique, that, combined with the GA, co-operatively groups together single phenotypes (primitives) to form more complex solutions (objects).

Also unseen previously, in the field of evolutionary design, is the use of geometric primitives and Boolean operators combined with edge blending, within a CAD system, to create original, aesthetically valid, 3D product representations. This representation is achieved using a small amount of data, and supports application to a range of products, without the need for additional programming.

Geometric Optimisation

The work on geometric optimisation has two clear benefits. It demonstrates that the integration of automatic fitness functions is readily achievable, and provides evidence that the GA is working effectively. When the system is running independently, with just geometric fitness functions employed, it has been reassuring to see convergence, although sometimes premature, and therefore successful optimisation, albeit to simple criteria. The limited work on automatic aesthetic assessment (e.g. fitness scores being penalised if objects are too fragmented) has not reached the stage where it is beneficial to the design system. As long as the user views all the objects in a population, they are able to identify aesthetically weak objects visually, and may as well be the sole aesthetic fitness provider. A further step would need to be taken if this aesthetic assessment was to provide benefit to the user.

Teamforming

The principles of the Teamforming method have been demonstrated, in that it can group similar or dissimilar primitives together (in terms of *size*, *sign* or *type*), making a more effective use of each population. The original objective was to enable the population to define the tactic, so that it may evolve to suit the user's current needs (as in self-adaptation in ES and EP, but applied to the Teamforming rule, and not mutation). At present though, the tactic is defined by the user at the beginning of each session, possibly restricting the technique's effectiveness.

Teamforming gives little visual inheritance between whole objects in subsequent generations, but features of objects are carried across effectively. The team tactic enables a variety of methods depending on suitability for application. It would seem that this initial implementation of Teamforming is not as elegant a process as that of the underlying Genetic Algorithm and as such does not exhibit emergent properties as was hoped. It is an interesting technique though, and when used within this research, provides an alternative method of evolution, which can be thought of as 'evolution by features'. The Teamforming method is particularly useful for designers looking for individual aesthetic features to apply to a variety of products.

6.3 Further Work

6.3.1 Practical improvements to the current EFD system

Avoiding Detrimental Boolean Operations

During object creation, there are two Boolean operations that, if executed as the last step, render the object useless - these two cases being the intersection of a small primitive wholly within the current body, or the union (or creation) of a large primitive that totally encloses the current body. In both cases the final object is left as a single geometric primitive. This occurrence could usefully be automatically predicted and avoided.

It should be noted that too much interfering with the fundamental processes of an evolutionary algorithm can be counterproductive, disrupting the 'natural' balance of the system. In the case of this research, objects having genes that result in destructive events are discounted by the user, letting evolution take its course. This process does take some time though, and, currently, discarding the object could result in the loss of a valuable potential object. Given the limited size of populations, on balance this addition to the software seems appropriate.

Improving Cohesive Objects

During the use of the software, most users have opted for the cohesive object creation mode. As previously explained, this mode was created rather simplistically, as a necessity for automatic geometric analysis. With hindsight, a more effective technique would use an adaptation of the post-creation Boolean process described in chapter 4: Isolated primitives could be deleted after the whole object had been created, rather than not being created at all. This would give isolated primitives, that were introduced early on (2^{nd}, 3^{rd}, or 4^{th}), a greater chance to interact. The problem of deciding how to deal with objects consisting of two separate bodies, each made up of two interacting primitives, can, no doubt, be overcome.

Interaction

For the progression towards an industrially applicable design tool, it is suggested that the designer should have greater control. This could involve the ability to explicitly select each pairing of parents, or the ability to maintain an accessible archive of favourite objects. These examples suggest moving away from the current, linear path, to a user-led, fluid process. Concerning user-interface design, the ability to view and rotate individual objects more conveniently, view objects from previous generations, and the automatic scaling of small objects would give the user more freedom, and speed up the evolutionary design process.

Automatic Aesthetic Assessment

Current aesthetic assessment is limited to reducing object fragmentation. Fragmented objects receive a fitness penalty, but no further action is taken. A further implementation step would be necessary, for automatic aesthetic assessment to be beneficial. It is suggested that objects with low aesthetic function values are either discarded (and other objects created in their place) or measures taken to 'improve' the objects (i.e. the size or positioning values of isolated primitives altered). This was seen as too prescriptive initially, the thought being that evolution would take its course through the user's fitness assessments. With further research into aesthetics however, these changes would be necessary.

Teamforming

Future development of the technique in this application may improve the low object-continuity situation. One possibility is that each member carries a 'preferred' order gene. This would restrict choice during team-member selection, hopefully reinstating some visual continuity between objects in subsequent generations. Retaining the '5 primitives per object', and with a population of 70 members creating 14 objects, 14 members would be 'first-created' primitives, 14 would be 'second-created' primitives and so on. Breeding would be restricted to members with the same creation order gene, implemented using the multi-species capability of the GA – each of 5 species being a creation order. It is noted that this approach would conflict with any work carried out on variable size teams.

6.3.2 Increasing Research Scope

While this research is undoubtedly useful for original form generation, future development could further increase the efficiency of the evolutionary process, provide additional functional capabilities, and even extend the point at which the designer takes over from the EFD system in developing the form manually (using traditional CAD techniques). Two of the original objectives, intended to expand the capacity of the software in this way, have not been implemented: Brief descriptions of each follow.

Internal Volumetric Constraints

An idea stated in the objectives, and introduced in [Graham5], but not implemented is an internal volumetric fitness function. Objects would strive to incorporate, within their boundaries, simple box models manually created by the designer, before the start of the evolutionary process. The principals of the technique are demonstrated in Figures 6.3.1 and 6.3.2, and outlined below:

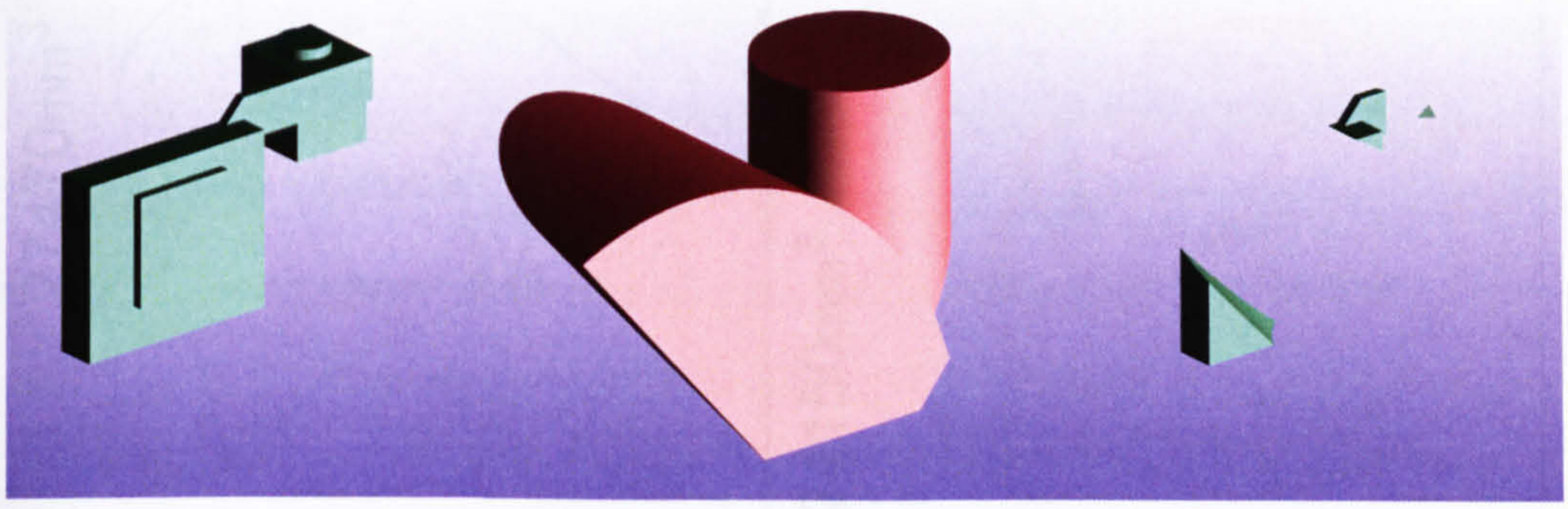
$$S_{object} = rV_{object-box} + \left(V_{box} - \left(V_{box \cap object}\right)\right)$$

- Before the evolutionary process is started, the designer constructs a simple box model of the internal space required (having volume V_{box})
- During fitness calculation, each evolved object's solid form (having volume V_{object}) is subtracted from the internal box model
- Any volume remaining counts heavily against the object's objective value (S_{object}), penalising the object for not totally enclosing the box model
- In a secondary test, the internal box model is subtracted from the object's form
- The volume remaining also counts against the object's objective value, but to a much lesser extent, penalising the object for not *efficiently* enclosing the box model

The relative importance, *r*, of these two tests would need to be established through experimentation (a value of 0.1 is suggested as a starting point).

Intersected volume = 1800mm^3

1970mm^3 - 1800mm^3 = 170mm^3

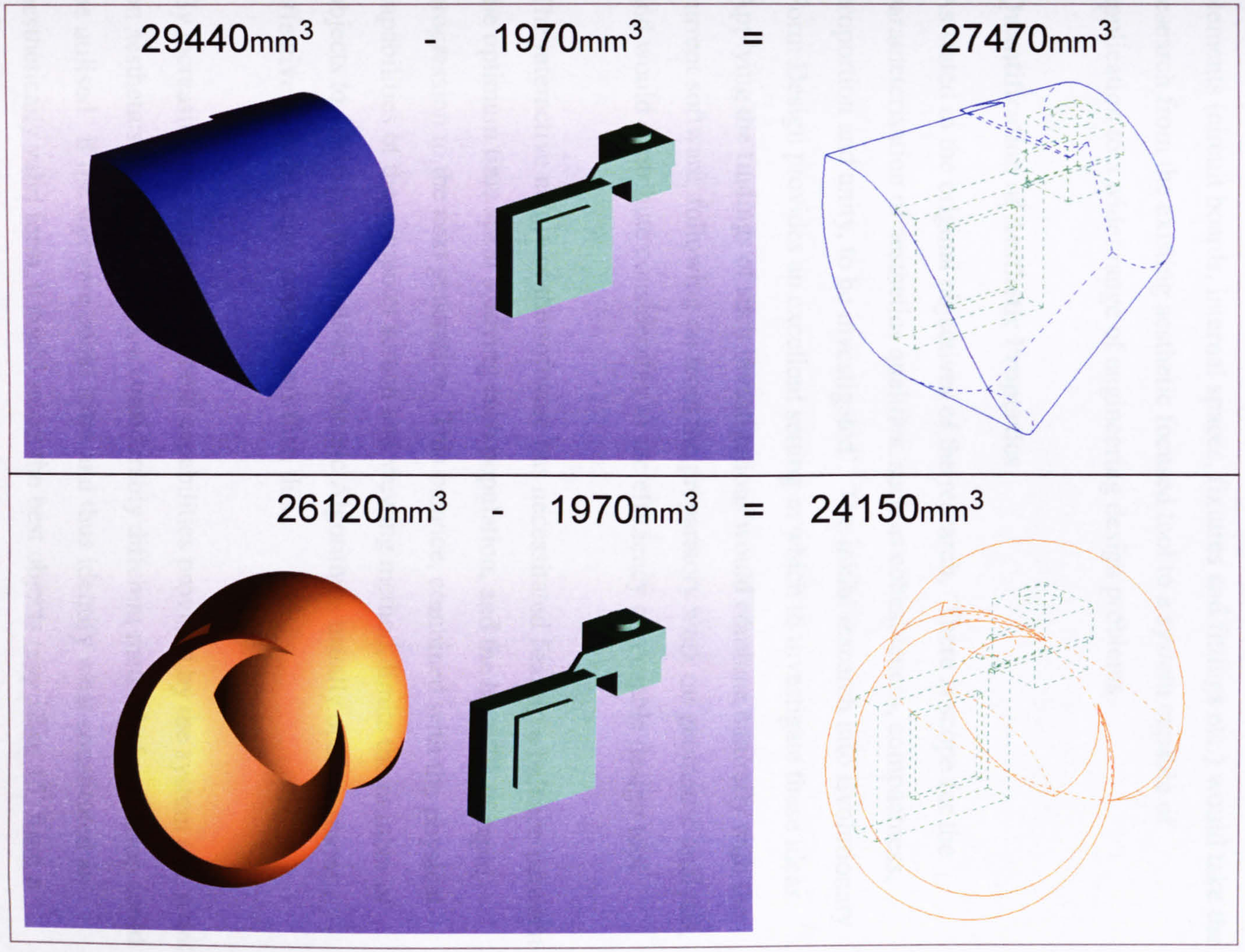


The evolved object does not enclose the box model (the remaining volume counts against the object's fitness), and would receive an 'enclose' objective function value of 170,

13230mm^3 - 1800mm^3 = 11430mm^3

The 'efficiency' objective function value would be 11430, the volume remaining after subtracting the box model from the object. This would be less influential than the 'enclose' objective function

Figure 6.3.1 - Demonstration of internal volumetric function



In these 2 cases, both objects completely enclose the box model, and would receive perfect 'enclose' objective values of 0.

In the lower example, the object has a smaller volume, so encloses more efficiently, and would therefore receive the better 'efficiency' objective value of 24150.

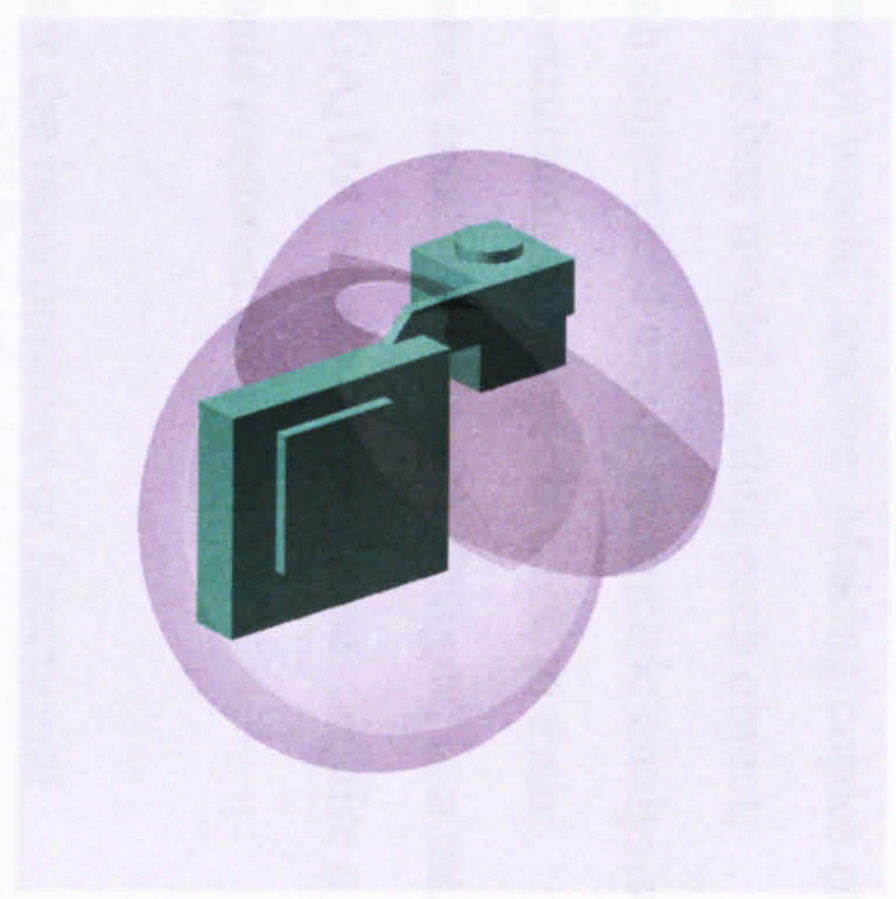


Figure 6.3.2 - Demonstration of internal volumetric function, efficiency aspect

The practicalities of programming this concept would involve: Making copies of the box model, finding the best position of the box model within each object, carrying out Boolean subtractions with each object and using geometric analysis to determine the remaining volumes - then returning the objects back to their previous state. The task of positioning the box model within the designs, is a task that would be best achieved using another GA, but would, in any case, require a lot of processing power to achieve the speeds required for usability.

The inclusion of internal size constraints for the incorporation of functional elements (circuit boards, internal spaces, fixtures and fittings etc.) would take the research from the existing aesthetic focused tool to a system capable of application to a wider range of engineering design problems.

Quantification of Aesthetic Properties

As stated in the original objectives of the research, 'There is scope for the parameterisation of aesthetics qualities, such as cohesiveness, compactness, proportion and unity, to be investigated'. The initial research into Evolutionary Form Design provides an excellent setting in which to investigate these ideas. Applying the findings of such investigations would combine naturally with the current software, following on from the preparatory work on geometric analysis, and would contribute considerably to the efficiency of a usable design tool.

The interactive nature of the software has necessitated finding a balance between the optimum time spent studying each population, and the benefits of rapid progression to the next generation. This balance, combined with the physical capabilities of the computer screen and viewing method, limits the number of objects to 12-16 per population. Genetic Algorithms usually operate (more effectively) with larger populations than this.

By increasing the automated fitness capabilities provided by the system (the work on aesthetics outlined above), a considerably different method of operation could be utilised. If the software could learn and thus identify what constituted an aesthetically valid form, it could select the best objects (say 12 or 16) from a

larger population (of around 30) to present to the user. This would perhaps double the efficiency of the software, drastically reducing the occurrence of unusable objects. This approach would provide a more rounded, more maintainable and more holistic route to efficiency than the general approach taken by this research (the trend towards detail changes and constraints during object creation).

The current software could be used to build up a database of 'good' objects, (limiting construction to just 2 primitives initially), which could then be analysed for parameters such as the following:

- ratio of sizes of primitives
- ratio of lengths within primitives
- which primitives work well together
- proportion of intersecting volume for each of the 3 Boolean operators

Teamforming

The ideas behind this technique could be applied to other problems where genotypes are made up of a number of repeated data-structures, and solutions are made up of collections of the same types of object, e.g.:

· In the case of the research discussed here, 5 identically structured groups of chromosomes were used in sequence to form the original genotype – solutions are constructed from 5 geometric primitives.

· In Bentley's GADES system, the genotype is made up of a variable number of blocks of 9 genes – solutions are constructed from a number of 'clipped stretched cuboids'.

· In the Emergent Design Group's Agency-GP, genotypes are made up of identically structured genes – scenes are constructed from collections of extruded NURBS curves.

6.4 Concluding Points

- Combining a small number of geometric primitives[1] using Boolean operations and applying edge blending, is a suitable technique for creating aesthetically interesting objects, and is capable of representing a range of products.
- By utilising the solid modelling capabilities of a CAD system, this product representation enables objects to be defined with a small amount of data, suitable for application by a Genetic Algorithm.
- A Genetic Algorithm can endow a design tool with usable evolutionary properties enabling the guiding of a population of objects towards an intended goal through intuitive interaction, resulting in the improvement of objects over a modest number of generations.
- Complementing user-supplied fitness with geometric and rudimentary aesthetic analysis can; increase the quality of populations presented to the user, confirm the successful optimisation capabilities of the GA, and demonstrate the potential for mechanical and further aesthetic automation .
- Treating constituent parts of an object (solution) as separate entities during the reproduction stage, and then combining them to form complete objects (solutions), in what has been termed a Teamforming stage, can provide a usable addition to the range of available techniques geared towards increasing GA efficiency.

---

[1] 5 or less

REFERENCES

affinnova — One Alewife Center, 4th Floor, Cambridge, MA 02140, USA
www.affinnova.com

Amin — **Amin S, Fernández-Villacañas J L,** "Dynamic Local Search", *Proceedings of the Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, pp 129-132, 1997

Anton — **Anton H,** "Calculus with Analytic Geometry, 4th edition", John Wiley & Sons Inc, 1992

Ashford — **Ashford F,** "The Aesthetics of Engineering Design", Business Books Ltd, 1969

Assad — **Assad A M, Packard N H,** "Emergent Colonisation in an Artificial Ecology", *'Toward A Practice of Autonomous Systems', Proceedings of the First European Conference on Artificial Life*, pp 143-152, 1992

Atick — **Atick J, Griffin P A, Redlich A N,** "The Vocabulary of Shape: Principal Shapes for Probing Perception and Neural Response", *Network: Computation in Neural Systems*, 7(1), IOP, 1996

Bäck — **Bäck T,** "Evolutionary Algorithms in Theory and Practice", Oxford University Press Inc, 1996

Baron — **Baron P, Fisher R, Tuson A, Mill F, Sherlock A,** "A voxel-based representation for evolutionary shape optimisation", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, 13(3), pp 145-156, 1999

Bentley1 — **Bentley P J, O'Reilly U M,** "Ten Steps to Make a Perfect Creative Evolutionary Design System", *GECCO 2001 Workshop on Non-Routine Design with Evolutionary Systems*
www.cs.usyd.edu.au/~josiah/gecco2001_workshop_schedule.html

Bentley2 — **Bentley P J, Wakefield J P,** "The Table: An Illustration of Evolutionary Design using Genetic Algorithms", *Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '95)*, pp 412-418, 1995

Bentley3 **Bentley P J**, "Generic Evolutionary Design of Solid Objects using a Genetic Algorithm", PhD thesis, University of Huddersfield, 1996

Bentley4 **Bentley P J**, "From Coffee Tables to Hospitals: Generic Evolutionary Design", Evolutionary Design By Computers, Morgan Kaufmann, 1999

Bentley5 **Bentley P J**, "An Introduction to Evolutionary Design by Computers", Evolutionary Design By Computers, Morgan Kaufmann, 1999

Byrne **Byrne M D, Mapfaira H**, "Assembly line balancing using genetic algorithms", *'Advances in Manufacturing Technology XIII', Proceedings of the Fifteenth National Conference on Manufacturing Research*, pp 119-124, 1999

Case **Case K, Graham I J, Wood R L, Abdul Karim M S**, "CAD Genetic Algorithms for Evolutionary Form and Function Design" , *'Advances in Manufacturing Technology XVI', Proceedings of the 18th National Conference on Manufacturing Research*, pp 103-107, 2002

Chouchoulas **Chouchoulas O**, "Shape Evolution: An Algorithmic Method for Conceptual Architectural Design Combining Shape Grammars and Genetic Algorithms", Artificial Intelligence in Design 2001, Poster Paper *www.bath.ac.uk/~abpoc/*

Collins **Collins R J**, "Ant Farm: Towards Simulated Evolution", *'Artificial Life II', Proceedings of the Workshop on Artificial Life February 1990*, pp 579-601, 1991

Cordón **Cordón O, José del Jesus M, Herrera F, Lozano M**, "An Evolutionary Paradigm for Designing Fuzzy Rule-Based Systems from Examples", *Proceedings of the Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, pp 139-144, 1997

Corne **Corne D, Ross P**, "Practical Issues and Recent Advances in Job- and Open-Shop Scheduling", *Evolutionary Algorithms in Engineering Applications*, pp 531-546, Springer, 1997

Corno **Corno F, Prinetto P, Rebaudengo M, Sonza Reorda M**, "Optimising Area Loss in Flat Glass Cutting", *Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, pp 450-455, 1997

Dahl **Dahl D W, Chattopadhyay A, Gorn G J,** "The importance of visualisation in concept design", *Design Studies*, 22(1), pp 5-26, Elsevier, 2001

Das **Das S, Franguiadakis T, Papka M, DeFanti T, Sandin D,** "A Genetic Programming Application in Virtual Reality", *Proceedings of the First IEEE Conference on Evolutionary Computing (ICEC '94)*, Vol 1, IEEE World Congress on Computational Intelligence, pp 480-484, 1994

Dasgupta **Dasgupta D, Michalewicz Z (Eds),** "Evolutionary Algorithms in Engineering Applications", Springer, 1997

Eby **Eby D, Averill R C, Punch III W F, Goodman E D,** "Optimal Design of Flywheels using an Injection Island Genetic Algorithm", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, 13(5), pp 327-340, 1999

Eckert1 **Eckert C, Kelly I, Stacey M,** "Interactive generative systems for conceptual design: A empirical perspective", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, 13(4), pp 303-320, 1999

Eckert2 **Eckert C, Stacey M,** "Sources if inspiration: a language of design", *Design Studies*, 21(5), pp523-538, 2000

EDG Emergent Design Group, Massachusetts Institute of Technology web.mit.edu/edgsrc/

Falkenauer **Falkenauer E,** "A Hybrid Grouping Genetic Algorithm for Bin Packing", *Journal of Heuristics*, 2(1), pp5-30, 1996

Forrest **Forrest S, Mayer-Kress G,** "Genetic Algorithms, Non-linear Dynamical Systems, and Models of International Security", *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, pp 166-185, 1991

Frazer **Frazer J,** "An Evolutionary Architecture", Architecture Association Publications, 1995

Furuta **Furuta H, Dogaki M, Teteishi K,** "Aesthetic design of arched bridges using genetic algorithms", *Proceedings of the Structures Congress*, Vol 2, pp 808-812, ASCE, 1997

GALESIA95 *First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '95)*, IEE

Conf Pub No 414, 1995

GALESIA97 *Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, IEE Conf Pub No 446, 1997

Garrett **Garret Jr J H**, "The computer-aided engineer: Prospects and risks", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, 12(1), pp 61-63, 1998

Gatarski **Gatarski R, Pontecorvo S**, "Breed Better Designs: the generative approach", *Designjournalen*, 6(1), SVID, 1999

Gen **Gen M, Cheng R**, "Genetic Algorithms and Engineering Design", John Wiley & Sons Inc, 1997

Gero **Gero J S, Kazakov V A, Schnier T**, Genetic Engineering and Design Problems, *Evolutionary Algorithms in Engineering Applications*, pp 47-68, Springer, 1997

Goldberg **Goldberg D E**, "Genetic Algorithms in Search, Optimisation and Machine Learning", Addison Wesley, 1989

Gonzalez **Gonzalez B, Torres M, Moreno J A**, "A hybrid genetic algorithm approach for the 'no-wait' flow-shop scheduling problem", *Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '95)*, pp 59-64, 1995

Graham1 **Graham I J**, "Development of a Multi-Species Genetic Algorithm", BEng Final year project report, Loughborough University, 1998

Graham2 **Graham I J, Case K, Wood R L**, "Evolutionary Form Design: The application of genetic algorithmic techniques to computer-aided product design", *'Advances in Manufacturing Technology XIII', Proceedings of the 15th National Conference on Manufacturing Research*, pp345-349, 1999

Graham3 **Graham I J, Case K, Wood R L**, "Evolutionary Computer Aided Design" , *2001 Congress on Evolutionary Computation (CEC 2001)*, Poster exhibit, 2001

Graham4 **Graham I J, Case K, Wood R L**, "Genetic Algorithms in Computer Aided Design", *Journal of Materials Processing Technology*, 117(1-2), pp 216-221, 2001

Graham5 **Graham I J, Case K, Wood R L,** "Genetic Algorithms in Computer Aided Design", *'Building on Manufacturing Advances of the Nineties, IMC-17', Proceedings of the 17th Annual Conference of the Irish Manufacturing Committee*, pp 47-53, 2000

Graham-Rowe **Graham-Rowe D,** "Designer genes", *Blueprint*, 186, ETP Ltd., August 2001

Grierson **Grierson D E, Prabhat H (Eds),** "Emergent Computing Methods in Engineering Design, Applications of Genetic Algorithms and Neural Networks", *NATO ASI Series, Series F*, Vol 149, Springer-Verlag, 1996

Guo **Guo B, Menon J,** "Local shape control for free-form solids in exact CSG representation", *Computer Aided Design*, 28(6/7), pp 483-493, 1996

Hemberg **Hemberg M, O'Reilly U M, Nordin P,** "GENR8: A Design Tool for Surface Generation", Late breaking paper at GECCO-2001, www.ai.mit.edu/projects/emergentDesign/genr8/

Holland1 **Holland J H,** "Genetic Algorithms and the Optimal Allocations of Trials", *SIAM Journal on Computing*, 2(2), pp 88-105, 1973

Holland2 **Holland J H,** "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975

Horn **Horn J, Nafpliotis N,** "A Niched Pareto Genetic Algorithm for Multi-Objective Optimisation", *Proceeding of the First IEEE Conference on Evolutionary Computing, IEEE World Congress on Computational Intelligence (ICEC '94)*, Vol 1, pp 82-87, 1994

Khatib **Khatib W, Fleming P J,** "An Introduction to Evolutionary Computing for Multidisciplinary Optimisation", *Proceedings of the Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, pp 7-12, 1997

Kephart **Kephart J O,** "How Topology Affects Population Dynamics", *'Artificial Life III', Proceedings of the Workshop on Artificial Life February 1992*, pp 447-463, 1994

Kumar **Kumar S, Bentley P J,** "Three Ways to Grow Designs: A Comparison of Embryogenies for an Evolutionary Design Problem", *Proceeding of the Genetic and Evolutionary Computation Conference (GECCO '99)*, pp35-43, 1999

Langton1 **Langton C G (Ed)**, *'Artificial Life', Proceedings of the First Workshop on Artificial Life*, Addison-Wesley, 1988

Langton2 **Langton C G, Taylor C, Farmer J D, Rasmussen S (Eds)**, *'Artificial Life II', Proceedings of the Workshop on Artificial Life February 1990*, Addison-Wesley, 1991

Langton3 **Langton C G (Ed)**, *'Artificial Life III', Proceedings of the Workshop on Artificial Life February 1992*, Addison-Wesley, 1994

Levy **Levy S**, "Artificial Life, The Quest for a New Creation", Penguin Books, 1993

Lewis **Lewis M**, "Aesthetic Evolutionary Design with Data Flow Networks", *Proceeding of the 3rd International Conference on Generative Art*, 2000 www.accad.ohio-state.edu/~mlewis/AED/Metavolve/Files/ga2k.pdf

Lindgren **Lindgren K, Nordahl M**, "Artificial Food Webs", *'Artificial Life III', Proceedings of the Workshop on Artificial Life February 1992*, pp 73-103, 1994

Macmillan **Macmillan S, Steele J, Austin S, Kirby P, Spence R**, "Development and verification of a generic framework for conceptual design", *Design Studies*, 22(2), pp169-191, 2001

Mill **Mill F, Sherlock A**, "Biological analogies in manufacturing", *Computers in Industry*, 43(2), pp 153-160, 2000

Nishino1 **Nishino H, Takagi H, Utsumiya K**, "A Digital Prototyping System for Designing Novel 3D Geometries", *Proceedings of the 6th International Conference on Virtual Systems and Multimedia*, 2000

Nishino2 **Nishino H, Takagi H, Cho S B, Utsumiya K**, "A 3D Modelling System for Creative Design", *Proceedings of the 15th International Conference on Information Networking (ICOIN-15)*, pp479-486, 2001

O'Reilly **O'Reilly U M, Testa P, Greenwold S, Hemberg M**, "Agency-GP: Agent-Based Genetic Programming for Design", Late breaking paper at GECCO-2001, www.ai.mit.edu/projects/emergentDesign/agency-gp/

Obayashi1 **Obayashi S**, "Aerodynamic inverse optimisation problems", Genetic Algorithms in Engineering Systems, IEE, 1997

Obayashi2 **Obayashi S, Tsukahara T, Nakamura T**, "Cascade Airfoil Design by

Multiobjective Genetic Algorithms", *Proceedings of the Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, pp 24-29, 1997

Ono — **Ono O, Watanabe G**, "Genetic Algorithms for Optimal Cutting", *Evolutionary Algorithms in Engineering Applications*, pp 515-530, Springer, 1997

Parmee — **Parmee I C, Bonham C R**, "Towards the support of innovative conceptual design through interactive designer/evolutionary computing strategies", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, 14(1), pp 3-16, 2000

Pearce — **Pearce R, Cowley P H**, "Use of Fuzzy Logic to overcome Constraint Problems in Genetic Algorithms", *First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '95)*, pp 13-17, 1995

Robinson — **Robinson G, El-Beltagy M, Keane A**, "Optimisation in Mechanical Design", *Evolutionary Design by Computers*, Morgan Kaufmann, pp 147-165, 1999

Rosenman — **Rosenman M A**, "The Generation of Form Using an Evolutionary Approach", *Evolutionary Algorithms in Engineering Applications*, pp 69-85, Springer, 1997

Rowbottom — **Rowbottom A**, "Evolutionary Art and Form", *Evolutionary Design by Computers*, Morgan Kaufmann, pp 261-227, 1999

Rowley — **Rowley T**, "A Toolkit for Visual Genetic Programming", University of Minnesota, 1994
http://www.geom.umn.edu/~trowley/genetic/report/report.html

Schroder — **Schroder P, Chipperfield A J, Fleming P J, Grum N**, "Multi-Objective Optimisation of Distributed Active Magnetic Bearing Controllers", *Proceedings of the Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, pp 13-18, 1997

Smith1 — **Smith R, Warrington S, Mill F**, "Shape Representation for Optimisation", *Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations*

and Applications (GALESIA '95), pp 112-117, 1995

Smith2 **Smith R E, Goldberg D E,** "Diploidy and Dominance in Artificial Genetic Search", *Complex Systems*, 6(3), pp 251-285, 1992

Smyth **Smyth S N, Wallace D R,** "Towards the Synthesis of Aesthetic Product Form", *Proceedings of the ASME DT Conferences*, DETC/DTM-14554, 2000

Soddu **Soddu C,** "Recognizability of the Idea: the evolutionary process of Argenia", *Proceedings of AISB'99 – Symposium on Artificial Intelligence and Scientific Creativity*, 1999

Taura1 **Taura T, Nagasaka I, Yamagishi A,** "Application of evolutionary programming to shape design", *Computer-Aided Design*, 30(1), pp 29-35, 1998

Taura2 **Taura T, Nagasaka I, Yamagishi A,** "Adaptive-growth-type 3D representation for configuration design", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, 13(3), pp 171-184, 1999

Todd **Todd S, Latham W,** "The Mutation and Growth of Art by Computers", *Evolutionary Design by Computers*, Morgan Kaufmann, pp 221-250, 1999

Tovey **Tovey M, Owen J,** "Sketching and direct CAD modelling in automotive design", *Design Studies*, 21(6), 569-588, 2000

Ulrich **Ulrich K T, Eppinger S D,** "Product Design and Development", McGraw Hill, 1995

Vihma **Vihma S,** "Products as Representations. A semiotic and aesthetic study of design products", Publication series of the University of Art and Design Helsinki, 1995

Wallace1 **Wallace D R, Jakiela M J,** "A Computer Model of Aesthetic Product Design: an approach to unify engineering and industrial design", Internal report, Massachusetts Institute of Technology, 1991

Wallace2 **Wallace D R, Jakiela M J,** "Automated Product Concept Design: Unifying Aesthetics and Engineering", *Computer Graphics and Applications*, 13(4), pp 66-75, 1993

Witbrock **Witbrock M, Neil-Reilly,** "Evolving Genetic Art", *Evolutionary Design by Computers*, Morgan Kaufmann, 1999

Wood1 **Wood R L,** "Genetic algorithm behaviour in the solution of an inverse thermal field problem", *Engineering Computations*, 13(5), pp 38-56, 1996

Wood2 **Wood R L,** "Genetic Algorithm Based Inverse Analysis", Internal report, Loughborough University, 1996

Yeager **Yeager L,** "PolyWorld: Life in a New Context", *'Artificial Life III', Proceedings of the Workshop on Artificial Life February 1992*, pp 263-297, 1994

Yee **Yee R,** "Golden proportion and aesthetic design of long-span bridges", *Transportation Research Record*, pp 36-46, National Research Council, 1998

Yu **Yu T, Bentley P J,** "Methods to Evolve Legal Phenotypes", Proceedings of the Fifth International Conference on Parallel Problem Solving From Nature", pp 280-282, 1998

Zalzala **Zalzala A M S, Fleming P J (Eds),** "Genetic Algorithms in Engineering Systems", IEE, 1997

APPENDIX A – PUBLICATIONS

Journal Papers

Graham, I.J., Case, K. and **Wood, R.L.**, "Genetic Algorithms in Computer Aided Design" , *Journal of Materials Processing* , 1171-2, November 2001, pp 216-221, ISSN 0924-0136

Conference Papers

Graham, I.J., Wood, R.L. and **Case, K.**, "Evolutionary Form Design: The Application of Genetic Algorithmic Techniques to Computer-Aided Product Design" , *'Advances in Manufacturing Technology XIII', Proceedings of the 15th National Conference on Manufacturing Research* , A.N. Bramley, A.R. Mileham, L.B. Newnes and G.W. Owen (Eds), Professional Engineering Publishing Ltd., University of Bath, September 1999, pp 345-349, ISBN 1-86058-227-3

Graham, I.J., Case, K. and **Wood, R.L.**, "Genetic Algorithms in Computer Aided Design", *'Building on Manufacturing Advances of the Nineties, IMC-17', Proceedings of the 17th Annual Conference of the Irish Manufacturing Committee* , P Donnellan (Ed), National University of Ireland, Galway, August 2000, pp 47-53, ISBN 0-9538974-0-0

Case, K., Graham, I.J., Wood, R.L. and **Abdul Karim, M.S.**, "CAD Genetic Algorithms for Evolutionary Form and Function Design" , *'Advances in Manufacturing Technology XVI', Proceedings of the 18th National Conference on Manufacturing Research* , K. Cheng, D Webb (Eds), Professional Engineering Publishing Ltd., Leeds Metropolitan University, September 2002, pp 103-107, ISBN 1-86058-378-4

Other Conference Contributions

Graham, I.J., Case, K. and **Wood, R.L.**, "Evolutionary Computer Aided Design" , Poster Exhibit, *2001 Congress on Evolutionary Computation (CEC 2001)* , Seoul, Korea, May 2001

APPENDIX B

DETAILED CROSSOVER EXAMPLE

iii. Table of decoded values for Parent and Child pairs

iv. Parents and Child pairs

v. Parent A

vi. Parent B

vii. Child α

viii. Child β

ix. Generation 4

x. Generation 5

Parent A

g4p1-blue

1	*cylinder*	43	48	19	1	0	1	57	65	44	*subtract*
2	*block*	48	30	37	0	0	-1	2	4	2	*subtract*
3	*block*	49	16	43	0	0	1	7	17	21	*unite*
4	*block*	39	19	48	1	-1	0	46	42	56	*create*
5	*block*	35	50	6	1	0	0	37	56	86	*subtract*
	PRIMITIVE	ORIGIN			VECTOR			LENGTHS			BOOLEAN

Parent B

g4p3-cyan

cylinder	43	48	19	1	0	1	59	27	53	*create*
block	38	30	5	0	0	-1	2	3	2	*subtract*
block	17	16	43	0	0	1	28	35	34	*intersect*
cylinder	55	59	48	1	-1	0	49	45	61	*create*
cylinder	35	50	6	1	0	1	54	45	67	*subtract*

Child α

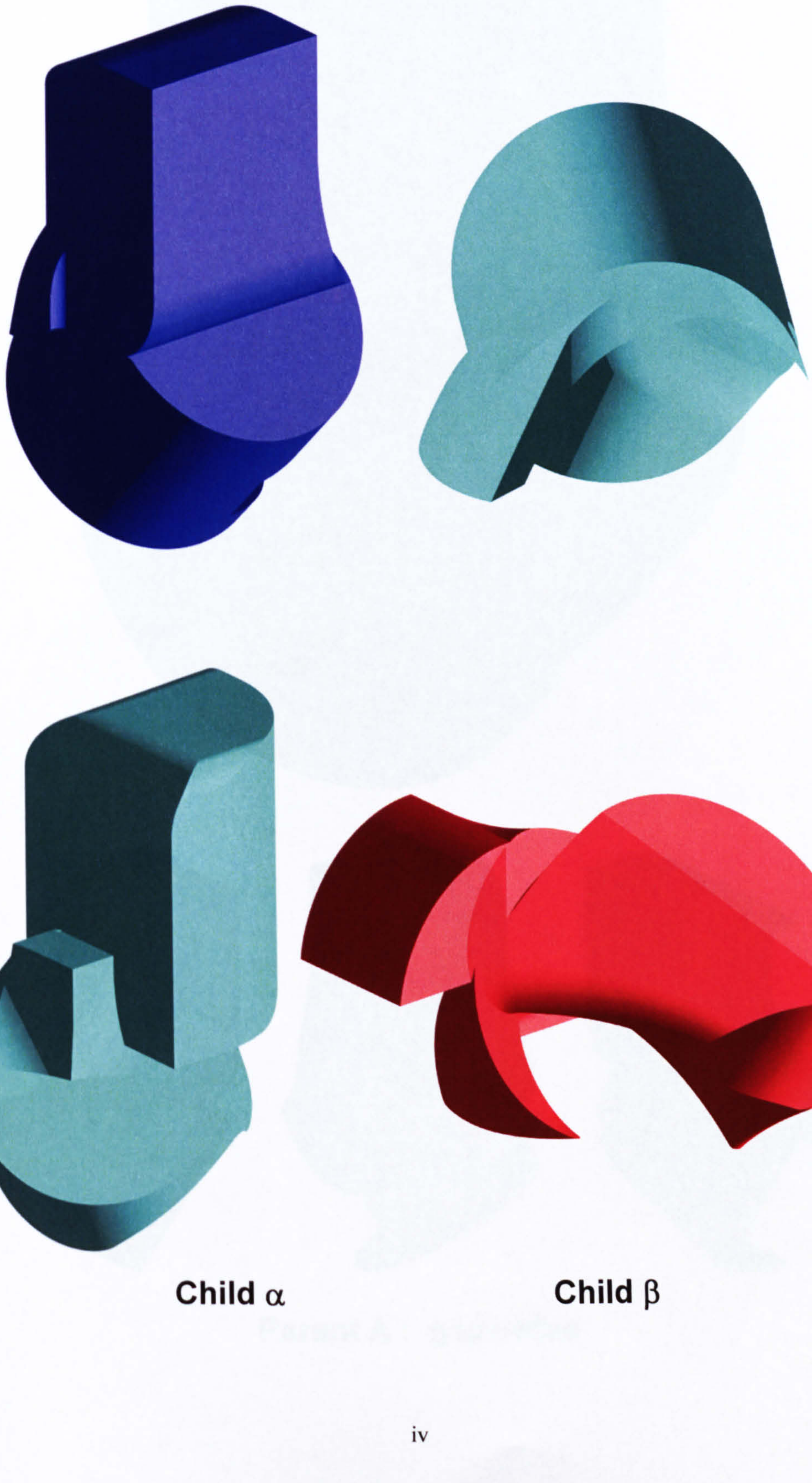
g5p3-cyan

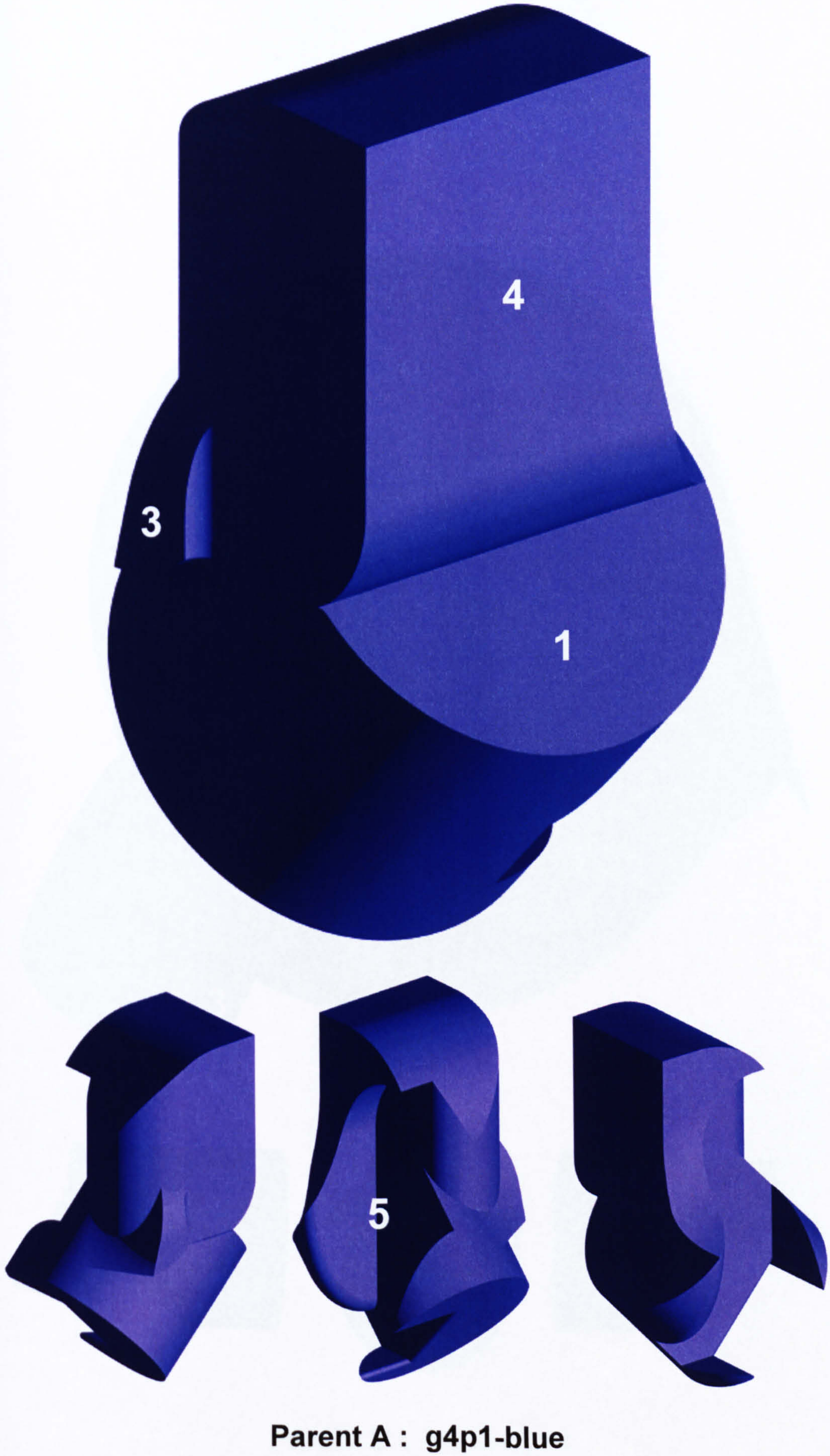
1	*cylinder*	43	48	19	1	-1	1	57	65	44	*create*
2	*block*	54	62	5	0	0	-1	2	3	2	*subtract*
3	*block*	49	16	43	0	0	1	19	17	23	*unite*
4	*block*	39	27	48	1	-1	0	49	45	61	*create*
5	*cylinder*	35	50	6	1	0	0	36	49	83	*subtract*

Child β

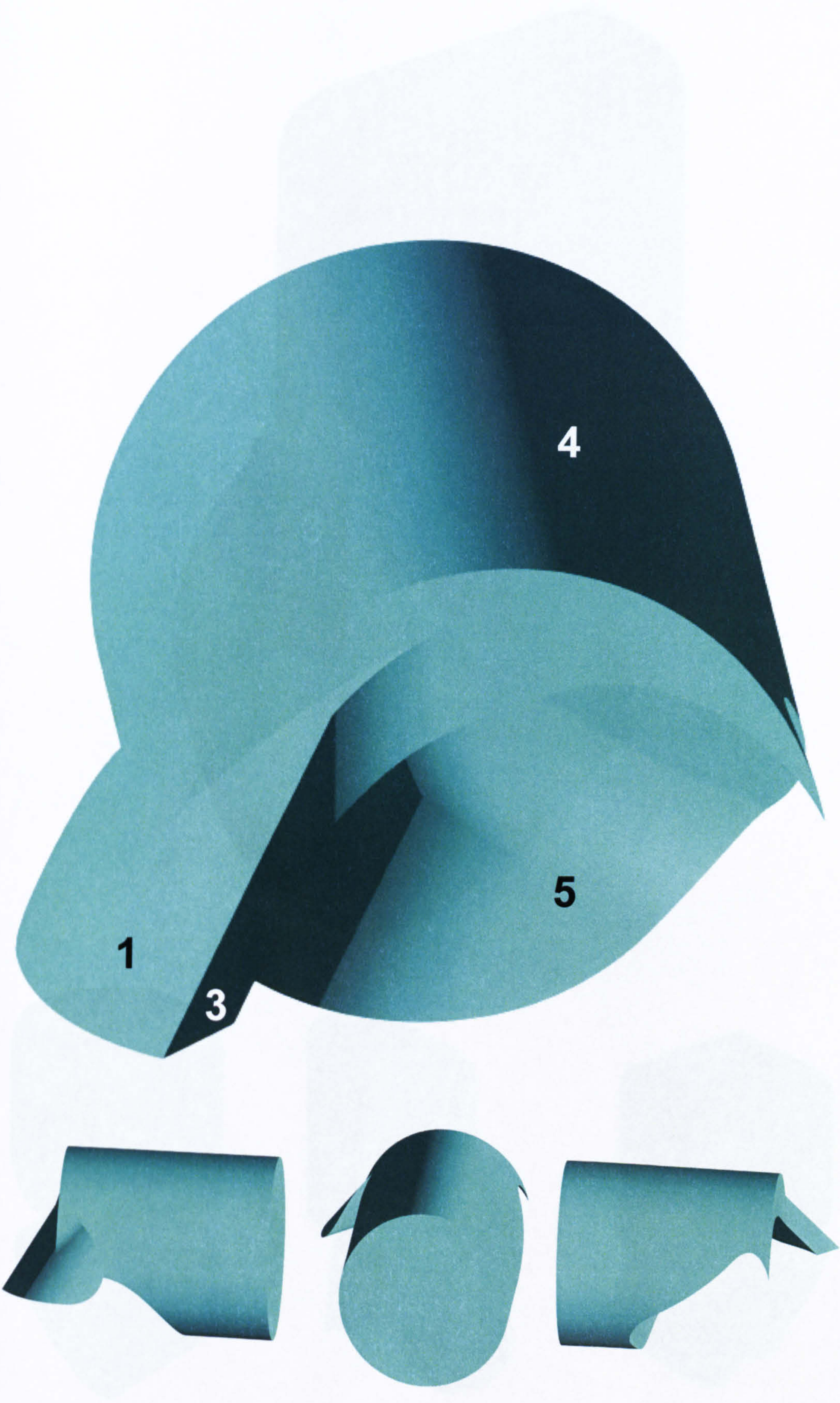
g5p4-red

cylinder	43	56	19	1	0	1	57	65	44	*create*
block	48	30	5	0	0	-1	2	4	2	*subtract*
block	17	16	43	0	0	1	26	24	21	*unite*
cylinder	55	51	48	1	-1	0	46	42	56	*subtract*
cylinder	35	50	6	1	0	0	52	48	65	*subtract*

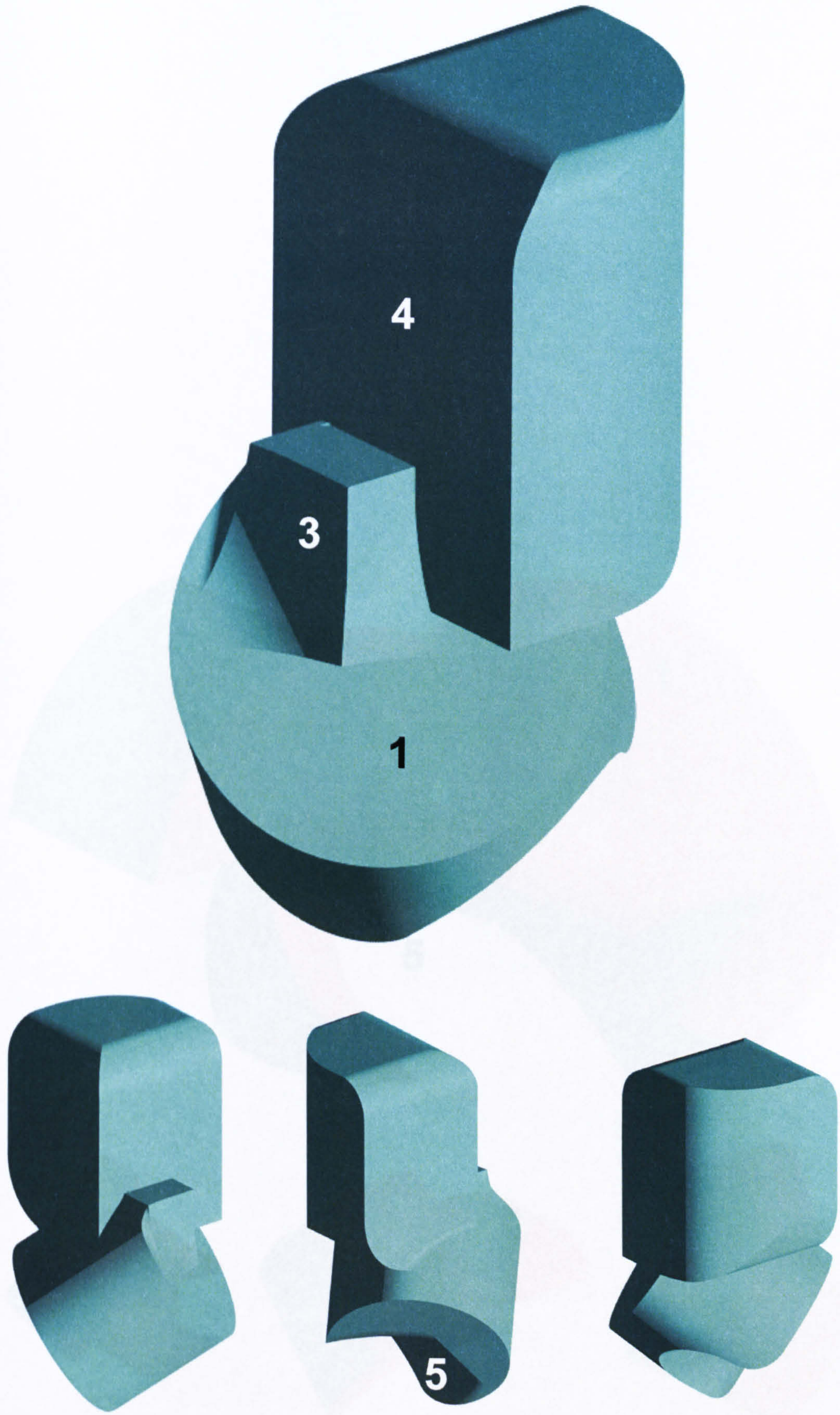

Parent A
Parent B
Child α
Child β



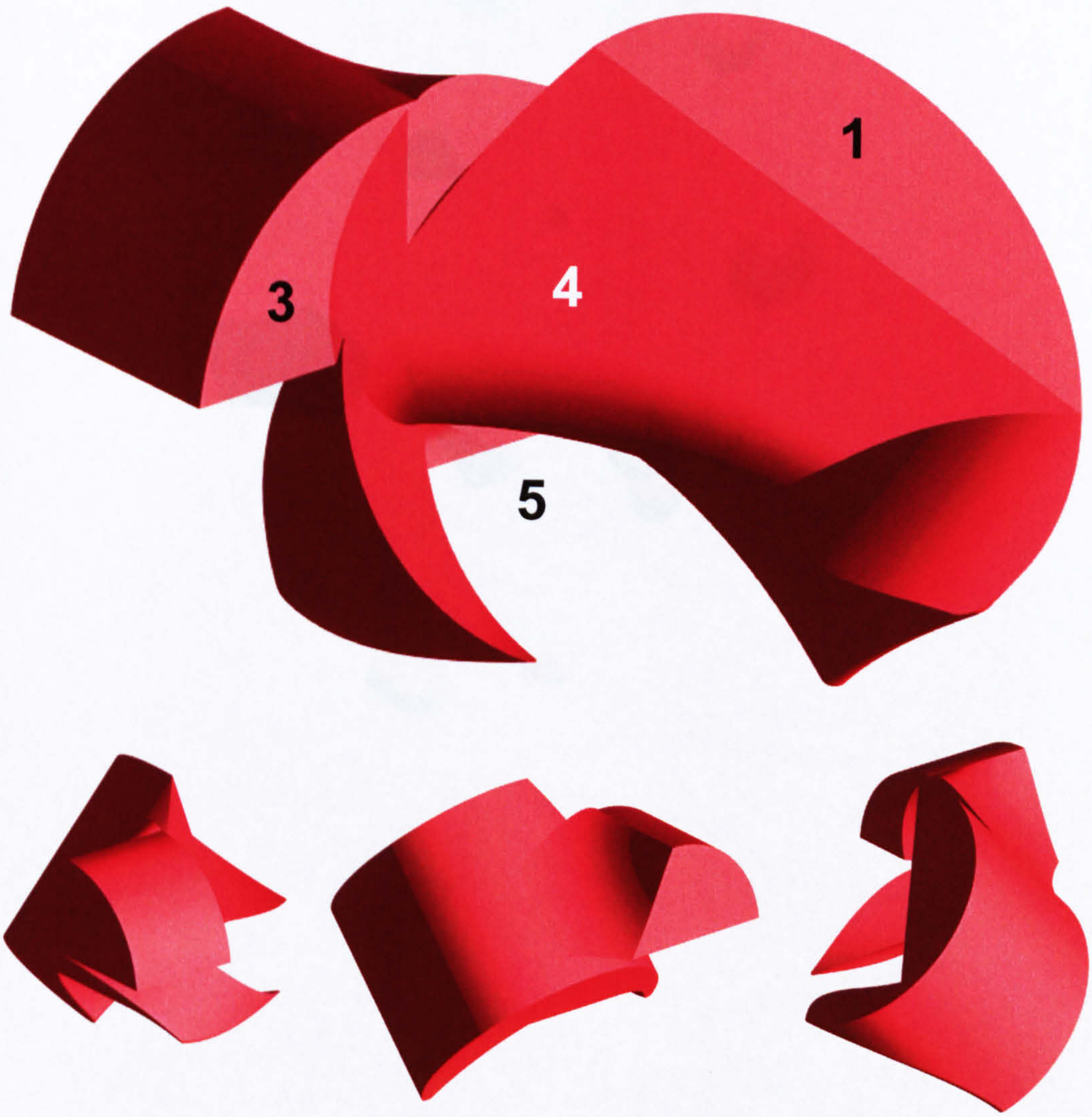
Parent A : g4p1-blue



Parent B : g4p3-cyan



Child α : g5p3-cyan



Child β : g5p4-red

Generation 4

Generation 5

APPENDIX C

PRODUCT CONCEPT ILLUSTRATIONS

xii. First three objects and associated populations

xiii. Yellow Plastic Office Chair

xiv. Brushed Stainless Steel Bottle Opener

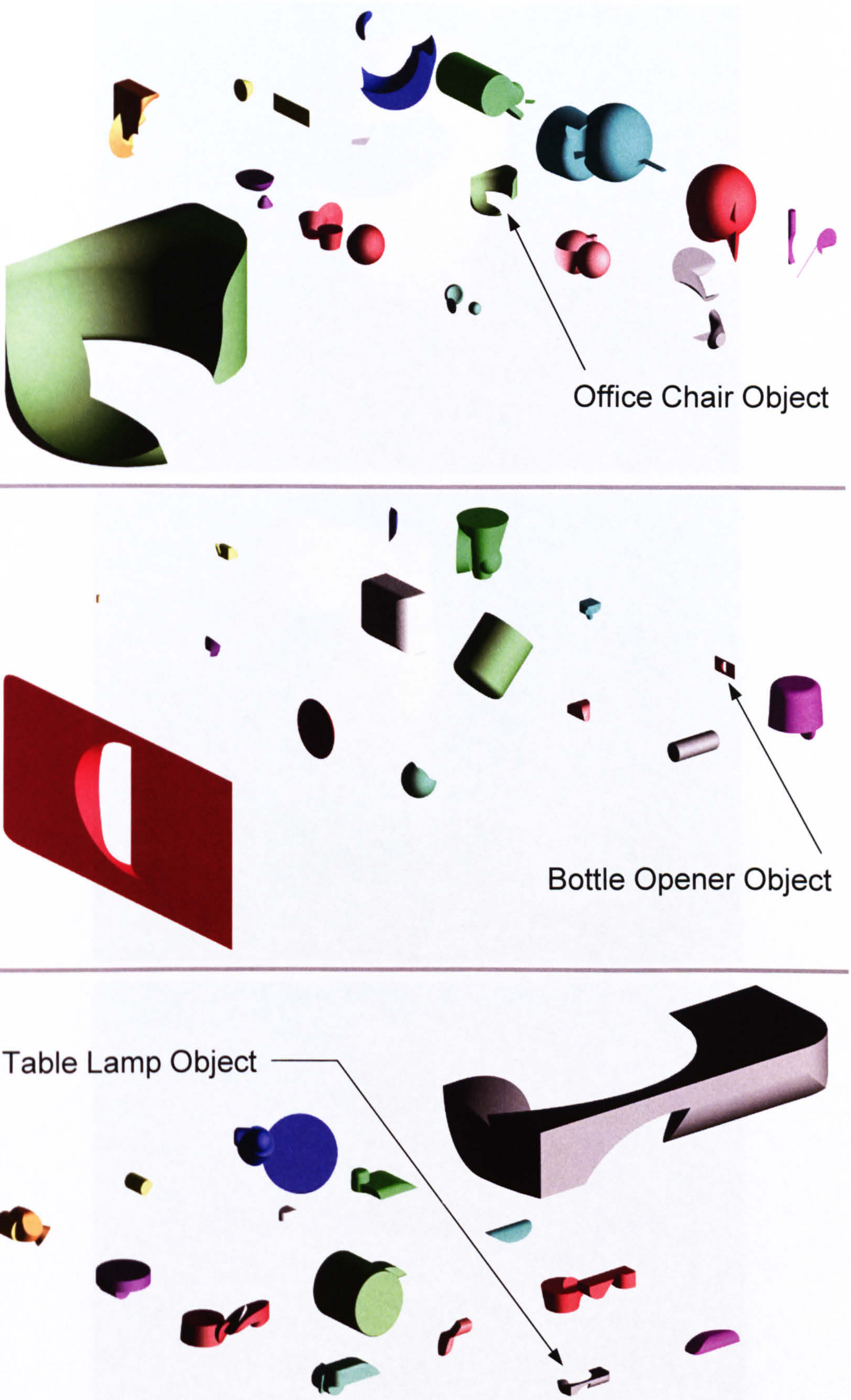
xv. **Blue Plastic Table Lamp**

xvi. Next three objects and associated populations

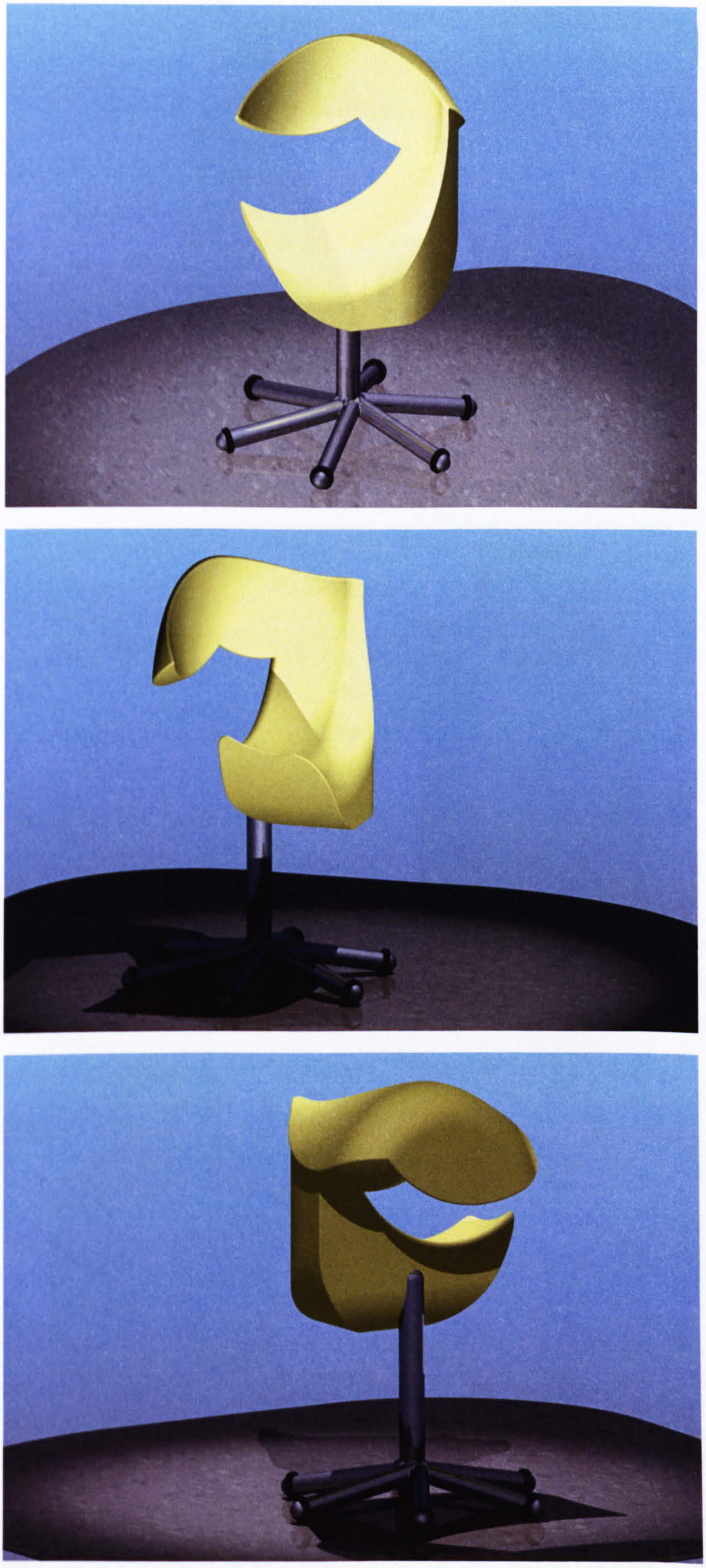
xvii. Compact Espresso Machine

xviii. Futuristic Mobile Phone Concept

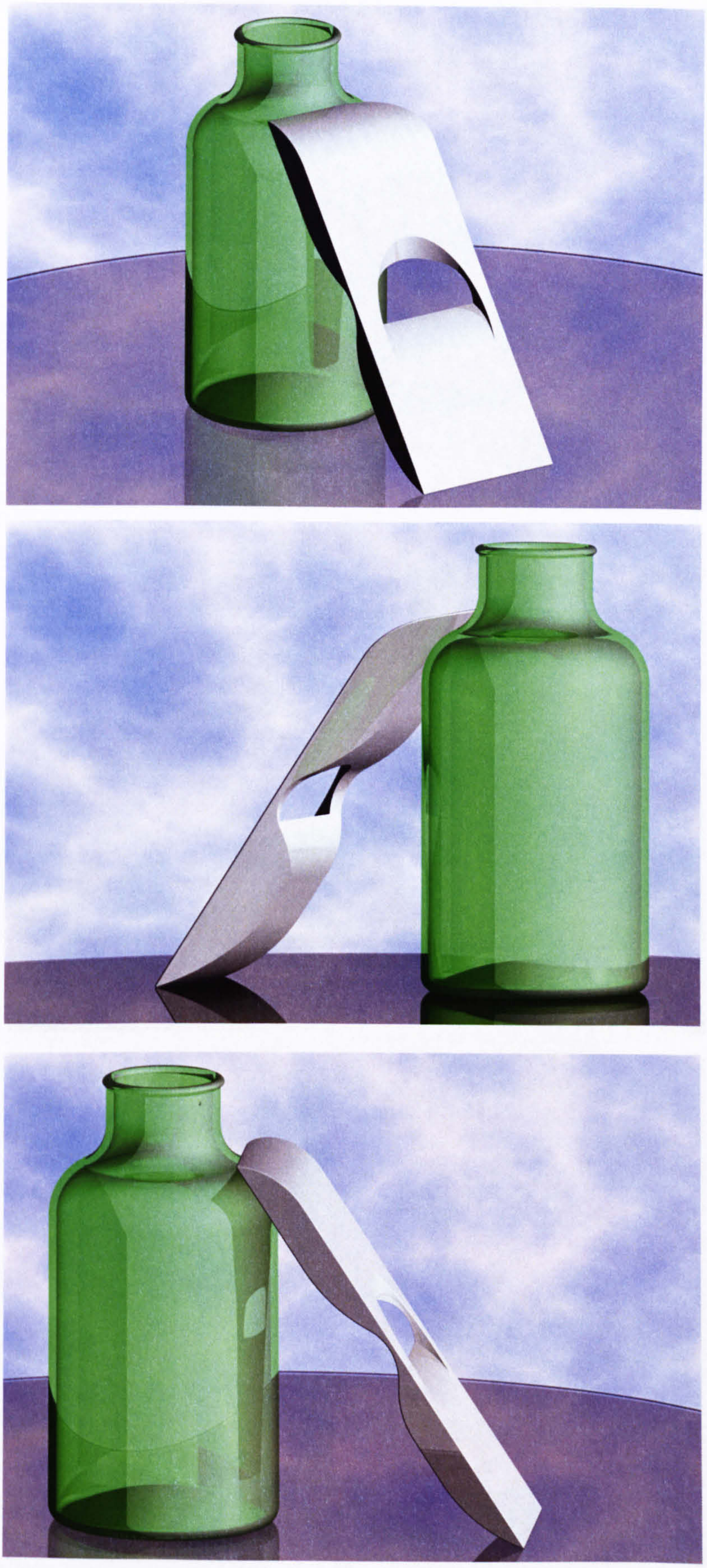
xix. White Ceramic Hand Basin



The evolved objects featured in the following product concept illustrations



Office Chair



Bottle Opener

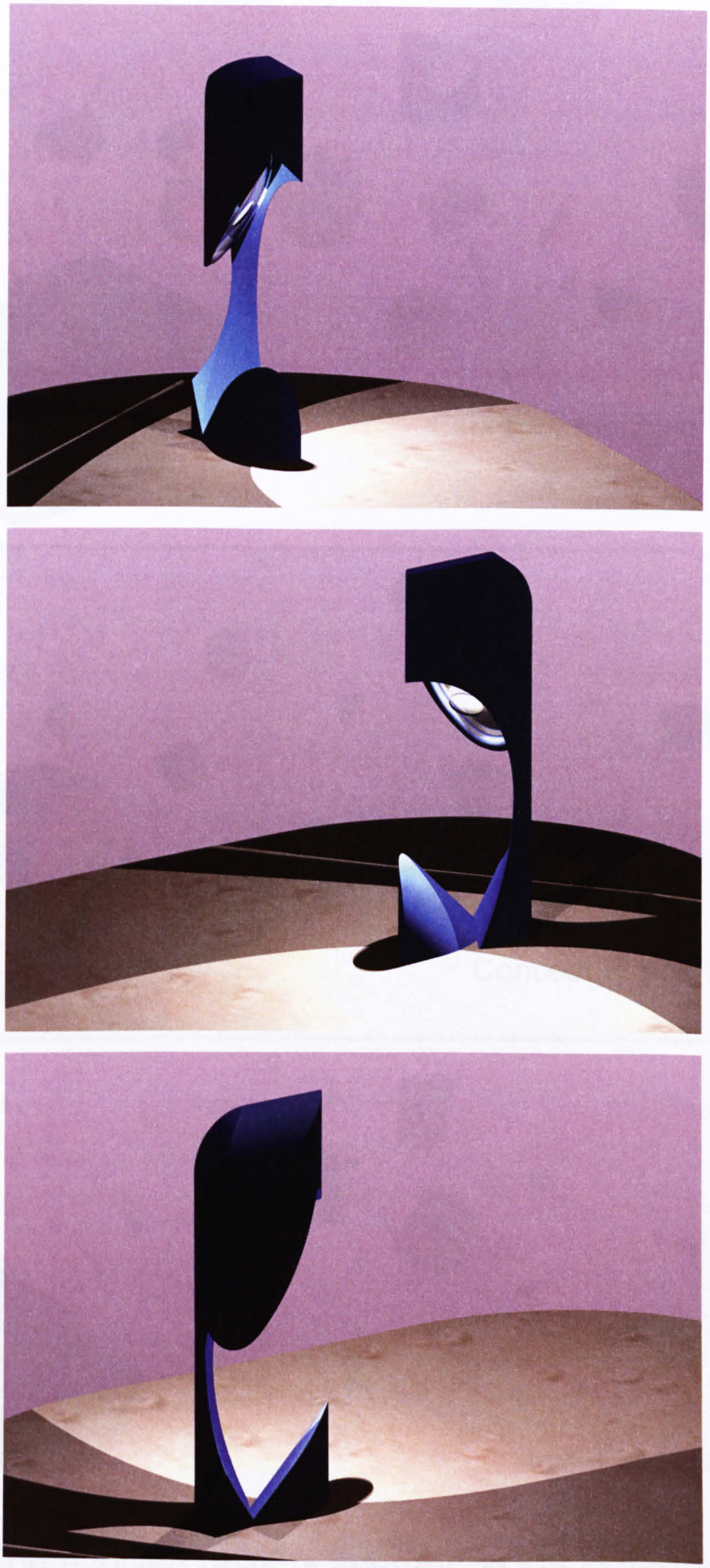
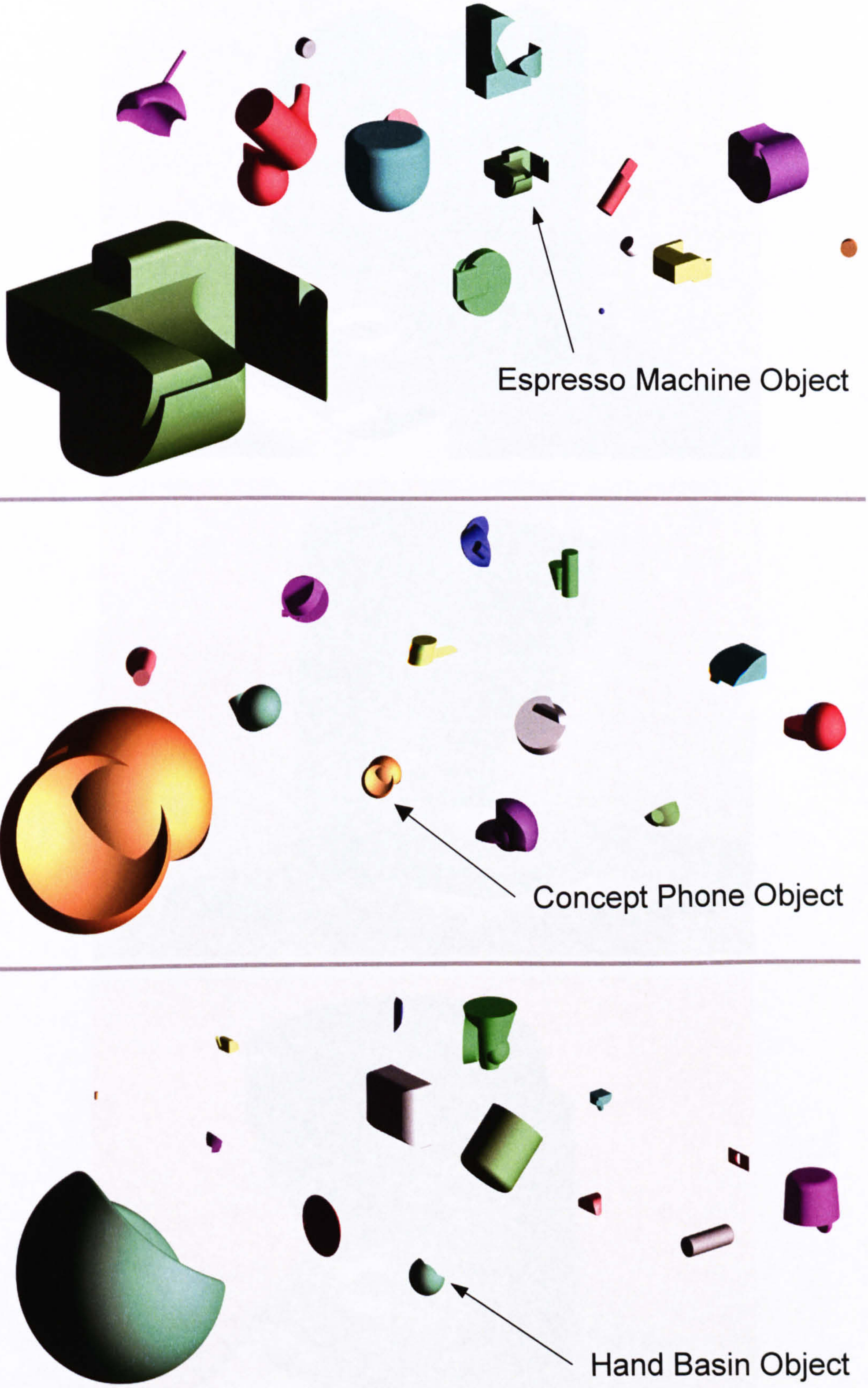
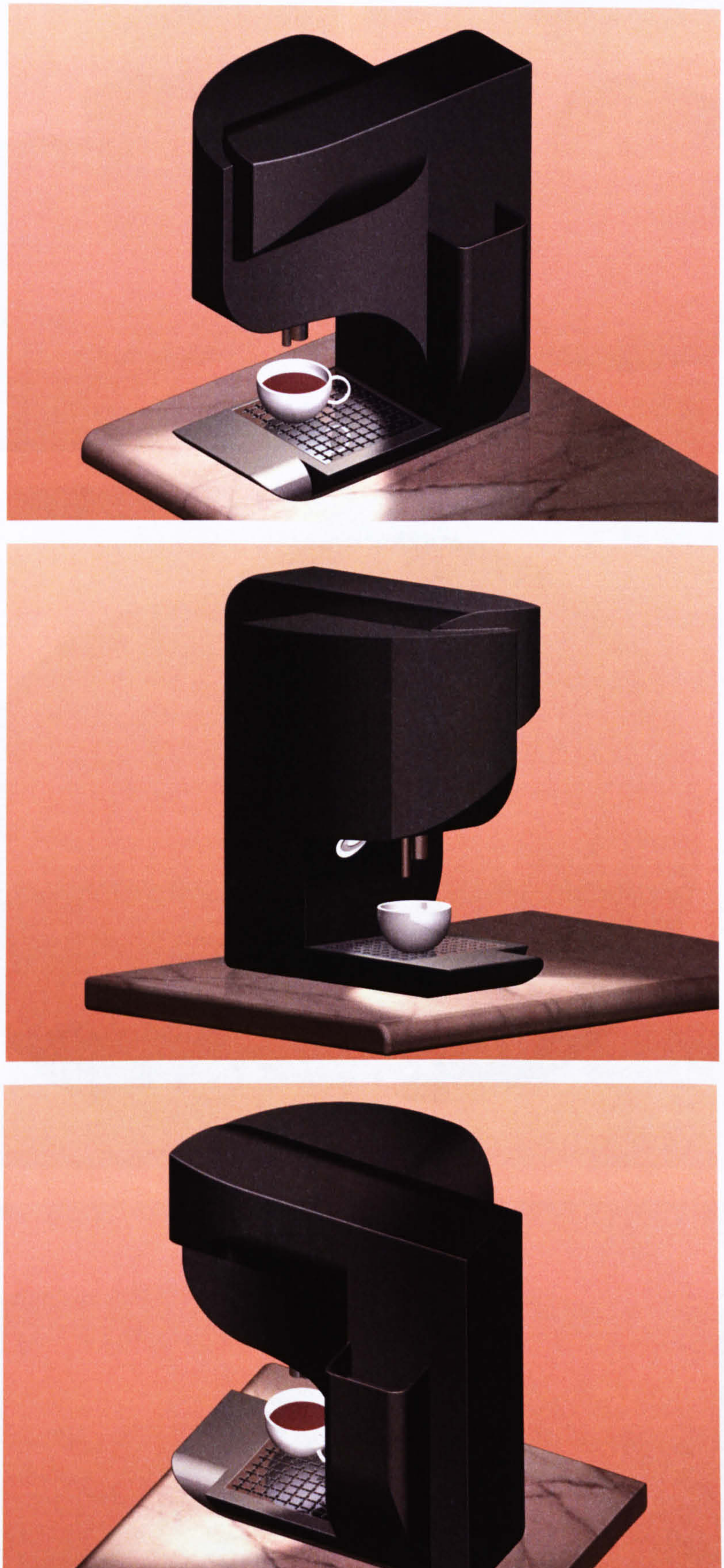


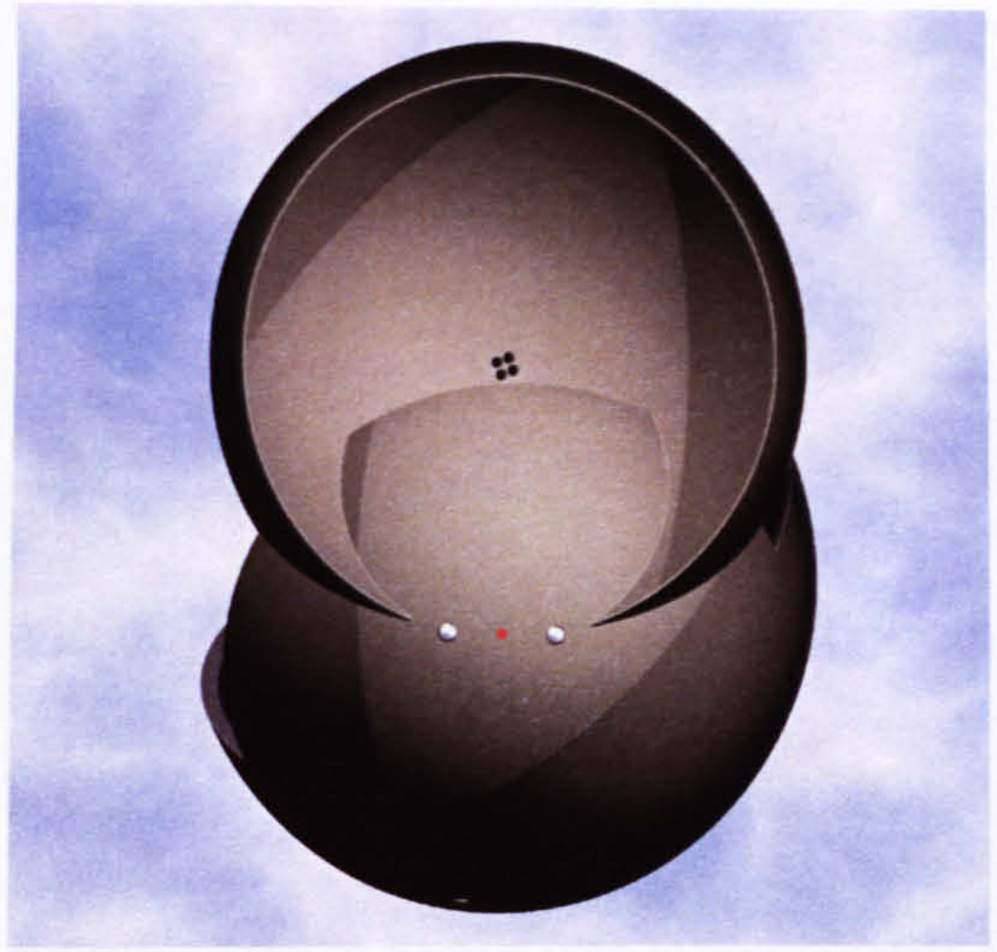
Table Lamp

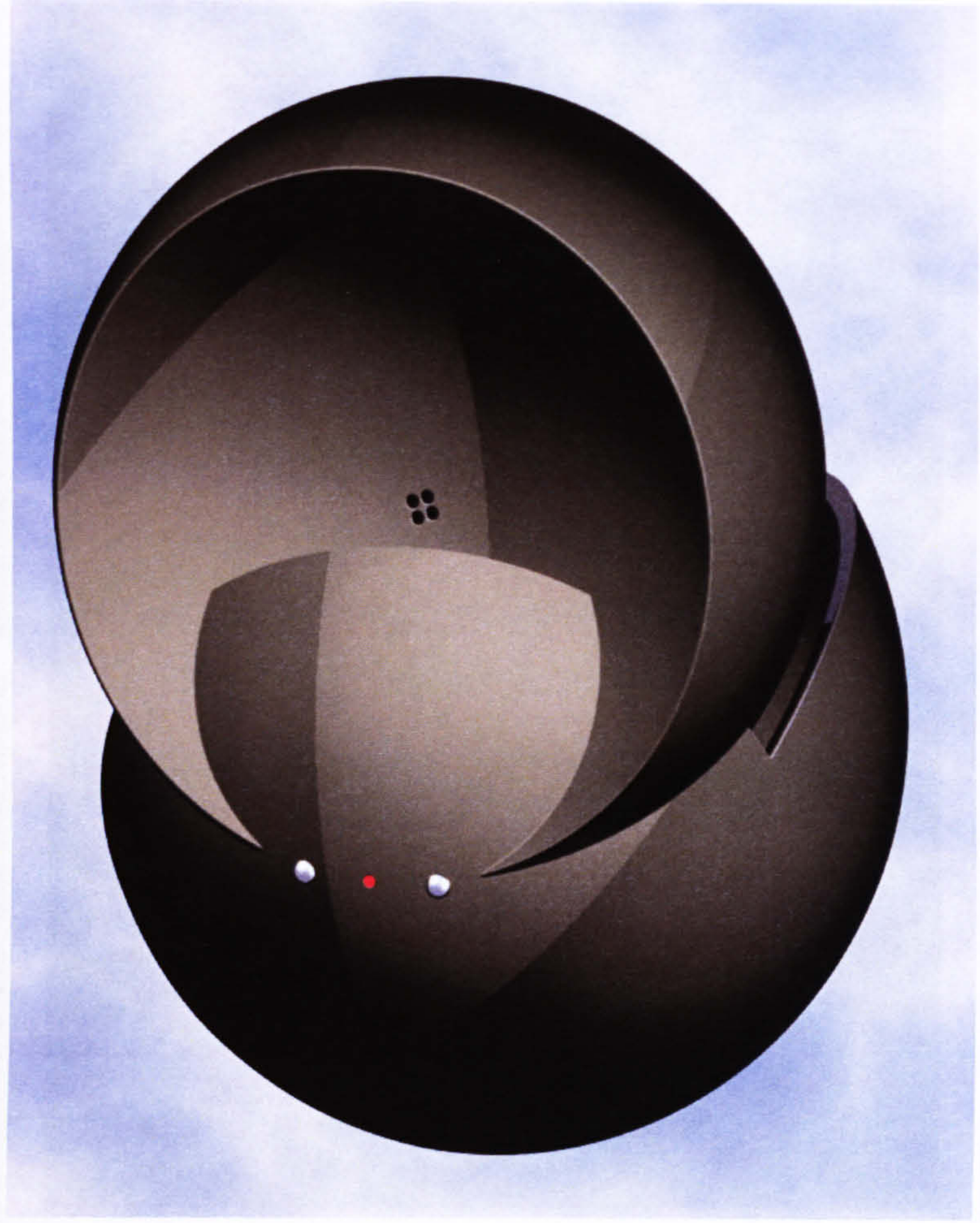


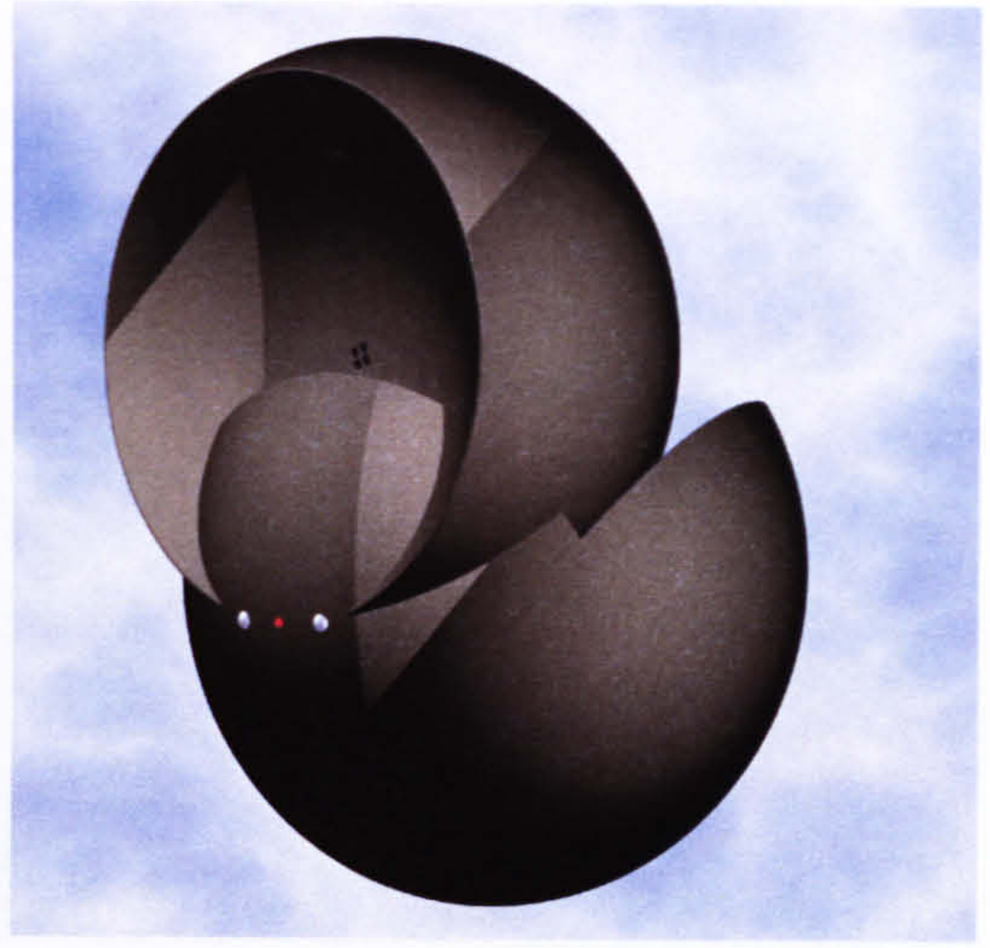
The evolved objects featured in the following product concept illustrations

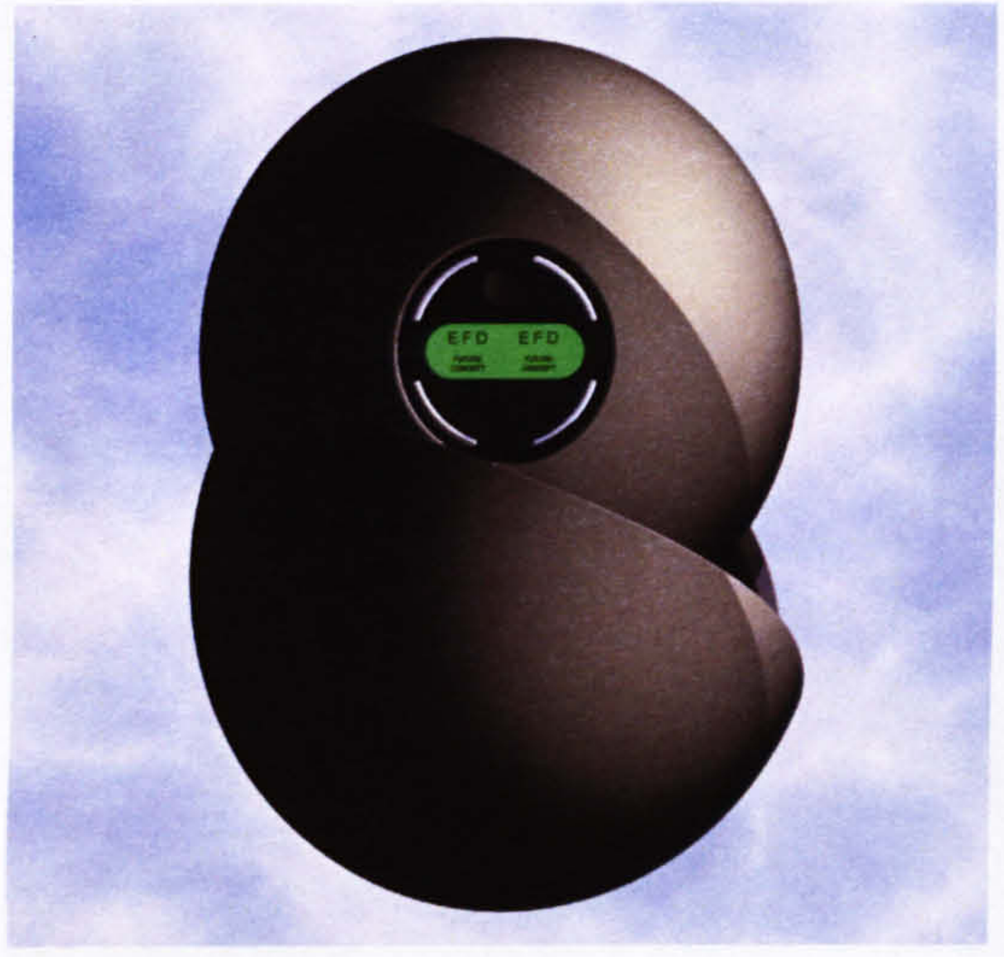


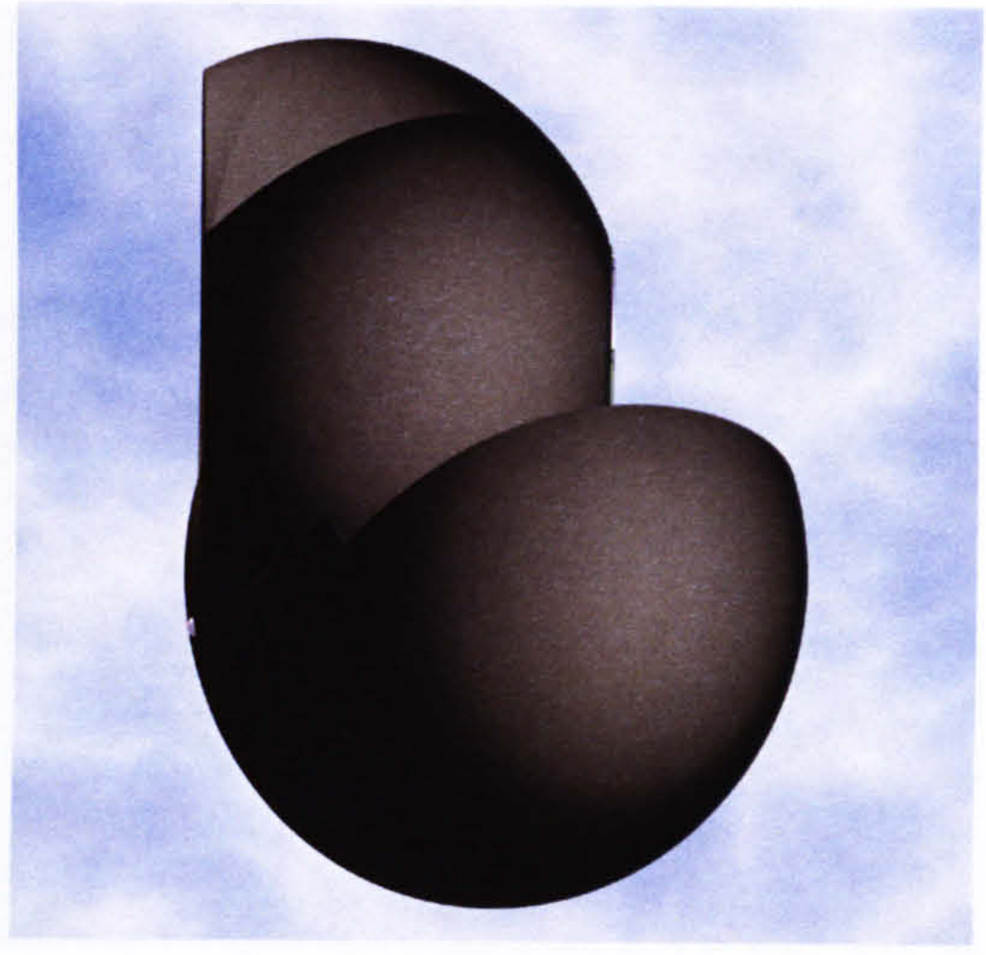
Espresso Machine



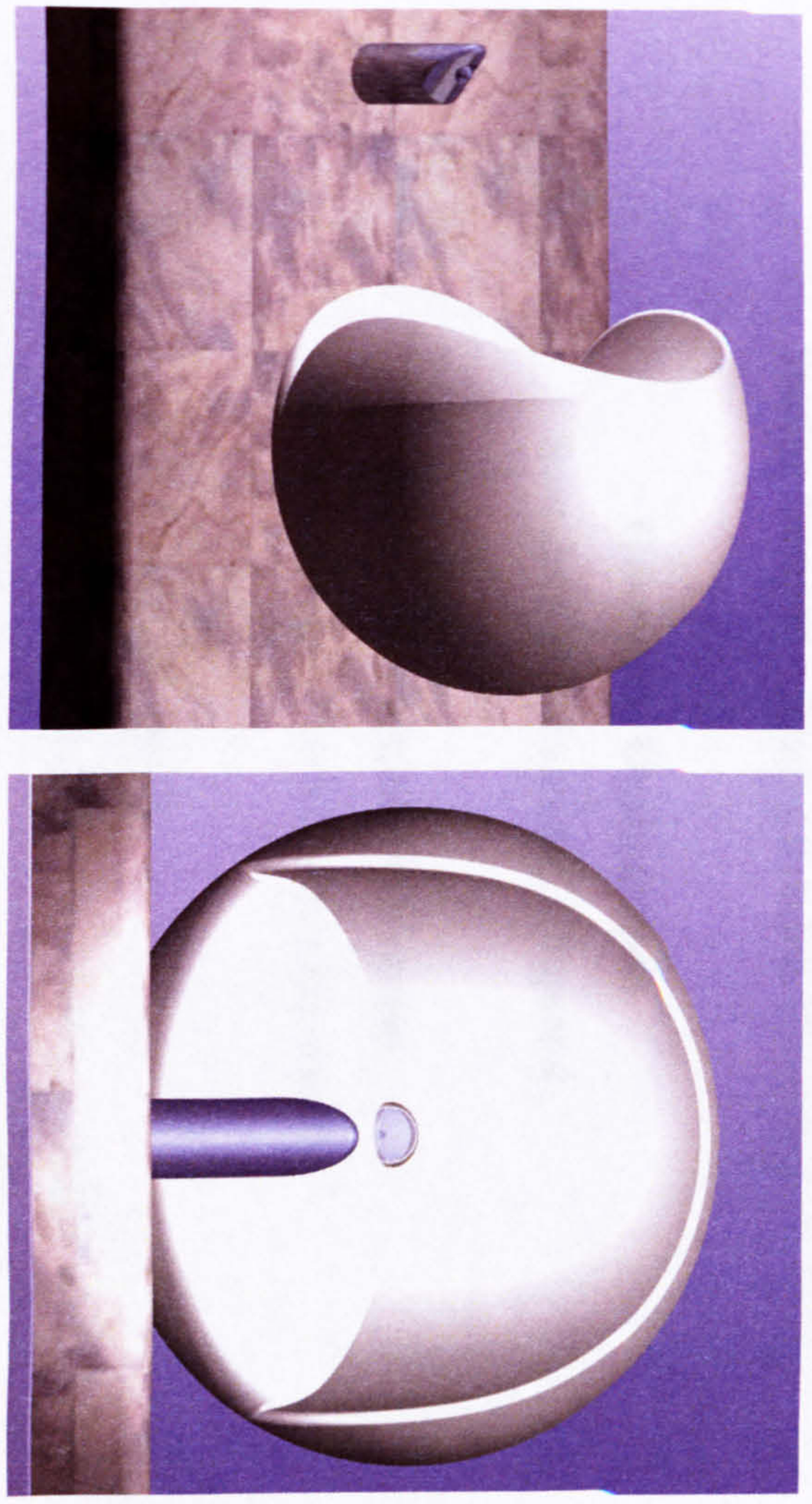








Concept Phone



Hand Basin

CD-ROM

EFD Executables

efd_menu_men

FirstEFD.dll

TeamForming.dll

Figures

LANDSCAPE.ppt

LANDSCAPE B&W.ppt

PORTRAIT.ppt

PORTRAIT B&W.ppt

Parts (Appendix B)

Detail40_004.prt

Detail40_005.prt

Slideshow Galleries

Animal Sculptures – CEC virtual sculptures, Chapter 5

EFD Products – Product concepts, Appendix C

Four Copper Objects – Freeform shape representation

Seating Designs – Seating designs, Chapter 5

Geometric Optimisation

Bounding Box Example, Chapter 4

Interactive Ancestor Diagram

Inheritance properties with blended objects

Thesis

Genetic Algorithms for Evolutionary Product Design