Integration of user-centred evolutionary computation with digital visualisation

Barry Dean
University of the West of England
bkd@barrydean.net

Ian Parmee
University of the West of England
ian.parmee@uwe.ac.uk

This research is concerned with applying evolutionary algorithms to explore the creation and optimisation of authentic and novel human character faces. Utilising computer software to create an authentic three-dimensional character face usually requires a graphic artist to spend a considerable amount of time and attention to painstaking detail to evolve authentic facial topology. Given that character faces share many common properties of proportion and relative positioning of facial components such as eyes, nose and mouth, then perhaps some novel face may be evolved from some standard base face through a user-centred application of a stochastic search process. Further, can such a system support the non-visually creative practitioner in the realisation of his or her own conceptual visualisations of an authentic character representation? An initial investigation applying these ideas within the context of a 3D modelling environment (Maya) to creating a human face and parametrically generate faces with varying characteristics for head shape, eye, nose, and mouth is reported. This work involves the integration of an interactive evolutionary process which utilises user-selected mesh components within the Maya environment.

Evolutionary computation. Interactive. Character face generation.

1. INTRODUCTION

Creation of 3D character art is time consuming for even seasoned practitioners. An experienced artist may subconsciously generate similar emergent features in facial characteristics; thereby unwittingly generate a familiar looking face in what intended to be an original work. A non-artist who lacks the required skills to create a character face manually may still have the ability to imagine and visualise varied and interesting human facial characteristics. Evolutionary computation may provide artists and non-specialists with a toolkit for exploration toward novel and authentic character face generation. The generation of authentic character faces has a number of applications. The notion of virtual characters within the context of real-time 3D environments, computer games and more recently film (Paouri, 1991) has led to an interest in the development of systems which can aid the production of authentic computer-generated characters. There is also a related application within the context of systems for the investigation of ageing (Wen-Bing, 2001). Research into facial recognition systems is a further related field (Yi, 2005).

2. EVOLUTIONARY COMPUTATION

The classic notion of a computer program is one that lists a sequence of instructions (input) that a computer processes to provide some usable output which has been derived from the input. This might be some text formatting task, solution to a mathematical equation, or a turn within the context of a computer generated game.

The written series of instructions is the *program*; the computer carrying out the set of instructions is the *process*. In classical computation the program code is fixed and unchanging during the duration of the process. Also the process does not have the facility to modify aspects of its internal algorithm's data unilaterally in an effort to direct it toward a new or improved result based on some internal or external feedback mechanism.

The internal processes of evolutionary computation possess the ability to modify the algorithmic data set in response to internal or external evaluative feedback.

Evolutionary computation attempts to mimic the evolutionary characteristics of populations based on parents and offspring as identified in natural biological systems. Possible solutions are

described as individuals. Each individual has genetic properties (genotype) that result in an emergent feature or quality (phenotype). New individuals (offspring) are generated by mating of parent individuals (crossover) which are then evaluated against some criteria (fitness) and included or discarded within the population accordingly. Mutation of genotype material also takes place to maintain some random aspects of phenotypic features within the population. How strongly the artificially computation seeks to mimic a natural system will vary depending on the type of evolutionary algorithm paradigm being used.

Evolutionary computation is a collective title that describes several well-established computer algorithms that can change their internal processes to search and evolve towards some predefined outcome or 'fitness'. Examples are Genetic Algorithms (GA), Evolutionary Programming (EP), and Evolutionary Strategies (Eiben, 2003). Such algorithms may be applied to a range of problems such as searching for optimal computer generated topology for an aircraft aerofoil design or a computer generated idealised representation of a human character face. The aerofoil design would be based upon a mathematical model describing, say, aerodynamic efficiency. This 'fitness function' determines the relative fitness of each design in each evolving 'population' of the evolutionary algorithm. However, the evolving character faces would typically be rated by a human user who would be monitoring and intervening with the search objectives of the evolutionary process. The latter example is the basic concept underpinning Interactive Evolutionary Computation (IEC), where a user-centred approach guides the evolutionary process toward some ideal goal (Takagi, 2001).

3. GENETIC ALGORITHMS

In general, evolutionary algorithms have a common framework upon which their processes are based (Dejong, 2005).

- A population of constant size m is evolved over time
- The current population is used as a source of parents to produce n offspring
- The expanded population is reduced from $m + n$ to m individuals.

One of the central evolutionary processes that are often deployed to generate an evolved set of optimisation scenarios with cost evaluation by user-centred ranking is the genetic algorithm.

Since their conception in the mid 1960's evolutionary algorithms have provided many novel and interesting solutions to problems in a wide

variety of domains. From the design of keyboards (Glover, 1987) and simple autonomous robots to the optimisation of gas pipelines (Goldberg, 1984) and evolution of computer generated art (Simms, 1991a & 1991b; Lutton, 2006), the fundamental principles involved in these examples are identical and are based on nature's own design strategy.

A genetic algorithm is constructed to follow nature's biological model of selection, recombination and mutation which is the basis of the Darwinian theory of evolution by natural selection, the survival of the fittest. The concept of a genetic algorithm is based on encoding a possible random solution as a string of values which model a biological chromosome.

The genetic algorithm (GA) has two basic implementations, the binary genetic algorithm, and the real-value genetic algorithm; some texts refer to the real-value genetic algorithm as the continuous genetic algorithm (Haupt, 2004).

The binary genetic algorithm is based on a bit-string of 0s and 1s of finite length which represents an individual from a randomly-generated population. As in the biological model each value of the bits in the string represents an allele in the chromosome. Depending on the problem domain, the numeric variables describing a random solution would be encoded as binary sub-strings in the chromosome and would be decoded to evaluate the relative fitness of that solution via the fitness function.

4. MODELLING A CHARACTER FACE

This section first outlines the process and methodology of changing a 3D character's facial characteristics by hand using the tools available in the modelling environment. These tools allow an artist to change the shape of a character's face manually by creating a copy of the original mesh and move groups of vertices in the copied mesh which, through associative connections, influence the topology of the original mesh. The degree of influence that the modified copy has over the original is under the manual control of the user. It then describes the process of integrating an evolutionary algorithm to control the degree of influence that each modified mesh copy will have on the original mesh.

When creating a character mesh in a 3D modelling environment there are choices to be made regarding the type of geometry used to create a mesh model – Polygons or NURBS (Non Uniform Rational Bi Splines). Through their underlying structure NURBS tend to yield more flowing organic shapes. However, they are not as adaptable as a polygon mesh structure especially when

considering the fine-grain modelling perceived to be required by an evolutionary system Polygon modelling has therefore been used for the creation and evolution of the character face. A base polygon character head mesh with a vertex count of 757, as shown in Figure 1 below, has been used in this first stage of the work.

Figure 1: Base Head geometry

4.1 Crafting facial characteristics

Each individual face shape and facial component characteristic is modelled by creating a copy of the original character geometry. For each required change of facial characteristic a copy of the original mesh is required. The characteristics of the copied mesh are then modified by hand selecting the required vertices to alter the topology of the mesh. Once the required vertex geometry has been modified the collection of modified vertices is then selected and a so-called vertex cluster is created, which has the effect of creating a selection set. The cluster weight influence may then be adjusted using a toolset which enables the area of selection to be more or less influenced by any transformation applied to the cluster such as scale, translation and rotation. Figure 2 below shows an example of cluster weights.

Figure 2: Cluster weights influence as shown in the environment.

The white area has maximum influence, black has zero influence. Using the tool the influence may be increased or reduced so to achieve fall-off on the

boundary where the geometry is only required to move over a small range as compared to the central areas of the cluster.

In the modelling environment the original character head mesh is referred to as the base geometry and each modified copy of the character head mesh is referred to as the target. Each target mesh created generates a so-called blendshape control slider to manipulate the range of change from the base character mesh to the maximum change in appearance under the influenced of the target character mesh. Each individual characteristic target mesh may influence the base mesh by moving the associated blendshape control slider. Figure 3 below shows an example of a base mesh (left) together with the result (right) of being influenced by a number of target mesh via the blendshape slider controls (bottom).

Figure 3: Base mesh (left), applied targets (right), blendshape control sliders (bottom)

A total of 15 individual target mesh shapes were created each with its own resulting blendshape slider. Seven targets for the individual head shapes, eye shape, eye rotation, eye position, nose size, nose shape, mouth shape, mouth size. A user could interactively move the blendshape sliders controls varying the amount of influence applied to the base mesh by dragging the respective slider using the mouse. However, the blendshape control sliders values may be also be accessed through the host environment's scripting language, the Maya Embedded Language (MEL). MEL is an interpreted language with c-like syntax. The entire Maya user interface, menu sets and GUI components are constructed via MEL statements and commands. The MEL script editor can be set to echo all commands, which is useful to see the appropriate MEL statement for generating a particular command or process. The general form of the MEL command for setting the value of a blendshape slider is achieved using the set attribute (setAttr) command with the appropriate numeric arguments. Thus it is at this interface that the output of an evolutionary algorithm was used to control the various facial characteristics in a parametric architecture. The number of possible

combinations of slider control positions for each target mesh is immense and it is this aspect that indicates a requirement for the integration of a user interactive stochastic search algorithm in order to efficiently explore and evolve 'good' solutions.

4.2. Integrating an evolutionary algorithm

In this work, given the implementation and natural biological characteristics of a GA which ostensibly map to the problem domain, the genetic algorithm has been chosen as the first evolutionary paradigm to be implemented in this research.

A Genetic Algorithm has been implemented using MEL. Each slider control is represented as a four-bit gene encoded as part of a 56-bit binary chromosome. The output of the GA is converted to its corresponding denary value; this denary value is used to control the respective blendshape target control between 0 and 1.

A fixed population size of ten was used where each individual resulting from a generation of the GA is rendered to its own window that includes cost buttons positioned at the bottom of the window. Each individual is given a cost weighting by the user interactively selecting the appropriate cost button from 1..10. The lowest 40% of the population is discarded and simply replaced with the offspring of the top four parents. There is no implementation of random pairing, weighting random pairing, or tournament selection required to converge to a preferred individual.

4.3 Implementing an EA using the Maya API

Given the restriction of MEL in terms of its lack of structured types and computational overhead as an interpreted language, the next stage of this work will involve working at the face and vertex-level generation of mesh geometry within a given upper and lower mesh envelope as shown in figure 4 below. These will act as upper and lower bounds for the GA. Some sub-envelopes of this volume such as the nose and mouth area will require the GA to have localised parameters. As the GA evolves a representative mesh structure through user centred interaction the solution envelope can be also manipulated to further guide and evolve the geometric form.

Figure 4: Central mesh show in green is contained within an inner and outer bound of vertex points which could be manipulated to guide the solution envelope of a GA

Maya, like other modelling environments, has an application programming interface (API) that provides 3rd party developers the opportunity to use C++ programming tools that extend the core functionality of the Maya application. Each version of Maya includes a software development kit (SDK) specific to a given platform e.g. Windows, OSX and Linux which features the library and header files necessary to build user designed modules known as plug-ins. These custom plug-ins can then be incorporated into the Maya software architecture. The SDK also includes template modules that fast-track development and provide developers with examples of various techniques and core functions.

The Maya software architecture is constructed using a data pipeline based on data and associated operations that are encapsulated as various individual nodes; the nodes are interconnected to form a dependency graph (Gould, 2003). Custom plug-ins can be created to implement custom commands and custom nodes. As nodes form the fundamental structure of the Maya software architecture then nodes provide the most sophisticated means for extending the applications functionality.

The node component which was chosen for the implementation of an api-based evolutionary algorithm was the custom deformer node. This node has functionality that can iterate over selected geometry and retrieve the individual vertex coordinate information for each value of x, y, and z in the selection set of vertices. The custom deformer node allows the developer to write an algorithm that acts on the acquired vertex data to deform the selected geometry in a desired fashion. Hence the custom deformer node is an ideal component for retrieving current vertex geometry that defines a facial structure in 3D coordinate space and to subsequently modify the coordinate

data via an evolutionary process interactively guided by a human evaluator.

The evolutionary algorithm has been implemented as a custom deformer node and communication with the process is achieved through the creation of various custom commands. The custom commands are themselves issued via a GUI interface which has been written using MEL. As illustrated in Figure 5 below.

Figure 5: command path to GA Node

Vertices in the facial area of the mesh are preselected via a MEL script as shown in Figure 6.

Figure 6: Selected vertex set

These 507 points will be collected as part of the coordinate data set once the deformer node is applied to the mesh. The associated coordinate points then form the basis of the data set that the initial GA population will act upon through modifying each original coordinate value by adding the current value of the GA population to each vertex x, y, z.

5. INITIAL RESULTS

Currently results are only available from the GA / MEL representation. Selecting Run GA from the control interface generates a new population and then selecting Render Face Shapes from the GA

control renders each member of the population to its corresponding window.

Figure 7 below shows individuals displayed from individual one (top left) then across and down for each row to individual ten. Individuals with obvious exaggerated features (4, 7, 9) were then given low ratings and others rated according to preferred facial individuality.

Figure 7: Rendered face shapes computed from the initial binary population

Each individual is given a cost weighting by selecting the appropriate cost button from 1..10. The lowest 40% of the population is discarded and simply replaced with the offspring of the top four parents; there is no implementation of random pairing, weighting random pairing, or tournament selection. Figure 8 below shows the output after five further generations of the GA.

Figure 8: Sixth Generation

The results from running the GA seem to converge as long as the mutation rate is set low, given the small population then a high mutation rate will have an adverse effect on the outcome. The initial population plays an important factor in determining the outcome and speed of convergence as an extreme initial set of initial individuals – where

values are at or near to the extreme settings, increases the time

5.1 GA representation design

In the second phase of this work an evolutionary algorithm has been integrated within the Maya API. However, the main issue surrounding using a GA approach is the large scale problem of dealing with a binary string which needs to be of a sufficient length to accommodate three vertex points (x, y, z) and the sign (+ or -) for every vertex in the selected set. Given there are 507 vertices in the selection set as shown in Figure 6 then for each vertex coordinate point there needs to be a gene of length 6 to modify existing vertex points in x, y, z between appropriate inner and outer bounds gives $6 \times 3 \times 507 = 9126$. Thus each individual (chromosome) requires a binary string of length 9,126. Such values yield considerably large-scale data sets on which to process and evolve. Alternative evolutionary algorithms such as evolutionary strategies working with real value data would have a more appropriate mapping of data to the problem domain.

6. CONCLUSIONS

The results from early work provide an example of how parametric facial features may be controlled by an evolutionary algorithm, in this case a GA. Such a system could be utilised in the context of existing character face customisation processes that facilitate the evolution of novel avatar characters which feature in virtual environments (Sony, 2010). However, a parametric approach constrains the possible exploration space of an evolutionary algorithm in a wider context. Despite a common set of geometric rules governing the positioning of facial components, the human face still varies considerably. The application of parametric values works better when the entire search space can be parameterised. An example would be using an evolutionary algorithm to search the possible parameter set for the application of control values in music synthesis such as the Bezier Synthesiser (Lang, 2004). In this context the solution space is closed and the evolutionary process will be able to search for all possible solutions. The human face is too varying and complex to parameterise fully and thus an evolutionary process will not have the opportunity to explore a wide and varied space of possible solutions. The complex nature of facial anatomy, as well as physiological factors which influence the user's choices over the selection of a particular individual do not exploit the full potential of evolutionary computation as a tool for evolved exploration. The integration of an API based evolutionary process working at the vertex component level provides a much richer exploration space but necessitates large scale data sets which

are not entirely suited to a genetic algorithm. Work is now involved with the integration of alternative evolutionary paradigms that better facilitate the processes and evolution of large-scale data sets. Currently, evolutionary strategies based on real number data forms the basis of activity in this work.

7. REFERENCES

- Eiben, A. and Smith J. (2003) *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin.
- Dejong, K. A. (2005) *Evolutionary Computation: a Unified Approach*. MIT Press, Cambridge Massachusetts.
- Glover, D.E. (1987). Solving a complex keyboard configuration problem through generalized adaptive search. In L. Davis (Ed.) *Genetic algorithms and simulated annealing*. pp. 12–31. Pitman, London.
- Goldberg D.E.(1984). Computer-aided pipeline operation using genetic algorithms and rule learning. Paper presented at the 1984 API Pipeline Cybernetics Symposium Houston Texas.
- Gould, D.A. (2003) *Complete Maya Programming*. Morgan Kaufmann, San Diego.
- Haupt, R. and Haupt, S. (2004). *Practical Genetic Algorithms*. John Wiley & Sons, Inc. New Jersey.
- Lang, L. (2004) Waveform Synthesis Using Bezier Curves with Control Point Modulation In *116th conference of the Audio Engineering Society*.
- Lutton, E. (2006) Evolution of Fractal Shapes for Artists and Designers. *International Journal on Artificial Intelligence Tools*, vol 15 no 4, pp. 651–672.
- Paouri, A.,Thalmann, N. and Thalmann D. (1991) Creating Realistic Three-Dimensional Human Shape Characters for Computer-Generated Films. In *Proc. Computer Animation '91*, Springer-Verlag.
- Sims K. (1991a), Artificial Evolution for Computer Graphics. In *Computer Graphics, ACM SIGGRAPH Proceedings*, vol 25, pp. 319–328.
- Sims K. (1991b), Interactive Evolution of Dynamical Systems. In *First European Conference on Artificial Life*, MIT Press.
- Sony PlayStation Home (2010) *User Avatar Interactive Character Face Definition Process*, Sony Corporation Inc.
- Takagi, H. (2001) Interactive Evolutionary Computation: Fusion of the Capacities of EC Optimization and Human Evaluation. In

Proceedings of the IEEE, vol 89, no 9, pp. 1275–1296.

Wen-Bing H., Cheng-Ping L. Chun-Wen C. (2001) Classification of Age Groups Based on Facial Features, *Tamkang Journal of Science and Engineering*, Vol. 4, No. 3, pp. 183–192

Yi , S. and Lijun Y. (2005) *A Genetic Algorithm Based Feature Selection Approach for 3D Face Recognition*. State University of New York at Binghamton, Binghamton, New York 13902.