Virtual reality's effect on parameter optimisation for crowd-sourced procedural animation

Henshall, Gareth; Teahan, William; Ap Cenydd, Llyr

The Visual Computer

DOI:
10.1007/s00371-018-1501-2

Published: 01/09/2018

Publisher's PDF, also known as Version of record

The Visual Computer (2018) 34:1255–1268
https://doi.org/10.1007/s00371-018-1501-2

ORIGINAL ARTICLE

Virtual reality's effect on parameter optimisation for crowd-sourced procedural animation

Gareth I. Henshall[1] · William J. Teahan[1] · Llyr Ap Cenydd[1]

Published online: 24 March 2018

Abstract
Procedural animation systems are capable of synthesising life-like organic motion automatically. However, due to extensive parameterisation, tuning these systems can be very difficult. Not only are there potentially hundreds of interlinked parameters, the resultant animation can be very subjective and the process is difficult to automate effectively. In this paper, we describe a crowd-sourced approach to procedural animation parameter optimisation using genetic algorithms. We test our approach by asking users to interactively rate a population of virtual dolphins to a prescribed behavioural criterion. Our results show that within a few generations a group of users can successfully tune the system towards a desired behaviour. Our secondary motivation is to investigate whether there are differences in animation and behavioural preference between observations made using a standard desktop monitor and those made in virtual reality (VR). We describe a study where users tuned two sets of dolphin animation systems in parallel, one using a normal monitor and another using an Oculus Rift. Our results indicate that being immersed in VR leads to some key differences in preferred behaviour.

Keywords Virtual reality · Procedural animation · Parameter optimisation · Genetic algorithm

1 Introduction and motivation

With advances in real-time computer graphics, virtual characters continue towards photo-realism. However, while rendering techniques allow for life-like visuals, the animation of the majority of complex characters are still heavily reliant on pre-created data such as key-frame or motion capture sequences.

While data-driven animation is capable of very convincing results, the motion is still inherently an illusion. Procedural animation can potentially produce life-like motion with the depth and breadth of behaviour, especially when the agent is reactive and embodied in the environment. With sufficient complexity, such systems could potentially produce animation with the range of behaviour comparable to real-life equivalents.

The advent of presence inducing VR gives us for the first time the opportunity to exist in the same space as virtual creatures at a convincing level of immersion. This opens challenges in many areas of animation, as now even the slightest hitch during motion blending can be enough to break immersion. The intimate nature of such experiences also requires greater attention to behavioural realism, player interaction and an emphasis on micro-animation such as eye contact.

One of the main challenges with procedural animation is the number of parameters that are required to successfully tune the system towards some goal motion or behaviour. As virtual creatures get more complex, so potentially does the number of parameters and therefore permutations. Furthermore, parameters are often interdependent and capable of producing emergent behaviour. A famous example of this is Craig Reynolds' Boids system [1], where three simple steering behaviours (separation, alignment and cohesion) combine to synthesise the natural phenomena of bird murmuration and flocking.

Effectively searching through this vast parameter space is therefore difficult for an individual or team to do. Further-

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s00371-018-1501-2) contains supplementary material, which is available to authorized users.

✉ Gareth I. Henshall
g.i.henshall@bangor.ac.uk

William J. Teahan
w.j.teahan@bangor.ac.uk

Llyr Ap Cenydd
llyr.ap.cenydd@bangor.ac.uk

[1] School of Computer Science, Bangor University, Bangor, UK

more, the appearance and anthropomorphisation of a virtual creature are not particularly conducive to automation as the outcome can often be very subjective.

The creation of a parameter space for a procedurally animated creature can also be subject specific. No two characters will be animated the same, and a parameter space that works for one creature will not be suitable for another [2].

The research outlined in this paper has two goals. Firstly, we aim to use genetic algorithms combined with crowdsourcing techniques to simplify, speed up and democratise the procedural animation optimisation process. Secondly, we are interested to see whether there are differences in an annealed animation system based on medium, specifically if optimal parameters and therefore desired creature behaviour differ based on whether the person guiding the system is using a desktop monitor, or are immersed with the creatures in VR.

2 Background

The creation of autonomous characters which have the ability to realistically navigate virtual environments remains a persistent challenge in real-time computer animation research. How natural a character's movement appears versus how much control can be exerted is also a common concern.

2.1 Animation optimisation

Blending a large library of motion captured sequences together using constraint-based optimisation continues to be one of the most widely used approaches within both commercial software and research [3,4]. Here, a motion database consisting of a set of animation clips is blended together to produce the illusion of natural movement [5,6].

Motion primitives such as the paths of end-effectors or joint rotation have frequently been used alongside inverse kinematic solvers to animated limbs [7–9]. Alternatively, physics-based approaches have been developed where an articulated figure is animated by applying force and torque on rigid bodies. The arthropod animation system described in [10] and the human movement system in [11] describe hybrid systems that aim to combine physically and kinematic based approaches. Optimisation techniques have also been used to derive parameters for physically simulated bipeds, such as deriving appropriate joint torques for realistic, expressive motion from motion capture data [12].

Grzeszczuk and Terzopoulos presented a learning technique capable of automatically synthesising realistic locomotion for animation of physics-based models of animals, being especially suitable for animals with highly flexible bodies such as snakes and fish [13]. Similarly, a realistic model of bird flight animation was developed Wu and Popovic [14] and later simplified for real-time applications [15]. A similar process of actuator-space optimisation was used by Karl Sims [16] to evolve a variety of morphologies and interesting associated locomotion styles through genetic algorithm optimisation.

Since the pioneering work of Ridsdale [17], neural networks have often been used as a tool for optimising animation motion control. Neural networks have been combined with inverted pendulum models [18], learning and optimisation strategies [19,20] and are found in some of the most advanced commercial animation systems such as the Euphoria Engine [21]. Neural networks are often used to evolve walking controllers in physically based animation systems [22]. Chao et al. proposed an adaptive genetic algorithm for optimising the specific driving characteristics of drivers for advanced traffic control [23]. Similarly to our own study, their GA uses roulette wheel selection, crossover and mutation. However, while their mutation operator simply inverted bits, this is not an applicable method for our study as the parameters are dynamic and not binary. Similarly, Ren et al. used a genetic algorithm to compute the optimal parameters for a dynamic model [24]. Their model simulated a swarm of flying insects and the way they interact with each other and their environment.

A fundamental issue with neural network-based approaches is that the optimised control systems are black boxes. There is no intuitive understanding of how the neural network is structured, which makes it difficult to tweak and apply to other models or environments.

Likewise while it is possible to fully automate the parameter optimisation process for tasks like walking in a straight line or across uneven surfaces, or for simple non-humanoid creatures, in more complex scenarios there remains the need for a human-in-the-loop approach. This is especially true for varied and dexterous systems (such as the movement repertoire of a virtual dolphin), where the resultant animation is quite subjective and not easily defined by heuristics.

2.2 Optimising for VR

The ultimate goal of VR is to produce an authentic experience of being "present"; within an artificial environment [25], which includes the need to simulate life-like motion and behaviour. Presence and immersion within a virtual world rely heavily on the use of sight and sound [26], so it follows that a greater focus on optimising animation systems is required for VR.

Kweon et al. [27] showed how the human brain responds differently to a stimulus presented in VR compared to a 2D Monitor. They found when comparing video between monitor and VR headset that β-wave vibrations are statistically significant between the two mediums. It is likely that a similar response can be found in real-time simulations and gaming.

Human-in-the-loop parameter optimisation techniques usually require the operator to have a proficient knowledge of the underlying system. Compounding this issue is that complex animation systems often consist of interlinked parameters, where altering a single parameter could produce unpredictable results. However, it has been argued that a human-in-the-loop method for selecting appropriate parameters is effective [28,29].

Our approach aims to make parameter optimisation abstract, where users rate the animation on its own terms, and do not see 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.

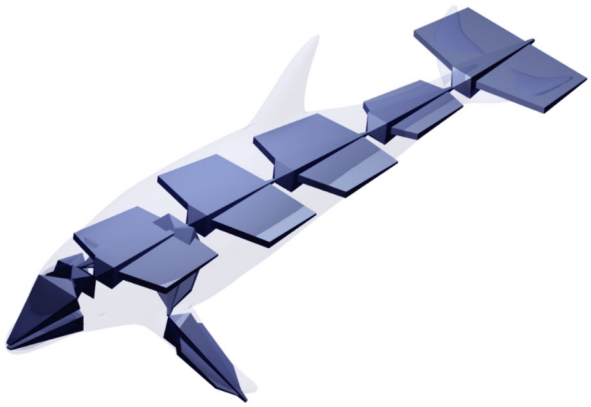


Fig. 1 Dolphin model consisting of a polygonal mesh and underlying skeletal rig. The skeleton consists of mouth and fin appendages attached to a main backbone chain

3 A dolphin model

There are three main types of animation systems available: kinematic; dynamic (physics); or a hybrid of the two. We chose our dolphin model (Fig. 2) as this was the most advanced system we had available. This is a complex kinematic system capable of synthesising acrobatic and dexterous behaviours. While we could have included a more accurate physics-based model to simulate water, hydrodynamics and virtual muscles, we chose not to do this as we wished to avoid the basic mechanics of motion and concentrate on the higher-level behaviours such as twisting, twirling, barrel rolling, and using tried and tested steering behaviours to control the global motion of the creature. This model also contained a set of hand-tuned parameters used commercially, which gave us a decent ground truth animation system to compare against. We also used the parameter settings from this tuned dolphin as a guide to inform the outer parameter bounds for this experiment. While the development of this dolphin required several months of refinement through experience and play testing, we aim to create a system which can tune a creature to a similar standard in a much shorter time frame.

The dolphin model used in our experiment consists of a polygonal mesh rigged with a skeleton (Fig. 1). The skeleton contains a backbone chain representing the torso and tail, with root bone at the head, and ancillary mouth and fin bones.

A target node is created for each bone in the backbone beyond the head and joined together to form a mass-spring system. During run-time, each bone in the backbone functions like a ball-socket joint, pointing towards their associated target node. The mass-spring system acts like an elastic guide for the creature's skeleton to follow, tuned so that it can stretch and contract slightly. The roll axis of all tail bones is limited in order to curb twisting of the backbone during more acrobatic motions.

With the mass-spring backbone in place, animation is generated using a combination of point-mass approximation for

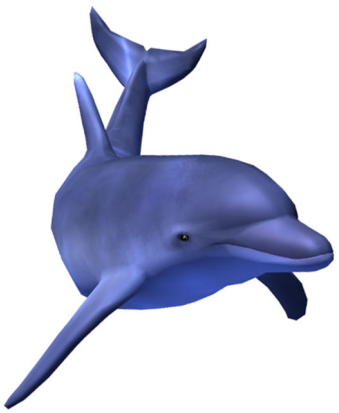


Fig. 2 Example of virtual dolphin used in our study. Each dolphin is controlled by 33 parameters describing its animation and behaviour

global translation and rotation, and local rotations of the backbone and appendages.

Global motion is instigated by moving the root bone through the water, guided by steering behaviours [30]. Basic Seek and Arrive behaviours allow the creature to swim towards and orbit around a target point in the water.

As the root bone moves around in 3D space, the backbone will smoothly follow as if flowing through water. Varying the parameters of the mass-spring system allows for control over the rigidity of this motion. For our experiment, the spring and damper values are set to approximate the elasticity of a dolphin.

Complex creature motion can be generated by mixing waves of various frequencies, amplitudes and axes on top of this underlying global baseline head motion. For example, mixing a sine wave of appropriate frequency and amplitude on top of the head's pitch rotation synthesises the classic up, down undulations of a dolphin as it swims towards the global target point. By applying various waves across pitch, yaw and roll axes, it is possible to synthesise anything from

Table 1 Parameter list and ranges for the dolphin animation system

Parameter	Range
Barrel roll chance	100–1500
Barrel roll speed range max	100–600
Barrel roll speed range min	100–600
Chattering speed max	1–50
Chattering speed min	1 50
Change time	1–10
Default rotate to target speed	0.0–10.0
Default speed	0.0–5.0
Default turn speed	0.0–5.0
Fastest speed	0.0–5.0
Acceleration	0.0–10.0
Speed decay	0.0–1.0
Speed change chance	0–500
Faithfulness	0–1000
Friendliness	0–1000
Near player distance	0.0–8.0
Near player min distance	0.0–3.0
Near player speed	0.0–defaultSpeed
Near player repulse force	0.0– 1.0
Mouth open chance	0–1000
Mouth open time range min	0–2
Mouth open time range max	0–2
Swim amplitude max X	0–100
Swim amplitude min X	0–100
Swim amplitude max Y	0–150
Swim amplitude min Y	0–150
Swim frequency max X	0–10
Swim frequency min X	0–10
Swim frequency max Y	0–10
Swim frequency min Y	0–10
Tail max amplitude	0–120
Tail rest amplitude	0–120
Tail amplitude decay	0–50

the stationary barrel rolling of a Humpback Whale, the over-steering of a thrashing shark, or the graceful twisting and twirling of a playful dolphin.

In procedural animation systems, there is often a balance to be found between realism and control. In our animation system, the steering behaviours represent full control over the motion, while the additive waves represent the deviation from this optimal path. However with appropriate parameters, it is possible to maintain a global heading while also synthesising dexterous and organic motion.

The full list of parameters used by our dolphin animation system is listed in Table 1. All aspects of a dolphin's behaviour are parameterised, including how they swim through the water, flick their tail, undulate their bodies, chatter, barrel roll, change target position and interact with the camera.

Several parameters represent the chance of triggering a behaviour—for example, the *friendliness* and *faithfulness* parameters describe how often a dolphin swims towards the camera and loses interest, respectively. Similarly, the variable *change time* controls how often the dolphin randomises the animation system's various frequency and amplitude parameters to elicit new undulation patterns.

This parameter space represents a vast amount of potential motions. Some will naturally have an optimal range (such as swim speed and acceleration), while others will combine in emergent ways to produce complex motions that can be described for human-in-the-loop purposes using verbs such as "relaxed", "friendly", "playful" and "aggressive".

The dolphin animation system described in this section consists of 33 parameters, many of which are interdependent. For example, the three rotational components blended on top of the global motion can combine to produce a variety of twists and turns. One of the challenges in optimising this type of complex animation system is verifying that the annealing process is working as expected, as realistic or desired motion patterns, or conversely errors, will emerge slowly through generations of user ratings.

We have previously conducted a similar experiment to the one described in this paper on a much simpler procedural animation system, consisting of seven parameters describing basic snake-like creatures [31,32]. In that experiment, we asked participants to rate randomly generated creatures on their physical attributes. Out of the seven changeable parameters, four were vital for the appearance of the snake (RGB colour elements & tail length). We were able to successfully conduce anonymous web users to optimise the system towards goal states such as "short purple snake" or "long erratic snake".

The experiment described in this work is a continuation of the snakes experiment. With our process verified, we aimed to use similar techniques on a much more complex dolphin procedural animation system. However as opposed to the snakes model which was largely morphological (tail length, colour), the dolphin animation system is entirely behavioural. As this is a much more subjective metric, there is likely to not be a perfect outcome, as an individual's perception of the "perfect" dolphin may differ. By using a dolphin's behaviour within the environment as the metric for this study, we are able to assess whether the user's perception of realism changes when the viewing medium changes. Unlike the snake experiment which was solely run on a 2D monitor through a web browser, the study described was run on both a monitor and a virtual reality headset. This means that there are multiple research questions for this study:

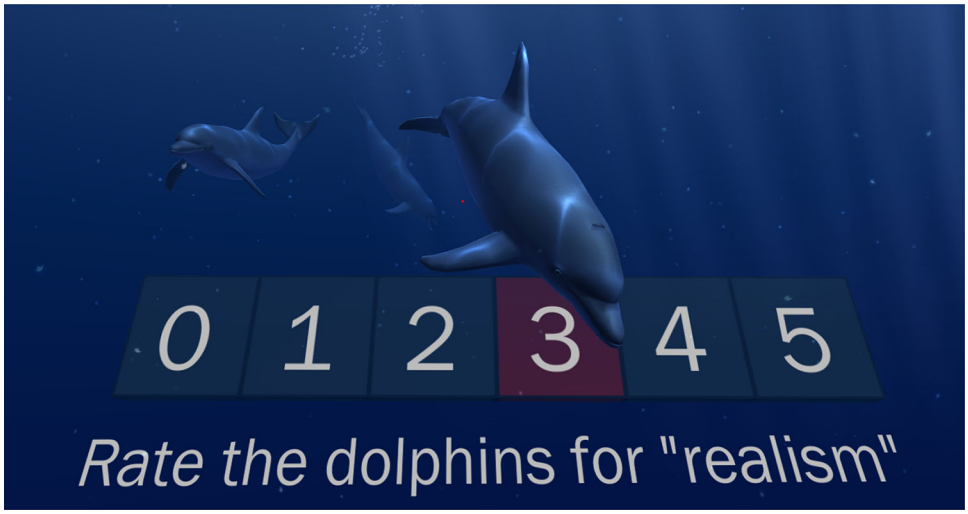


Fig. 3 A screenshot of our application showing the virtual environment, dolphins and embedded 0–5 rating system

- Does the GA and system still effectively optimise a more complex animation system?
- Can we tune a creature to behave in a particular way as opposed to simply altering its appearance?
- Does the platform in which the creatures are viewed alter the user's perception of realism?

The application described in this paper was developed in the Unity Game Engine. The virtual environment consists of the user floating in open water surrounded by detritus particles and light shafts, with a shader-based wave plane above. The dolphins (Fig. 2) swim around with a maximum radius of 100m from the user. The desktop part of the experiment was conducted on a standard 24-inch monitor, while the VR experiment used a consumer Oculus Rift headset.

3.1 Creature initialisation

At the start of the optimisation process, a script creates the first generation of creatures. For both desktop and VR populations, we started with an identical parameter file, consisting of 100 dolphins with random parameters.

The parameters have varying levels of granularity. For example, *defaultSpeed* has a range from 0.0 to 5.0 with a granularity of 0.01, while *barrelRollChance* has a range from 100 to 1500 with a granularity of 1. Across the 33 parameters, there is a total of 3.67e−101 possible dolphins, with the initial generation encompassing 2.721e−100 of all possible permutations.

3.2 Rating system

Upon starting the application, users find themselves underwater with three dolphins (see Fig. 3). Each trio of dolphins is identical (adopting the same parameters) and swim around indefinitely. We chose three dolphins for the study because by random chance each dolphin could be swimming far away from the player. By instantiating three identical dolphins at a time, we increased the chance of the user being able to perceive and rate the dolphin's behavioural repertoire faster and more accurately.

Each participant was told that they were rating dolphins for realism in an entertainment application, rather than for scientific accuracy. While it would be possible to run this experiment with dolphin experts, for this study we expected no prior deep knowledge of dolphin movement or behaviour and wanted to minimise any bias due factors such as model, lighting or shader quality.

Users were asked to rate the dolphins on how realistic they appeared on a 0–5 scale. If a dolphin appeared inactive or broken, then it would receive a rating of 0, with progressively higher ratings awarded for greater realism. In the desktop environment, users manipulated the camera and rating selection using the mouse. On the Rift, users looked around using headtracking and swivel chair rotation and chose appropriate ratings using a combination of gaze tracking and an Oculus remote.

Users were asked to rate on the desktop and VR in random order. Each user was given five minutes in each medium to rate as many or as few dolphins as they liked. We recommended trying to rate at least five dolphins in this time so that they could build a more informed idea of the range of phenotypes. There was no quota of ratings, and the application continued to instantiate new dolphins to rate indefinitely. Once the five minutes had elapsed, the user moved to the second part of the study.

We used a server to store and update the generated parameter files and associated ratings. Using a server potentially allows for multiple users to run the program and rate creatures at the same time, as demonstrated in our prototype experiment, though here users rated dolphins one at a time. When a creature is rated, the time-stamped parameters and asso-

ciated rating are recorded. Dolphins are selected randomly from the parameter file until there are sufficient ratings for a new generation. As the selection process is random, a user could rate the same dolphin multiple times during the study. As this study does not need any details from a participant, there is no signing-up or logging-in process. Instead, they are automatically connected to the server and can start rating the creatures immediately.

3.3 Participation data

To gain as many participants as possible, our experiment was advertised internally through the University emailing system. We tried to encourage participation by offering a spot prize for two participants upon completion of the whole experiment. We then relied on word of mouth to get as many participants as possible to come and take part in the study. The experiment had 26 participants in total which meant on average it took 4.33 participants to fill a generation. Most participants were Computer Science students and 84.6% male. The average age was 24 years old but ranged from 18 to 42. 53.8% of the users taking part in the study had never used a VR headset before. According to a feedback questionnaire we conducted after the users completed the study, the VR experience received a higher enjoyability rating over the desktop monitor. The VR experience received 4.42 out of 5 for intuitiveness of the controls, whereas only 3.65 was given for the desktop experience, likely due to the difference between mouse-look and head tracking controls. The participants also felt it was easier to judge the dolphins in the VR experience.

3.4 Subsequent generations

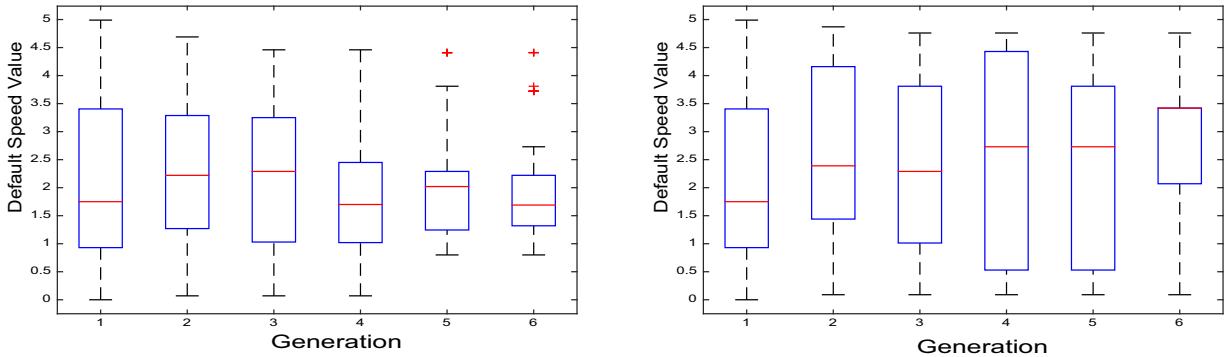
For this study, we adopted a genetic algorithm as our optimisation method between generations. A genetic algorithm is a meta-heuristic inspired by the process of natural selection. In simple terms, a genetic algorithm can be likened to a simulation of Darwin's Theory of Evolution whereby the fittest survive, first introduced by John H. Holland in his book "Adaptation in Natural and Artificial Systems" in 1975 [33]. Alternatives to using a genetic algorithm include a Bayesian optimisation approach [34], where a model is updated and queried to drive optimisation decisions [35]. However, Bayesian optimisation is most commonly used in systems where a human in the loop is not required. A study conducted in 2005 showed that through using a genetic algorithm, parameters were optimised with far fewer generations needed to obtain the optimal outcome [36].

When a generation of dolphins has been sufficiently rated, a script automatically produces the next generation. The genetic algorithm is described in Algorithm 1. When creating a new generation, the current parameter files with their respective ratings are time-stamped and archived on the server (Line 2) for further analysis later. We decided that 100 ratings would be enough for each generation, as it gives a reasonable chance that most dolphin variants have been rated at least once per generation. Once the dolphins have been placed in order of rating (Lines 3–5), the top 25% are automatically selected for the new generation as the strongest candidates (Line 6); this ensures that the fittest creatures are always pushed into the next generation to be rated again. All dolphin ratings are then given a fitness value, and using a roulette wheel selection method (Lines 8–9), two dolphins are randomly chosen to be one of two parents which are used to generate two children for the next generation. By using a roulette wheel selection method, a creature is rewarded for being a stronger candidate and therefore has a much higher chance of being selected as a parent for the next generation.

Algorithm 1: Pseudo-code for our human-in-the-loop optimisation algorithm.

```
1  if at least 100 dolphins have been rated then
2      download parameter and saved ratings file for current
       generation
3      merge any duplicate dolphins
4      calculate the average score for each dolphin
5      sort the dolphins into descending score order according to
       average score
6      copy top 25% rated dolphins into a new generation
7      for remaining 75% of new generation do
8          use roulette wheel selection to find parent1
9          use roulette wheel selection to find parent2
10         perform single-point crossover with parent1 and
           parent2 to produce child1 and child2
11         perform mutation chance for child1 and child2
12         add child1 and child2 to the new generation
13     create new parameter file from the new generation
14     upload new parameter file to server
```

Using a single-point crossover method (Line 10), a random place within the parameters of the parents is selected as the crossover point. The children of these two dolphins are then formed by combining the first portion of parent A and the second part of parent B. The opposite operation is performed to create the second child. These two children are then added to the new generation (Line 18). The process is repeated until the next generation's parameter file is full. Each individual parameter of a creature is given a 1% chance of mutating (Line 11). If it is selected to mutate, then a new value is calculated between the upper and lower bounds of that parameter. This is done to ensure genetic diversity from one generation to the next. After the new generation file is complete, it replaces the previous parameter file on the server (Lines 13–14). This process is automatic and seamless, and



(a) Desktop **(b)** VR

Fig. 4 Value of default speed parameter over the generations

the end user will not notice any difference when rating. Users can start off rating one generation and finish rating a different generation.

4 Results

As previously mentioned, we asked participants to rate the dolphins on how realistic they appeared. In this section, we present analysis of how the 33 parameters change over six generations across both mediums. We also compare differences between the final generations average parameter using Euclidean and Manhattan distances, and how the average rating changed over time.

To compare how the sixth-generation parameters differ across desktop and VR, we have conducted a direct comparison of some of the more divergent or interesting parameters—*defaultSpeed*, *barrelRollChance*, *faithfulness*, *friendliness* and *mouthOpenChance*. The latter four parameters represent odds that a specific behaviour is activated or deactivated, with large numbers representing less chance of something occurring. For example, using our fixed frame-rate of 90 frames per second, a *barrelRollChance* of 300 means that on average the dolphin will perform a barrel roll every 3.33 s, while a value of 900 would trigger a barrel roll on average every 10 s.

4.1 Default swim speed

The parameter *defaultSpeed* represents the minimum speed a dolphin can swim. While dolphins will periodically kick their tail and accelerate, they will naturally slow down to this default speed at a rate of *SpeedDecay*. As shown in Fig. 4, the default speed parameter was in the range of 0–5, roughly representing speed in metres per second. The greater the *defaultSpeed* value, the faster on average the dolphin will swim around the environment. On the desktop, the parameter on average remains fairly stable at ≈ 1.75 with outlying values starting to converge on this value. However, in VR the average default speed rises from ≈ 1.75 to 3.45 across the six generations. This indicates that users preferred significantly faster dolphins in the VR simulation. One potential reason for this is that VR allows users to better keep track of dolphins as they swim around, while faster dolphins might be more difficult to track on the desktop to the more cumbersome mouse-based camera controls. This result could also suggest that users can better judge realistic speeds in VR compared to a 2D image on a monitor. Using a t test, we can assess whether there is a statistical difference between the final generation parameter space for both desktop and VR. As shown in Table 2, the P value produced is a lot smaller than the alpha level of 0.05. Therefore for default speed, we are able to reject the null hypothesis as the data sets are significantly different.

Table 2 Default speed t test: two-sample assuming unequal variances

	VR Default Speed Gen 6	2D Default Speed Gen 6
Mean	2.723	1.899
Variance	2.085	0.773
Observations	105	
Mean diff	0	
df	172	
t Stat	4.998	
$P(T \leq t)$ two-tail	1.41861E–06	
t Critical two-tail	1.974	

4.2 Friendliness and faithfulness

By default, each dolphin swims towards a randomly changing target position in 3D space. However, under certain conditions they will target the user. The closely linked *friendliness* and *faithfulness* parameters determine how often the dol-

Table 3 Friendliness *t* test: two-sample assuming unequal variances

	VR Friendliness Generation 6	2D Friendliness Generation 6
Mean	654	467
Variance	67742	67779
Observations	105	
Mean Diff	0	
df	208	
t Stat	5.192	
$P(T \leq t)$ two-tail	4.92914E−07	
t Critical two-tail	1.971	

phins will swim up to the user, and when they will decide to go back to their usual routine, respectively. A low value for the friendliness parameter means dolphins are more likely to swim up to the player, while lower values of faithfulness indicate a shorter attention span. As shown in Table 3, we can reject the null hypothesis as the P value is less than our alpha of 0.05. Therefore between VR and 2D desktop, there is a significant difference in the data sets.

As the plots in Fig. 5 show, the faithfulness value in both desktop and VR climb similarly over the generations, with a final average of $\approx$ 600 and $\approx$ 550, respectively. These values represent an attention span of around 6.39 s. In both, the outlying values start to compress around these averages, suggesting that users were happy with this behaviour.

The *friendliness* parameter tells a different story. On the desktop, the value trended downwards to $\approx$ 300, meaning that the dolphin would approach the user approximately over 3.33 s. In VR however, the *friendliness* value steadily increased to $\approx$ 700, indicating that users preferred it when the dolphin came up the them far less regularly. Again, this could partly be down to the ability to track the dolphins more easily in VR, but could also be a result of a more keen sense of personal space in VR. The difference between mediums could also be explained by a co-dependency with the rise in average *defaultSpeed*, with higher average speeds allowing dolphins to reach the player faster when the player is selected, thereby increasing the number of close encounters. Unlike default speed and friendliness, we cannot reject the null hypothesis as the P value (Table 4) is greater than 0.05. Therefore, the data sets are not significantly different. This echoes our results shown in Fig. 5 where the faithfulness level for both VR and 2D desktop displays is very similar (Fig. 6).

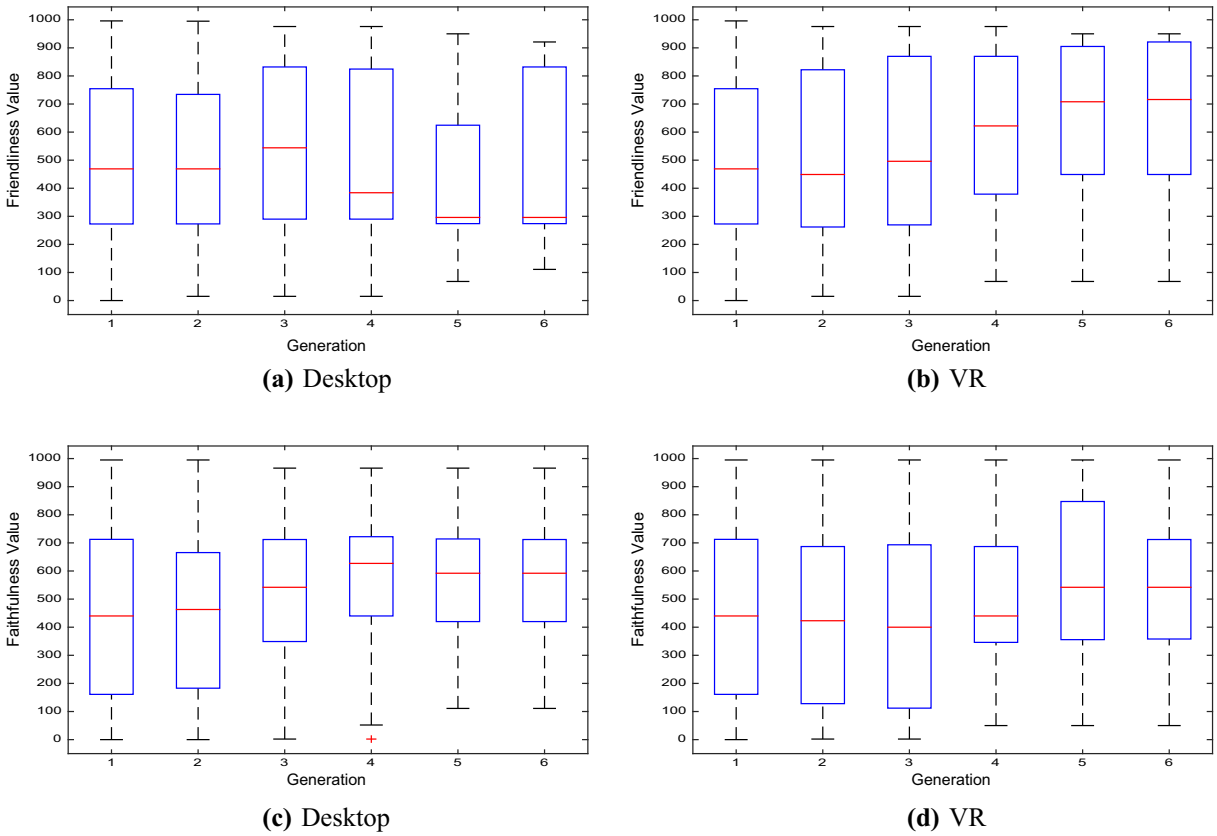


Fig. 5 Friendliness **(a), (b)** and faithfulness **(c), (d)** parameters over the generations. Dolphins with higher values for friendliness are less likely to approach the user

Table 4 Faithfulness *t* test: two-sample assuming unequal variances

	VR Faithfulness Generation 6	2D Faithfulness Generation 6
Mean	551	558
Variance	68206	47218
Observations	105	
Mean Diff	0	
df	201	
t Stat	−0.221	
$P(T \leq t)$ two-tail	0.825	
t Critical two-tail	1.972	

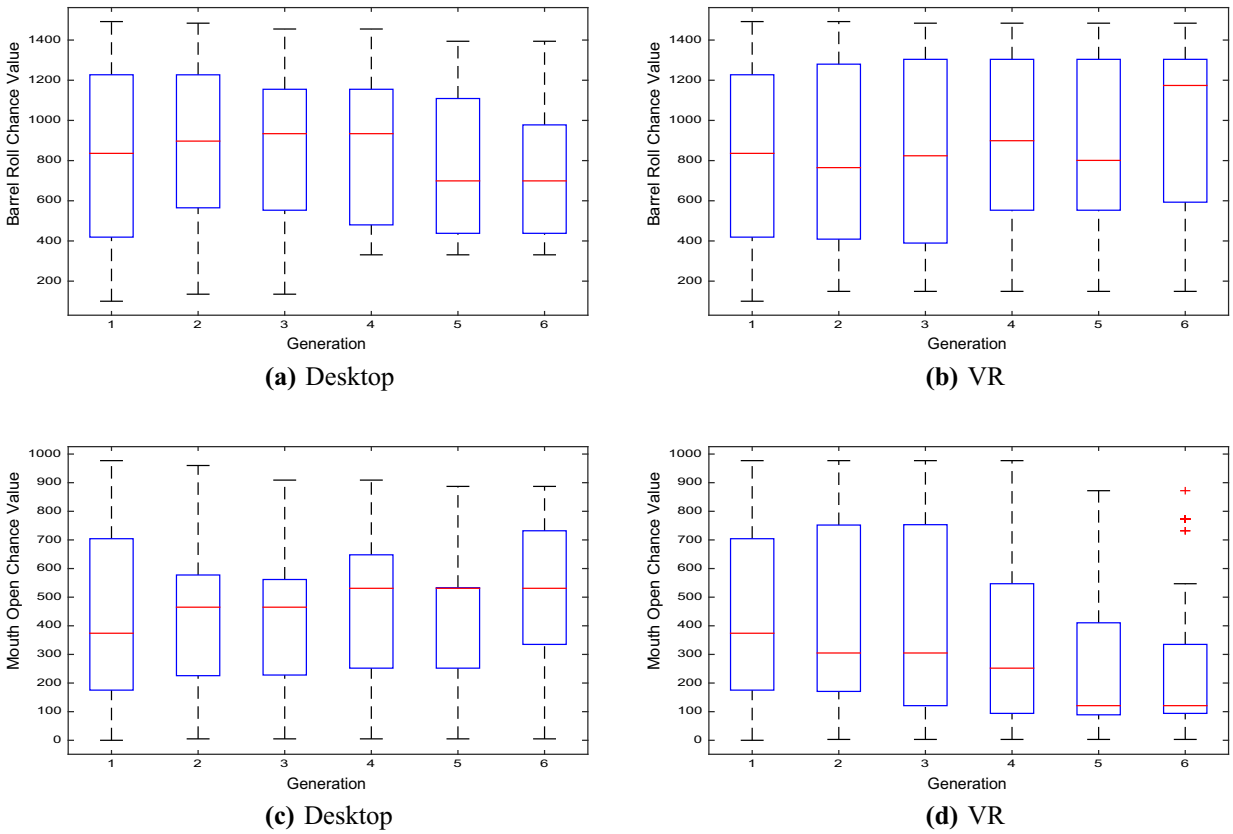
4.3 Barrel rolling and chattering

Two other parameters to consider are *barrelRollChance* and *mouthOpenChance*. Both work in the same fashion as *friendliness* and *faithfulness*, where higher values mean behaviours occur less often.

The parameter *barrelRollChance* represents how often a dolphin will perform a barrel roll. After a slight increase, the desktop dolphins ultimately decreased the barrel rolling to on average every 7.78 s. However, the VR dolphins steadily increased the gap between barrel rolls to an average of 13.30 s. While there is a spike in the last generation, the outlying parameters are also trending upwards in VR, indicating that VR users prefer the dolphins to perform barrel rolls less often.

The parameter *mouthOpenChance* represents the chance a dolphin will open its mouth to "speak", a behaviour which also triggers clicking and chirping sound effects. While this is not realistic behaviour, it does give the dolphins extra character, especially as users will tend to look at the creature's head more than anywhere else. When *mouthOpenChance* triggers, the animation system will use a further four parameters to control how long the mouth stays open and how fast it oscillates. Looking at the experimental results, the value of *mouthOpenChance* increases over the generations on the desktop while the VR value decreases, suggesting that users prefer chattier dolphins in VR. While this is likely due to a variety of factors including friendliness, faithfulness and how close the dolphins appear in VR, it could also be that the sound of dolphin chatter helps enhance immersion and therefore yield higher ratings.

As with default speed and friendliness, we can reject the null hypothesis for both barrel roll chance (Table 5) and mouth open chance (Table 6) as they both have a *P* value



(a) Desktop **(b)** VR **(c)** Desktop **(d)** VR

Fig. 6 Barrel roll (**a**) (**b**) and mouth open chance (**c**) (**d**) parameters over the generations. Lower values give higher chance of performing barrel rolls and dolphin chatter respectively

Table 5 Barrel roll chance *t* test: two-sample assuming unequal variances

	VR Barrel Roll Chance Gen 6	2D Barrel Roll Chance Gen 6
Mean	760	997
Variance	121318	186342
Observations	105	
Mean Diff	0	
df	199	
t Stat	−3.818	
$P(T \leq t)$ two-tail	0.000	
t Critical two-tail	1.972	

Table 6 Mouth open chance *t* test: two-sample assuming unequal variances

	VR Mouth Open Chance Gen 6	2D Mouth Open Chance Gen 6
Mean	493	256
Variance	63831	59420
Observations	105	
Mean Diff	0	
df	208	
t Stat	6.923	
$P(T \leq t)$ two-tail	5.39388E−11	
t Critical two-tail	1.971	

less than the alpha of 0.05. Therefore, there is a statistically significant difference between the data sets when comparing VR with 2D desktop displays.

4.4 Other notable parameters

Other noteworthy parameters include *speedChangeChance*, which controls how often the dolphin will change speed and randomise associated undulation frequency and amplitudes. In VR, this value decreases over the generations, whereas it stays relatively level on desktop. The lower value in VR could be due to participants wanting the creature to have a more consistent speed and behaviour while swimming around. Conversely the higher chance on desktop could also be due to the use of the more cumbersome mouse controls, where changes in speed and motion behaviour are less obvious.

Finally, the *changeTime* parameter, which denotes how often the dolphin changes its goal position, differs between the two simulations—increasing across the generations in VR while decreasing on desktop. This could also be down to aforementioned factors such as a deeper level of immersion in VR, and more natural camera control allowing for a greater sensitivity to changes in dolphin movement and swimming direction.

4.5 Statistical analysis using two-way ANOVA with replication

The two-way analysis of variance (ANOVA) examines the influence of two different independent variables (IV) on the dependant variable (DV). The two-way ANOVA assesses the main effect of each independent variable as well as testing the interaction between them. We can use this test to determine whether the null hypothesis that the mean (average value of the dependant variable) is the same for all groups can be rejected or not.

When choosing to analyse data using a two-way ANOVA, your data need to "pass" the six assumptions (such as continuous dependent variable, no significant outliers, etc.) that are required to give a valid result [37].

Through the use of two-way ANOVA, we can determine the statistical significance of the 2D monitor and VR parameter spaces. This test requires two independent variables (2D Generation 6 & VR Generation 6) and multiple dependant variables (the 33 parameters). As listed in Table 7, the *P* value of our sample is 0.021 which is less than the significance value of 0.05 proving the results are statistically significant. Our *f Value* is larger than the *F-Crit Value* further proving our results are significant and we can reject the null hypothesis.

4.6 Euclidean and Manhattan distances

We also compared the parameter space between the two mediums by calculating the Euclidean and Manhattan distances. By calculating the Euclidean (Eq. 1) and Manhattan (Eq. 2) distances using these formulas, we can compare the two mediums average dolphins through the generations; therefore, we can observe how close the two parameter spaces are to each other. Before the calculations are made, the parameters were normalised to ensure that there was no bias between parameters. The Manhattan distance shows the distance between two points on a grid-based horizontal and vertical path, while the Square distance shows the distance between the two points in a diagonal or in a "*as the crow flies*" manner.

$$\sqrt{\text{dist}(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2} \tag{1}$$

$$\text{dist}(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{2}$$

As shown in Fig. 7, as the study progressed both the Euclidean and Manhattan Distances gradually increased. This indicates that the parameter space for both the 2D and VR dolphins has been tuned differently. As the study has progressed through the generations, the distances between the

Table 7 Two-way ANOVA with replication test: comparing 2D monitor and VR final generations

	Sum of squares	*df*	Mean square	*F*	*P* value	*F* Crit
Sample	65332.671	1	65332.671	5.348	0.021	3.842
Columns	315272801.426	32	9852275.045	806.499	0.000	1.445
Interaction	7680472.786	32	240014.775	19.647	0.000	1.445
Within	83851313.496	6864	12216.100			
Total	406869920.379	6929				

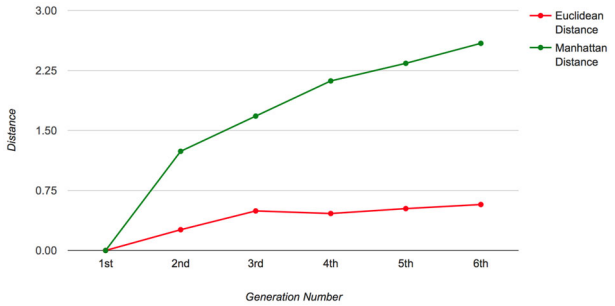


Fig. 7 Euclidean and Manhattan distances at each generation

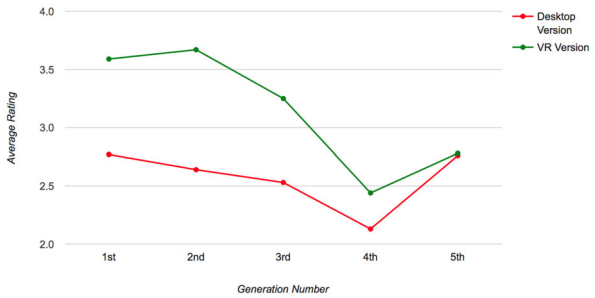


Fig. 8 Average ratings for each generation

parameter files for the two creatures are trending in different directions. This could indicate that different parameters come to prominence on different mediums, or that animation systems might need to be optimised differently across platforms.

4.7 Average ratings

Figure 8 shows the average rating given to each generation of dolphin. The average VR rating starts significantly higher than for the desktop, which could be partly due the fact that for 53.8% of users, this was their first VR experience. This could also reflect that virtual creatures are inherently more realistic in VR, which corroborates with the user feedback.

Somewhat surprisingly for both the desktop and VR dolphins, the average rating trends downwards over the generations, meaning that on average users are rating dolphins as being less realistic with each generation. In both mediums, the average rating decreases between generations 1–4 with a sharp uptick in generation 5.

There are a number of reasons that could explain this trend. Perhaps as more generations are produced, the difference in realism between the "best" dolphin and the "worst" is less apparent, so users become harsher and more discerning with ratings and continue to use the scale effectively. As the dolphins become increasingly realistic through the generations, they are held to a higher standard, so a creature that might have received a rating of 5 in generation 1 now only receives a rating of 3 in generation 5. As the participants only came in to rate the dolphins once, most only saw a single generation of dolphins and therefore could not compare the current generation to previous generations.

5 Conclusion

The process of tuning a procedural animation system is very subjective and difficult to automate as there is no calculable "perfect" behavioural profile. Our aim is to develop techniques that allow users to intuitively guide the development of a complex procedural animation system towards an ideal controller. Our results indicate that while our system can successfully anneal the animation system towards preferred parameters, there are complexities in tuning a virtual creature's parameters to cover both 2D desktop and VR environments, and that certain parameters and behaviours might need to be individually tuned to suit the viewing method.

As mentioned in introduction, the intimate nature of VR experiences requires greater attention to animation quality and behavioural realism. Users are better able to judge the motion accuracy in VR and the experience of rating is more intense. Creatures in close proximity have a real sense of presence and volume and are perceptually quite different to the same scenario on a normal monitor.

There is strong evidence to suggest that VR is capable of a much greater sense of immersion compared to viewing the same simulation on a 2D monitor [38]. The same study indicated that users found the VR version of a simulation more intuitive and were able to perform the tasks with less training or practice. However, we must be cautious not to over interpret the results outlined in the previous section as proof that users prefer different behaviours across mediums. Behavioural preference will also be influenced by the

difference between monitor detachment and VR immersion, with the latter offering a clearer and more intuitive view of the creature. Our results do, however, suggest that virtual creatures are perceived differently in VR, which will affect optimal parameters and therefore the preferred behaviour profile.

It has been found that previous experience with computer games and virtual environments can also vastly affect the performance of a participant when given a virtual task [39]. A further adaption of the experiment could be to use two separate participant groups—gaming proficient and computer illiterate, which would allow us to compare the preferred characterisations of a virtual creature across the two groups.

While this system could also use a similar crowd sources approach to optimise a physics-based animation, it is outside the scope of this paper.

Finally, while this study focuses on the differences between 2D monitor displays and VR, it would be interesting to see whether similar discrepancies in behaviour can be seen in stereoscopic displays such as 3DTV and Z-Space. Similarly superimposing virtual creatures onto the real world is an increasingly common feature of augmented reality (AR) applications. A user's perception may differ when using AR, especially as head-mounted displays improve, due to issues such as real-world distractions and higher levels of immersion.

5.1 Towards real-time parameter optimisation in VR

Our ultimate goal with this work is to develop a system that allows users to naturally tailor virtual creatures towards a goal behaviour. This system could potentially be on an online server, where a group of like-minded people or anonymous users could tune the preferred virtual creature behaviour in parallel.

Our system is designed in a way that allows for any generic procedural animation system or virtual creature to be optimised. As this utilises the participants' perception only, any hypothetical real or mythical creature can be optimised, as all that is required is user interpretation and perception of the resultant behaviour.

We intend to speed up this process through modifications to the underlying genetic algorithm, and the granularity of the rating system. While clear trends appeared in our results, after six generations many parameters had not yet plateaued. In future work, we will endeavour to speed up the process of optimising the system over generations. One of the issues we found in analysing the results was how to interpret parameter trends in isolation. One potential solution to these issues is to have the ability to lock individual or groups of parameters, allowing users to temporary or permanently focus the parameter search space.

Finally, we are planning to extend the VR application so that users can optimise the parameters of their creature in a more natural manner. The Oculus Touch motion controller allows users to gesture, point, thumb up, thumb down and interact in a much more natural manner, which combined with multiple users and parameter locking could allow for faster, more intuitive and collaborative exploration of a parameter space.

Acknowledgements The dolphin animation system used in this paper is based on the system used in the VR application Ocean Rift by Picselica Ltd.

References

1. Reynolds, C.W.: Flocks, herds and schools: a distributed behavioral model. ACM SIGGRAPH Comput. Graph. **21**(4), 25–34 (1987)
2. Hodgins, J.K., Pollard, N.S.: Adapting simulated behaviors for new characters. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co., pp. 153–162 (1997)
3. Fang, A.C., Pollard, N.S.: Efficient synthesis of physically valid human motion. ACM Trans. Graph. **22**(3), 417–426 (2003). ACM
4. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. ACM Trans. Graph. **21**(3), 473–482 (2002). ACM
5. Lee, Y., Wampler, K., Bernstein, G., Popović, J., Popović, Z.: Motion fields for interactive character locomotion. ACM Trans. Graph. **29**(6), 138 (2010). ACM
6. Lee, J.M.: Fast K-Nearest Neighbor Searching in Static Objects, Wireless Personal Communications, pp. 1–14 (2017)
7. Burrell, T., Montazeri, A., Monk, S., Taylor, C.J.: Feedback control-based inverse kinematics solvers for a nuclear decommissioning robot. IFAC-PapersOnLine **49**(21), 177–184 (2016)
8. Meredith, M., Maddock, S.: 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)
9. van Basten, B.J., Stüvel, S.A., Egges, A.: A hybrid interpolation scheme for footprint-driven walking synthesis. In: Proceedings of Graphics Interface 2011. Canadian Human-Computer Communications Society, pp. 9–16 (2011)
10. Cenydd, L.A., Teahan, W.: An embodied approach to arthropod animation. Comput. Anim. Virtual Worlds **24**(1), 65–83 (2013)
11. Bowman, C., Fujita, H., Perin, G.: Towards a knowledge based environment for the cognitive understanding and creation of immersive visualization of expressive human movement data. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer, pp. 182–192 (2016)
12. Liu, C.K., Hertzmann, A., Popović, Z.: Learning physics-based motion style with nonlinear inverse optimization. ACM Trans. Graph. **24**(3), 1071–1081 (2005)
13. Grzeszczuk, R., Terzopoulos, D.: Automated learning of muscle-actuated locomotion through control abstraction. In: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive

Techniques, ser. SIGGRAPH'95. New York, NY, USA. ACM, pp. 63–70 (1995). [Online]. https://doi.org/10.1145/218380.218411
14. Wu, J.-C., Popović, Z.: Realistic modeling of bird flight animations. In: ACM SIGGRAPH 2003 Papers, ser. SIGGRAPH'03. New York, NY, USA. ACM, pp. 888–895 (2003). [Online]. https://doi.org/10.1145/1201775.882360
15. Zhu, C., Muraoka, K., Kawabata, T., Cao, C., Fujimoto, T., Chiba, N.: Real-time animation of bird flight based on aerodynamics. J. Soc. Art Sci. **5**(1), 1–10 (2006)
16. Sims, K.: Evolving virtual creatures. In: Proceedings of the 21st annual conference on Computer Graphics and Interactive Techniques. ACM, pp. 15–22 (1994)
17. Ridsdale, G.: Connectionist modelling of skill dynamics. J. Visual. Comput. Anim. **1**(2), 66–72 (1990)
18. Coros, S., Beaudoin, P., van de Panne, M.: Generalized biped walking control. ACM Trans. Graph. **29**(4), 130 (2010). ACM
19. Wang, J.M., Fleet, D.J., Hertzmann, A.: Optimizing walking controllers. ACM Trans. Graph. **28**(5), 168 (2009)
20. Yin, K., Loken, K., van de Panne, M.: Simbicon: simple biped locomotion control. ACM Trans. Graph. **26**(3), 105 (2007). ACM
21. Natural Motion. (visited on 15.03.2017) Dynamic Motion Synthesis (2011). [Online]. http://www.naturalmotion.com/middleware/euphoria/
22. Geijtenbeek, T., van de Panne, M., van der Stappen, A.F.: Flexible muscle-based locomotion for bipedal creatures. ACM Trans. Graph. **32**(6), 1–11 (2013)
23. Chao, Q., Shen, J., Jin, X.: Video-based personalized traffic learning. Graph. Models **75**(6), 305–317 (2013)
24. Ren, J., Wang, X., Jin, X., Manocha, D.: Simulating flying insects using dynamics and data-driven noise modelling to generate diverse collective behaviors. PLoS ONE **11**(5), e0155698 (2016)
25. Herbelin, B., Salomon, R., Serino, A., Blanke, O.: 5 Neural mechanisms of bodily self-consciousness and the experience of presence in virtual reality. In: Human Computer Confluence Transforming Human Experience Through Symbiotic Technologies, pp. 80 (2016)
26. Cummings, J.J., Bailenson, J.N.: How immersive is enough? A meta-analysis of the effect of immersive technology on user presence. Media Psychol. **19**(2), 272–309 (2016)
27. Kweon, S.H., Kweon, H.J., Kim, S.-J., Li, X., Liu, X., Kweon, H.L.: A brain wave research on VR (virtual reality) usage: comparison between VR and 2D video in EEG measurement. In: International Conference on Applied Human Factors and Ergonomics. Springer, pp. 194–203 (2017)
28. Johansen, R.S.: Automated Semi-Procedural Animation for Character Locomotion. Aarhus Universitet, Institut for Informations Medievidenskab (2009)
29. Yin, K., Coros, S., Beaudoin, P., van de Panne, M.: Continuation methods for adapting simulated skills. ACM Trans. Graph. **27**(3), 81 (2008). ACM
30. Reynolds, C.W.: Steering behaviors for autonomous characters. Game Dev. Conf. **1999**, 763–782 (1999)
31. Henshall, G.I., Headleand, C.J., Teahan, W.J., Ap Cenydd, L.: Towards crowd-sourced parameter optimisation for procedural animation. In: 2015 International Conference on Cyberworlds (CW). IEEE, pp. 379–381 (2015)
32. Henshall, G.I., Teahan, W.J., Ap Cenydd, L.: Crowd-sourced procedural animation optimisation: comparing desktop and VR behaviour. In: 2017 International Conference on Cyberworlds (CW). IEEE, pp. 48–55(2017)
33. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor (1975)
34. Brochu, E., Brochu, T., de Freitas, N.: A Bayesian interactive optimization approach to procedural animation design. In: Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Eurographics Association, pp. 103–112 (2010)
35. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., De Freitas, N.: Taking the human out of the loop: a review of Bayesian optimization. Proc. IEEE **104**(1), 148–175 (2016)
36. Mori, N., Takeda, M., Matsumoto, K.: A comparison study between genetic algorithms and Bayesian optimize algorithms by novel indices. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation. ACM, pp. 1485–1492 (2005)
37. Laerd Statistics. (visited on 21.10.2017) Dynamic Motion Synthesis (2011). [Online]. https://statistics.laerd.com/spss-tutorials/two-way-anova-using-spss-statistics.php
38. Hupont, I., Gracia, J., Sanagustin, L., Gracia, M.A.: How do new visual immersive systems influence gaming QoE? A use case of serious gaming with oculus rift. In: 2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX). IEEE, pp. 1–6 (2015)
39. Murias, K., Kwok, K., Castillejo, A.G., Liu, I., Iaria, G.: The effects of video game use on performance in a virtual navigation task. Comput. Hum. Behav. **58**, 398–406 (2016)

Gareth I. Henshall is a PhD student at the School of Computer Science, Bangor University, UK. He achieved his Degree in Computer Information Systems BSc in 2014. He is now researching into the process of optimising procedurally animated creatures parameter space.

William J. Teahan is a lecturer in the School of Computer Science, Bangor University, UK. He obtained a D.Phil. in 1998 from the University of Waikato, NZ. He leads the Artificial Intelligence and Intelligent Agents research group at Bangor and has broad interests in many areas including natural language processing, data mining, multi-agent systems, machine learning and artificial life.

Llyr Ap Cenydd is a lecturer at the School of Computer Science, Bangor University, UK. He holds a PhD in Computer Graphics and Animation, and his research interests include real-time graphics, virtual reality, procedural animation and artificial life.