Towards Crowd-Sourced Parameter Optimisation for Procedural Animation

Gareth I. Henshall, Christopher J. Headleand,
William J. Teahan and Llyr Ap Cenydd
School of Computer Science
Bangor University
Bangor, Wales, UK
Email: g.i.henshall@bangor.ac.uk

***Abstract*—Procedural animation systems are capable of synthesizing vivid organic character motion automatically. However, these systems can consist of hundreds of interlinked parameters, yielding a very large search space of potential animations and emergent behaviours. Parametrised Animation is also notoriously difficult to automatically optimise, as the resultant motion or behaviour can be subtle, complex and subjective.**

We describe a web-based simulation that enables multiple users to interactively rate the animation of a randomly generated population of virtual creatures based on a prescribed criteria. A record of each individual rating is stored and used to seed subsequent generations, thereby guiding the system towards exhibiting a desired type of motion or behaviour.

We start by introducing issues inherent to procedural animation techniques and survey related work. We then describe our prototype implementation that allows multiple users to interactively shape the parameters of a snake-like animation system in real-time. We conclude by discussing the next steps of our research, including the optimisation of procedurally animated dolphins.

I. Introduction and Motivation

Advances in graphics allows for the rendering and simulation of increasingly realistic characters. However, these are still predominately animated using pre-determined data. Whilst this is adequate for a specifically designed environment or scene, the interaction between the creature and the environment is still primarily an illusion.

Procedural animation potentially allows for the automatic generation of limitless and situated organic animation in real-time, facilitating a much more diverse series of actions not possible using a predetermined set of sequences. Furthermore, such systems allow for the complex animation of non-human characters, where motion captured or key-framed sequences are not as readily available.

However, as procedural animation systems become more complex, the parameter space can very quickly become vast and difficult to optimise. It is therefore hard for a single person or development team to find suitable parameters that produces a desired or optimal behaviour. It is also very difficult to automate this process because the appearance and anthropomorphication of a virtual creature can be very subjective, especially if the system is complex enough to evoke emergent behaviour.

Formally, if each parameter has a value range of r, then the number of possible variations is r^n where n is the number of parameters. For example, assuming each single parameter has an integer value range from 1–20, then even with only three parameters there would be 8000 variations. With nine there would be 512 billion possible variations. Complex animation systems usually contain hundreds of parameters and clearly a more effective way of searching this parameter space is needed rather than by using a trial and error approach. In reality, the range of values for many parameters are best represented using a continuous scale rather than an integer scale. This then means the possible number of variations is only bounded by the precision of the floating point representation.

II. Background

The creation of autonomous characters which are able to navigate around a virtual environment realistically continues to be a challenge in real-time computer animation research. The issue of how much control can be exerted on a character versus how natural a movement appears also remains very relevant.

The conventional constraint based, data-driven approaches of blending a large library of motion captured sequences is still one of the most common techniques in research and commercial software [1], [2]. Here a particular action or movement of a character can be extracted from a motion database and blended together to create the impression of natural movement [3].

Parametrised formulas are commonly used to create motion primitives in procedural animation, which can describe a joint rotation or the paths of end-effectors. This is commonly used alongside inverse kinematics solvers [4], [5]. Another option is to use a physically based approach, where the direct application of torque and force on the articulated figure generates animation. Systems have also been developed that aim to combine the relative advantages of both kinematic and physically based techniques, such as the Arthropod animation system described in [6].

Natural Motions Euphoria Engine [7], inverted pendulum models [8] and learning and optimisation strategies [9], [10] show examples of using neural networks to control animation. Other optimisation techniques include the use of motion capture data to enhance procedural animation [11] and active learning methods such as Bayesian optimisation [12].

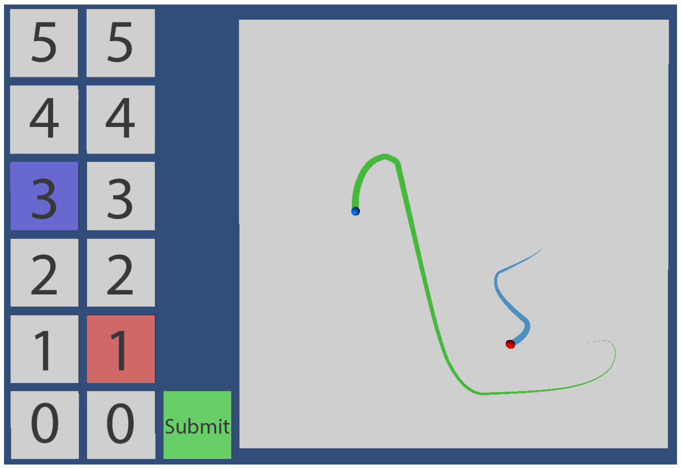

Fig. 1: A screenshot of our prototype client. Each snake-like creature (red, blue) has a corresponding rating system.

Research conducted in [13], [14] suggest that choosing the appropriate set of optimised parameters requires a human in the loop method. A common issue with this kind of optimisation is that it can be difficult to know how best to modify parameters in order to produce a desired effect, especially if the user lacks knowledge of the underlying system. This is further confounded by the fact that complex systems often exhibit emergent behaviour where several parameters interact in wholly unpredictable ways.

Our approach aims to make parameter optimisation completely abstract, where the user makes their judgement based purely on the perceived animation, and never actually sees the numbers being adjusted or how they relate to one another. Furthermore, our system is automatically updated based on crowd-sourced data, and takes advantage of both human and computer based optimisation strategies.

III. Prototype Implementation

As part of a pilot study to test the validity and robustness of our technique, we have created a prototype system in the Unity game engine that uses crowd-sourcing to adjust the parameters of simple snake-like creatures (Figure 1).

The system consists of seven adjustable parameters, yielding 77520 potential snake animations. The snakes are physics driven with *MoveSpeed* and *TurnSpeed* parameters controlling the speed and turning rate respectively. Each snake steers towards an invisible target which resets to a new position at a time set by a *TargetMove* parameter. *TrailTime* describes how long the tail of each snake is, while a parameter called *TrailColour* encodes the RGB colour of the tail.

In each simulation instance, the user is shown two snakes instantiated with randomly chosen parameters. They are then asked to rate the snakes based on a description (e.g. an erratic mover with a long green tail), using a rating system described in section B. The main aim of the prototype is to ensure that we can successfully allow users to anneal the appearance and behaviour of the snakes to a desired state.

A. *Initial Population*

At the start of our simulation, an initial random population of n candidates are generated. Each candidate represents a set of random values within the parameter space of all snakes. These values are generated within a maximum and minimum value taken either from the full scope of the parameter (e.g. 0-255 for colour values), or pre-constrained within a range. A granularity level is also set for each parameter. The candidates are stored collectively on the host server in a database.

During each instance of the experiment, the client machine retrieves and checks out two candidates. These candidates are read into the simulation and two snakes are instanced with the parameters. The user is then tasked with ranking the snakes based on a subjective evaluation of their behaviour and characteristics.

For the purpose of ranking, each snake is identified by a coloured marker. This can be seen in Figure 1, where each snake has a different coloured head that corresponds to a blue ranking scale (left) and a red ranking scale (right).

B. *Ratings System*

The candidate creatures are ranked based on a 0 to 5 scale, with 0 representing low quality, and 5 representing high quality.

Using 0 – 5 ensures that the system always has a value above or below the median, forcing the user to make a positive or negative judgement. Additionally, it provides 3 quantitative ranges for further statistical testing if required during post assessment analysis.

0 - 1	: fail
2 - 3	: undecided
4 - 5	: pass

Through the combination of the raw score (value) and qualitative rank, information can be garnered not only on the overall success of an instance, but also the overall confidence in that scoring.

Each generation terminates when either two criteria are reached: (a) a minimum number of simulations have been run, ensuring all candidates have been assessed at least once; and (b) the simulation has run for a minimum time period.

At the end of the generation, all raw scores are averaged, resulting in an overall fitness for each candidate.

C. *Subsequent Generations*

Subsequent generations are produced through the process of a simple genetic algorithm. The candidates are selected to breed the next generation using roulette wheel selection [15]. The intention is that through each subsequent generation, the algorithm homes-in on suitable parameters.

This process also allows us to pre-seed the algorithm with pre-designed, high quality candidates to further optimise or simply focus the exploration of the parameter space. The system also allows for the design of various “personalities” by selecting multiple high fitness candidates from the final generation.

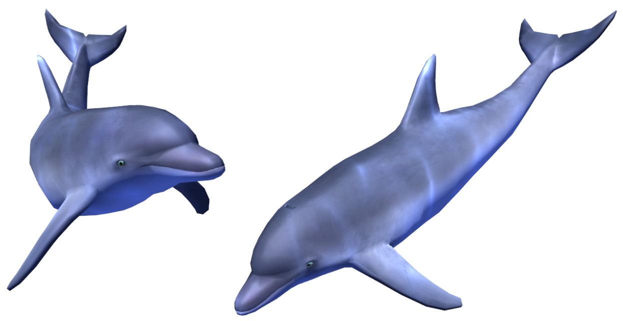

Fig. 2: Two dolphins controlled by our procedural animation system. The system consists of over 70 parameters that govern everything from eye blink and saccade rate to barrel roll frequency.

D. Server

The simulator is run in a browser as a client application using Unity's web player. Each user is automatically connected to the server and the parameters for their individual snakes are pulled in. On each rating submission, the time-stamped parameters and rating are recorded in the server database and a new set of snakes are generated on the client. The system allows multiple users to receive and submit ratings simultaneously.

IV. Conclusion

The process of tuning a procedural animation system can be very subjective and difficult to automate, with no 'perfect' ground truth animation or behavioural profile. Our aim is to develop a technique that allows users to interactively rate and thereby guide an arbitrarily complex procedural animation system into producing a desired set of motions or behaviours however realistic or stylized.

The next step is to conduct a pilot study to ensure that we can guide the optimization of our simple snake-like animation prototype towards desired results. This will include testing the robustness of the system driven by both a small closed group of experts and laymen over the internet.

Once our prototype system is running to our satisfaction, we intend to use it to anneal a much more complex creature animation system currently being developed in tandem (figure 2). The system consists of over 70 parameters and is capable of producing life-like and diverse behaviour for a variety of marine animals. This represents a vast increase in variations compared to our prototype, and the large parameter space will allow us to test if it's possible to crowd-source different types of animation behaviours by asking users to rate based on adjectives like friendlyness, erraticness, dexteriousness or playfulness.

A further goal of this research is to test whether virtual reality affects the user's perception of a creatures' organicity and behavioural realism — for example, the relative importance of dolphin eye contact, blinking, chattering and desired proximity to the user. We intend to run two sets of simulations with the same brief of rating how realistically the creatures are animating, one using regular 2D monitors and another using VR headsets. Comparing the two will give an insight into whether different parameters or animation features affect realism, which features are most prominent and how the behaviour of procedurally animated creatures can be altered depending on how they are being viewed.

Acknowledgement

The authors extend their thanks to HPC Wales for their ongoing support.

References

[1] A. C. Fang and N. S. Pollard, "Efficient synthesis of physically valid human motion," in *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3. ACM, 2003, pp. 417–426.

[2] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," in *ACM transactions on graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 473–482.

[3] Y. Lee, K. Wampler, G. Bernstein, J. Popovic, and Z. Popovic, "Motion fields for interactive character locomotion," in *ACM Transactions on Graphics (TOG)*, vol. 29, no. 6. ACM, 2010, p. 138.

[4] M. Meredith and S. Maddock, "Real-time inverse kinematics: The return of the jacobian," Technical Report No. CS-04-06, Department of Computer Science, University of Sheffield, Tech. Rep., 2004.

[5] B. J. van Basten, S. A. Stüvel, and A. Egges, "A hybrid interpolation scheme for footprint-driven walking synthesis," in *Proceedings of Graphics Interface 2011*. Canadian Human-Computer Communications Society, 2011, pp. 9–16.

[6] L. a. Cenydd and B. Teahan, "An embodied approach to arthropod animation," *Computer Animation and Virtual Worlds*, vol. 24, no. 1, pp. 65–83, 2013.

[7] N. Motion, "Dynamic motion synthesis, (april 2011)." [Online]. Available: http://www.naturalmotion.com/middleware/euphoria/

[8] S. Coros, P. Beaudoin, and M. van de Panne, "Generalized biped walking control," in *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4. ACM, 2010, p. 130.

[9] K. Yin, K. Loken, and M. van de Panne, "Simbicon: Simple biped locomotion control," in *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 105.

[10] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Optimizing walking controllers," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, p. 168, 2009.

[11] C.-H. Liang and T.-Y. Li, "Enhancing procedural animations with motion capture data (a156)."

[12] E. Brochu, T. Brochu, and N. de Freitas, "A bayesian interactive optimization approach to procedural animation design," in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 2010, pp. 103–112.

[13] K. Yin, S. Coros, P. Beaudoin, and M. van de Panne, "Continuation methods for adapting simulated skills," in *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3. ACM, 2008, p. 81.

[14] R. S. Johansen, "Automated semi-procedural animation for character locomotion," Ph.D. dissertation, Aarhus University, 2009.

[15] T. Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.