DOCTOR OF PHILOSOPHY

Optimising Kinematic Systems Using Crowd-Sourcing and Genetic Algorithms

Henshall, Gareth

Award date:
2019

Awarding institution:
Bangor University

PRIFYSGOL
BANGOR
UNIVERSITY

School of Computer Science and Electronic Engineering
College of Environmental Sciences and Engineering

Optimising Kinematic Systems Using Crowd-Sourcing and Genetic Algorithms

---

Gareth I. Henshall

Submitted in partial satisfaction of the requirements for the
Degree of Doctor of Philosophy
in Computer Science

Supervisor Dr. Llyr ap Cenydd & Dr. Bill Teahan

April 2019

Acknowledgements

> *When anything new comes along, everyone, like a child discovering the world, thinks that they've invented it, but you scratch a little and you find a caveman scratching on a wall is creating virtual reality in a sense. What is new here is that more sophisticated instruments give you the power to do it more easily. Virtual reality is dreams.*
>
> **— Morton Heilig**

One of the hardest parts of this thesis to write is this page, it is a near impossible task to thank everyone who has helped, influenced or guided me over the last few years. With that being said I would like to extend my most sincere thanks to the following people for all their support during my PhD.

- Dr. Llyr ap Cenydd and Dr. William Teahan as without their continued guidance and support this research would have never reached level it did. I thoroughly enjoyed working with you these last few years and hope we continue to collaborate in the future.

- To HPC Wales and The Lever Hulme Trust for their separate funding through some of my years as a researcher. Without them much of this would not have been possible.

- Dr. David Perkins, you were always there to offer some advice, mentor and help me through my teaching. I feel if I had not worked with you over the past few years my teaching would not be at the level it is now. Also, you provided a safe

place to come and chat about any worries and concerns and I thank you for that. I trust we will work together again in the future.

- The Academics and Staff in the Bangor University Computer Science department who have made me feel so welcome and made it a great environment to work in for the duration of my studies. I have some very fond memories of the department and it will forever be a pleasant place to come visit.

- Cameron Gray, James Jackson & Joseph Mearman where to start with you three. Cameron, for being a LaTeXwizard and fixing the silly mistakes I make when days are slow. James & Joe, it was a pleasure to work with you and the friendship you all provided was made the experience far more bearable. I wish you all the best in your future studies, I am only a message away if you need me.

- To my friends and family who without your continued encouragement and support much of this wouldn't have been possible. Whether it was cheering my up on a low day or motivating me to get back on track you have all helped in your own way.

- Finally, to Mum and Dad. You have always pushed and encouraged me to achieve my best in life and this was no different during my PhD years. Your support is always unwavering, and I would not have achieved much of the things I have without your love and guidance. Some much-needed frank talks were had along the way but these have only helped me become a better individual. So, for all of your love and help I thank you.

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.

The research presented in this thesis is divided into three stages. Our first motivation is to examine whether artificially intelligent characters appear more or less human-like in virtual reality (VR). Our results indicate that there is a clear split in how we perceive an artificial character depending on viewing method and game type.

Our second motivation is to assess whether anonymous individuals can anneal a procedurally animated creature towards a desired outcome. To do this we present an online system which used crowd-sourcing to direct a genetic algorithm. This methodology is further tested 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 an animation system toward a desired behaviour.

Our final motivation is to investigate if there are differences in animation and behavioural preference between observations made across 2D screens and 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.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

As real-time computer graphics continues to advance, virtual characters continue towards photo-realism. Although rendering techniques allow for the development of life-like visuals, the majority of complex characters still rely heavily on pre-determined data such as key-frame or motion capture sequences for their animations. While this remains the standard method of animating characters, especially humans, the creation and use of data-driven animation can also be expensive and arduous for developers especially as they constantly try to keep up with the advances in technology and the demands of the consumer market.

Whilst the methods behind data-driven animation produce very convincing results, the motion is still inherently an illusion. Data-driven animation alone can struggle to produce natural behaviour in increasingly dynamic environments. Procedural animation can potentially overcome this issue by producing situated like-life motion, especially when the agent is reactive and embodied in the environment. If the systems produced have sufficient complexity, they could potentially produce animations with the range of motions and behaviours comparable to their real-life counterparts.

Due to the immersive nature of virtual reality (VR), we perceive things differently to a conventional 2D monitor. The advent of consumer grade VR capable of inducing a deep sense of *'being there'*, known as presence, gives us for the first time the opportunity to exist in the same space as virtual creatures at a convincing level of immersion. This presents new and exciting 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. Furthermore, the recent introduction of more sophisticated hand controllers aids the immersion even more as users are now able to use their hands and soon individual fingers to interact with the environment. This rapid pace of development means that it is now getting increasingly easier to break immersion in VR, and any motion or behavioural glitch can break the illusion and therefore sense of presence.

1.2 Problem Statement

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 inter-dependant and capable of producing emergent behaviour. A famous example of this is Craig Reynolds' Boids system [86], where three simple steering behaviours (Separation, Alignment and Cohesion) combine to synthesize the natural phenomena of bird mumuration and flocking. Effectively searching through this vast parameter space is therefore difficult for an individual or team to do. Furthermore, the appearance and anthropomorphisation of a virtual creature is 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 [43].

The research outlined in this thesis has two goals. Firstly, we aim to use genetic algorithms combined with crowd sourcing techniques to simplify, speed up and democratise the procedural animation optimisation process. Secondly, we are interested to see if 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.

1.3 Hypothesis

The overarching hypothesis for this research is as follows:

> *The optimisation of the parameter space for procedurally animated creatures can be achieved through a combination of genetic algorithms and crowd sourcing. Such a system could be useful for tuning procedural animation systems across different viewing mediums.*

1.3.1 Research Questions

As this topic area is fundamentally exploratory, a traditional hypothesis is not sufficient, as there is no clear prediction to be made. Therefore, the following research questions form the basis for this research.

1. Do people perceive the same animation systems and their behaviour differently in Virtual Reality?

2. Can crowd sourcing be used to optimise the parameter space of a procedurally animated creature towards some desired goal?

3. Is there a difference in the ideal parameter space and therefore procedural animation and behaviour between 2D and Virtual Reality?

4. If so, what key attributes are required for a specific viewing platform?

1.3.2 Objectives

From the research questions stated above the following objectives can be proposed:

1. Examine whether crowd sourcing can be used to optimise the parameter space of a procedural animation system through an online experiment.

2. Conduct an experiment comparing the optimisation of an identical starting parameter space across multiple viewing platforms assessing the changes produced over multiple generations — 2D Monitor and Virtual Reality.

3. Evaluate the differences in desired animation and behaviour attributes based on viewing platform.

1.4 Scope & Limitations

Our research is limited to crowd sourcing as a means for data collection. As this is the main source of data collection, we have not considered using a neural network as an alternative method. Due to the resources and time needed to create a suitable neural network, this had to be ruled out as an option for this research, and this has meant that crowd sourcing became the only of data collection.

Secondly, our implementation is limited to creatures with relatively simple animation systems. While future work could explore more complex articulated figures such as quadrupeds and bipeds, and physically simulated articulated figures, this work concentrates on kinematic animation systems only.

1.5 Contributions

The following contributions are claimed from the work presented within this thesis:

Contribution 1 — Virtual Reality's Influence on the Perception of Artificial Intelligent Characters

Two experiments were designed to test a user's perception of an AI character across 2D and virtual reality. Very little research has been conducted into this area and the system proposed demonstrates that non-playable characters (NPC) are perceived differently depending on their viewing platform. Chapter 3 describes how the perception of an NPC alters depending on the viewing platform. Secondly, the level of immersion provided through VR also appears to alter the way we perceive AI characters. There is also a link to the way a game is played and how VR affects the players perception of that world.

Contribution 2 — A Parameter Optimisation Tool for Crowd Sourced Procedural Animation Systems

A system based on a genetic algorithm was developed to tune the parameter space of a procedurally animated creature towards a specific goal. In Chapter 4 we describe a novel online system that uses crowd sourcing and genetic algorithms to optimise a parameter space for a particle-based animation system. We demonstrate how this approach can be used to anneal a simple animation system towards a desired animation and behaviour. This is further reinforced by demonstrating statistical difference between a starting parameter space and its end.

Contribution 3 — A Dolphin Models Adaptation to Fit a Parameter Optimisation Tool

A procedurally animated dolphin model was altered to fit the parameter optimisation tool used in Chapter 4. In Chapter 5 we extended the previous approach to a more complex dolphin animation system from a commercial game. By adapting the methods used in Chapter 3 to fit a more complex animation system we demonstrate its robustness and flexibility. The dolphin model was adapted to suit the system and was then optimised towards a desired goal.

Contribution 4 — Evaluating Parameter Spaces Procedural Animations Creatures

We demonstrate the need for differing parameter spaces for a procedurally animated creature when being viewed on a different platform. In Chapter 6 we provided definitive proof that there is a statistically significant difference between a procedurally animated creatures parameter space when they are tuned for a specific viewing platform, whether it be within VR or on a 2D monitor. Moreover, this suggests that the creation of a procedurally animated creature should be implemented with the viewing platform in mind and tested from its early conception.

1.6 Publications

The following are publications which were written as a part of the research conducted for this thesis. Some publications have been extended and form the main Chapters of this thesis, where-as others are referenced accordingly within the text.

[32] C. J. Headleand, G. Henshall, L. Ap Cenydd and W. J. Teahan, 'Randomised Multiconnected Environment Generator', Bangor University, 2014 (pp. 6, 7, 31).

[33] ——, 'Towards Real-Time Behavioral Evolution in Video Games', in *Artificial Life and Intelligent Agents Symposium*, Springer, 2014, pp. 3–16 (pp. 6, 7, 31).

[34] ——, 'The Influence of Virtual Reality on the Perception of Artificial Intelligent Characters in Games', in *Research and Development in Intelligent Systems XXXII*, Springer, 2015, pp. 345–357 (pp. 6, 7, 31).

[35] G. Henshall, C. Headleand, W. Teahan and L. ap Cenydd, 'Towards Crowd-Sourced Parameter Optimisation for Procedural Animation', Cyberworlds, 2015 (pp. 6, 7, 52, 113, 149).

[36] G. Henshall, W. Teahan and L. ap Cenydd, 'Crowd-Sourced Optimisation of Procedural Animation Systems', Artificial Evolution (EA), 2017 (pp. 6, 7, 52, 113, 150).

[37] ——, 'Crowd-Sourced Procedural Animation Optimisation: Comparing Desktop and VR Behaviour', Cyberworlds, 2017 (pp. 6–8, 83, 92, 113).

[38] ——, 'Towards Real-Time Animation Optimisation in VR', Computer Grahpcis & Visual Computing (CGVC), 2017 (pp. 6–8, 83, 113).

[39] ——, 'Virtual Reality's Effect On Parameter Optimisation for Crowd-Sourced Procedural Animation', The Visual Computer, Springer, 2018 (pp. 6–8, 83, 113).

For publications [32], [33] & [34], I co-developed the two experimental games used within the study and was in charge of both running the study and dealing with participants. Contributions were also made to the papers themselves. For publications [35], [36], [37], [38] & [39], I was the lead developer across all experiments. The Dolphin animation system used in these studies was implemented by Dr. Llyr Ap Cenydd, as part of the commercial VR application Ocean Rift [81]. The experiments in these publications

were designed, developed and ran by myself, and I am the primary author of all five papers and was second author on the remaining three.

1.7 Structure of Thesis

The remainder of this thesis is structured as follows:

Chapter 2 - Contains an extensive review of the literature in the areas of animation optimisation, and perception in virtual reality. This chapter also aims to cover the main theories and techniques used throughout the thesis.

Chapter 3 - Describes two pilot studies into the way we perceive a non-player character within a gaming environment across both virtual reality and standard 2D display.

Publications: This chapter is primarily made up of work published in [34], with additional work coming from [32], [33].

Chapter 4 - Describes a prototype system which uses a crowd sourced parameter optimisation tool and a genetic algorithm to tune the parameter space of a simple snake like creature towards an optimal outcome.

Publications: Parts of this chapter were primarily published in [35], [36], where the former received the award for 'Best Poster'.

Chapter 5 - Explains the process extending the system developed in chapter 4 for a much more complex dolphin animation system. The chapter also describes a study where users were asked to rate dolphins across multiple evolved generations, in order to compare differences in behaviour and motion preference across 2D monitor and VR.

Publications: Parts of this were primarily published in [37], [38] with [39] forming the backbone of this chapter.

Chapter 6 - A full analysis of the results gathered during Chapter 5's study are provided here. We also provide an in-depth breakdown of the key parameters, and how they altered across generations and differ between viewing platforms.

Publications: As with the previous chapter most of the work here was published in [39], with additional work coming from [37], [38].

Chapter 7 - Evaluates the overriding success and outcomes of this thesis. Following this a reflection of the objectives is carried out, with the main findings and contributions presented. The chapter concludes by discussing limitations and possible avenues for future research.

1.7.1 Thesis Structure Overview

Figure 1.1 shows the overall structure of the thesis. Chapters 3 and 4 describe two separate pieces of work that were conducted in parallel, this provided insight and groundwork for the system and experiment detailed in Chapter 5. The results of this study are then analysed in Chapter 6.

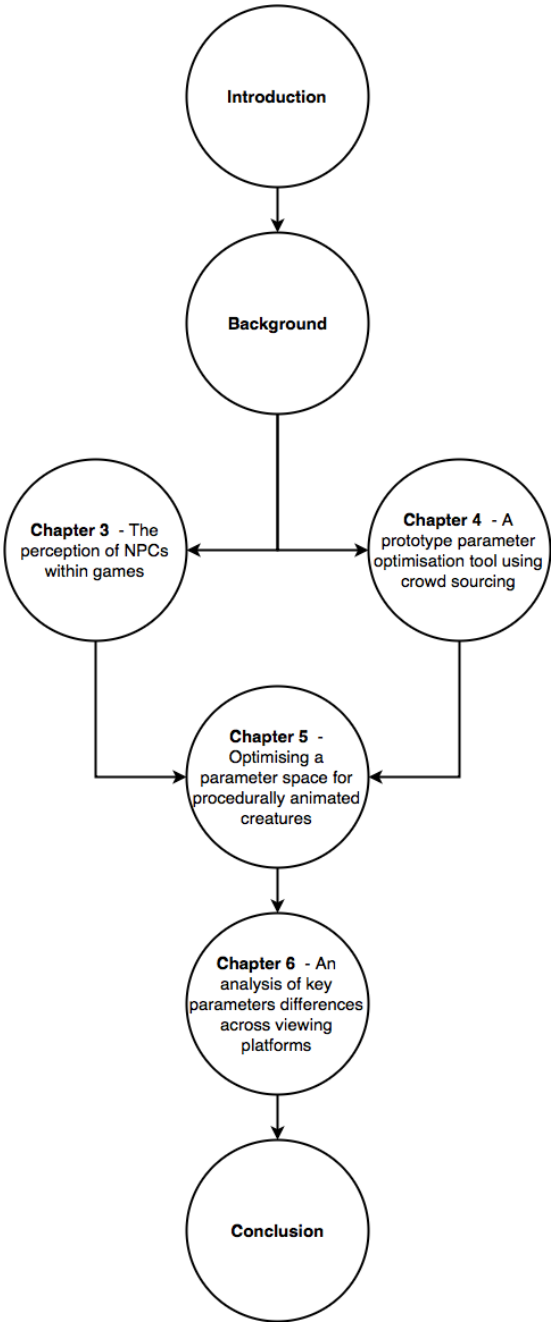


Figure 1.1: A schematic diagram showing the overall structure of this thesis.

Chapter 2

Related Work

In this chapter we provide an extensive review of the literature into two main areas: Animation Optimisation and Optimisation for Virtual Reality. These works provide a grounding for which the research presented within this thesis is based on.

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 Techniques for Solving Optimisation Problems

2.1.1 Genetic Algorithms

A genetic algorithm (GA) is a method for solving optimisation problems which can be both constrained and unconstrained based on natural selection (the process that drives biological evolution) [69]. A GA is designed to continuously modify a population of individual genomes towards a particular end goal. Through each iteration of the GA a random selection of the genomes from the current population are chosen to reproduce therefore creating offspring (children) for the new generation. The intention is that over successive generations the population will evolve towards the optimal solution. Although the complexity of the problem trying to be solved by a GA can vary greatly, they all follow a similar base structure (see Algorithm 1) which can serve as a blue print for the development of a new GA.

Algorithm 1: Basic Genetic Algorithm Pseudo-Code.

```
1 initialise population
2 while new population not complete do
3     calculate fitness
4     selection
5     crossover
6     mutation
7     add to new population
```

A GA consists of three main types of rules at each step when creating a new generation from the current population (different methods for selection, crossover and mutation are discussed in Section 4.4):

Selection: This selects the individuals from the parent population to be used in the creation of offspring.

Crossover: A crossover is the method that combines the selected parents to form the offspring.

Mutation: Applies a random change to individual chromosomes of each offspring.

A GA can often be viewed as function optimiser even though the range of problems to which GA's can be applied to is quite board [109]. The description of a GA is intentionally vague as in some sense the term 'genetic algorithm' can have two meanings. In a strict sense the term refers to a model investigated and introduced by John Holland in 1975 [45]. It is still the case that much of the existing theory behind a GA is based solely or mostly on the model introduced by Holland. Secondly, they can be referred to as the canonical genetic algorithm [106]. A GA's success as an approach for optimisation are most effective when applied to a problem with no available derivatives and the fitness landscape suffers from ill-conditioned parts [57].

2.1.2 Alternative Methods to Genetic Algorithms

There are many alternative methods to a GA which perform similar types of operation but are better suited to different applications, some of these and their uses are discussed below.

Tabu-Search

This method was first introduced by Fred W. Glover in 1986 and formalised in 1989 [26]. Similarly, to a GA it is a meta heuristic search method employing local search methods. Glover's initial research showed that this search method could be used in treating classic problems such as graph colouring and the travelling salesman. Tabu-Search (TS) uses a local or neighbourhood search procedure to move from a potential solution to a new improved solution within the neighbourhood until the stopping criteria has been satisfied. TS carefully explores the neighbourhood of each solution during the search in order to avoid common pitfalls such as score plateaus or exploring poor-scoring areas. The system relies on three main memory structures (tabu list) which consist of a set of banned solutions and rules:

1. **Short-Term:** The most recently considered solutions with strategic forgetting implemented.

2. **Intermediate-term:** Records and compares features of a select number of best trial solutions.

3. **Long-term:** The goal of this is to diversify the search by employing principles which are roughly the reverse of the intermediate-term memory.

Essentially, TS aims to avoid repetitious moves such as, when two moves are located near to one another the search can get stuck in locally optimal solutions. These searches are marked as 'tabu' and therefore will not be returned to whilst in the surrounding search space. The goal of the intermediate and long-term methods is to break out of these local searches whilst maintaining possibility of finding an optimal solution.

Simulated Annealing

Simulated Annealing (SA) is a method for solving unconstrained and bound-constrained optimisation problems [68]. Simply, this is the process of not always improving the output but sometimes extending the search to find alternatives. It exploits the local space to search for optimum solutions for the problem but will also explore the wider search area to allow for any over-looking. This method was developed by Kirkpatrick et al. in 1983 [54], they state that there is a deep and useful connection between statistical

mechanics and multivariate or combinatorial optimisation. This combination along with annealing provides a framework for optimisation of very large and complex system. Their method provided a new perspective on traditional optimisation problems and methods. Goffe et al. proposed using SA on four econometric problems [27]. They determined that SA is a very robust algorithm as it is less likely to fail on difficult functions and could also find the global optimum.

Artificial Neural Network

McCulloch and Pitts [70] used a basis of mathematics and algorithms called threshold logic to create a computational method for neural networks. Their pioneering research paved the way for neural networks and has since been split into two approaches. One focuses on biological process whilst the other is the application of neural networks within artificial intelligence. In optimisation an ANN repeats a two-phase cycle (propagation and weight update). When an input vector is presented to the ANN, it is propagated through the network layer by layer until the output layer is reached. A loss function is then used to compare the output layer with an optimal layer providing the ANN with an error value for each neuron in the output layer. These error values are then propagated back through the network until each neuron has an associated error value which represents its contribution to the original output. A study conducted by Villarrubig et al. [105] used an ANN to analyse its performance on different optimisation problems. Their solution was able to approximate objective functions with a multi-layer perceptron and then use those approximations to solve optimisation problems.

2.1.3 A Comparison of Optimisation Methods

Many studies based on comparing optimisation methods have been conducted and some of these will be discussed below: Westhead et al. compared a genetic algorithm, evolutionary programming, simulated annealing and tabu search for molecular docking [108]. They determined that the genetic algorithm performed the best in terms of the median energy of the solutions located. However, their results showed that tabu searching performed better in terms of locating solutions close to crystallographic ligand conformation. They conclude by suggesting that a combination of a genetic algorithm and tabu search could provide the optimal performance and would outperform any method alone. A genetic algorithm, simulated annealing and tabu search have also

been compare when predicting the effort component of software projects [104]. Uysal states that all three methods are capable for estimating the optimal parameters of the effort components of software projects. However, unlike Westhead et al. it was found that simulated annealing outperformed the rest on this type of searching. A genetic algorithm outperformed other methods when performing warehouse scheduling [110]. The purpose of this study was to compare several techniques for scheduling shipments for a brewing warehouse.

As the research presented in this thesis focuses on the trying to emulate nature in the optimisation of a creature's parameter space it was decided that the best optimisation method would be a genetic algorithm as this best suit this type of research. The nature of a GA means that it can be simple to implemented but adapted to suit many more complex scenarios, such as complex procedurally animated creatures. Therefore, this is the optimisation method that has been used for future studies contained within this thesis.

2.2 Animation Optimisation

Early techniques for the optimisation of motion included motion warping, retargeting and various signal processing operations. Motion warping is a simple technique based on the warping of motion parameter curves for keyframes or captured animation [111]. The animator derives a smooth deformation which preserves the structure of the original motion through keyframe-like constraints. This technique allows for reusable 'clip motions' to be stored into libraries for use when required. Retargeting motion is the process of adapting animated motions from one character for another. Gleicher presented a method for finding the adaptions needed to retarget motions from one character to another [25]. To achieve this, he used geometric constraints and a simple objective function. While the practicality of the method works, a cost is paid towards the quality of the resulting motions. Lee & Shin present a similar technique for adapting pre-exisitng motions for a character to have the desired features which have been specified by a set of constraints [64]. Their approach used a hierarchical curve fitting technique alongside a new inverse kinematic solver. This method greatly reducing the burden of a numerical optimisation to find the solutions, the technique was demonstrated to be appropriate for retargeting a motion to compensate for geometric variations. Finally, motion signal

processing can be successfully applied to designing, modifying and adapting animated motion [8]. They assembled a library of signal processing techniques applicable to animated motion. The technique provided a rapid interactive loop and facilitated reuse and adaptation of motion data.

Parameter optimisation is an efficient way to generate new animations from a minimal amount of data. However, physically based techniques can be difficult to scale for more complex animation systems. Fang & Pollard describe a set of constraints and objective functions which lead to linear time analytical first derivatives [21]. Realistic motion is an important part of video games and simulations, the more life-like a character's movements, the more immersive the environment appears to be. The blending of large libraries 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. While motion capture is a reliable way for reproducing motion in virtual environments, this data has proven to be difficult to modify [56]. If the data provided through motion capture is insufficient then more will need to be acquired which is both a time consuming and expensive process. Kovar et al. present a framework for generating controllable and realistic motion through a database of motion capture. The approaches encapsulate connections by automatically constructing a graph comprised of different motions, then searching that graph for any motions which satisfy the user constraints.

The illusion of natural movement has been produced through blending together a set of animation clips from a motion database. Lee et al. proposed a novel representation of motion data and control which enabled characters with both highly agile response to user input and natural handling of arbitrary external disturbances to continuously flow through the configuration space of character poses [65]. This methodology addresses some of the key issues inherent in graph-like representation such as, responsiveness, to start with arbitrary poses and responses to perturbations. Lee showed that pre-built k-nearest neighbour searching is 2-3 times faster than the kd-tree method in the Point Cloud Library [63]. This method proposes an efficient method for searching static objects.

2.2.1 Inverse Kinematics

Inverse Kinematics (IK) has been used heavily in the animation of characters in real time applications. It is a mathematical process of recovering the movements of an object in a world from some other stored data. In other words, the angles of a kinematic chain of joints are calculated to achieve a desired pose [22]. The only requirement is to define a position as the input, IK then calculates what the pose is. An IK solver attempts to minimise the positional error between two points by providing a target position and a position for the chain end-effector. New techniques, such as Jacobian matric [71], analytical [12] and iterative-based [20] approaches were formed as a result of the problem posed by IK. Forward Kinematics (FK) is essentially the reverse of IK and is often used in predicting movements in non-living objects. This method takes the angles of each joint within a chain and calculates the target position of a figure (such as a hand). This could be stated as: given the four leg lengths, find the current pose of the character once the leg lengths have been measured. Two novel feedback control-based IK solvers were developed by Burrell et al. [9]. The first method has distinct similarities to other feedback control-based solvers and borrows ideas from the Cyclic Coordinate Decent and the Jacobian Transpose methods [10]. This produced a straightforward algorithm with unstable proportional integral derivative gains to determine its performance. Their second approach solves the IK problem through a discrete time state space modelling framework. The second method is more complex to implement but can converge more quickly and has improved immunity to the kinematic singularities which occur in the Jacobian based methods [9].

Unlike Burrell et al. [9] which attempts to find new methods for IK solvers, Meredith & Maddock [71] utilise the Jacobian based solution in their techniques for an efficient solving real-time IK solver. They present a real-time application which drives a walking character around rough terrain, thus demonstrating the effectiveness of their Jacobian interpretation. The technique used incrementally changed a joint's orientation from the stable starting point towards a configuration state. The study concludes that there is a definite argument for using the half-sized Jacobian when only the end-effectors position is required. Existing motions are blended in a linearised representation to guarantee exact control. By concatenating the parameters, van Basten et al. generated highly-constrained animations in real time [3]. Their method consisted of two novel

concepts; a hybrid interpolation scheme which relies on rotational and Cartesian interpolation of joints. This results in exact positioning of limbs because of the linear blend weights. The second method utilises a blend candidate selection schema and soft constraints. This technique is fully automatic and guarantees exact limb positioning.

A real-time system which uses a set of constraints to produce the most likely pose was developed by Grochow et al [29]. Their system can replace style-based IK whenever it is used within computer animation and vision. The model uses training data to determine the preferred pose through the probability model (Scaled Guassian Process Latent Variable Model). All parameters of the models are automatically learned which ensures no manual tuning is required. There are a number of potential applications where a game or simulation will require the motions of a character to look both realistic and satisfy very specific constraints (e.g, moving towards a target) in real-time.

2.2.2 Physics Based Approaches

Physics-based approaches to animation can produce life-like visuals, as any in real life is likely to adhere to the laws of physics. However, designing controllers for producing stable and life-like motion of physically-based articulated figures is incredibly difficult and almost impossible to design by hand, most use a user in the loop-based approach to deal with this issue [55], [62].

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 [30]. By using a multilevel learning process, it first performs repeated locomotion trails in search of actuator control functions. Secondly, a short-time Fourier analysis is applied. This learning process abstracts control functions which produce effective locomotion into a compact representation. Finally, the creatures then put into practice the new compact controllers they have learned.

Similarly, a realistic physics-based method of bird flight animations was developed by Wu and Popovic [113]. Their method enables a bird to follow a specified trajectory by computing a realistic set of wing beats. Elastically deformable feathers are added to a bird's articulated skeleton and each wing beat is optimised separately through optimising

its parameters for the most natural motion resulting in a final series of concatenated optimal wing beats. Although this method worked well, it was later simplified for real-time applications [117]. Their method includes taking into account the aerodynamics of the bird flight model. Target nodes are set into the space which the bird passes through in sequence while adjusting the orientation and flapping continuously. This method allows for a bird to fly along an arbitrary path.

A non-linear inverse optimisation algorithm for estimating optimisation of parameters from motion capture data was used to derive realistic character motion for physically simulated bipeds [66]. This method was used to derive appropriate muscle and ligament stiffness at joints depending on the task. New motions were created through space-time optimisation which minimises the total muscle torques according to the prescribed preferences.

A similar process of creating virtual creatures which move and behave in a simulated three-dimensional physical world was developed by Karl Sims [92]. Genetic algorithm optimisations techniques were used in the morphology of creatures and their neural systems which control the muscle forces. He was able to direct the simulated evolution's towards specific behaviours such as swimming, walking and jumping through fitness evaluation functions. All of this resulted in a system which generated autonomous three-dimensional virtual creatures without the need for user specifications, design efforts or knowledge of algorithmic details.

2.2.3 Neural Network Animation Controllers

Gary Riddle pioneered early developments of neural networks [88]. He described a model for skilled action for synthetic actors in a virtual environment. The method implemented a collection of trained neural networks which guided lower-level motors skills from a connectionist model. Although his models held considerable promise, they were still a long way from achieving a level of sophistication required for simulated actors to behave as real people do.

Neural networks have been used in physically-simulated walking motions [14]. This control is real-time and requires no motion-specific or character-specific tuning. The method works by integrating an inverse pendulum model, adjustments for gravity and

velocity errors and tracking by using a proportional-derivative control, using Jacobian's transpose control. Instant realisation of a suitable controller can be achieved when the character proportions and motion styles authored interactively. Overall, the method generalises across gait parameters, motion styles, character proportions and locomotion tasks. The motion styles, gait parameters and character proportions can be interactively authored.

The SIMBICON controller [115] is a simple controller strategy which was used to generate a large variety of gaits and styles in real-time. A small number of parameters or the construction of a controller can be authored and informed by motion capture. Direct transitions between controllers are demonstrated as well as parameterised control of changes in direction and speed. A modified version of SIMBICON was developed through using a method for the optimisation of a parameter space for a physics-based controller was developed by Wang et al. [107]. Their method demonstrated that even subtle details in control parameterisation can have a significant impact of the style of motion. The Euphoria Engine is a system which combines physics-based simulation, artificial intelligence and hand-made animations created by NaturalMotion [76]. It has been used to create convincingly reactive game characters such as those which appear in the Grand Theft Auto and Red Dead Redemption game series by Rockstar Games [47]. The system allows an infinite number of ways for a game's conditions and applied forces to influence a character's animations, which allows them to be knocked off balance, stumble and fall like digital stunt actors, creating unique end results every time.

Neural networks are often used in muscle-based control methods for simulating bipeds where the optimisation of both muscle routing and control parameters occurs [24]. A variety of bipedal creatures can be supported by a generic locomotion control method. As a result, biomechanical constraints and incorporated into torque patterns. The method described has the ability to support a variety of creatures, a range of speed, turning behaviours and robustness to external perturbations and variations in terrain.

Chao et al. proposed a video-based adaptive genetic algorithm which learns specific driving characteristics of drivers for advanced traffic control [11]. The vehicle's specific driving characteristics are calculated with an offline learning process. Their approach can vividly recreate the traffics flow in a sample video with very high accuracy by using

the vehicle's initial status and personalised parameters as an input. This system out performed existing methods for model calibration as the adaptive crossover and mutation rates improve the overall search capabilities. 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 [85]. Their model simulated a swarm of flying insects and the way they interact with each other and their environment. They also presented a novel evaluation metric and statistical validation approach which took into account the various characteristics of insect motion.

A common problem in computer animation is how to enable virtual characters to perceive and respond to a 3D virtual environment in a way similar to how biological humans or creatures perceive their physical surroundings. This could potentially yield autonomous virtual creatures which are able to behave more like their biological counterparts. Nakada et al. proposed developing a general-purpose artificial vision system which will more accurately model a biological system [78].

Due to their scalability and high run-time performances, neural networks are gaining increasing attention, especially with the advent of deep reinforcement learning. Neural networks can learn from massive amounts of data, while keeping their own data sizes smaller. Zhang et al. produced a novel neural networks architecture called Mode-Adaptive Neural Networks for the control of a quadruped character [116]. Their system consists of the gaiting and motion prediction networks. As the system is far more flexible than alternatives, it can learn consistent expert weights across a wider range of actions in an end-to-end fashion.

Results for evolutionary algorithms are often difficult to describe through text and far too complex to show using images. Many researchers provide video clips to demonstrate the system's functionality, but this does not allow the viewer to adjust the viewing angle to their preference. Clark et al. present a web-based application for sharing interactive animations [13]. Their system allows researchers to generate animation log data which can be loaded into any modern web browser. The camera in these animations can be controlled by the users such that then can pan, move and zoom in and around the scene.

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. For articulated figure animation in particular it can also be difficult to develop flexible controllers that can perform multiple actions. For example, while a neural network can learn to balance, walk or jump with accuracy, smoothly switching or blending between behaviours can be challenging.

2.2.4 Human-in-the-Loop Optimisation

An alternative to automatic optimisation techniques such as neural networks and evolutionary algorithms is to include a human operator in the optimisation process. However 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.

Generalising controllers for new situations are very important for the development of larger skills sets for physically-simulated characters [114]. Adapting controllers to large environmental controllers can be provided by continuation-based methods. These methods can provide surprisingly large motion adaptions. Johansen's method used a set of example motions in the form of keyframes or motion captured walk and run cycles [53]. The system then automatically analyses each motion extracting required parameters. Both of these studies argue that a human-in-the-loop method for selecting appropriate parameters is effective.

Whilst 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 largely 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.

2.3 Optimising Characters for VR

While Ivan Sutherland published his pioneering essay "The Ultimate Display" over 50 years ago [100], it has only been relatively recently that technology capable of producing a 'looking glass into the mathematical wonderland' has been possible (see Blascovich & Bealenson for a historical account [6]). Sutherland's first head-mounted display (HMD), nicknamed 'The Sword of Damocles' gave user's fears of bodily harm, as it had to be bolted to the ceiling due to its size. However, with recent advances in technology, especially in the mobile phone industry, HMD's are now viable consumer products. As this industry grows, it is increasingly crucial to understand how this rapidly evolving technology effects immersion across diverse applications like video games, training, marketing, communication and education.

2.3.1 The Concept of 'Presence'

Slater & Usoh suggested that user's can maintain a sense of presence that their real bodies are located within a virtual environment (VE), created through a human's perception of visual, audio and kinetic environments generated by the computer. This is described as the 'suspension of disbelief' [94].

Lombard & Ditton began to describe the concepts of 'presence' in 1997 as an illusion that a mediated experience is not mediated [67]. They describe six interrelated, but distinct conceptualisations of presence found in a diverse set of literature:

1. The presence as social richness or the extent to which a medium is perceived as sociable, warm, sensitive, personal or intimate when used to interact with other people.

2. The accuracy of object, people and events representations within a medium is known as presence in realism. This describes how close the the 'real' thing the representation looks, sounds and feels.

3. Presence in transportation can be broken down into three categories; 'you are there', in which a user is taken to another place; 'it is here', in which another place and/or the object within it are transported to the user; and 'we are together',

in which two or more communicators are transported together to a place which they share.

4. Immersion presence can be described as "The degree to which a virtual environment submerges the perceptual system of the user" [5]. This can be determined by assessing how many of the user's sense are provided with an input and the 'degree' to which those senses are 'shut out' from the real world. For example, the eyes are covered by a Head Mounted Display (HMD), the ears are covered by headphones muffling or blocking out any external sounds and the hands are covered with gloves which enable the user to touch virtual objects.

5. Social actors use direct address camera views to give the feeling that they are looking at the viewer directly. This is referred to as presence as a social actor within medium. user's respond to social cues presented by the actor in parasocial interaction media even if it is illogical to do so.

6. Social responses to cues provided by the medium itself and not people or characters is referred to presence as a medium for social actors. Alan Turing's 'Turing test' sparked debates in the 1950s around the potential for modern computers to mimic humans. Whilst science fiction evokes social responses between humans and computers, robots and androids (e.g. C3P0 and R2D2 in Star Wars and Data in Star Trek) the phenomenon also seems to exist in today's less sophisticated computers.

The ultimate goal of VR is to produce an authentic experience of being 'present'; within an artificial environment, which includes the need to simulate life-like motion and behaviour. Classically, VR technologies have been created using a cybernetic approach. This uses multimedia technologies to allow subjects a way of removing all interaction from the external world by being placed at the core of the feedback control loop [40].

2.3.2 Immersion Within a Virtual Environment

Video games and simulations allow user's the ability to become fully immersed within a virtual world. However, there is some debate on what a truly immersive environment really is. Two prominent terms used within gaming to describe a user's presence within

gaming are ‘flow’ and ‘immersion’. While, both of these are separate aspects of a gaming environment, if one is not correctly implemented then the user’s presence can be very easily broken. Flow is a psychological theory and when implanted within a game or simulation presents players with a challenge and through engaging with the challenge the player can achieve a flow state, the game then tries to maintain this flow state throughout. Flow state is can be described as ‘hyper focused’ where the individual is fully engaged in the game play at that moment, the awareness of their surroundings falls away as each action within the game flows from one to the next almost naturally. Flow has been referred to as the ‘optimal’ experience when nothing else matters [51]. Where as immersion is, ‘the sense of being in the virtual world’ [96]. It has been argued that total immersion is not always achievable [52] as it can be classed as ‘sub-optimal’. Both positive and negative experiences within gaming have been shown to provide an engaging experience. There is little evidence to show that developers should prioritise flow over immersion or visa-versa, therefore the terms of flow and immersion can be used interchangeably [73].

Trying to produce a simulation that allows for a user to transition seamlessly from reality to virtual reality has proved to be difficult. The idea is that a user can interact with the virtual world as effortlessly as they would the real world. Some try to get around this through the use of augmented or mixed reality, where-by the user actually touches something physical which has a virtual overlay on it to give it the appearance of something else [4].

Presence and immersion within a virtual world rely heavily on the use of sight and sound so it follows that a greater focus on optimising animation systems is required for VR. It is often assumed that a higher level of immersion leads to a greater level of presence within VEs. Cummings and Bailenson suggested that immersion was found to have a medium-sized effect on presence, while individual features were found to affect a user’s presence in varying sizes [15]. A two-step formative process helps to achieve presence; a user constructs a spatialise mental model of the mediated, and then accepts this environment over reality as their primary sense of self-reference.

Being able to provide realism at interactive rates still remains a major challenge in generating high-fidelity VEs. High-fidelity simulations of light and sound are still

unachievable in real-time, though it is likely inevitable. Harvey et al. investigated the effect spatialised directional sound has on the visual attention of a user towards a rendered image [31]. Their results showed that this method performs only significantly better than simple image saliency or acoustic intensity maps when used as a rendering strategy.

A Controller's Effect on Immersion

Over the last three decades a vast amount of research has been performed into how we interact with computers. For many years the adopted style for human-computer interaction (HCI) was through windows, icons, menus and pointers. Technology and a better understanding of the psychological and social aspects of HCI advances have led to an array of post-WIMP ("windows, icons, menus, pointer") styles [91]. An early adopter of these methods was the Nintendo Wii controllers released in November 2006. These controllers allowed for consumers to interact with a virtual environment in a completely new way. They not only relied on visual and audio signals to arouse a user's sense, but also introduced sense to the experience. Furthermore, these new control methods now allowed users to interact with the virtual world in three dimensions using their hands.

A decade later VR headsets started to appear with motion controls offering six degrees of freedom. While Nintendo Wii controllers only accurately tracked rotations, with some control over positional depth, devices like the Oculus Touch and HTC Vive controllers allow for full positional tracking with sub-mm accuracy. The prototype Valve Knuckles EV2 [98] took interaction with hand controls in a virtual world to a new level, by also tracking the position of each finger as the hand grips the controller. The controllers also allow the user to 'let go' of them due to a strap that attaches them to the hand. While they still rely very heavily on toggles and buttons for most of the control mechanisms, the ability to grab or drop objects in a VE by closing or opening your hand is a big step forward. This feature is also present in other controllers (e.g. the Oculus Touch Controllers) but these still relied on a button being pressed to simulate the action. Secondly, the EV2 contains pressure sensors which allows the user to 'squash' or 'crush' an item in the virtual world if the game allows it.

As motion controllers continue to develop, the ability to stay 'truly' present within the virtual world will increase. This idea of 'hand presence' will further reduce the chances of immersion being broken due to a mismatch between the user's real-world actions and the visual and physical response in VR.

2.3.3 Photo-Realism's Effect on Presence in Virtual Environments

When developing characters for virtual environments, their design and appearance needs to be considered. For example, while a photo-real style might be desirable in standard console video games played on a 2D screen, a character that appears to be almost photo-realistic can produce negative responses from user's in VR. This effect ('The Uncanny Valley') was first proposed by M. Mori in 1970 [75]. Mori demonstrated this theory through his work in robotics. He showed how as human likeness of a character; the familiarity we have with it increases up to a point. A hypothetical graph (Figure 2.1) of the viewers impression of pleasantness as a function of the degree of realism was introduced. If the level of human likeness goes above ~65% then the familiarity level rapidly decreases meaning that immersion levels could deteriorate. It should be noted that if human likeness gets above ~85% then the familiarity level will start to increase again but achieving those levels when developing a virtual character is proving more and more challenging as HMD technology advances. Zibrek et al. attempted to overcome this challenge by altering the rendering style of a character whilst keeping the realism or motion and shape constant [119]. They found that rendering styles did not overly affect the appeal of a character. Their results showed that a realistic appearance was found to increase co-presence when a characters's behaviour was also realistic.

Demand for effectively building virtual characters for communication in shared virtual worlds is also rapidly increasing. This requires that we need a better understanding on how a character in a virtual world is perceived, such as simulated (or eventually tracked) facial expressions and micro movements, eye contact, and body language, as a poor design choice could result in a negative reaction from user's [118].

Slater et al. designed a study to asseses the level of presence in a VE when influence by a variety of factors [95]. They tested 24 subjects split into two groups, half of which were transported between VEs by going through doors and the other half by using virtual HMDs. Their analysis showed that a positive and significant association with stacking

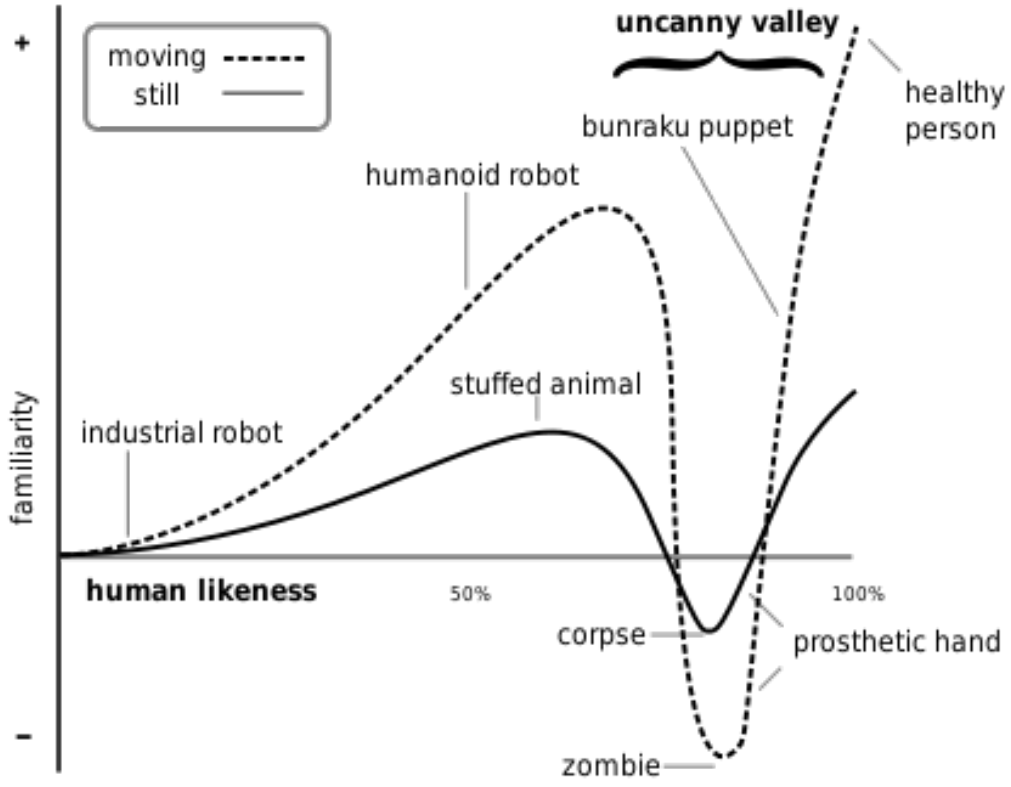


Figure 2.1: Mori's hypothetical graph of the uncanny valley's effect on human perception of robots [90]

level depth for those participants who used virtual HMDs for their transportation, and a negative association for those who went through doors.

It can often be seen that to be truly present within a VE, a user must be reporting that they feel the sensations of being in the virtual world. The term presence comes from 'telepresence' which is a psychological state or subjective perception in which part of a human's perception fails to accurately acknowledge the role that technology has on the experience and instead perceives it to be real [89]. Presence is often measured on one overriding factor over others, with attention being one of the most common. Subjective measures (i.e, questionnaires) are the most commonly used metric. As Witmer and Singer correctly stressed, any measurement of presence should be both valid (i.e, measuring what is needed to be measured and doing it well) and reliable (i.e, only dependant on the characteristics under consideration) [112]. An advantage to questionnaires is that as well as subjective sensations during the VR experience being measured, you are also able to ask for descriptions of the VR experience and their own personal behavioural and psychological responses. Other subjective metrics

for measuring presence include continuous measure, with Ijsselstein and de Ridder proposing a measure which involved participants adjusting a slider during their VR experience to indicate the levels of presence experienced in real time [49]. Presence counters assess the number of transitions from the real to virtual world during an experience. Slater & Steed's method relies on data being collected during the virtual experience itself by counting the number of transitions from the virtual to real world. From this a Markov chain model was constructed. This model could then be used to estimate probability of being 'present' in the VE [93]. On three measures — vividness, interactivity and user characteristics. Evidence was found to show these played a major role in providing a user with presence in a VE.

2.3.4 The Brain's Response to VR Compared to a 2D Monitor

Kweon et al. conducted a study to assess how the human brain responds to a stimulus presented in VR compared to a 2D monitor [58]. Their experiment tested 20 subjects' brain waves whilst viewing content across both mediums. They found when comparing video between monitor and VR headset that β-wave vibrations are statistically significant between the two mediums. Overall, β-waves were higher for all genre's (sports, news & advertisements) when viewed in VR. Videos felt more realistic to viewers when viewed in VR as more sense and thinking was stimulated making the videos feel more realistic, as a result this enhanced the experience of the video itself.

2.4 Chapter Summary and Conclusion

This research has shown that as rapidly increasing VR immersion requires far more attention to the complexity of animation systems. There is evidence showing that how we perceive characters in VR is different to a 2D environment. This means that considerations into how a virtual character is implemented needs to be taken. Procedural animation systems can potentially be optimised towards a desired outcome effectively by using the wisdom of the crowd, combined with traditional optimisation systems like evolutionary algorithms.

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. In the next chapters we describe two prototype studies into how we perceive artificial characters across multiple view platforms and whether a simple genetic algorithm and rating system can be used to tune a creature's parameter space towards an intended outcome.

Chapter 3

The Influence of Virtual Reality on the Perception of Artificial Intelligent Characters in Games

The previous Chapter reviewed much of the research done in fields relating to our work, covering advances in Procedural Animation, Optimisation and Virtual Reality.

One prominent question for the gaming industry is whether it is easier to identify an AI character through VR or a traditional desktop monitor. For example, many modern collaborative on-line games seamlessly replace human controlled companions with AI ones if they leave the game, allowing the player to continue without breaking immersion. Virtual characters are now at a stage where it is increasingly difficult to make human/not human distinctions in virtual environments [79], and AI is an important component of this façade. If VR had an effect on this, developers may need to reconsider their development choices, or how they implement their AI characters.

For this study we proposed two main research questions:

1. Does virtual reality change the way we perceive non-player characters in games?

2. Do AI controlled characters appear more or less life-like through this viewing medium?

Essentially, we will explore whether virtual reality make the synthetic behaviour of AI characters more noticeable.

This Chapter describes a study designed to test the influences of virtual reality on the perception of identifying an AI character within games. This was done through two experiments both of which involved the participant playing a game (one racing and the other a first-person shooter) on a 2D monitor and in VR. The task for each participant was to play the game and determine if their opponent was a human or a non-player character (NPC). In this Chapter we go into detail describing the set-up of each experiment and the considerations taken with each.

The following Chapter was primarily published in [34], with additional content coming from [32], [33].

3.1 Racing Game

To implement the racing game, we used two sample projects available on the Unity asset store. The first *Car Tutorial* [102] is a complete package including a track and a physics driven player car. We augmented this package with the AI car from *The Vehicle Physics Toolkit (VPT)* [50] package, which is also freely available. The underlying physics of both AI and player-controlled cars were based on the same model [102]. The goal of the racing game was to complete one lap of the track before your opponent. Once the player reaches the finish line the game is complete, and the result of the race is displayed on the screen.

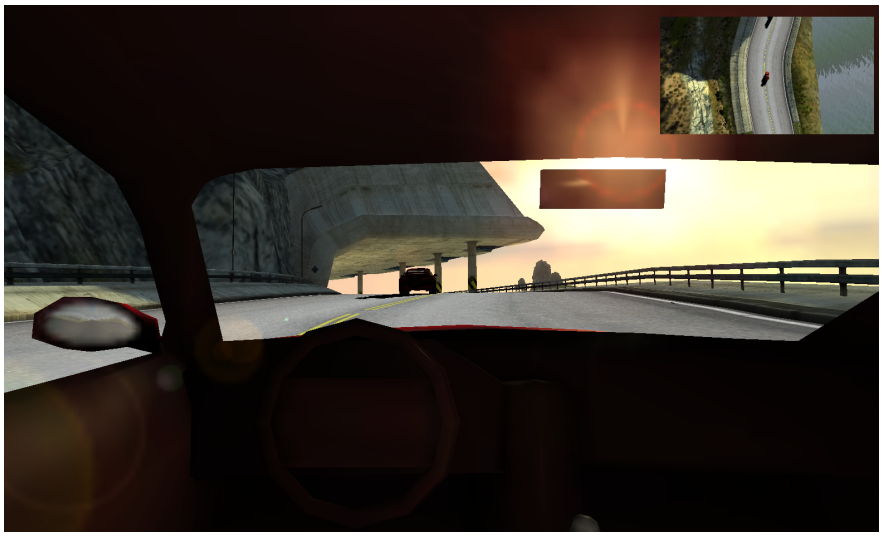


Figure 3.1: Players view from inside the car during the game.

The AI car follows a predetermined path along the center of the virtual racetrack. A variable in the AI car script determines how much it can deviate from that path before it

needs to correct. Varying this value allows us to produce a relatively realistic driving style. The path itself is constructed from a series of game objects linked together to form a complete circuit. Within the VPT package, breaking zones (see Figure 3.2) are placed throughout the track so that the AI car will slow down at sharp corners. These had to be placed strategically to give the AI car the appearance of a realistic race car. How much the car slowed down in these sections had to be thoroughly tested as each corner needed a different speed for the car to successfully make it round, generally the sharper the corner the more the car was slowed down. If the car either didn't slow down enough or slowed down too much it might appear unrealistic to the participant. Each braking zone was implemented using a box trigger collider, which when triggered by the AI car would adjust its $maxBreakTorque$ causing the car to slow down. A reset function was also added so that any AI or player-controlled car that crashed or flipped over would automatically reset its position and rotation to the middle of the road after two seconds. This also ensured that neither of the two cars could get stuck on the outer bounds of the track.

Figure 3.2: The breaking zones along the track which cause the AI car to slow down.

The majority of racing games have a natural rhythm, with players regularly changing position and overtaking each other, rather than one player dominating the race. This is a commonly sought-after mechanic by game designers and is usually achieved by applying a 'rubber banding' function that speeds or slows down the AI car if too far behind or ahead. If the player car gets too far ahead the AI car will temporarily go above its usual top speed to help it catch up, and if the player falls too far behind then the AI car will curb its speed to give the player a chance to catch back up. Both of

these allow for closer racing and can prevent the participant getting bored due the AI car getting too far ahead or falling behind. The same mechanic tends to naturally arise in similarly-skilled player controlled games due to driver error etc. Rubberbanding was implemented into our game for the same reason, ensuring that the AI was constantly battling the player for position regardless of the players ability. A variable was also added to the AI car to act as 'human error', this would trigger at random stages during the race and caused the car to make a mistake, whether this was not decreasing its speed enough for a corner and going off track or slowing down too much and allowing the player to catch up or get away. This made the AI car seem more realistic to the player and made the racing more enjoyable.

The original camera position followed above and behind the car (third person perspective). This camera angle was not suitable for VR as it is both unnatural and would likely lead to nausea [84]. The camera was moved to be inside the car, providing a much more natural perspective. Being surrounded by the car's interior also helps to reduce nausea as it provides a static frame of reference. We also added a mini-map (see Figure 3.3) in the top right of the players vision, this allowed the player to preview the road ahead and plan their strategy accordingly.

Figure 3.3: The implemented mini-map for participants to get a bird's eye view of the track

3.2 First Person Shooter

The FPS game required the player to explore a randomly generated maze and destroy an opponent. The player and NPC used an identical character (see Figure 3.4) during the

game. The character was a simple sphere with a turret weapon. The drone fired smaller spheres which acted as bullets which reduced the enemy's health upon a collision. A small health bar indicated the players health and decreased each time they were shot. Upon entering the game, the players were placed on opposite corners of the environment ensuring there would be no advantage to the placement of a player's character. The game ended once either player ran out of health (i.e. has been hit by a bullet 10 times).

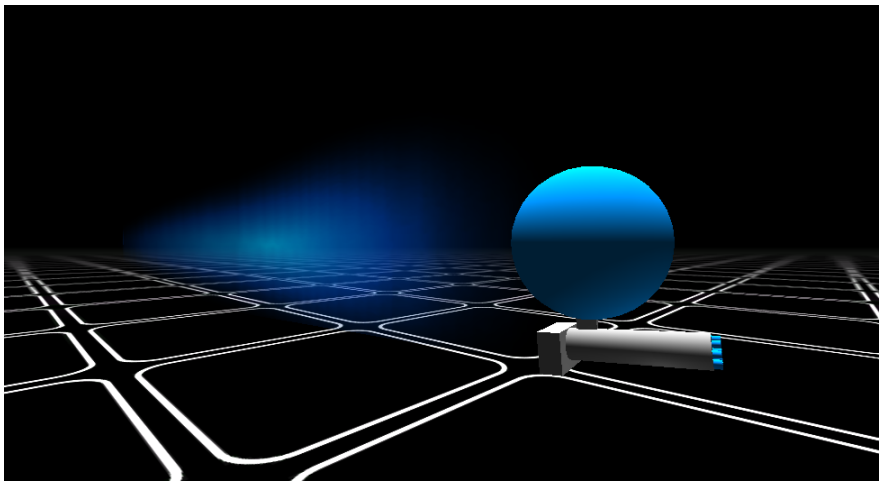


Figure 3.4: The player and opponent character in the FPS game

To generate the environment, we implemented the Randomised Multiconnected Environment Generator (RMCM) algorithm **headleand1** in Unity. Each environment (see Figure 3.5) was 50 units square, with one unit equal to the diameter of the enemy and the player's avatar colliders.

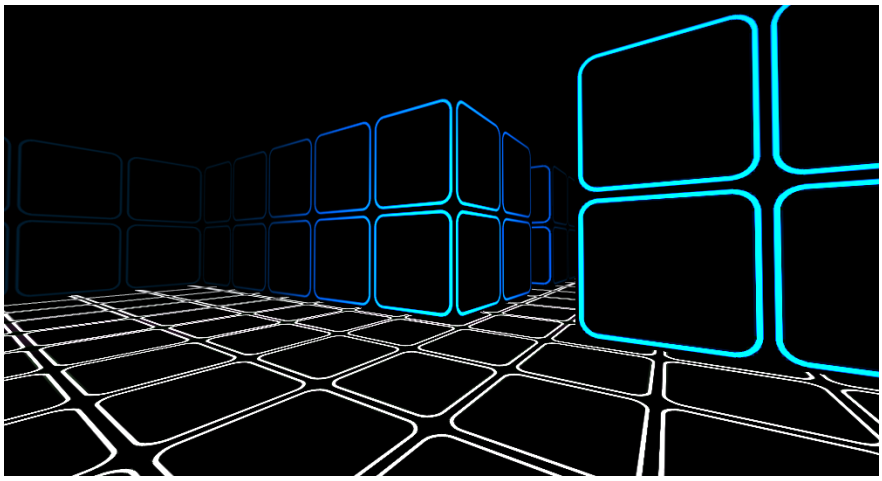


Figure 3.5: The player's view within the FPS maze. This is randomly generated using the RMCM algorithm.

3.2.1 Randomised Multiconnected Environment Generator

Multiconnected environments (also known as conjunct environments, non-perfect mazes or labyrinths) are good candidates for use as a testing environment. This is because much of the real-world is a multiconnected environment, including buildings and road networks. In contrast there are relatively few, real-world examples of linear environments or perfect mazes.

However, few environment generators in the public domain generate a truly stochastic environment. In fact, most public domain algorithms produce a relatively standardised configuration of rooms connected by corridors, conforming to a style which could, in principle be learnt. While the possibility of learning an environment is unlikely in most cases, we must always consider external factors, and clearly with embodied agents, the environmental factor cannot be ignored.

There is also an argument that by conforming to a standard method of generating test environments, benchmarking exercises could be undertaken.

3.2.2 Procedural Dungeon Generation Algorithms

In the previous section (section 3.2.1) we discussed a multi-connected environment comprised of a variety of open spaces and connections. The games industry typically refers to this type of environment as a 'dungeon'. The Future Data Lab website [59] provides a list of various procedural dungeon generation algorithms used in the games development industry, for which we will provide a brief overview.

Random Room Placement: Described as one of the most common dungeon generator algorithms. This places a room of random size randomly on a grid ensuring there are no overlaps. The algorithm then loops over the rooms creating connections using a variant of A* algorithm.

Cellular Automata: This method uses a cellular automaton to create a natural looking cave system. The main difference between this approach and the others described is that it avoids the room and corridors archetype, and instead grows a single, connected space.

BSP Tree: The approach begins with a rectangular, blank dungeon template, which is then subdivided into two spaces of non-equal size, then these new spaces are also subdivded into two, with this process continuing for a set number of iterations. Within each of the newly created sub-spaces, a room is randomly placed, and connections are made between each of the split rectangles. While this approach makes a good use of the initial space, the number of rooms and connections are constrained bu the initial parameters, meaning that the environment is not truly stochastic

Procedurally Built: This approach tries to model the way a man-made dungeon may actually be built. First an initial room is created, from this room a random number of walls are selected, and a door placed along their edge. On the other side of this door, a feature is placed, either a room or a corridor. This grows the environment until a terminal condition is reached (such as the generation of a desired number of rooms). One criticism of this approach is that it makes poor use of space and leads to a very linear environment with few interconnections.

Custom Dungeon Algorithm

On the back of the research conducted into current algorithms available a new version was devised for use within this study (see Figure 3.6).

Our new algorithm begins by generating a number of voxels in a 2-dimensional grid, the number of voxels generated being the product of the width and length of the environment the user specifies. This block of voxels provides us with a blank environment that we can develop from.

The next phase involves stamping random room outlines into the blank environment. Each room is generated at a random initial x, y coordinate, with a size generated as a random sample between a minimum and maximum room size (that is user defined). All the voxels within the room are given a label of 'Room' and given an ID which represents the order in which the rooms were generated. The immediate border voxels around the room are labelled as 'Wall' and given a null ID. In Figure 3.7a the voxels labelled as 'Room' are displayed in grey, whereas the voxels labelled as 'Wall' have been left black.

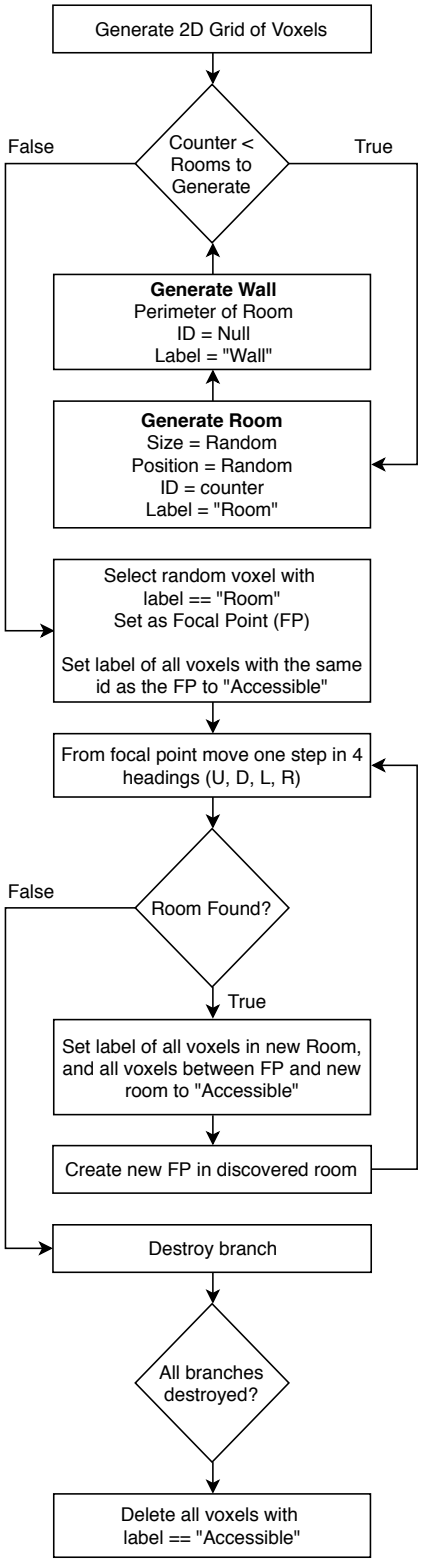
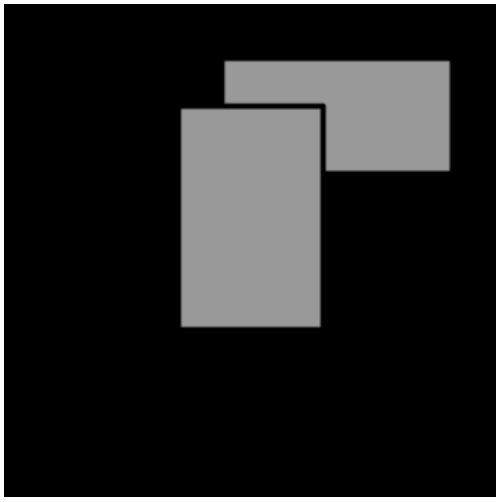


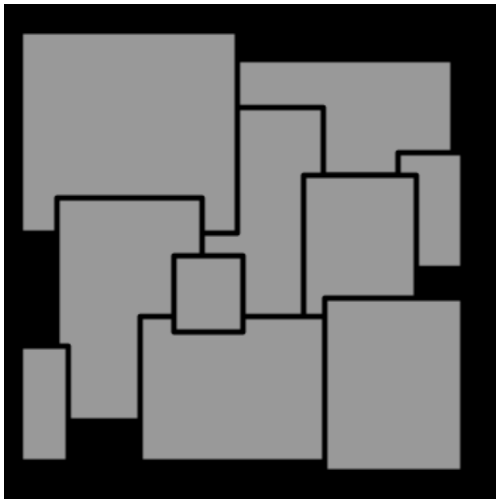
Figure 3.6: The used algorithm as a flow diagram

The algorithm now loops through generating random rooms. Each new randomly generated room overwrites any voxel data previously defined. In Figure 3.7b we can see a second generated room which has overwritten the voxel data in the bottom left of the first room. We can consider this to be overlaying new rooms to generate a patchwork configuration.

(a) The blank environment with a single room.



(b) A second room added to the example environment.



(c) The result of the random room generation.

Figure 3.7: A visual representation of a random room generation in stages.

This process continues generating rooms randomly within the blank environment. This provides us with a floor plan which resembles Figure 3.7c. The method provides us with something which typically fills the majority of the available space organically, without the need for complex space filling algorithms, or the resultant predictable layouts generated by the BSP tree approach.

Once the room generation algorithm has concluded, the next phase is the door placement, ensuring that all rooms within the final environment are accessible to the agent.

This is achieved by first selecting a random voxel with the label ‘Room’ and accessing its ID. In figure 4, this initial voxel has been identified with a black circle, and we will

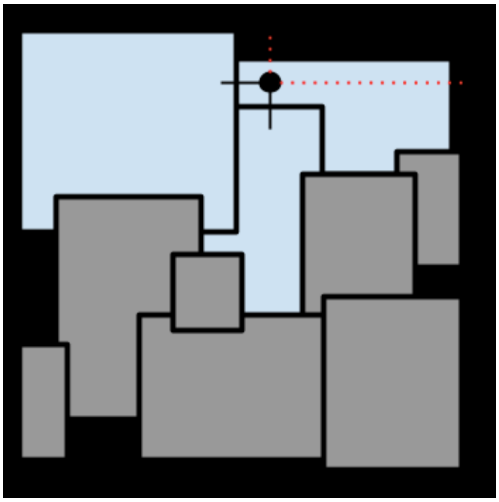
refer to this as the ‘focal point’. Then all voxels which share the same ID as the focal point are instructed to change their label to ‘Accessible’.

Four paths are then generated by stepping through the voxel array in four directions, left, right, up and down. For each step, the current voxel is sampled, and if its ID is the same as the ID of the focal point, or if the it’s label is ‘Wall’, then the steps in that direction continue. If the current voxel is unlabelled, then that path is destroyed, in Figure 3.8a, this is represented by the dashed red lines.

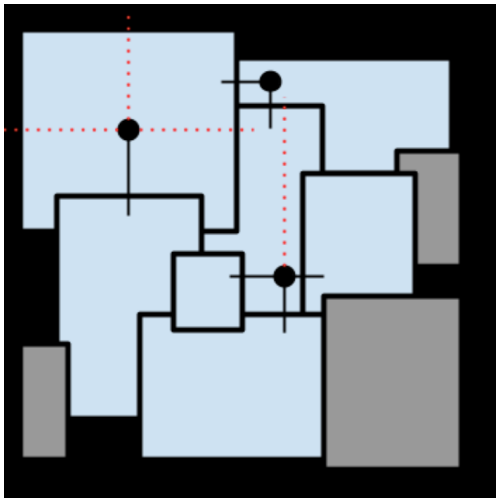
Alternatively, if the label of the current sample voxel is ‘Room’, then the path has found a new room, and all voxels with the same room ID as the sampled voxel, have their label changed to ‘Accessible‘. To determine where a door should be placed, we simply back track along the path until we find a voxel or voxels with the label ‘Wall‘ and change their label to ‘Accessible‘. As shown in Figure 3.8, all rooms which have had their label changed to ‘Accessible‘ are coloured blue, and the paths which have created this route are coloured black.

In each new room discovered, a new focal point is spawned within that room, and the process is repeated, as can be seen in Figures 3.8b and 3.8c. Eventually, a focal point will be spawned, and no new rooms will be found. This has been highlighted in Figure 3.8c with two white focal points which have four red, dashed lines from each. This does not necessarily mean that there isn‘t a connected room (as with the bottom left focal point) but could simply mean that the spawned focal point is just not in a position to discover it.

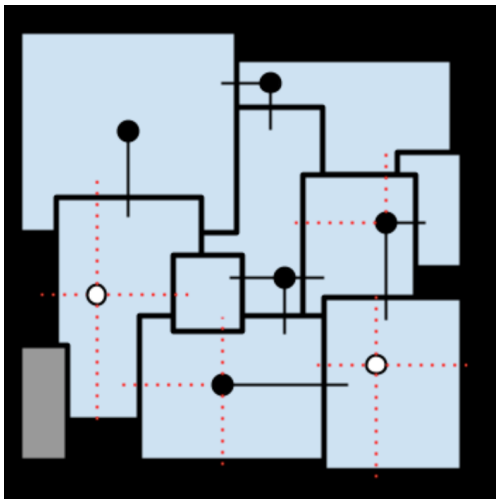
There are a few approaches which can be taken to solve this issue. However, the focus of this algorithm is that the environment is randomised and non-uniform, and a few missed rooms can add to this effect. However, too many missed locations could be detrimental to the environment generation. A compromise solution we implemented was as follows - If a new focal point found no new rooms, it was re-spawned at a different voxel in the environment with an accessible label. If both the first and second attempts failed, then that branch of the tree was destroyed.



(a) The first randomly placed focal point, showing the exploration in four directions.



(b) A second set of focal points are created from the successful branches of the first focal points.



(c) A second room added to the example environment.

Figure 3.8: Door placement through the random placement of a focal point in each room to ensure player accessability.

The search continues until either all the branches from each generated focal point have been killed or, alternatively the user can set a search depth to limit the size of the environment generated.

The final process involves iterating through each voxel in the environment. If the voxel has a label of Accessible, then it is deleted. This removes all the space created by the rooms and connections. The final result is a multiconnected environment, constructed out of rooms with non-uniform layouts and a varying number of connections such as the example in Figure 3.9. The full process can be seen in the flow diagram in Figure 3.6.

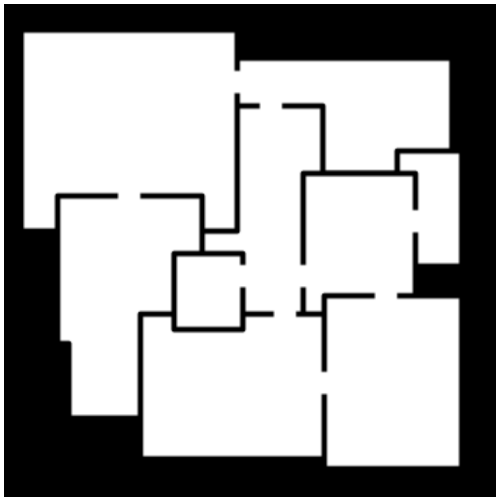


Figure 3.9: The final generated environment

Algorithm Summary

Our approach generates an environment which is clearly stochastic, with a large number of possible rooms and connection configurations. By generating the rooms in a patchwork manner, we remove the possibility that all rooms will be rectangular with a standard number of connecting features. This ensures that all generated environments are truly random and eliminates the possibility that an agent could simply learn a standard configuration.

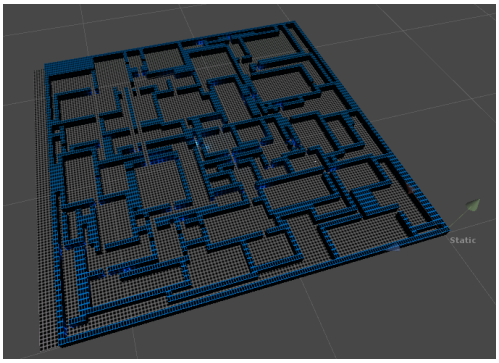
Online Environment Generator

Figure 3.10 contains a web-playable example of the environment generator which can be explored in first person. It also contains a download link for a Unity3D [2] project which includes the scripts to generate the environment. The script is released under a standard BSD licence for use by other projects with attribution. It is expected that we will later include the Random environment generator within a larger benchmarking software, so this release should be considered a community beta release (V0.1). Updates to the software will be indexed on this page and maintained for legacy purposes.

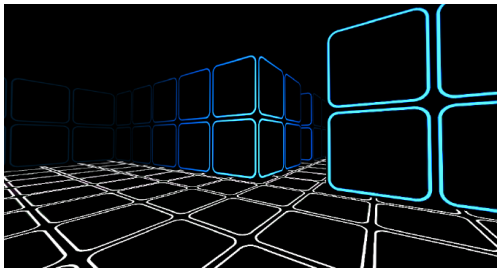
An example generated environment and a first-person rendering can be seen in Figures 3.10a and 3.10b respectively.

Limitations in 3D Rendering

As with any voxel-based algorithm draw calls can be high, especially in environments which utilise dynamic lighting. There are solutions to this issue, such as implementing



(a) An examp,e environment generated with the new algorithm.



(b) A first person rendering from inside the environment.

Figure 3.10: An example of a generaed environment and the first person redndering.

occlusion culling, but this may not be suitable for all applications. Another option is to merge the voxels to create a single geometry or replace large areas of connected voxels prefabricated units.

However, it should be noted that most modern game engines have built-in optimisation procedures which solve most of these issues without the need for a further computational step. In large environments that we experimented with (22500 initial voxels), the Unity engine was able to render to the player at between 9 and 15 draw calls per frame (without dynamic lighting).

3.3 The AI Opponent

The player and the opponent were placed at opposite corners of the environment. The player and the opponent both controlled an identical character, ensuring equivalent speed and manoeuvring ability. The two entities in the game were armed with the same projectile weapon and had the same number of lives.

The AI was controlled by a simple finite state machine (see Figure 3.11) operating in one of three states:

Wander

In the wander state, the AI randomly explores the environment, sensing the world ahead of it with a vision cone of 120°. The vision range is limited only by occlusion from walls in the environment. As the character moves into a new room, it will detect any exits it can see, select one at random and steer towards

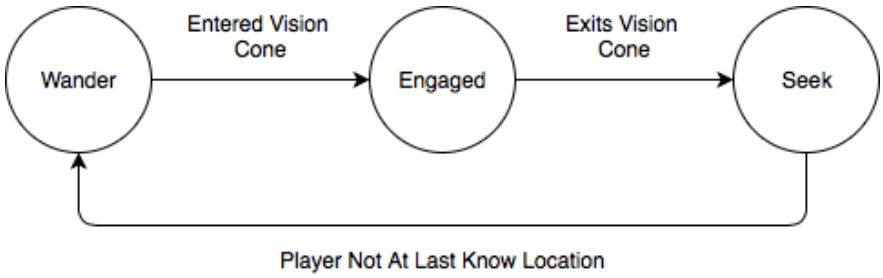


Figure 3.11: The finite state machine controlling the AI character.

it. If it does not see an exit directly ahead, it explores the room using a wander steering behaviour **reynolds1**.

Engaged

If the player enters the opponent‘s vision cone, the AI enters the engaged state. In this state, it will move towards the player firing its weapon. The AI stops moving forward if the player is within a ‘close range’ vision cone, where the length of the close-range vision cone is equal to the diameter of the enemy and the player’s character collider.

Seek

If the AI is in the engaged state, and the player exits its vision cone, it enters a seek state. The seek state causes the AI to turn and move towards the last point at which it saw the player. Once there, if it has not seen the player again (activating the engaged state), it returns to the wander state.

Each time the player was shot, a small health bar provided the player with a visual indicator of the damage. The game was a single round ending with either the player or opponent being destroyed.

3.4 Monitor, Headset and Input Device

We used the same camera position and field of view for the VR and monitor versions of both games. A standard Unity camera was used to render for the monitor versions, The VR versions used the stereoscopic camera implementation provided by Oculus, that renders a separate image for left and right eyes.

The Oculus Rift is capable of tracking the player in 3D space using an infrared camera that tracks the headset's position. This allowed the player to lean around in the VR car game, but not in the 2D monitor version. While this could have a small impact on performance, we deemed positional tracking to be an integral part of VR as it has an important effect on both nausea reduction and immersion.

As stated in section 3.5 half of the four experiments were run on the Oculus Rift, the games were controlled using a gamepad (specifically the Microsoft Xbox gamepad). Once the headset is on, the participant is unable to see the keys of a keyboard, so a tactile controller is more suitable.

3.5 Experimental Methodology

To explore the research questions, 16 participants (12 male, 4 female) were tasked with playing two types of game: a racing game, and a first-person shooter (FPS). Both game types were played through two viewing mediums, an Oculus Rift DK2 and a standard PC gaming monitor. Every person played all four games.

Games were played in a mixed order, and it was ensured that the same viewing medium was not repeated twice (resulting in 8 possible orders of play). Each order was therefore played by two participants

Each participant was told that during the four games, they would play two rounds against a human, and two against an AI opponent. They were told that their task in each game was to identify whether the identity of the opponent was an AI or human.

Participants were placed in a segregated booth, unable to see other people during the experiment. The beginning of each game included a splash screen which implied the game was connecting to a multiplayer server, in order to add to the facade.

However, regardless of whether the person played through VR or monitor, their opponent was the same AI (one AI for the racing game, another for the FPS). The purpose of this deception was to ensure that the players were competing against opponents of identical competence and that in-game ability was not used as a flag to differentiate between opponents. This removes one confounding variable from the experiment and

is consistent with the experimental design of studies with similar objectives [72]. By following this approach, we only identify differences in the perception of AI through the two-viewing media.

At the end of the experiment, participants filled out a survey. For each of the games, they were asked to make an assessment of the identity of the opponent they competed against. This was done on a 1 to 5 scale, with 1 representing high confidence that the opponent was human and 5 representing high confidence that the opponent was AI, a score of 3 indicated that the player was unsure either way. The player was then also asked to rate their enjoyment of the game, and a free text response provided the participant with the opportunity to provide qualitative data.

3.5.1 Ethical Consideration

There are two principle ethical considerations in this study. The first is that the experimental design involves deceiving the participants. However, in this case, the harm caused from the deception is minimal, and it was deemed to be the only practical method of achieving the goals of the study. The second consideration is that video games have been shown to induce motion sickness [99] and that use of the Oculus Rift compounds this nausea in users [17], [18], a condition known as cybersickness. To reduce the risk, we designed both games to adhere to VR best practices [80]. For example, low frame rates are the most common source of sickness, and so we ensured both game types were played on a machine capable of producing a constant 75fps (Frames Per Second), the native frame-rate of the Oculus Rift DK2 [97]. Efforts were also made to alleviate motion sickness due to vection [42], including limiting movement speed in the FPS game.

All participants were informed of the risk before they entered the study and had to read the health and safety information (produced by Oculus) and sign a consent form. Participants were also told that if they felt sick, they could ask to end the study at any time. Additionally, participants were given a short break between each game, and the participants never played two VR games in a row, limiting extended exposure.

3.6 Experimental Results

The results showed a clear split between the monitor and VR based games. However, what makes the results particularly interesting is that the split is inverted for the two game types.

In the VR racing game, players typically reported that the opponent was human. The mean result for all the racing games played through VR is 2.56, placing the average player opinion between human and undecided. This is an inconclusive score by itself, but the mode score of 2 provides further insight into the perception of the players. We can also observe an obvious bias towards the players reporting the opponent as human controlled in the distribution of scores (seen in Figure 3.12).

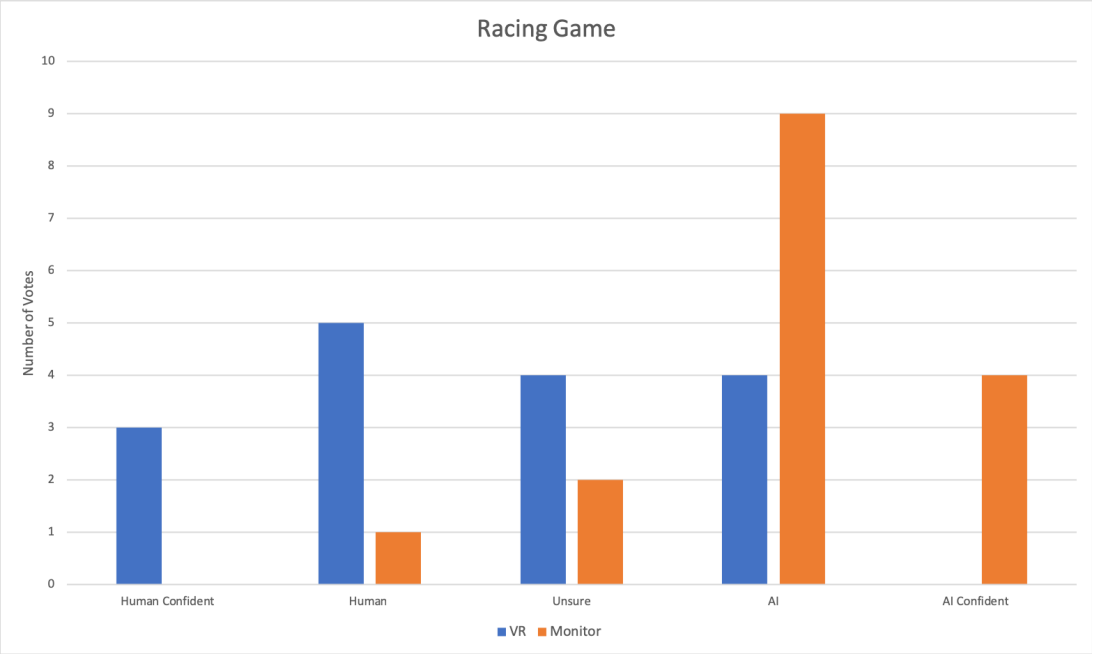


Figure 3.12: The participant's assessment of their opponent's identity for the First-Person Shooter game.

When playing the racing game through a monitor, the players typically reported that the opponent was an AI. The identification here was more statistically obvious, with a mean score of 4 and a mode of 4 (13 of the 16 participants voted that they believed the character was AI). When played in VR, the majority of participants trended towards believing that their opponent was human controlled (mean score of 2.56 and a mode of 2). However, when playing the game through the monitor, the participants were more likely to report a belief that the opponent was AI controlled (mean score of 4 and a mode of 4).

However, we observe an inverse trend in the FPS game. While playing the game in VR, the majority of participants reported that the opponent was AI controlled (mean score of 3.68 and a mode of 4). This falls into a similar distribution to the AI reporting in the racing game, but through the alternate viewing medium (VR rather than monitor).

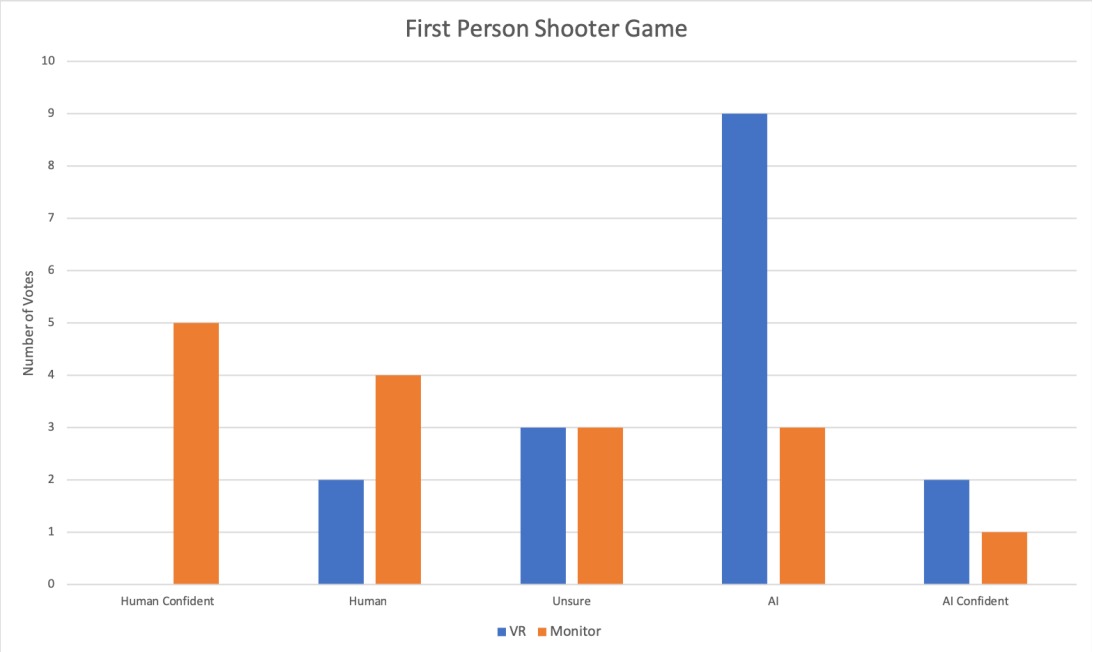


Figure 3.13: The participant's assessment of their opponent's identity for the First-Person Shooter game.

When the participants played the FPS game through the monitor, they trended towards reporting that the opponent was human controlled. As with the racing game, this was reported with significantly less confidence than the reporting of the AI character (mean score of 2.43 and a mode of 1). When played in VR, the majority of participants trended towards believing that their opponent was AI controlled (mean score of 3.68 and a mode of 4). However, when playing the game through the monitor, the participants were more likely to report a belief that the opponent was human controlled (mean score of 2.43 and a mode of 1).

It is perhaps unsurprising that where participants reported that they believed the opponent was an AI (monitor for the racing game and VR for the FPS), they responded with higher confidence. The AI designed for both the games was relatively simple and contained little sophistication or artificial stupidity to make it respond more like a human controlled character. However, as the participants voted with relatively strong conviction in these cases, it is interesting to see participants trending towards reporting

that their opponent was human controlled in the alternative viewing medium (even though this was with less confidence).

We also asked the players to rate their enjoyment of each of the four games. We were expecting to see a correlation between the games where the participant believed they were playing against a human and higher enjoyment. No such correlation existed, with the games that received the highest rating being the ones played in VR. However, we do not assume that this trend necessarily means that players will enjoy VR games more than their monitor based equivalent. The majority of the participants had not played games through VR before the study. As such, the novelty of the new viewing medium likely contributed to the enjoyment results we have reported.

We also captured data regarding who won the game, the player or the opponent. In the racing game, a win was recorded if the player completed two laps in the shortest time. In the FPS, a win was recorded if the player successfully destroyed the opponent before they themselves were destroyed.

As can be seen in Figure 3.14, the player was not particularly successful in either game (6 wins recorded in the racing game, 1 win in the FPS). We assume that this was because the participants were not provided with the opportunity to practice the game before the study, and conversations with participants after the experiment adds evidence towards this suspicion. However, in the racing game, it did appear that the player performed moderately better through VR. We gained further insight through the free text response. In eight of the games, the player reported that the game was easier in VR, with several mentioning that the ability to look around freely was a positive experience.

3.7 Chapter Summary and Conclusion

The important conclusion to draw from this study is that the level of immersion provided through Virtual Reality appears to clearly impact how we perceive AI characters. Despite the study being undertaken with a relatively small number of participants, the results show a clear split in the player's perception of their in-game opponent. We felt that the results derived from 16 participants were showing a clear enough trend that we could progress with further research in this area without having to run a second instance

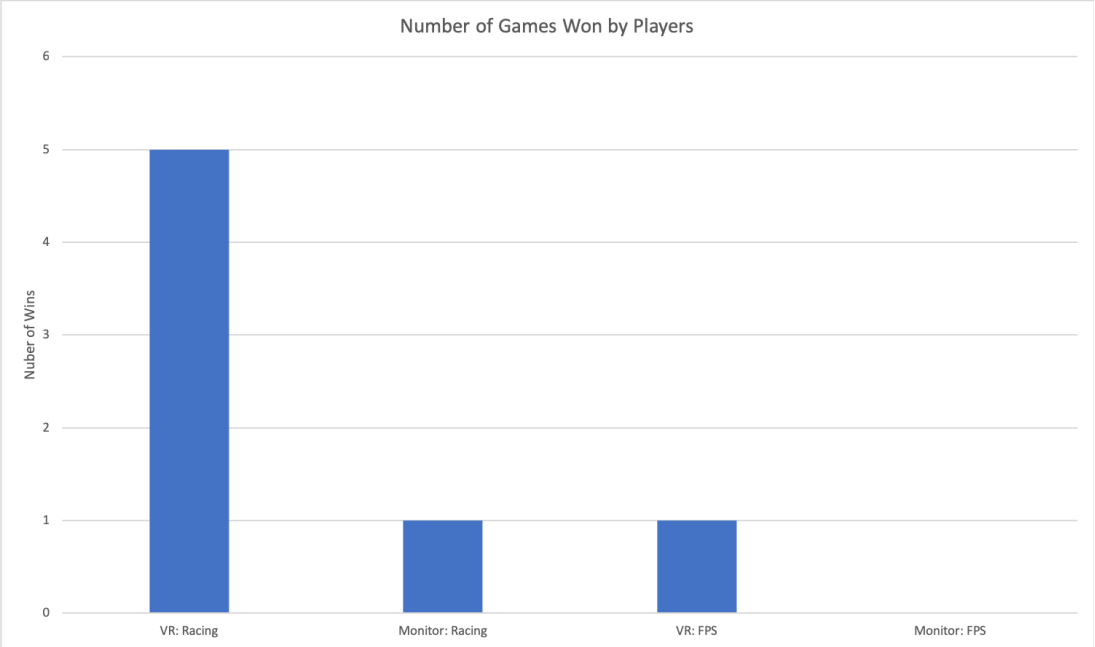


Figure 3.14: Number of players wins for each of the four game instances.

of this study. Clearly, more participants would have given more accurate data to take forwards into future studies but as this was conducted very early on in the research, we prioritised further research into the optimisation of a creature's parameter space. This study could be replicated with up to date more advanced VR headsets to assess if the advances in technology have altered our perception of an NPC or not.

We anticipated that the results for both game types would be the same, demonstrating that VR either makes AI characters more or less obvious to a human player; clearly, from our results, this is not the case. This study indicates that there is likely a link between the way a game is played, and how VR affects the player's perception of the world. Further studies need to explore this in greater detail.

Perception through virtual reality could have clear implications for the future development of AI in games. It appears that VR could have the effect of making AI characters more or less life-like during play, and this will impact how we design them. If prolonged presence is the ultimate goal of VR, it is clear that AI will have a significant role to play. One further consideration is that the Rift DK2 hardware used in this study did not invoke a sense of presence in most people beyond fleeting moments. However, in future experiments newer models of the Oculus Rift were used as they were released, finally ending up with the consumer Oculus Rift headset used in Chapters 5 and 6.

In the next Chapter we describe a second study which uses a specifically designed and developed crowd sourcing tool to tune a simple procedural animation system towards a prescribed and desired outcome.

Chapter 4

An Initial Evaluation of Crowd Sourced Procedural Animation Optimisation for a Simple Animation System

In the previous chapter we described two experiments which aimed to evaluate how players perceive AI characters differently between a standard 2D monitor and VR. To do this we designed and implemented two games, one where the player races a car around a track and a first-person shooter set in a randomly generated maze, where players aim to track down and shoot an AI player. Our results showed that there was a clear split between the monitor and VR based games. However, the results were inverted for the two game types. In the VR racing game, the player typically reported that the opponent was human where as they thought the opponent was AI in the VR first person shooter. The opposite was true for the 2D monitor versions of the games where the player perceived the opponent to be AI in the racing game and human in the first-person shooter. This indicates there is a link between how a game is played and how VR affects the players perception of the world.

Procedural animation systems are capable of producing vivid organic character behaviour and motion automatically. However, these systems can consist of hundreds of interlinked parameters yielding a very large search space for potential animations and emergent behaviours. This chapter describes the creation of a crowd-sourced optimisation tool that uses genetic algorithms to anneal the parameters of a simple procedural animation system. There were three main parts to implement for this study; the procedurally

animated creature (see section 4.1), the rating system (see section 4.2) and the genetic algorithm (see section **??**). Great consideration had to be taken with the design of all of these sections as small alterations could potentially alter the results drastically.

The opening sections of this chapter discuss the overall design and setup of this study with some of the techniques also used later on in Chapter 5. The final part of this chapter discusses the results of the study and how our findings informs future research.

Parts of the following Chapter were published in [35], [36], where the former received the award for 'Best Poster'.

4.1 A Prototype Snake Model

For this prototype study a primitive snake like creature was created in the Unity 3D game engine. The idea behind this was to create a very simple creature which can be physically altered to test the robustness of our genetic algorithm and ratings system. The snake itself is a sphere game object with a trail renderer attached to it. These combined give the appearance of a snake like creature (Figure 4.1). The snake is physics driven with two parameters controlling the speed and turning rate of the creature respectively. Each snake turns towards an invisible target which is reset to a new position (within a -20.00 to 20.00 scale on both the x & y axis) at a time scale determined by either the $targetMove$ parameter or the snake itself colliding with it. $trailTime$ describes the length of each snake's tail, using a trail renderer explained in the next section. Finally, the RGB elements control the colour of the trail renderer itself. The three colour elements and $trailTime$ work together to create the appearance of the snake itself whereas, $topSpeed$, $turnSpeed$ & $targetMove$ adjust the movement of the snake.

4.1.1 Trail Renderers

Unity's trail renderer generates a trail of polygons behind a moving game object. This can be used to give a moving object a comet-like tail, and highlight the position or path taken by that object. A real-world example of a trail renderer would be the contrails given off by a plane's engines whilst in flight. It is good practice to have a trail renderer as the only renderer attached to a game object.

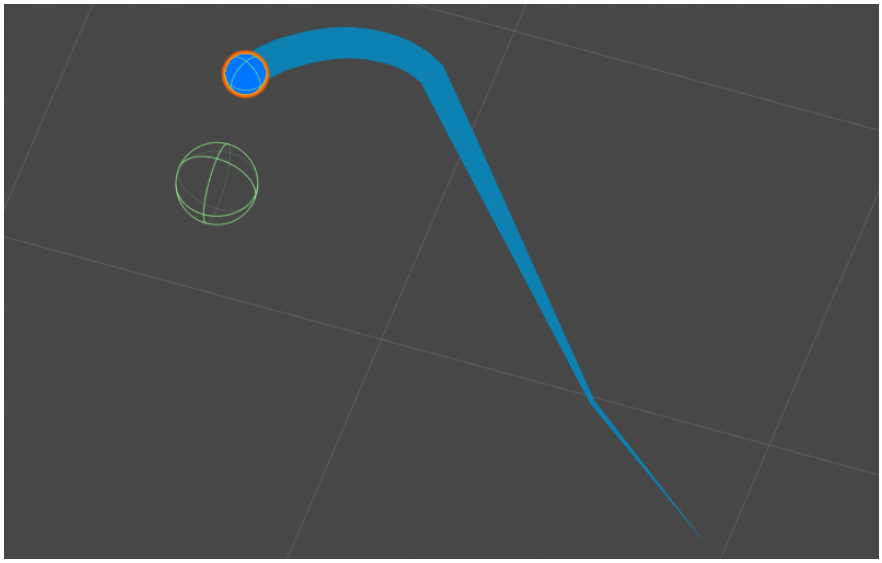


Figure 4.1: A snake model comprised of a sphere game object and a trail rendered to give the appearance of a tail. Also, an example of a snake moving towards the sphere target within the scene.

Our trail renderer uses the setup shown in Figure 4.2. In Table 4.1 we describe the different properties of this trail renderer, and how it effects the appearance of the snake's tail. We have only highlighted the properties which have had an effect on our snake. Some of these properties have been left to the default value as these were the most appropriate for our model. All definitions for the properties have been taken from the Unity Documentation [103].

Table 4.1: Trail Renderer Properties

Property	**Function**	**Use In The Snake**
Cast Shadows	Determines whether the trail casts shadows, whether they should be cast from one or both sides of the trail, or whether the trail should only cast shadows and not otherwise be drawn.	We set this to "On" as it gives the appearance of a more realistic snake.
Materials	These properties describe an array of Materials used for rendering the trail. Particle Shaders work best for trails.	We created a material (*Trail1*) which was then accessed through a script to alter the colour of the trail renderer.

Continued on next page

Time	Define the length of the trail, measure in seconds.	This is one of our adjustable parameters (Section 4.1.2) to control the length of the tail. Can be anywhere within 1.00 - 5.00.
Min Vertex Distance	The minimum distance between anchor points of the trail.	This is as small as possible as we wished the renderer to appear like it was part of the snake's body. By making a smaller number this reduced any visible gap.
Width	Define a width value and a curve to control the width of your trail at various points between its start and end. The curve is applied from the beginning to the end of the trail and sampled at each vertex. The overall width of the curve is controlled by the width value.	The trail gradually decreases in width towards the end, like a snake's tail would. With a max width of 1.00 as this was the size of the sphere itself.
Colour	Define a gradient to control the colour of the trail along its length.	This property gets adjusted through code to match the RGB values pulled in from our server

Continued on next page

Texture Mode	Control how the Texture is applied to the Trail. Use Stretch to apply the Texture map along the entire length of the trail or use Wrap to repeat the Texture along the length of the Trail. Use the Tiling parameters in the Material to control the repeat rate.	We stretched the material to cover the renderer just once, as its a solid colour this was this was the most appropriate selection.

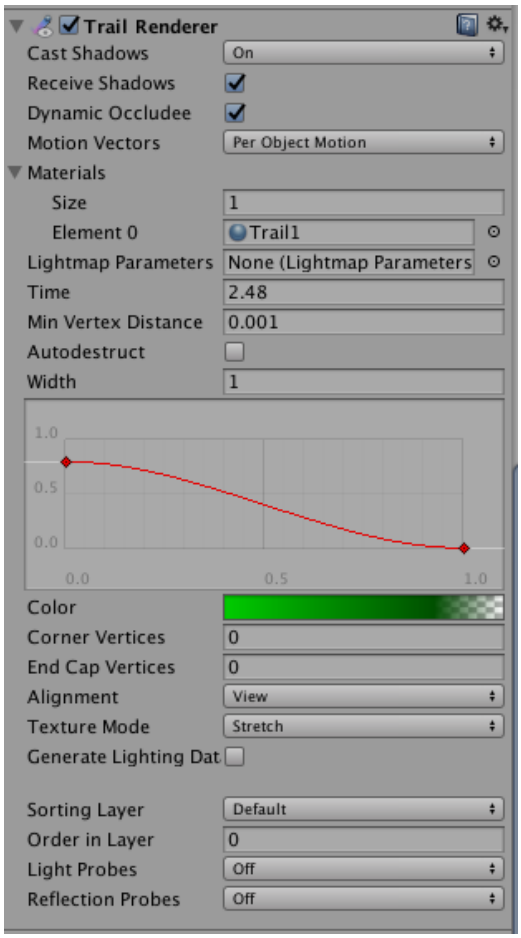


Figure 4.2: Trail Renderer settings as used with the snake

The snake moves in 2D across a 5x5 plane and can never fall off the edge. This background plane was given a bright grid texture so that the snakes were easily viable within the environment. We also included text to remind the participant on the purpose of the study.

Each snake has 7 adjustable parameters (see section 4.1.2). Outer bounds had to be set of each individual parameter, and these were initially set using our best judgement as to what would be perceived as realistic. This was applicable for all parameters apart from the colour elements (RGB) which have a normalised scale of 0.00 - 1.00.

When initialising the snakes, a parameter file is created and uploaded to the server. The seven adjustable parameters with the ranges allocated yield 2.4E16 possible combinations. To test every single possible creature would be impossible so we chose to take a sample of this for our study. 100 total snakes were created with each parameter given a random value within the bounds allocated (Section 4.1.2). This means that these 100 snakes from the initial generation represent 4.166E-15% of all possible permutations. 100 snakes were used per generation as we aimed to obtain 100 ratings before the genetic algorithm processes the data.

4.1.2 Snake Models Parameters List

Below is a complete breakdown of the parameters used in the creation of the snakes. The parameters name as used within the program is stated. The range and granularity describe the lower and upper bounds and the scale of the parameter respectively. Associated parameters are any of the other parameters which directly impact or are impacted by this. Finally, the description provides a summary of what the parameter is used for.

Top Speed:

Range - 5.00 - 20.00

Granularity - 0.01

Associated Parameters - Turn Speed & Target Move

Description - This parameter controls the speed in which the snake moves around the environment. The larger this value becomes the faster the snake will move around. This speed-based value controls the translation of the sphere.

Turn Speed:

Range - 4.00 - 6.00

Granularity - 0.01

Associated Parameters - Top Speed & Target Move

Description - As this value controls the turning angle of the snake. Therefore, it will be able to perform a sharper turn as the value is increased. This is a speed controller and overall controls the rotation of the sphere.

Trail Time:

Range - 1.00 - 5.00

Granularity - 0.01

Associated Parameters - RGB

Description - The value describes the length of the snake's tail. The tail length is controlled in seconds, this means that with a value of 5 the tail renderer will last for 5 seconds before phasing out.

Target Move:

Range - 1.00 - 5.00

Granularity - 0.01

Associated Parameters - Turn Speed & Top Speed

Description - Target move links to an invisible sphere collider within the environment. The snake is always trying to reach this target. The value controls how frequently this target is moved, and it can be placed anywhere from -20.0 to 20.0 on both the *x* & *y* axis. If the snake reaches and collides with this target this also causes it to move to a new location.

RGB:

Range - 0.00 - 1.00

Granularity - 0.01

Associated Parameters - Trail Time

Description - These control the Red, Green & Blue elements of the trail renderer. A combination of these 3 values gives the snake its tail colour. The higher the value the stronger that colours pigment will appear. For example, if RGB = 0.01, 0.01, 0.99 then the snake will have a very Blue tail.

4.2 Ratings System

The study relies on crowd sourcing as its method for optimising the parameter space. We have done this as we wish to obtain as wide a demographic as possible and optimise towards this. The system is designed to aid developers in creating virtual creatures which appear realistic to the wider population and not just experts in the field. By acquiring the views of a wider demographic only further tests the robustness of our system.

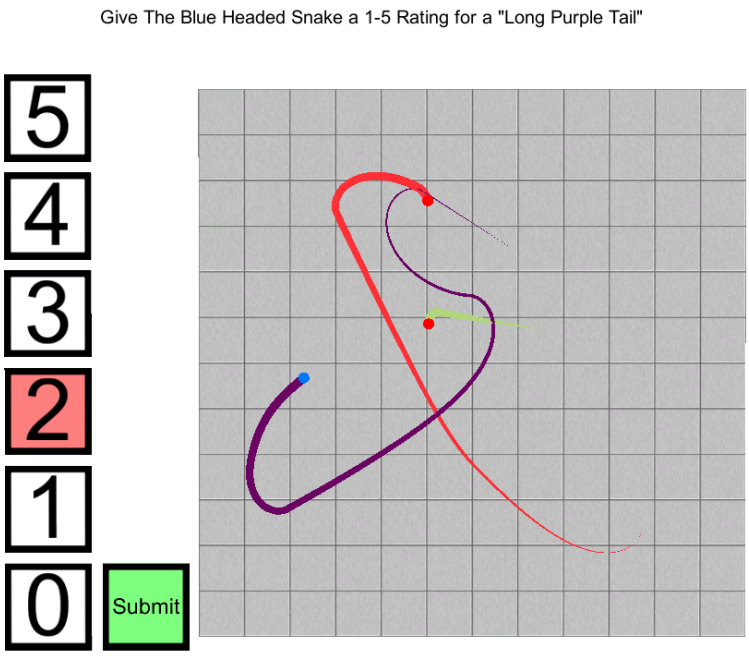


Figure 4.3: The 0 - 5 rating system used within the prototype experiment.

This study used a 0 - 5 rating scale (Figure 4.3). When choosing a rating system, all possible options had to be considered. A rating scale is a set of categories designed to draw out information about a qualitative or quantitative attribute. These require a user to assign a value to an object which is to be rated based upon an attribute. Rating scales can be classified into three categories:

- Numeric Rating Scale
- Graphical Rating Scale
- Descriptive Rating Scale

We analysed many relevant ratings systems which could be used within our experiment (Table 4.2). It was concluded that the most appropriate system was a slight variation of the Likert Scale. We ask the users to rate the snake on a scale of 0 - 5 (Figure 4.3). This is similar to the functionality of a Likert Scale as 0/1 represent to the strongly disagree portion, 2/3 are more neutral with 4/5 being strongly agreeing.

Table 4.2: A Comparison of Different Rating Scales

Rating System	Definition
Likert Scale	This is a scale used to determine the attitudes of a participant towards a topic.
Guttman Scale	A Guttman scale presents a number of items to which the person is requested to agree or not agree. This is typically done in a 'Yes/No' dichotomous format.
Thurstone Scale	A Thurstone scale has a number of agree or disagree statements. It is a unidimensional scale to measure attitudes towards people.
Osgood's Semantic Differential Scale	A user is asked to choose on a position on a scale between two polar opposite adjectives "Good - Evil" or "Hot - Cold").

4.2.1 Snake Rating System

As discussed in the previous section we decided upon a 0 - 5 rating scale. As shown in Figure 4.4 the ratings numbers are textured cube game objects. These acted as the buttons a participant could use to select their rating. As this study was run in a web browser, the rating system was designed to be scrolled through using the arrow keys on the user's keyboard. Users can scroll up and down and once happy with their selection move across to the submit button and click enter. We implemented a submit button as this would reduce the amount of accidental ratings given by participants.

4.2.2 A Users View of the Application

Upon starting the application, users find themselves viewing three snakes roaming around the environment with a fixed ratings scale on the left (Figure 4.3). In the Figure the user was required to rate the blue headed snake on how long and purple its tail

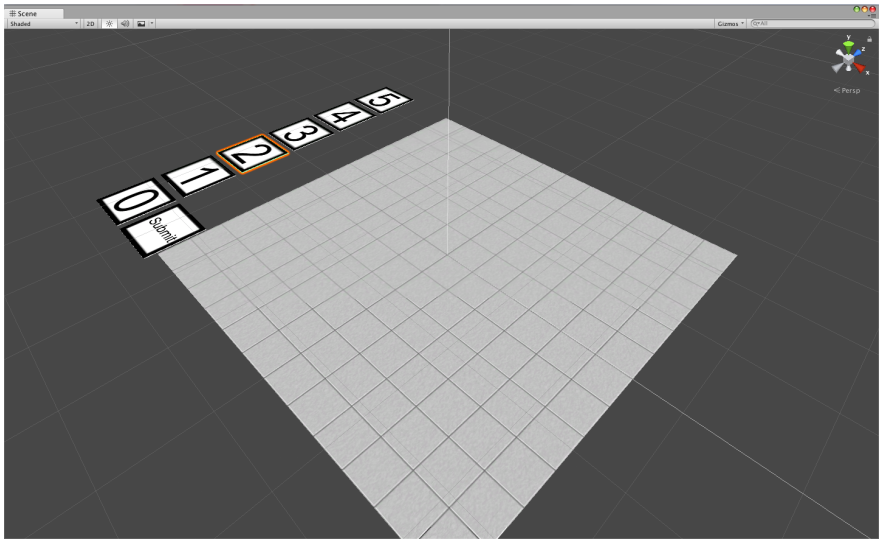


Figure 4.4: The cubes and plane used within the environment for rating selection and snake's platform.

is using the aforementioned ratings scale. Each of the three snakes have different parameters and therefore varying behaviour and appearance. The program randomly chooses the snakes, one of which will have a blue head and two with red heads. The red headed snakes are in the environment to be used purely as a comparison.

Each participant was asked to rate each snake on a 0 - 5 scale. If a snake appeared inactive or broken, then it would receive a rating of 0, with progressively higher ratings awarded for a longer and more purple tail. We highlighted that there was no right or wrong answer with this test as we wished to acquire the opinions of the participants throughout. The ratings could be selected through the use of the arrow keys on a keyboard, hitting enter once the submit button was highlighted. There was no end point to this experiment and users could rate snakes for as long as they wished, but we advised to perform the task for a minimum of 2 minutes as they would be able to rate enough snakes in this time to form a fair opinion.

We used a server to store and update the generated parameter files and associated ratings. Using a server allowed for multiple users to run the program and rate creatures at the same time. When a creature is rated, the time-stamped parameters and associated rating are recorded. Snakes 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 snake 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.

4.3 Analysis of Experiments Participants

A seasoned gamer could have a biased opinion on what they perceive to be realistic as they may have been influenced by previous experiences, whereas a non-gamer could have a very different opinion. It has been found that video game players can often out perform a non-gamer on measures of perception and cognition [7]. If gamers are recruited for their experience, they may expect to perform the task well given their expertise. However, a belief that you should perform well can influence performance within some environments [61]. Individuals with a longer history of video gaming have better performance levels within virtual tasks. Those with a navigational gaming background performed far better to those with a more rounded gaming history when undertaking a navigational based virtual task [77]. In order to gain as many participants as possible, our experiment was advertised primarily though social media including Facebook, LinkedIn, Reddit and Twitter. As this experiment was online and anonymous, we are unable to map any details about each participant. One issue with anonymous online crowd-sourcing is that it is very difficult to capture and remove 'false' data. Filtering is one of the more commonly used methods during crowd sourcing applications and systems [82], [83]. While, these methods are suitable for some experimentation this would not have been effective for this study as it would have had to be a manual process of checking each individual result as they came in and this would not have been feasible given the resources available. Ways to have improved the quality of data could have included a brief registration system so that once a participant has 'logged in' their data is stored separately until it can be verified and used for the experiment or removed without influencing the experiments outcome. By implementing a flagging system which can detect potential anomalies in the data collected the burden on checking each individual's data would be reduced. Something to note is that even with an in person experiment it is difficult to capture 'false' data as it is the perception of the participant that is being assessed which is very subjective and unique to each person. In that case it would be very difficult to distinguish between someone's actual opinion and what we may perceive to be 'false' data.

4.4 Populating Further Generations using a Genetic Algorithm

A genetic algorithm is a meta heuristic inspired by the process of natural selection. The idea behind a genetic algorithm is to simulate what nature does. In simple terms a genetic algorithm can be likened to a simulation of Darwin's Theory of Evolution whereby the fittest survive. Genetic algorithms were introduced by John H. Holland in his book "Adaptation in Natural and Artificial Systems" originally publised in 1975 [44].

A genetic algorithm will vary greatly depending on the complexity required but they all follow a similar structure (Figure 4.5). Firstly, a population of individuals are randomly generated at the start of the process. These individuals are then allocated a fitness value and ranked accordingly. These individuals are then selected depending on their fitness value. Crossover and mutation procedures are applied to individuals in the population of each generation according to some probability in order to create genetic variation. The original population is then replaced by the new generation and this process continues until some termination is met.

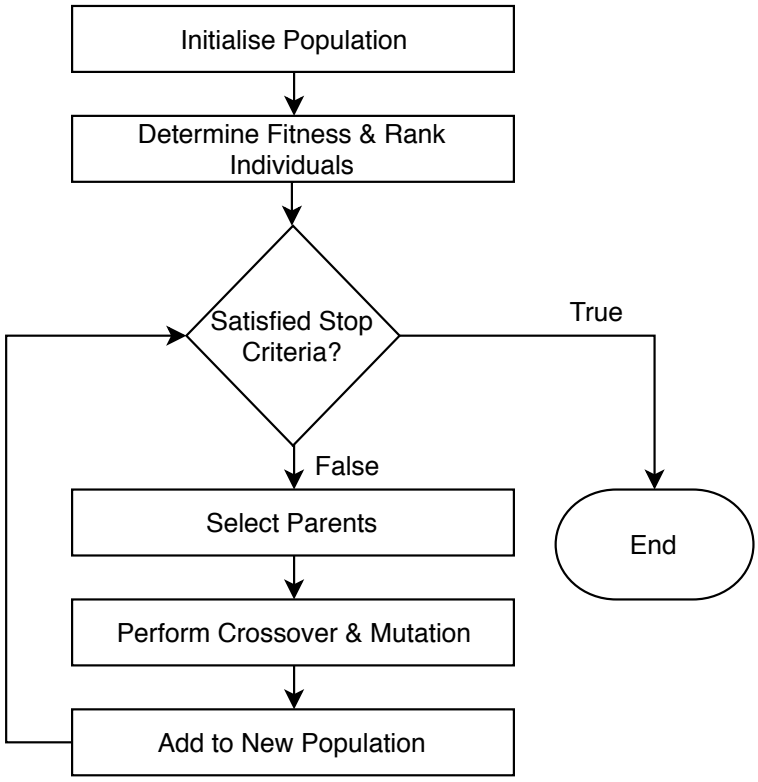


Figure 4.5: Flowchart of the general structure of a Genetic Algorithm.

A genetic algorithm usually comprises of three main parts:

- Selection Method
- Cross Over Method
- Mutation Method

Each of these parts have been explored in greater detail in their relevant sections below.

4.4.1 Selection Methods

An important part of any genetic algorithm includes a selection method. This is where individual genomes are chosen from a population for breeding using a crossover operator. Whilst there are multiple options, it was decided that using Roulette Wheel Selection was the most appropriate for these studies; this and other selection methods are discussed below.

Roulette Wheel Selection

Roulette Wheel Selection is generally regarded as one of the most commonly used chromosome selection methods. The fitness allocated to each creature is used to associate the probability that it will be selected through this methodology. If f_i is the fitness of creature i, the probability (p_i) of it being selected is

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{4.1}$$

where N is the number of individuals in the population. Within our study this means that the higher the average rating received for a parameter row the more likely it is going to be selected as a parent for the cross over method. This can be viewed similarly to a roulette wheel in a casino. Each parameter row is given a portion of the wheel and that is sized based on their fitness value (Figure 4.6).

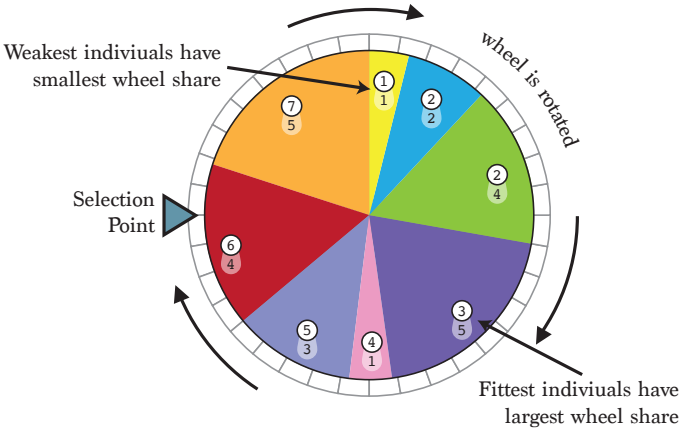


Figure 4.6: A visual representation of Roulette Wheel Selection

Tournament Selection

Another more commonly used selection method is tournament selection. In its simplest form it selects two chromosomes and puts them against each other to determine which one gets selected. The winner is determined through its fitness, therefore the fittest of the two candidates gets selected for the crossover. A more common version of this selection method chooses N individuals at random from a population, the individual with highest fitness value from these is selected as a candidate for crossover. The tournament size is the number of individuals taking part. An advantage of tournament selection is the computational time required is reduced using this method, strong individuals do not take over the cross over procedure and no need for a fitness scale or ranking. In the example (Figure 4.7) the tournament size is three, which means that these three individuals compete against each other to be selected. The larger the tournament size the greater the chance of a loss of diversity as individuals with lower fitness values will struggle to win larger tournament. An issue with this selection is that the weakest candidate will never be selected as it would never win a tournament unless it was able to be selected multiple times and was the only one competing in the tournament. This could mean that the weakest candidate is automatically ruled out of future populations.

Linear Ranking Selection

Ranking selection was originally introduced by Baker in 1985. It was created as a way to deal with inhibiting premature convergence through the use of the adaptive selection methods [2]. As with the other selection methods the population is first sorted according to their fitness value. Each individual is then given a rank, with the fittest individual

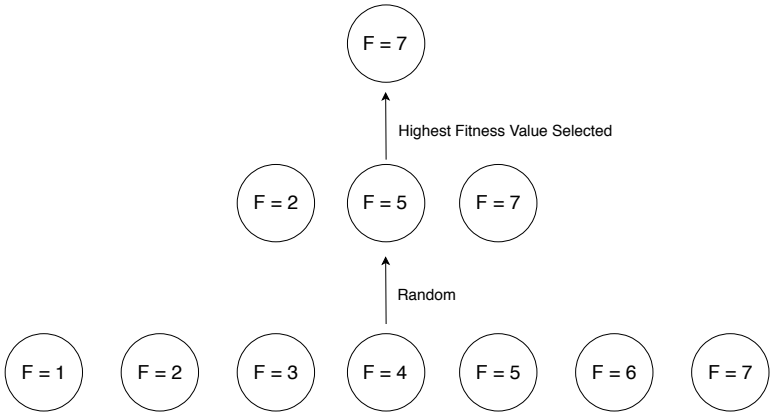


Figure 4.7: A visual representation of Tournament Selection

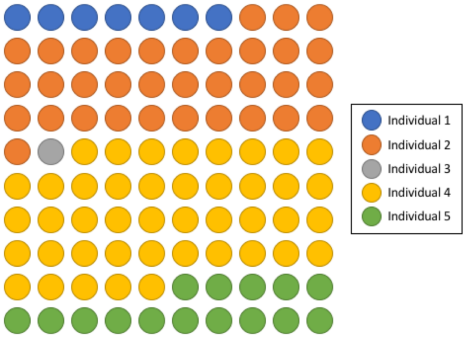
given rank N and the least fit receiving rank 1. Each individual is then given a portion of the selection probability based on its rank (Figure 4.8). Prior to ranking there is a disproportionate advantage for the higher rated individuals giving the lower rated individuals a 1% chance of being selected.

$$p_i = \frac{1}{N}\left(N^- + (n^+ - n^-)\frac{i-1}{N-1}\right) \tag{4.2}$$

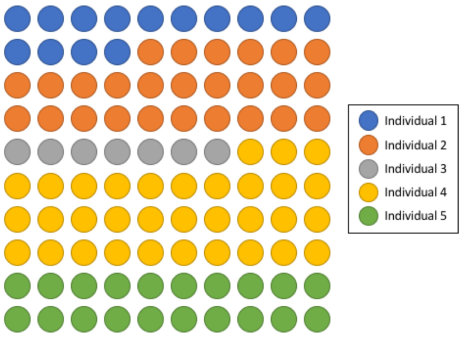
Where p_i is the selection probability of ith individual. n^+/N is the probability of selection of the best individual and n^-/N is the probability of the worst individual being selected. An example of rank selection is given below (Table 4.3). Each individual has been given a hypothetical rating and then placed into rank order based on this rating. After ranking each individual has a portion which represents their order in the ranking and not their individual rating, therefore giving a more proportionate chance of being selected.

Table 4.3: Example data for rank selection.

	Rating	***Rank***	***Portion Allocated***
Individual 1	15	2	7%
Individual 2	73	4	34%
Individual 3	2	1	1%
Individual 4	93	5	43%
Individual 5	33	3	15%



(a) Probability of selection prior to ranking.



(b) Probability of selection after ranking.

Figure 4.8: A visual representation of Rank Selection

4.4.2 Fitness Function Variation

A selection method within a GA can take many different forms as described previously (Section 4.4.1. For our study we have relied on the data collected from crowd sourcing to act as the fitness functions. There are some ways in which the effectiveness of our genetic algorithm could be improved through the implementation of variants. Fours approaches of implementing variants into a genetic algorithm have been considered:

1. **Elitism:** This is a 'best must survive' approach. It works by selecting the best or fittest individual(s) from the parent population then adding it directly into the next generation. This method can have a dramatic impact on the performance of a GA because it can ensure that time is not wasted re-discovering previously discarded solutions. Candidates selected through elitism also remain eligible for selection as parents when breeding the remainder of the new generation.

2. **Adaptive Genetic Algorithms:** These are GA's whose parameters such as the population size, crossover method, or mutation chances vary whilst the genetic algorithm is running. An example of this could be: The mutation chance is altered according to the changes of the new generations. If the population does not improve at a suitable rate, then the mutation rate could increase to try and prompt the system to increase its progress. This could then decrease again once improvements are back to a satisfactory level.

3. **Hybrid Genetic Algorithms:** These are commonly used when additional auxiliary information such as derivatives or other knowledge is already known

about an objective function. The optimisation task could be divided into two complementing sections: The global optimisation which is conducted by the genetic algorithm itself and a more local refinement is done by the conventional method. An example of this could be: The GA performs its population of the new generations. This is then followed by the local method which for every n generations the population is compared to and bread with a locally optimised individual.

4. **Self Organisation Genetic Algorithms:** These methods do not only rely on data as its object for evolution. Parameters of the GA such as its genetic operators and coding functions are optimised too. If this is implemented correctly, the GA could find its own optimal way for representing and manipulating the data automatically.

The fitness function value for this study and future studies is based on the rating that the creature receives. To determine this, we begin by averaging out the ratings for each individual creature as they could be rated more than once. They are then sorted into descending order according to their average score with the top 25% automatically being selected for the new generation. All of the creatures are then used within the selection method with their average rating acting as their fitness function, i.e. the higher their average rating then the greater the chance they will be selected to be a parent for use in the crossover and mutation methods. Algorithm 2 describes the fitness function method of the optimisation algorithm in pseudo-code.

Fitness Function Algorithm Pseudo-Code

Algorithm 2: Fitness function pseudo-code used within the optimisation algorithm.

```
1 merge any duplicate creatures
2 calculate the average score for each creature
3 sort the the snakes into descending score order according to average score
4 copy top 25% rated creatures into a new generation
5 use all creatures within selection method based on fitness function
```

4.4.3 Cross-Over Method

This is a genetic operator used to vary the programming of a chromosome(s) from one generation to the next. It is based upon biological crossover and reproduction. A cross

over takes more than one parent and produces one or more offspring from them. We considered 3 types of cross over method: Single-Point, Two-Point and Uniform Point crossover. All methods described use two parent chromosomes and swap data points between them to create two children which get put into the new population.

Single Point Crossover

As the name suggests this method selects a random point within the two parents and swaps their genomes around after this point, therefore creating two new children (Figure 4.9).

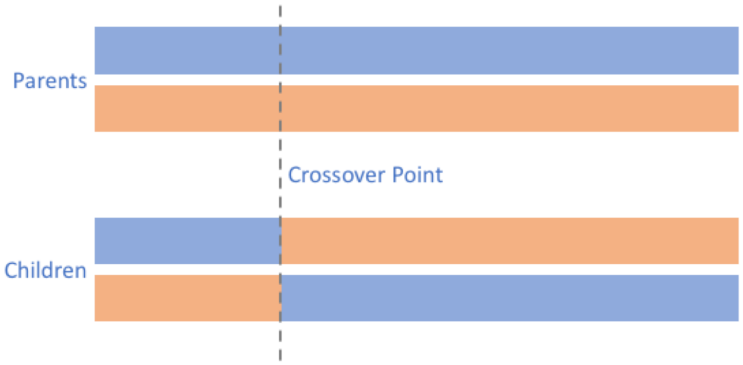


Figure 4.9: A visual representation of Single Point Crossover

Two Point Crossover

Similarly, to the above random points are selected for the crossover, this time however it uses two points to decide where to be crossed therefore producing a more mixed child (Figure 4.10).

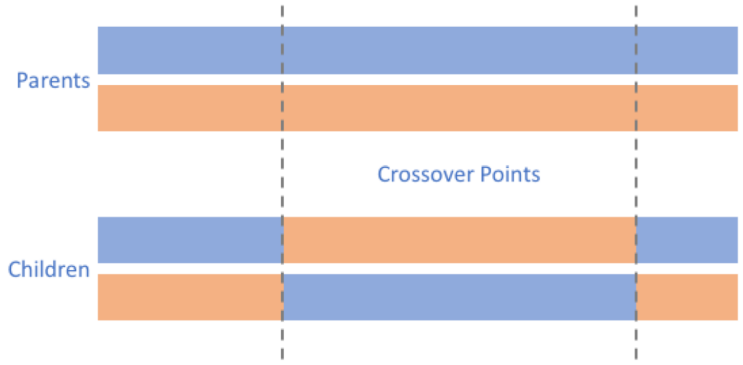


Figure 4.10: A visual representation of Two Point Crossover

Uniform Crossover

Finally, uniform crossover looks at each section of the parents individually and random decides which child it should be placed into. This method creates the most mixed children and the offspring could be vastly different from the parents (Figure 4.11).

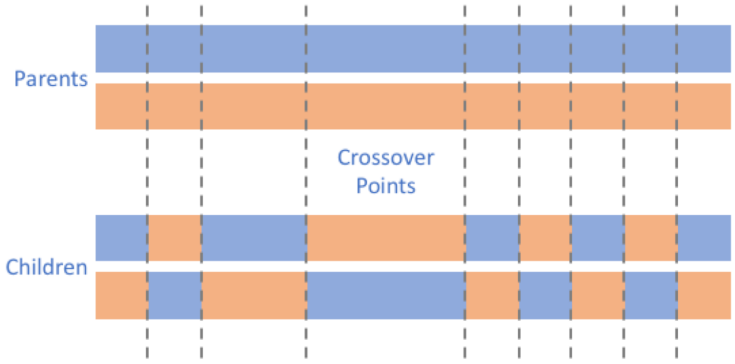


Figure 4.11: A visual representation of Uniform Crossover

4.4.4 Mutations

A mutation is a key part of any genetic algorithm as it maintains genetic diversity between generations of chromosomes [28], [45]. The role of a mutation is to potentially restore lost or unexplored genetic material. This should help to prevent premature convergence of a genetic algorithm to sub-optimal solutions. Generally speaking, the chance of a mutation happening is set to be very low otherwise the algorithm could turn into a generic random allocation. A mutation operator only works on one gene at a time and this is then given a chance to be modified separately from the rest of the genes [19]. Many studies have shown that having an ever-changeable mutation rate is preferable when compared to a fixed rate [1], [23], [41]. Having an adaptable mutation rate has clear benefits as it would allow for a larger parameter space to be searched early on in the study to ensure population coverage. On the other hand, we prefer more exploitations towards the end of the study or in later generations would ensure the convergence of the population towards a global optimum. Due to the simplicity of the GA required for this study it was concluded that a fixed mutation rate was applicable for this application. The more important factor for this research was the chance a mutation would occur. When implemented within our GA the mutation rate was set to 1%. This was due to the reasons discussed in Section 4.3, it can be problematic attaining enough data to be able to ensure a large enough population coverage, therefore by having a higher mutation chance we can allow for much of the parameter space to be tested through each

generation. Therefore, it was determined that a mutation rate of 1% would be small enough to not cause any major discrepancies within our results but also large enough to provide us with a good population coverage. A mutation can be implemented in many different forms to suit the algorithms needs, below are discussed the ones which are applicable to our research.

Uniform Mutation

The chosen value (parameter) will be replaced by a random uniform value within the selected upper and lower bounds of that gene. A random number r is generated, this is then compared with the mutation rate P_m (a user specified parameter). If $r > P_m$ then this gene is mutated. When used with floating point numbers the gene is given a random value within the bounds of that gene. For example, if gene 3 (G_3) is selected where G = [2.312, 4.265, 6.231, 1.423], and the bounds of G_3 is [5.000, 7, 000], then after a mutation has occurred the chromosome may now be G = [2.312, 4.265, **5.923**, 1.423].

Non-Uniform Mutation

To reduce the randomness of a Uniform Mutation, Michalewicz presented a dynamic mutation operator called non-uniform mutation [74]. This works as follows, if gene G_i was selected for mutation, the offspring would be determined using the following formula:

$$G_i = \begin{cases} x_i + \Delta(t, UB_i - X) & \text{if random digit (a) is} > 0.5 \\ x_i - \Delta(t, X_i - LB_i) & \text{if random digit (a) is} \leq 0.5 \end{cases} \tag{4.3}$$

where a is random float between 1 and 0. LB_i and UB_i represent the lower and upper bounds or the gene respectively. The function $\Delta(t, X_i)$ is defined by

$$\Delta(t, X_i) = X\left(1 - r^{(1-\frac{t}{T}^b)}\right) \tag{4.4}$$

where r is also a random number between 0 and 1. T is the maximal generation number, and b is is pre-determined degree of dependency on iteration number t. The formula $\Delta(t, x)$ returns a value in the range of [0, x] such that the probability of $\Delta(t, x)$ being

close to 0 increases as t increases. This property ensures that the newly generated gene is within the feasible range. Furthermore, it enables the operator to explore the parameter space very broadly initially with a large mutation step, decreasing to a more local area at later stages of the process once a suitable gene has been determined.

Boundary Mutation

Boundary mutations works in a similar way to uniform mutation in that a gene is random given a new value based upon its upper and lower bounds. But with this mutation method, the new value is set to either the upper of lower bound with equal probability. For example,

$$G_i = \begin{cases} UB_i & \text{if random digit (a) is } > 0.5 \\ LB_i & \text{if random digit (a) is } \leq 0.5 \end{cases} \tag{4.5}$$

where the each of the parameters have the same meaning as in non-uniform mutation.

This operator is especially useful when the optimal solution for a gene lies near too one or the other boundary in the search space. However, this method would not be useful if the whole gene space needed to be tested for optimisation.

Creep Mutation

The creep mutation operator was introduced by David in his GA handbook [16]. This function enables the selected gene to be incremented or decremented by the creep value (C). The selected gene X_i is mutated using the following formula:

$$X_i = min(iMax, max(iMin, X_1 + rs(iMax - iMin))) \tag{4.6}$$

where $iMin$ and $iMax$ are the lower and upper bounds of the gene respectively. r is a random number between [-1, 1] and s is the creep factor which is a user defined number. There is an equal chance that the gene's value will be increased or decreased, and the value is adjusted to be the boundary value if it falls outside of the valid range for that gene.

4.4.5 Our Genetic Algorithm

Throughout our prototype snake and dolphin experiments (described in the next chapter), we used the same genetic algorithm to test the validity across different levels of animation system complexity.

Once a generation has been successfully rated a python script described in Algorithm 3 automatically produces the next generation to be rated. When a generation has been sufficiently rated, a script automatically produces the next generation. When creating a new generation, the current parameter files with their respective ratings are time-stamped and archived on the server (Line 2). We decided that 100 ratings would be enough for each generation, as it gives a reasonable chance that most snakes variants have been rated at least once per generation. Once the snakes have been placed in order of rating (Lines 3 to 5), the top 25% are automatically selected for the new generation as the strongest candidates (Line 6). All snake ratings are then given a fitness value and using a roulette wheel selection method (Lines 8 to 9), two snakes are randomly chosen to be one of two parents which are used to generate two children for the next generation. Using a single-point cross-over method (Line 10), a random place within the parameters of the parents is selected as the crossover point. The children of these two snakes 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. After the new generation file is complete, it replaces the previous parameter file on the server (Lines 13 to 14). This process is automatic, seamless, and the end user will not notice any difference when rating. Users can start off rating one generation and finish rating a different generation.

4.4.6 Human-in-the-Loop Algorithm Pseudo-Code

4.5 Server Side

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

Algorithm 3: Pseudo-code for our human-in-the-loop optimisation algorithm.

```
if at least 100 creatures have been rated then
    download parameter and saved ratings file for current generation
    merge any duplicate creature
    calculate the average score for each creature
    sort the the snakes into descending score order according to average score
    copy top 25% rated creatures into a new generation
    for remaining 75% of new generation do
        use roulette wheel selection to find parent1
        use roulette wheel selection to find parent2
        perform single-point crossover with parent1 and parent2 to produce
          child1 and child2
        perform mutation chance for child1 and child2
        add child1 and child2 to the new generation
    create new parameter file from the new generation
    upload new parameter file to server
```

at the same time. When a creature is rated, the time-stamped parameters and associated rating are recorded. Snakes 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 creature 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.

4.6 Results

As previously mentioned, we asked participants to rate the snakes based on the description "purple snake with long tail".

To judge how successful the system was, we only need to look at the trends in four of the parameters: TrialTime, and the RGB components of TrailTime. These four parameters will be analysed in greater detail, whereas the other three parameters (TopSpeed, TurnSpeed & TargetMove), whilst still changeable, do not directly impact on the generated specification and will only have an overview of their results. Looking at the aforementioned four parameters, the model score is represented by the outer bounds of the parameters *model*:

$$TrailTime = 5, Red = 1, Green = 0, Blue = 1$$

Therefore, the closer a snake's parameters match to this the longer and more purple its tail will be therefore being closer to the "perfect" creature.

To quantify the results, we can observe the trends of these parameters in a number of ways over the generations. Firstly, we can see how the different parameters change individually over the generations.

4.6.1 Trail Time

Trail Time is the parameter which controls the length of the creature's tail. This works by increasing the duration of the $trailRenderer$ attached to the sphere game object. The range for this parameter was from 1 - 5 seconds. As previous stated we expected the results for the trail time to increase across the generations. Figure 4.12 shows that this has occurred in our experiment. From generation 0 to 9 the length of the snake's tail increases from just about 3.1s to roughly 4.4s. There was a short plateau during generations 3 and 4, which could be down to participants focusing their attention on the colour of the snake's tail and not its length, therefore allowing for some shorter tailed snakes to receive higher ratings than normal.

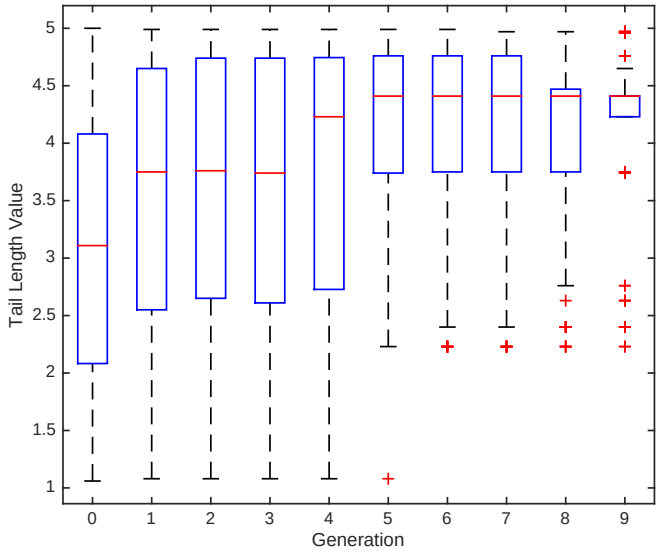
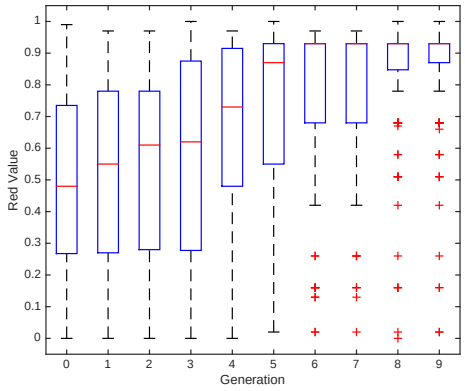


Figure 4.12: Tail Length ($trailTime$) trends across the generations

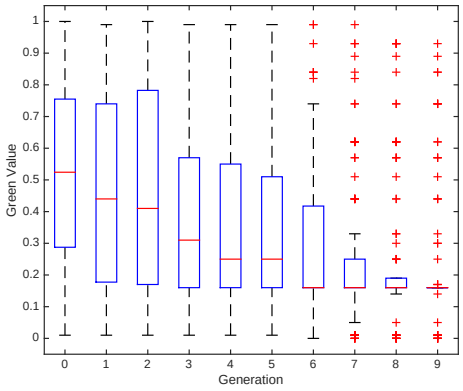
4.6.2 RGB

We can analyse the RGB elements of the snake together as these are all linked. They control the colour of the snake's tail and all work on a scale from 0 - 1. We expected

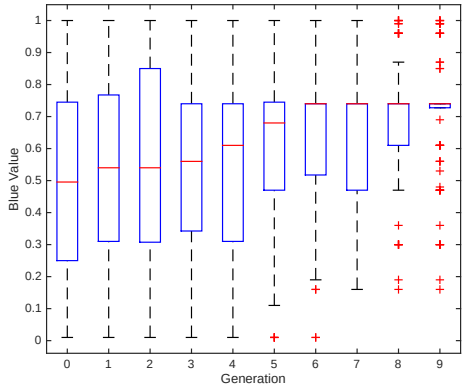
both the Red and Blue elements to increase in value while the Green element decreases as this would produce a more purple appearance. As Figure 4.13 shows that occured. The Red element increases from approximately 0.5 to over 0.9. Similarly, with the Blue element also increasing from 0.5 to over 0.7. The Green element however decreases from over 0.5 to under 0.2. Overall, this will produce a tail which appears to be more purple over time.



(a) Red trends across the generations



(b) Green trends across the generations



(c) Blue trends across the generations

Figure 4.13: RGB trends across the generations

4.6.3 Other Parameters

Throughout the generations the $TopSpeed$ took a steady decline from 12.10 to 9.38 where as the $TurnSpeed$ and $TargetMove$ both stayed fairly stable throughout (4.97 to 5.19 and 2.04 to 1.74 respectively). So even though $TopSpeed$ was not vital to the ratings description, it did slowly decrease over subsequent generations.

4.6.4 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 if 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, the data needs to "pass" six assumptions (as described below) that are required to give a valid result [60].

Assumption 1: Your dependent variable should be measured at the continuous level (i.e., they are interval or ratio variables).

Assumption 2: Your two independent variables should each consist of two or more categorical, independent groups.

Assumption 3: You should have independence of observations, which means that there is no relationship between the observations in each group or between the groups themselves.

Assumption 4: There should be no significant outliers. Outliers are data points within your data that do not follow the usual pattern.

Assumption 5: Your dependent variable should be approximately normally distributed for each combination of the groups of the two independent variables.

Assumption 6: There needs to be homogeneity of variances for each combination of the groups of the two independent variables.

The results of a two-way ANOVA with replication are displayed in a table format as shown in Table 4.4. The sum of squares (SS_T) is the total variation of the data can be calculated as follows (Equation 4.7).

$$SS_T = \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{4.7}$$

Where: s is the standard deviation, y_i is the ith observation, n is the number of observations and $\bar{y}$ is the mean of the n observations.

From this we are then able to calculate the mean square (MS_T) of the data set through the following formula shown in Equation 4.8. The denominator if this relationship is the number of degrees of freedom associated with the sample variance. Therefore, the number of degrees of freedom associated with $dof(SS_T)$ is $(n - 1)$. It is also known as the mean square as it involves dividing the sum of squares by the respective dof.

$$MS_T = \frac{SS_T}{dof(SS_T)} = \frac{SS_T}{n - 1} \tag{4.8}$$

The F-value in ANOVA is a tool which helps to determine whether the variance between the means of two populations is significantly different. This is the also used to determine the P-value; the P-value is the probability of getting a result at least as extreme as the one that was actually observed given that the null hypothesis is true. The F-value is calculated by using the formula shown in Equation 4.9 and the P-Value as seen in Equation 4.10, where $P(F \leq f_i)$ is the cumulative distribution function for the F distribution.

$$F(n) = \frac{MS(n)}{MS(error)} \tag{4.9}$$

$$p = 1 - P(F \leq f_i) \tag{4.10}$$

Through the use of two-way ANOVA, we can determine the statistical significance of the first and final generations parameter spaces. This test requires two independent variables (Generation 0 & Generation 9) and multiple dependant variables (the 7 Parameters). As shown in Table 4.4, the P-Value of our sample is 0.003 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.

Table 4.4: 2-Way ANOVA with Replication Test: Comparing Generation 0 and Final Generation

	Sum of Squares	*df*	*Mean Square*	*F*	*P-Value*	*F Crit*
Sample	17.118	1	17.118	9.114	0.003	3.848
Columns	17526.981	6	2921.163	1555.326	0.000	2.105
Interaction	449.195	6	74.866	39.861	0.000	2.105
Within	2734.613	1456	1.878			
Total	20727.906	1469				

4.6.5 Euclidean and Manhattan Distances

We can also compare the parameter space between the two mediums by calculating the Euclidean and Manhattan distances. By calculating the Euclidean (Equation 4.11) and Manhattan (Equation 4.12) distances using these formula's, we can compare the two mediums average snakes through the generations therefore we can observe how close the parameter spaces are to each other. The Manhattan distance shows the distance between two points on a grid based horizontal and vertical path, while the Euclidean distance shows the distance between the two points in a diagonal or in a "*as the crow flies*" manner. Before we could calculate these all the data had to be normalised using Equation 4.13. We compared the data from each generation against the model answer for the snake like creature.

$$\sqrt{dist(x,y) = \sum_{i=1}^{n}(x_i - y_i)^2} \qquad (4.11) \qquad dist(x,y) = \sum_{i=1}^{n}|x_i - y_i| \qquad (4.12)$$

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)} \qquad (4.13)$$

As shown in Figure 6.6, as the study progressed both the Euclidean and Manhattan distances gradually got closer to 0. This indicates that the parameter space for the snake was trending towards the model answer for this study, and therefore verifying that our system is capable of tuning parameters for a procedurally animated creature.

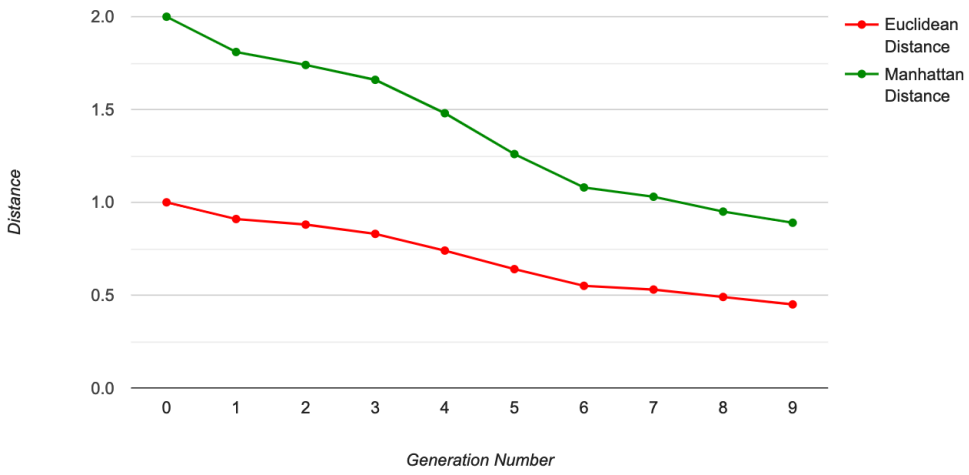


Figure 4.14: Euclidean and Manhattan distances for each generation.

4.7 Chapter Summary and Conclusion

In this Chapter we discussed a prototype study designed to test the robustness of a crowd sourced parameter optimising system using a genetic algorithm. The first step was the creation a suitable creature which could be optimised in a short time frame. A snake-like creature was chosen due to being able to easily optimise and measure the physical appearance of it. The main characteristics the study was interested in was the length and colour of the snake's tail ($TrailTime$ & RGB). From this a genetic algorithm was designed to use crowd sourcing as a means of parameter optimisation. The results show that after a few generations, anonymous users could successfully trend the average snake towards a 'perfect' outcome, thereby proving the validity of our prototype system. Even with a limited number of generations and the potential for false data to be added to the system, the snake has trended towards the desired 'perfect' outcome. It would have been interesting to run this experiment for a second time with an altered GA and a larger data source to thoroughly test our methodology, but we felt that the results from this study were substantial enough for this exact system to be tested on a more complex animation system.

However, the process of tuning a procedurally animated systems is still very subjective and difficult to automate, and there is no 'perfect' behavioural profile. Our aim is to develop techniques that allow users to interactively rate a creature's behaviour, appearance, etc. and therefore guide the development of a complex procedural animation system. By altering what aspect, the user is rating, the system can produce vastly

different results. The system could also allow for the simultaneous development of multiple complex systems. As you can see in Figure 4.15, (where Generation 1 is represented by the red headed snake, Generation 5 the orange headed snake and Generation 10 by the green snake), this prototype experiment has proven that the genetic algorithm combined with the ratings system study has been successful and we can now apply it to more complex systems.

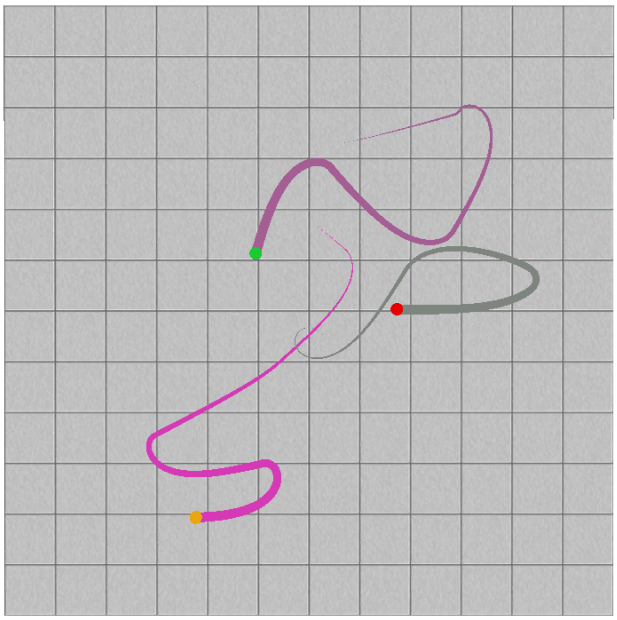


Figure 4.15: The appearance of the average snakes for Generations 1 (red head), 5 (orange head) & 10 (green head).

The next step is to implement the same system within a much more complex procedurally animated system which we have been developing in tandem. This system consists of 37 adjustable parameters and is capable of producing life-like and diverse dolphin motion and behaviour. The complexity and connectedness of the parameter space of this system vastly increases the variations of behaviour for each creature compared to our prototype. This larger parameter space will allow for us to thoroughly test if it is possible to crowd-source different types of animation behaviours by asking the user to create a creature based on adjectives such as ‘playful’, ‘angry’, ‘calm’ or ‘friendly’. As these creature swim through a 3D virtual environment, they will be much harder to represent in image form, therefore using the same method as described above should allow us to ascertain the robustness of our system.

A further goal of this study is to test whether a user’s perception of the procedural creature is altered depending on whether it is viewed on a monitor (2D) or within

Virtual Reality (VR). Using our crowd-sourcing method, we can see how the dolphins' parameter space alters through the generations giving developers a better idea as how to optimise a creature's parameters in the future. This experiment could potentially reveal if certain motion characteristics or even single parameters are more or less important across 2D and VR and provide insights into how best to develop and tune procedural animation systems for different mediums.

Chapter 5

Adapting a Dolphin Animation System for Crowd Sourced Procedural Animation Optimisation

Chapter 3 described a prototype study to establish if humans perceive a non-playable character (NPC) differently depending which medium they are viewed in. This was achieved through two games (first person shooter and a racing game) running two versions, one in VR and one on a 2D monitor. Users ran all four versions of the experiment in a random order and feedback was gathered to establish whether they perceived their opponent to be human or AI. The experiment showed that there was a clear split in the player's perception of their in-game opponent, however these results were inverted. For the two games with the racing game users typically reported that the opponent was human when played in VR where as the opposite was true for the first-person shooter.

The previous chapter describes our prototype experiment in which an online study was conducted to test the robustness of our crowd sourced parameter optimisation system. This was assessed through a study which used a suitably simple snake-like creature which could be optimised in a short time frame towards a desired outcome. We then developed an online genetic algorithm driven application where users could rate snakes for a specific appearance. The results gathered proved the validity of our prototype system, as anonymous users were able to successfully trend the average snake towards a 'perfect' outcome of a 'long tailed purple snake' after a few generations.

This chapter utilises the findings from the previous two chapters to create a study which tests two things - if it is possible to tune a much more complex animation system towards a behavioural goal, and if there is any difference in preference across different viewing platforms. This chapter utilises the methods used in Chapter 4 for the experimental set up and puts the principles from Chapter 3 through more rigorous testing. This chapter also describes a study designed to test whether a user perceives a dolphin differently on a 2D monitor and VR. We are interested to see if 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.

Parts of the following chapter were primarily published in [37], [38]. With [39] forming the backbone of this chapter.

5.1 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 (Figure 5.1) 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 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 (Figure 5.2). The skeleton contains a backbone chain representing the torso and tail, with root bone at the head, and ancillary mouth and fin bones.

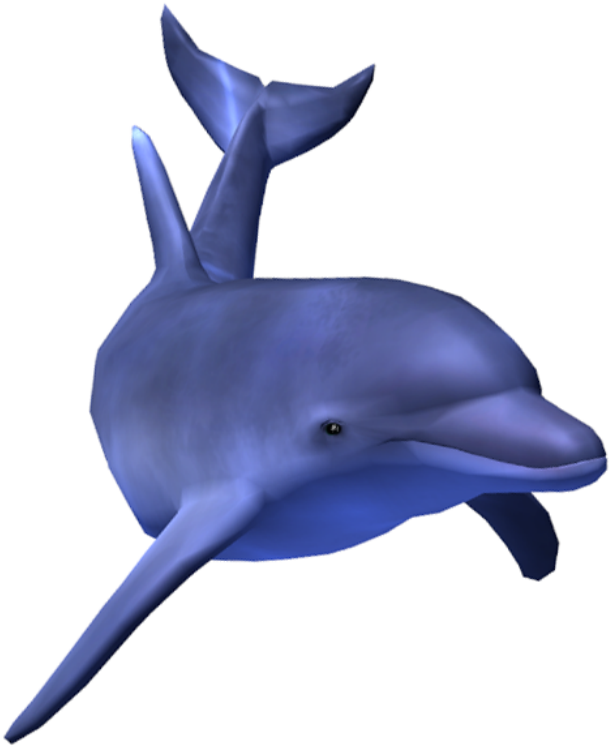


Figure 5.1: Example virtual dolphin used in our study. Each dolphin is controlled by 33 parameters describing its animation and behaviour.

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 are 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 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 [87]. Basic Seek and Arrive behaviours allows 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

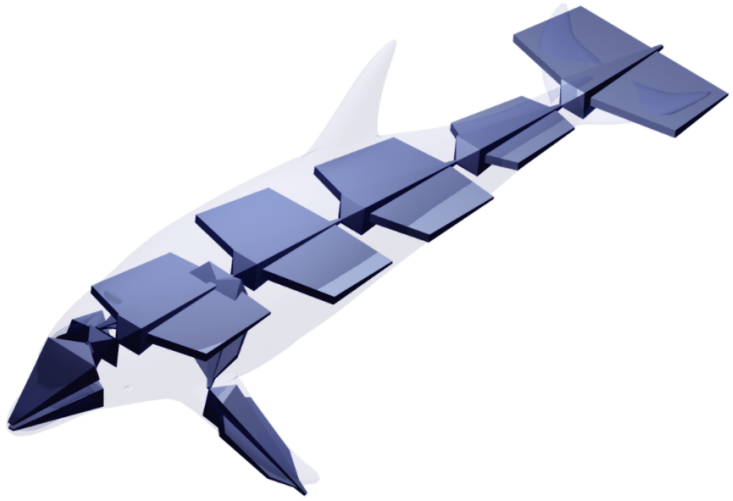


Figure 5.2: Dolphin model consisting of a polygonal mesh and underlying skeletal rig. The skeleton consists of mouth and fin appendages attached to a main backbone chain.

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 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 can be seen in Table 5.1 (for a full detailed list see Appendix B). All aspects of a dolphin's behaviour is 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.

Table 5.1: Parameter list and ranges for the dolphin animation system.

Parameter	*Range*	*Granularity*
Barrel Roll Chance	100 - 1500	1
Barrel Roll Speed Range Max	100 - 600	1
Barrel Roll Speed Range Min	100 - 600	1
Chattering Speed Max	1 - 50	0.01
Chattering Speed Min	1 - 50	0.01
Change Time	1 - 10	0.01
Default Rotate To Target Speed	0.0 - 10.0	0.01
Default Speed	0.0 - 5.0	0.01
Default Turn Speed	0.0 - 5.0	0.01
Fastest Speed	0.0 - 5.0	0.01
Acceleration	0.0 - 10.0	0.01
Speed Decay	0.0 - 1.0	0.01
Speed Change Chance	0 - 500	1
Faithfulness	0 - 1000	1
Friendliness	0 - 1000	1
Near Player Distance	0.0 - 8.0	0.01
Near Player Min Distance	0.0 - 3.0	0.01
Near Player Speed	0.0 - defaultSpeed	0.01
Near Player Repulse Force	0.0 - 1.0	0.01
Mouth Open Chance	0 - 1000	1
Mouth Open Time Range Min	0 - 2	0.01
Mouth Open Time Range Max	0 - 2	0.01
Swim Amplitude Max X	0 - 100	0.01
Swim Amplitude Min X	0 - 100	0.01
Swim Amplitude Max Y	0 - 150	0.01
Swim Amplitude Min Y	0 - 150	0.01
Swim Frequency Max X	0 - 10	0.01
Swim Frequency Min X	0 - 10	0.01
Swim Frequency Max Y	0 - 10	0.01
Swim Frequency Min Y	0 - 10	0.01

Continued on next page

Tail Max Amplitude	0 - 120	0.01
Tail Rest Amplitude	0 - 120	0.01
Tail Amplitude Decay	0 - 50	0.01

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 inter-dependent. For example, the three rotational components blended on top of the global motion can combine to produce a variety of twists and turns. The 33 parameters were chosen after some manual alterations testing how they affect animation system and discussions with Dr. Llyr ap Cenydd (the animation systems developer). It was determined that the chosen parameters covered a large enough proportion of the dolphin's characteristics that the experiment could be run. We felt that has more or all of the parameters been included in the experiment then the number of generations needed to see drastic improvements is the dolphin's realism would be increased to an unattainable level. Once the parameters were chosen then the range and granularity also needed to be established. Some of the outer bounds for a parameter were restricted by the animation's setup, such as Faithfulness and Friendliness which both work on a range of 0 – 1000, these are used in a percentage chance approach. Whereas, Acceleration (a range of 0.0 – 10.0) could have a higher upper bound but after manual experimentation it was concluded that 10.0 was a high enough mark as if a dolphin accelerated at this rate then it was too difficult to keep track of the creature in both environments. As the results for Acceleration show the outer bounds were large enough for the participants to view a wide range of dolphins with both mediums preferring a max acceleration speed

of between 4 – 5. 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.

5.2 Underwater Environment Setup

The environment for the dolphin simulation consists of open water, with the user floating in the middle of the dolphin's habitat. It is important that the virtual environment makes the user feel like they are underwater, to both increase immersion, and to match the potential perceived realism of the dolphin motion and behaviour.

5.2.1 Water fog effect

A standard exp2 fog effect is added with an appropriate blue colour, representing the user's underwater visibility. The camera's clear colour is also set to this colour value, and the density of the fog is set so that the fog fades to a vanishing point just before the far clip plane, so that the water effect fades entirely to the base fog colour before clipping can be seen. This is important to ensure that the water plane fades completely before the far clip plane, and that dolphins swimming close to the far clip plane do likewise.

5.2.2 Atmospheric Scattering

An atmospheric scattering effect is added which aims to approximate how light behaves as it penetrates the surface and into the deep. This is achieved using a Unity asset called Fog Volume 3 [101]. Fog Volume 3 is an external Unity asset plugin which aims to model a wide range of atmospheric effects, including water sub-surface scattering, using volumetric lighting. The effect is that the colour of the water is brighter at the water surface and darkens with depth, giving a more realistic underwater effect.

5.2.3 Water Surface

The surface of the water is about 10m above the camera and helps immerse the user in the environment. It consists of a flat plane with a 100x100 polygon dimension. During runtime a shader modifies the vertex positions of the plane to produce a subtle wave

effect. Two normal maps are also combined and scrolled (the first on the u axis, the second on the v axis) in the fragment shader, to simulate the finer details on the water surface. The shader also applies specular highlighting and diffuse noise to the final colour of the water surface. The shader is a modified version of a now discontinued Unity asset store shader.

5.2.4 Detritus

A cloud of small particles such as dust motes have long been used in VR to provide a sense of depth to a scene. In the underwater environment a particle system is used to create a cloud of detritus billboard particles which randomly float in the water. A wind modifier is used with a low frequency and amplitude in order to simulate slow moving currents in the water.

5.2.5 Lighting

The scene is lit using three lights. The first is a light blue directional light that acts as the main light source of the scene. This is accompanied by a blue ambient light value which simulates the bouncing ambient light underwater. Finally, a water-specific directional light is used to light the water surface appropriately.

5.2.6 Light Shafts

To break up the environment and give a greater sense of depth, a crude approximation of light shafts has also been added to the scene. These consist of a quad mesh with a light shaft texture, shaded with an additive transparency shader. The light shafts are spawned randomly around the user, with a minimum and maximum distance of 6m and 12m respectively. The origin of the quad is offset so that the light shafts appear to start at the height of the water surface. Light shafts are made to fade in and out after 15-20 seconds by altering the alpha channel, and their positions are randomly updated between cycles. While there are more physically accurate methods of simulating light shafts, such as volumetric lighting, the effect given by a quad with additive shader is adequate for the purposes of the dolphin scene, and the light shafts are far enough away from the player that their lack of volume is not apparent, which is especially important in VR. The light shaft effect can be seen in Figure 5.3.

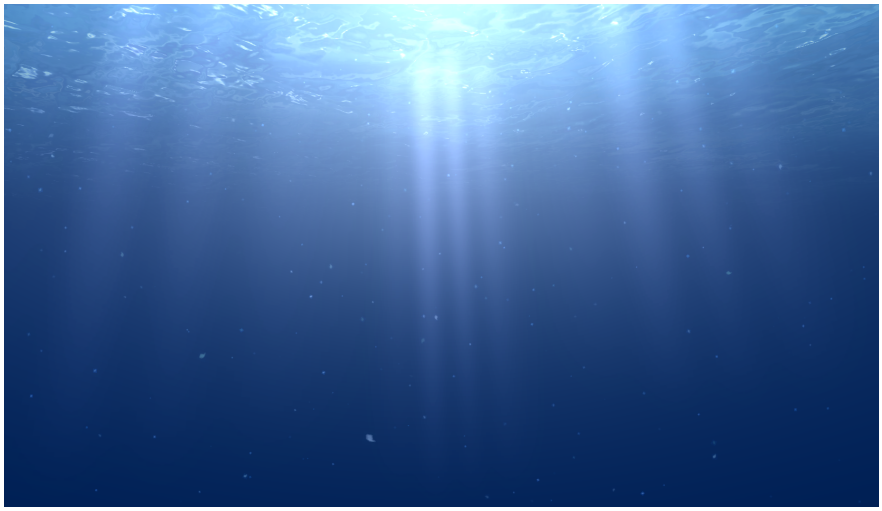


Figure 5.3: Screenshot of underwater environment demonstrating sub-surface scattering, water surface shader and light shaft effects

5.2.7 Caustics

Another effect used to add detail to the scene is the crude simulation of water caustics, which is where light appears to dance on the surface of underwater objects due to refraction at the water surface. This is achieved using a Unity projector, which orthographically projects a tiled and looping animated caustics texture (32 images) downwards onto anything in the scene. The caustic effect is mostly seen on the bodies of the dolphins, which helps break up the surface of the skin and greatly enhances the underwater effect. The caustics are also seen on the surface of the ratings system. The caustics effect can be seen in Figure 5.4.

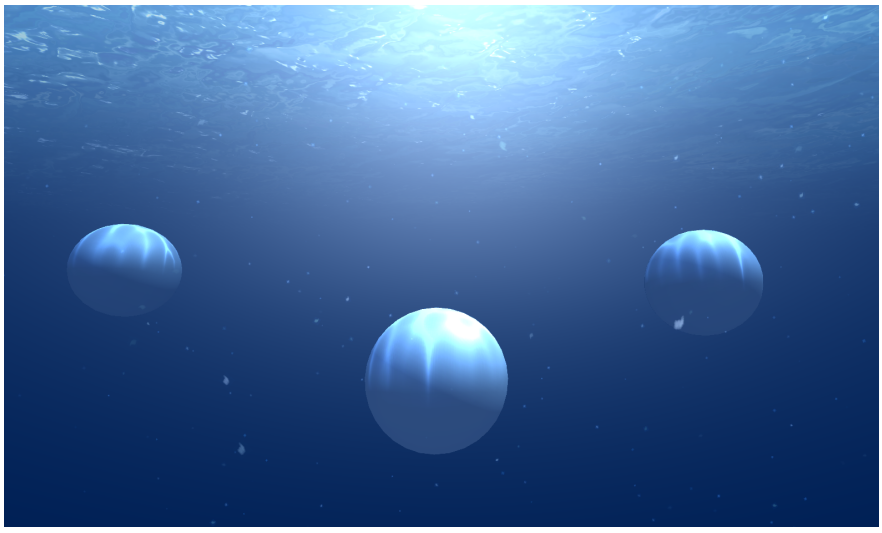


Figure 5.4: Screenshot of debug view of underwater environment. Projected caustic effects can be seen on surface of the spheres

5.2.8 Underwater Sound

An ambient underwater sound plays on a loop during the simulation, in order to give a sense of being submerged in water. Similarly, a scuba breathing audio loop is synched with two bubble particle systems in order to simulate the user breathing underwater.

5.2.9 Completed Underwater Environment

A screenshot of the completed underwater environment can be seen in Figure 5.5. While stylised, the effects combine to produce an environment we deem capable of a degree of realism and immersion that is conducive with optimising a procedural animation system towards "realistic" motion and behaviour.

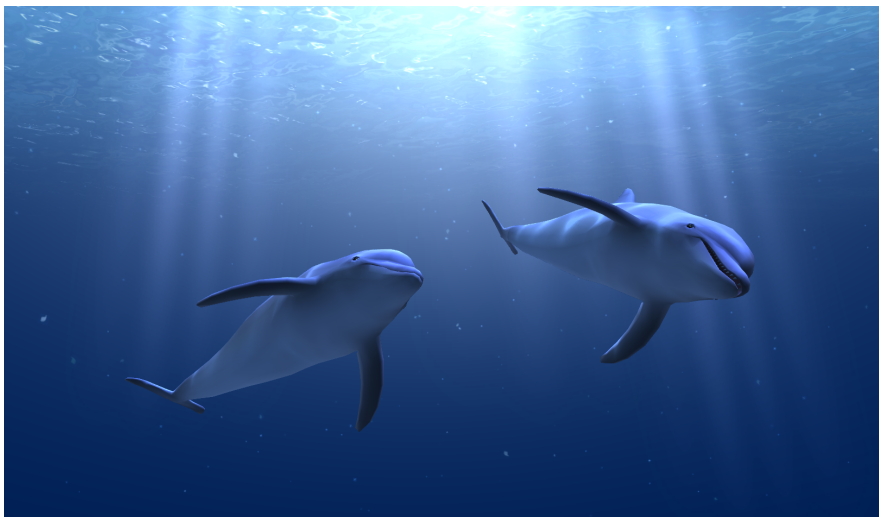


Figure 5.5: Screenshot of dolphin models placed in completed underwater environment

5.3 Experimental Methodology

To explore the research questions a study was set up which asked participants to rate a dolphin on how realistic they appear. There were two versions of the application, one in VR using a consumer Oculus Rift headset and the other on a standard 2D monitor. Both versions ran in the exact same way the only differences being the viewing platform and the procedure for submitting a rating. When using the 2D monitor, participants used a standard mouse to pan around the environment and then clicked on the rating they wished to submit. Within VR, they would look around with their head movements and once they have decided upon a rating, they looked at the corresponding number and clicked the middle button on the Oculus Remote. In both versions, once a rating is selected new dolphins would appear in the environment, and the process continue until

the time limit had expired. Each participant was given 5 minutes in each environment with a short break in between. They played the two versions in a random order as to avoid any bias towards either medium. If at any point a participant started to feel unwell, they could end the experiment. During both versions they wore over ear headphones, this was done for two reasons; to drown out any background noise in the room and to aid immersion in the environment as they could hear dolphin noises, water, breathing, etc.

In the previous chapter we decribed a much simpler procedural animation system (see Chapter 4), consisting of seven parameters **henshall2015towards**, [37]. 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).

The experiment described in section is a continuation of the snake's experiment. With our process verified, we aimed to use similar techniques on the much more complex dolphin procedural animation system. However, as opposed to the snake's 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 if 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 were multiple research questions for this study:

- 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?

5.3.1 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. This was done to keep it inline with the previous study conducted in Chapter 4. As we are potentially limited by the number of users willing to participate in the study, we have restricted each generation to 100 ratings therefore, creating any more creatures to start with would result in too many not being rated per generation.

The parameters have varying levels of granularity. For example, *defaultSpeed* has a range from 0.0–5.0 with a granularity of 0.01 while *barrelRollChance* has a range from 100–1500 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. A list of the parameters, their range & granularity can be seen in Table 5.1 and for a more extensive breakdown please see Appendix B.

5.3.2 Rating System

Upon starting the application, users find themselves underwater with three dolphins (see Figure 5.6). Each trio of dolphins are 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,

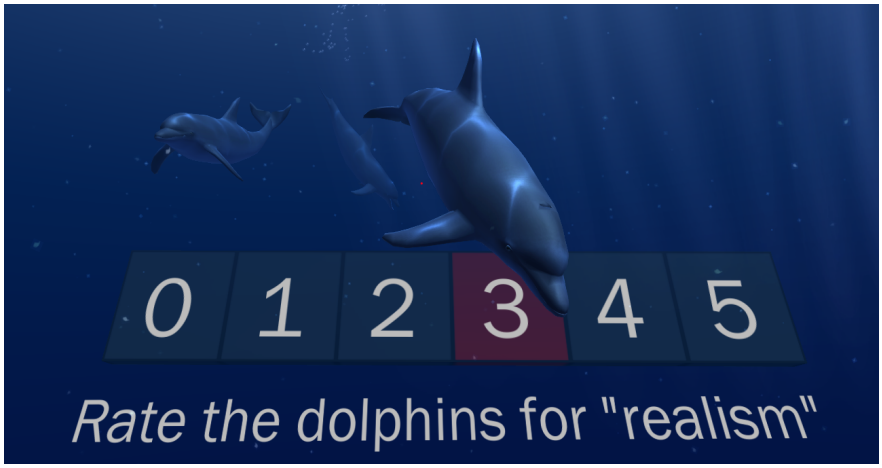


Figure 5.6: A screenshot of our application showing the virtual environment, dolphins and embedded 0-5 rating system.

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 associated 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.

5.4 Subsequent Generations

For this study, we adopted the same genetic algorithm as our optimisation method between generations (Algorithm 3). When a generation of dolphins has been sufficiently rated, a script automatically produces the next generation. 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. Once the dolphins have been placed in order of rating (Lines 3 to 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 to 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.

Using a single-point cross-over 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 to 14). This process is automatic, seamless, and the end user will not notice any difference when rating. Users can start off rating one generation and finish rating a different generation.

5.5 Chapter Summary and Conclusion

In this Chapter we have discussed a study designed to test the system against a much more complex procedurally animated system which was developed in tandem. The system consisted of 33 adjustable parameters and is capable of producing a life-like dolphin in

both its behaviour and motion. The process of tuning a procedural animation system is very subjective and difficult to automate as there is no calculable 'perfect' behavioural profile. One of our aims is to develop techniques that allow users to intuitively guide the development of a complex procedural animation system towards an ideal controller. As mentioned in the 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.

We were able to gather 26 participants in total all of which ran both (2D monitor and VR) versions of the experiment in a random order. In the next chapter we discuss the study itself and what insights can be drawn from the results.

Chapter 6

An Evaluation of Crowd Sourced Procedural Animation Optimisation for the Dolphin Animation System

Chapter 3 described a prototype study designed to assess the influence of VR on the perception of an AI character in a game. This was achieved through the creation of two games (racing & first-person shooter). The two games were playing in both VR and on a 2D monitor and the participant had to determine whether the opponent was human or an NPC. Our results showed that in the racing game participants typically thought the opponent in VR was a human whereas the opposite was true in the shooter. This indicates that there is a link between how a game is played and the players perception of the world.

In Chapter 4 we created a study designed to assess the robustness of an online crowd sourced parameter optimisation system. Users were tasked with rating a snake-like creature on how long and purple its tail appeared. Through the crowd sourcing and a genetic algorithm, the system was able to tune the creatures towards the intended goal. The results gathered from this proved the validity of the system and this was tested further in Chapter 5.

The previous Chapter described the set-up procedure of our final study. By bringing together the findings from both Chapters 3 and 4, a much more complex animation system was extended to test the system against. Instead of a user tuning a creature towards a physical attribute however, we asked users to rate the dolphins based on their motion and behaviours.

6.1 Experimental Methodology

In this chapter we evaluate the results gathered from the study and the findings that can be drawn from these. This study was run offline as it required participants to use a VR headset. However, in future it would be perfectly possible for many anonymous users to optimise a population of VR dolphins remotely, as the rating and evolutionary system is server-based.

For this study we were attempting to tune the parameter space of a procedurally animated creature through a genetic algorithm and crowd sourcing. The dolphin model itself is a kinematic animation system with 33 adjustable parameters. There were 100 unique dolphins at the start of the study all of which had been randomly generated with each individual parameter being within an upper and lower bound. The participants were asked to rate the dolphins for how realistic they perceived them to be. They are looking at the dolphins for realism in a gaming or simulated environment and not for biological accuracy. They used two versions of the same experiment for the same length of time and had to rate as many dolphins as they wished on how realistic they appeared to be. The two version of the experiment were identical apart from the viewing platform, one was on a standard 2D monitor and the other in VR using a consumer Oculus Rift. In the 2D version the user used the mouse to navigate the scene and save the ratings. In VR an Oculus Remote was used as a clicker to save ratings and their head movements panned around the scene. This study was conducted to assess if we perceive procedurally animated creatures differently depending on the viewing platform, and if so, how does this affect the creature's parameter space.

6.2 Participant 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. Ideally, we would have liked to have far more participants than this, but it proved very difficult to encourage participation even

with a sport prize incentive. Due to the nature of the experiment where we require participants to attend in person it is very difficult to gain interest, crowd sourcing tends to be far more successful when used in online studies which can be conducted in the comfort of the participants own home in their own time, but even then it proves to be difficult [48]. However, we were still able to see trends and alterations in our parameter spaces even with the limited number of participants. It is expected that even with far greater results the same trends would still occur, the biggest difference would be that each generation could be far larger before a new generation is populated, this would enable a wider population of creatures to be assessed. Most participants were Computer Science students and 84.6% male. The average age was 24 years old but ranged from 18–42. 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.

6.3 Results

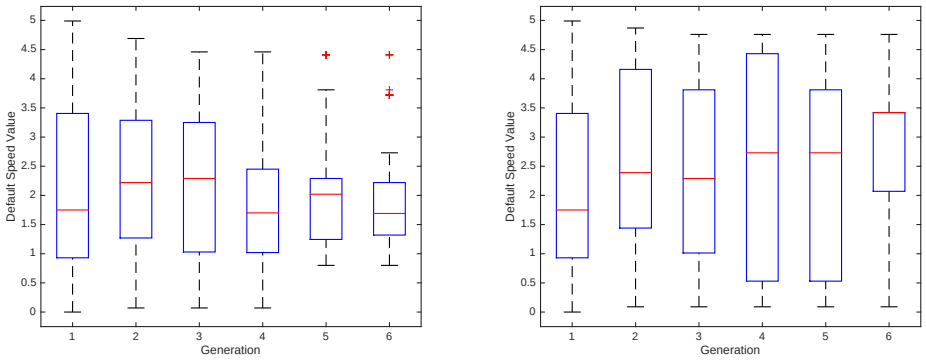
As previously mentioned, we asked participants to rate the dolphins on how realistic they appeared. In this section, we present analysis of how the 33 parameters change over six generations across both mediums. We also compare differences between the final generations average parameter using Euclidean 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* which controls the base speed of the dolphin, *barrelRollChance* which is the change of a barrel roll triggering each frame, *faithfulness* is the chance that the dolphin will become interested in the player, *friendliness* is the chance that the dolphin will lose interest in the player and *mouthOpenChance* which is the chance every frame that the dolphin opens its mouth. 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 seconds, while a value of 900 would trigger a barrel roll on average every 10 seconds.

6.3.1 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 determined by the *SpeedDecay* parameter. As shown in Figure 6.1, the default speed parameter was in the range of 0–5, roughly representing speed in meters 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 if there is a statistical difference between the final generation parameter space for both Desktop and VR. As shown in Table 6.1, 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.



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

Figure 6.1: The Default Swim Speed parameter over the generations. Lower values represent a slower swim speed and higher is faster.

Table 6.1: Default Speed t-Test: Two-Sample Assuming Unequal Variances

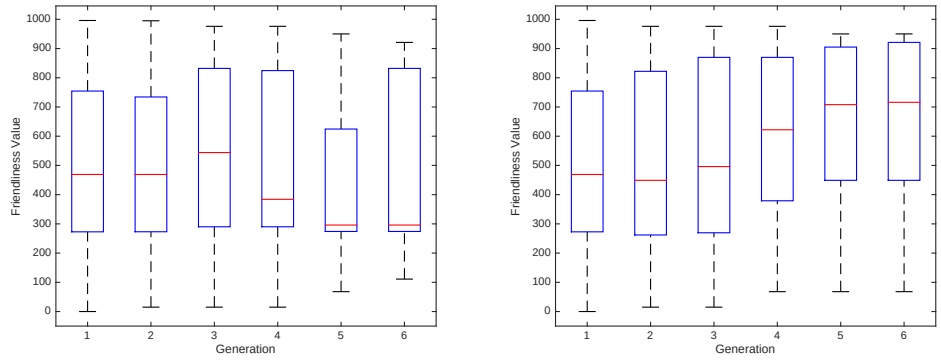
	VR Default Speed Gen 6	*2D Default Speed Gen 6*
Mean	2.723	1.899
Variance	2.085	0.773
Observations	105	
Mean Diff	0	
df	172	
t Stat	4.998	
P(T<=t) two-tail	1.41861E-06	
t Critical two-tail	1.974	

6.3.2 Friendliness and Faithfulness

By default, each dolphin swims toward 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 dolphins 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 indicates a shorter attention span. As shown in Table 6.2, 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 Figure 6.2 show, the faithfulness value in both desktop and VR climb similarly over the generations, with a final average of ≈ 600 and ≈ 550 respectively. These values represent an attention span of around 6.39 seconds. 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 ≈ 300 meaning that the dolphin would approach the user approximately over 3.33 seconds. In VR however, the *friendliness* value steadily increased to ≈ 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 keener 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



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

(c) Desktop **(d)** VR

Figure 6.2: Friendliness *(a)(b)* and faithfulness *(c)(d)* parameters over the generations. Dolphins with higher values for friendliness are less likely to approach the user.

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 6.3) is greater than 0.05. Therefore, the data sets are not significantly different. This echos our results shown in Figure 6.2 where the faithfulness level for both VR and 2D Desktop displays are very similar.

Table 6.2: 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<=t) two-tail	4.92914E-07	
t Critical two-tail	1.971	

Table 6.3: Faithfulness t-Test: Two-Sample Assuming Unequal Variances

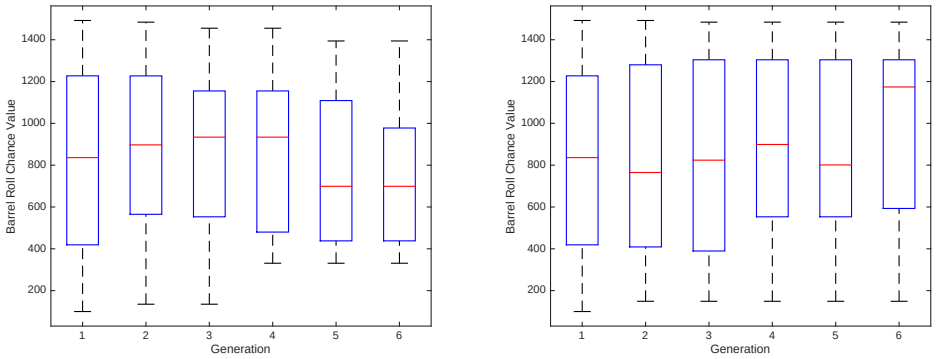
	VR Faithfulness Generation 6	*2D Faithfulness Generation 6*
Mean	551	558
Variance	68206	47218
Observations	105	
Mean Diff	0	
df	201	
t Stat	-0.221	
P(T<=t) two-tail	0.825	
t Critical two-tail	1.972	

6.3.3 Barrel Rolling and Chattering

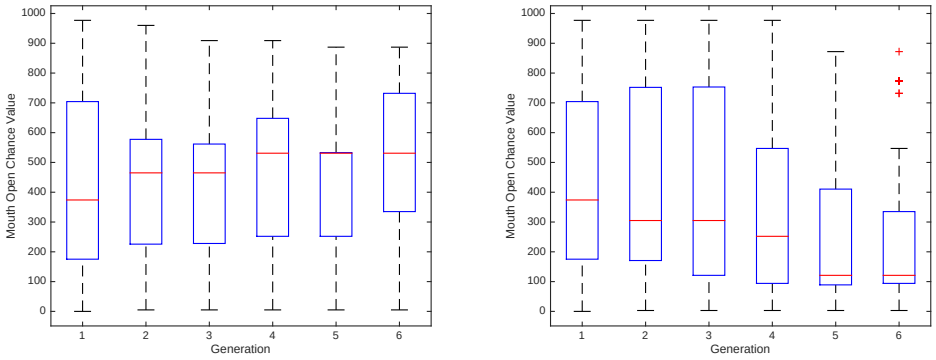
Two other parameters to consider are *barrelRollChance* and *mouthOpenChance*. Both work in the same fashion as *friendliness* and *faithfulness*, where higher values mean behaviours occur less often.

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 seconds. However, the VR dolphins steadily increased the gap between barrel rolls to an average of 13.30 seconds. 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.



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



(c) Desktop **(d)** VR

Figure 6.3: 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.

As with Default Speed and Friendliness, we can reject the null hypothesis for both Barrel Roll Chance (Table 6.4) and Mouth Open Chance (Table 6.5) as they both have a P-value 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.

Table 6.4: 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<=t) two-tail	0.000	
t Critical two-tail	1.972	

Table 6.5: 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<=t) two-tail	5.39388E-11	
t Critical two-tail	1.971	

6.3.4 Other Notable Parameters

In this section we will discuss some other notable parameters, graphs for all 33 parameters across both medums can be found in Appendix D. 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 whilst 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.

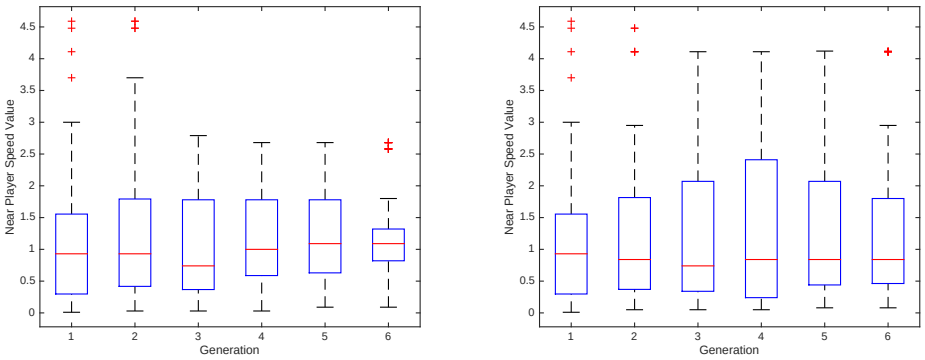
Finally, the *changeTime* parameter, which denotes how often the dolphin changes its goal position, differs between the two simulations — increasing across the generations in VR while decreasing on desktop. This could also be down to aforementioned factors such as a deeper level of immersion in VR, and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

6.3.5 Similar Parameters Across Mediums

Although we have discussed many parameters which showed significant differences between the two viewing platforms, there are some which displayed very little change both across the generations and from 2D to VR.

The parameter $nearPlayerSpeed$ is the dolphin's target speed when near to the player, this is typically much slower than the dolphin's default speed. As shown in Figure 6.4

the average parameter value stays relatively stable across all generations and viewing platforms. The could be down to a few reasons, participants could have been focusing on other aspects of the dolphin so may not have noticed the speed change when the dolphin approached. Secondly, participants may have preferred a slightly slower dolphin when it was close to the camera as it was easier to track with the mouse or headset.



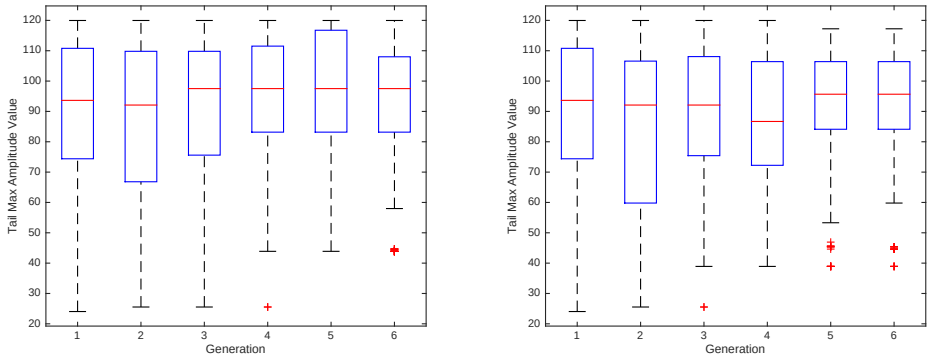
(a) Desktop **(b)** VR

Figure 6.4: The Near Player Distance parameter over the generations. Lower values represent a slower swim speed and higher is faster

We compare both $tailMaxAmplitude$ and $tailRestAmplitude$ as they are both very heavily linked. Although the graphs (Figure 6.5) show that the overall values between the two parameters are very different, the changes across generations and viewing platforms are fairly static. The amplitude in which the tail is kicked is controlled by $tailRestAmplitude$, this controls the amplitude the tail is kicked at while to dolphin is in a rested state. $tailMaxAmplitude$ however, activates when $changeTime$ is triggered causing the dolphin will accelerate to its maximum speed with its tail kicking at this amplitude. These two parameters may have stayed static due to the participant not knowing how much a realistic dolphin kicks its tail, other attributes being much more prominent, or that the initial values were adequate and therefore remained relatively stable.

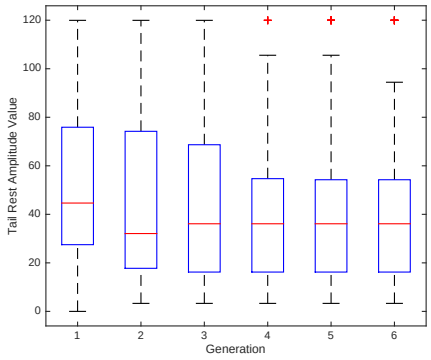
6.3.6 Statistical Analysis using Two-Way ANOVA with Replication

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 if the null hypotheis, that the mean

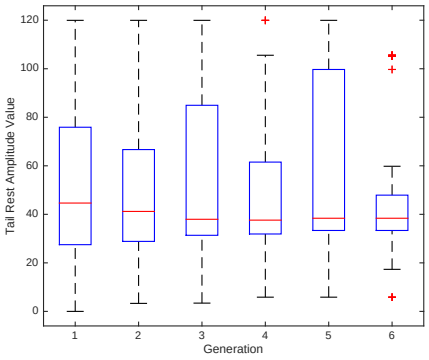


(a) Desktop

(b) VR



(c) Desktop



(d) VR

Figure 6.5: Tail Max Amplitude *(a)(b)* and Tail Rest Amplitude *(c)(d)* parameters over the generations. Lower values represent a smaller tail amplitude.

(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 needs to "pass" the six assumptions (such as continuous dependent variable, no significant outliers, etc.) that are required to give a valid result [60].

As with the previous studies the use of two-way ANOVA can determine the statistical significance of the 2D Monitor and VR parameter spaces. The two independent variables for this study were 2D Generation 6 & VR Generation 6 and the 33 Parameters representing the multiple dependant variables. As shown in Table 6.6, 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.

Table 6.6: 2-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				

6.3.7 Euclidean and Manhattan Distances

As with the previous studies we also compared the parameter spaces of both versions by analysing the Euclidean & Manhattan distances. As shown in Figure 6.6, 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 have been tuned differently. As the study has progressed through the generations, the distances between the 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.

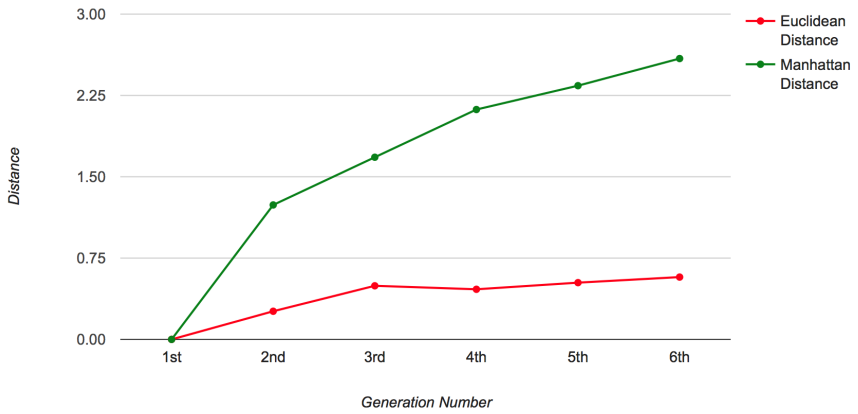


Figure 6.6: Euclidean and Manhattan distances at each generation.

6.3.8 Average Ratings

Figure 6.7 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.

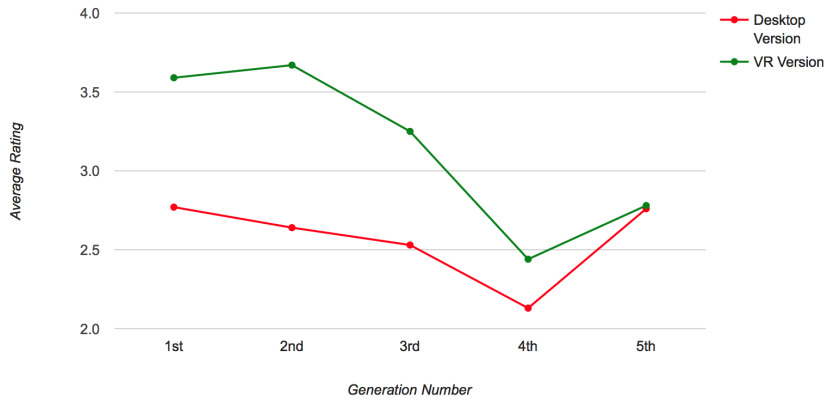


Figure 6.7: Average ratings for each generation.

6.4 Chapter Summary and Conclusion

This Chapter fully analyses the results from the study described in Chapter 5. The study was conducted on two viewing platforms, a consumer Oculus Rift headset and a standard 2D monitor. The 26 participants were asked to rate the dolphins on screen for how realistic they appeared. Each participant ran the two versions in random order as to avoid any bias towards one or the other.

As our results show there were some big differences between parameters when viewing in VR and on a 2D monitor. For example, the optimised rate that dolphins opened their

mouths to chatter was very different across the two mediums, with the value increasing over the generations when viewed in VR and decreasing on the 2D monitor. Similarly, with the dolphin's base speed value, its value in VR decreased where as it increased on a 2D monitor and through a t-Test we showed that there was a significant difference between the two data sets.

We also ran t-Tests on the entire parameter space for the final generation of VR and 2D to assess the differences between them. Similarly, with the individual parameters the parameter spaces had a P-Value less than 0.05 which proved there was a significant difference and furthermore, the f-Value was larger than the F-Crit value emphasising the difference between the parameter spaces. We also assessed the parameter spaces as a whole using Euclidean and Manhattan distances, which again proved significant differences between the two mediums.

A possible impact on experimental results could be the size, quality and resolution of the monitor used during the experiment. As discussed in the experimental methodology (Section 5.3) participants used a standard 2D monitor, a mouse and keyboard and headphones for one part of the study. Had the monitor been small in size or have low resolution this could alter the results significantly as it could prove more difficult for the participants to become immersed within the 2D environment. A consideration for future experiments of this nature need to be the quality of the 2D monitor that is being used. To be able to perform a direct comparison to the VR version the monitor should be the same or very similar resolution to the VR headset being used. This would help to eliminate the bias towards one medium over the other cause through poor image quality. Hou et al. showed that when a game was played on a larger screen the levels of physical and self-presence was increased compared to a small screen. They also found that the larger screen produced a more favourable impression on the game character. They determined that the interaction between human influence and technological factors can determine the sense of presence within a game [46].

Ultimately, the results from this chapter and the previous studies indicate that we do view animation and behavioural attributes differently in VR. This might indicate that when developing virtual characters for simulations or games, the subtleties of animation need careful consideration depending on viewing method.

While we prescribed ‘realism’ as the animation and behavioural metric for this study, the system could also be used to optimise for other descriptions such as ‘playful’, ‘erratic’, ‘fun’, ‘shy’, etc. It would be interesting to see how changing the description for the dolphin would alter the overall parameter space. Furthermore, would we see the same disparity between the 2D monitor and VR? We assume that many of the main parameters would still have similar differences between the two mediums. For example, how close a dolphin comes to the main camera would likely not change depending on the ‘mood’ of the creature, as it was likely that this parameter is set to a level where the user’s personal space is not invaded. On the other hand, some which may alter between moods could be the dolphin’s faithfulness and friendliness. If we tune towards a ‘sad’ or ‘shy’ dolphin then we would assume that the dolphin would not swim up to the user as often as a ‘playful’ one for example. These questions and presumptions could be assessed with future developments of this system.

Chapter 7

Final Summary and Conclusions

7.1 Introduction

In this final chapter we will be concluding all of the previous work, by revisiting the original thesis objectives and the issues raised through the literature review. Following this the main findings and contributions of this research will be broken down and some limitations will be addressed. Finally, we conclude with some final thoughts on the research and suggest possible avenues for future work.

7.2 Reflection of the Thesis Objectives

At the start of this thesis we introduced the following research objectives:

1. Examine whether crowd sourcing can be used to optimise the parameter space of a procedural animation system through an online experiment.

2. Conduct an experiment comparing the optimisation of a parameter space across multiple viewing platforms — 2D Monitor and Virtual Reality.

3. Evaluate the differences in desired animation and behaviour attributes based on viewing platform.

The first objective was primarily addressed within Chapter 4. We were able to successfully create a system where by crowd sourcing could be used to optimise the parameter space of a procedurally animated creature. This was done through the means of an online experiment in which participants were asked to rate a snake on a scale of 0-5 for how long and purple its tail was. After a few generations our results showed

that anonymous users could anneal a creatures parameter space towards the 'perfect' outcome. This was backed up further by the following study described in Chapter 5 with results from Chapter 6. This study was not anonymous but still produced statistically significant results showing that a parameter space could be tuned towards a behavioural metric as apposed to a physical one. This research was submitted [35], [36] with the former claiming the award for best poster.

Our second objective was explored in Chapter 5 where we conducted an experiment which compared the parameter space of a procedurally animated creature across viewing platforms (2D monitor and VR). We created the experiment on the back of the results found in Chapters 3 & 4. By bringing together the findings of both of these studies a much more complex animation system was created and tested against the optimisation system. This study looked into comparing the parameters spaces for the dolphin on both a 2D monitor and within VR using a consumer Oculus Rift. Participants ran two studies in a random order to avoid ant bias and they were asked to rate the dolphins for how realistic they appeared to be.

The final research objectives concerned establishing if and what the differences in desired animation and behavioural attributes were based on the viewing platform. Chapter 6 fully evaluated the results from Chapter 5's study. The result showed that there were statistically significant differences between a procedurally animated creatures parameter space when tuned within different viewing platforms. Parameters were compared between the two mediums through box plots which gives a visual representation of each generation's values for a specific parameter. Through the running of multiple t-Tests we could also prove there was a difference between the two data sets. Although these tests only compared individual parameters, we conducted a 2-Way ANOVA with Replication Test for the whole parameter space to assess how significant the differences were. Ultimately, the results from this study shows that we do perceive a parameter space differently depending on the viewing platform. This means that when developing games or simulations careful consideration into a character's motions and behaviours needs to be taken. This research was primarily published in a journal [39] with other papers coming from early stages of the research [37], [38].

7.3 Main Findings and Contributions

In this section we present the main findings and contributions of this research.

Contribution 1 — Virtual Reality's Influence on the Perception of Artificial Intelligent Characters

Our first contribution comes from the two experiments which were designed to test the user's perception of an AI character across viewing platforms (2D and VR). Chapter 3 described the study and how our perception of an NPC does alter depending on the viewing platform. Secondly, the level of immersion within VR seems to alter our perception of an AI character. We also found a link to how a game is played and how VR then affects a player's perception of that world.

Contribution 2 — A Parameter Optimisation Tool for Crowd Sourced Procedural Animation Systems

A procedurally animated creature's parameter space was altered towards a prescribed goal through our system. This system was developed first as a prototype anonymous online study (see Chapter 4). The system was then extended to be tested against a far more complex animation system on both desktop monitor and VR. Both of these systems used a genetic algorithm along with the online and offline crowd sourcing to successfully anneal an animation system towards a desired goal.

Contribution 3 — Adapting a Commercial Animation System for use with our Parameter Optimisation Tool

To thoroughly test the optimisation process we adapted a procedurally animated dolphin model so that we could interactively tune parameters using our system. By testing the system against a more complex animation system, we demonstrated that our optimisation tool is capable of scaling up to commercial sized animation systems, and that we can use it not only to optimise for morphology (Chapter 4), but also the more subjective motion and behaviour (Chapter 5).

Contribution 4 — Evaluating Parameter Spaces Procedural Animations Creatures

In Chapter 6 we provided definitive proof that there is a statistically significant different

between the parameter space of a creature when they have been tuned for a specific viewing platform whether, it was on a 2D monitor or VR. Moreover, this indicates that the design and creation of a virtual character's animation system should take the viewing platform into consideration from the early stages of implementation.

7.4 Limitations

One limitation of this research is that the method of data collection is restricted to crowd sourcing. As this was the main source for the collection of data, we have not considered implementing neural networks as an alternative method. Although, neural networks have been discussed in the literature review (see Chapter 2) they were not implemented as a part of this research. We ruled this out as an option due to the vast amount of time and resources needed to create a neural network, their black box nature and the difficulty of having a testable metric for something as subjective as a dolphin's motion and behaviour. We deemed offline and online crowd sourcing to be a distributed and powerful form of data collection which minimised these problems. However, in general crowd sourcing at this level can be a very slow process, and getting enough participants proved to be difficult. In future such a system could be available for anyone to use online, however the system would need to be designed to counter malicious or noisy results.

There is potential for the base optimisation process to be faster by altering the genetic algorithm or generation cycles. Both of these could potentially get similar results faster, with far fewer participants. Another method could be to have 'lockable' parameters or parameter groups, which would allow for a significant narrowing of the search space, facilitating the ability to tune the animation system in stages, or delegating the task to different people.

Throughout the snake and dolphin experiments we kept the genetic algorithm the same. This was done to test if the simple GA was capable of tuning a more complex animation system. Work into different versions of the algorithm (changing the selection method for example) could have been valuable in optimising the algorithm to become its most effective and efficient state. Doing this however would have taken a lot of time as each version of the GA would need to be thoroughly tested with a set creature. Ideally, you

would create a base starting point for a creature and then tune it across n generations for each iteration of the GA. This would require a lot of participants to be able to collect enough data or a rudimentary neural network (or other automated system) to be implemented.

We limited our research to creatures with relatively simple animation systems. Future work could explore more complex articulated figures such as quadrupeds and bipeds, and physically simulated articulated figures. If this system were to be evolved for more complex characters, it would likely need to be enhanced with some form of automated system such as a neural network for evolving the base walking motions. Human-in-the-loop methods could still be used for the optimisation of the higher-level behaviour and emotions of the characters, with the sub-conscious ambulatory behaviour being controlled by an automated evolutionary process.

Finally, we could alter the user interface to better suit motion controllers like the Oculus Touch, which would further enhance the user's presence within the simulation. By using the headset to look at a rating and the using the clicker to save the chosen rating users could have found immersion breaking as the clicker would not have seemed like a natural thing to be doing within that virtual environment. The use of hand gestures could have provided greater levels of immersion, and enabled users to rate the dolphins more effectively due to a more natural user interface. This could even be expanded to allow users to select the dolphins they wish to tune, or even the part of the dolphin they want to lock, fine tune or concentrate on (personality, tail animation, speed), which would facilitate faster and more accurate optimisation.

7.5 Future Work

This thesis has concluded by addressing the research objectives and while these have all been met there is still scope for further research into this area. On the back of this research we could now propose a new set of research questions and objectives. Below we discuss potentially directions this research could be taken in.

7.5.1 Framework Modification to Allow Modularisation and Unified Development

A natural development of this research would be to further enhance the framework described to enable developers to use it as a whole or integrate their own data collection and manipulation methods. Allowing a developer to add a custom GA which would better suit their needs could be a valuable addition to the framework. As shown in this thesis, a one method fits all approach to the framework may not be the most effective or efficient way to optimise characters for different situations. For example, whilst the current framework works for the optimisation of a 'simple' animation system with only 33 adjustable parameters. It may not be as usable for a creature with 100+ parameters as this could require a more advanced crossover method in the genetic algorithm or a more suitable form of data collection such as a neural network (see Section 7.5.2). By allowing developers to alter the framework to suit their needs this will further open up the possible applications of this system.

7.5.2 Neural Networks as a Means for Data Collection

As previously discussed, we used crowd sourcing as the sole means of data collection. Whilst this proved to be a valid method for this research it proved difficult to attain enough willing participants to take the study. A way to overcome this would be to develop a neural network system which utilised massive amounts of data while keeping its own data sets much smaller. While this could be a superior method for data collection, careful consideration into the exact methodology used for this would need to be taken. There are many different ways to implement a neural network and each of them have their own benefits and intricacies.

7.5.3 Optimising Creatures Towards Different Behaviours

While our research only used the realism of a creature as its animation and behavioural metric, it could be optimised for other descriptions simultaneously. It would be interesting to see how a parameter space differs between two dolphins one of which was tuned to be 'playful' and the other 'shy'. While we assume that many of the smaller less significant parameters would stay the same some of the larger ones such as $friendliness$ or $faithfulness$ could have big differences depending on the portrayed emotion. This

could also potentially allow us to tune a dolphin to have multiple emotions which can be blended between. For instance, a dolphin could appear to be very happy and playful for the first part of a simulation but if more users or creatures were to appear within the virtual environment it could switch to a more shy or vice-versa. Similary for other underwater animals, this type of system could be used to tune behaviours like predation, schooling and socialising.

7.5.4 More Complex Animation Systems

The research presented concentrates on relatively simple animation systems (the snake-like creature and dolphin) but further work into this area could test the system against a far more complex creature. Would our system be capable of tuning the parameter space of a quad or biped creature? Initial development of a suitable creature would be required with considerations into which of its parameters should be open for tuning. If we are able to amend the system to fit a more complex animation system this could open it up for use with any creature whether it is hypothetically real or mythical, as the system relies on the user's perception only. All that is required is user interpretation and perception of the resultant behaviour.

7.5.5 Natural Parameter Manipulation using Motion Controls

Our system allows for multiple users to tweak a parameter system in parallel, either locally or over the internet. However, while crowd-sourcing like this is powerful, one of the key weaknesses is that it can require a large number of users in order to achieve desired results. We are currently exploring techniques for speeding up this process by modifying the behaviour of our underling evolutionary algorithms, and the granularity of the rating system. Our overall aim is to allow a single user or small group of developers to enter VR and tune a virtual creature's behaviour in a single session.

Key to achieving this goal is to consider ways of speeding up the process of intelligently searching a potentially vast parameter space. One approach we are currently researching is the ability to lock individual or sets of parameters, in order to either temporarily concentrate on a parameter sub-space or to mark certain parameters as optimal. This could also allow users to split the optimisation task into a series of smaller parameter spaces.

The Oculus Touch motion controllers facilitate ‘hand presence’, allowing users to see their hands in VR and perform gestures like pointing, waving and thumb motions. In our initial system as described, users simply selected ratings using arrow keys or a remote control and had no other control over the optimisation process. With our VR system, we are also extending our application to support Oculus Touch, allowing users to interact using context sensitive menus and natural hand gestures.

Our current prototype allows users to select a dolphin using a pointing gesture (see Figure 7.1), which causes them to swim up to the camera for closer inspection. With a dolphin selected, users can then bring up an interactive menu and isolate parameters or tweak parameters using physical sliders. We are also exploring techniques of using other natural gestures like waving to dismiss and a ‘thumbs up’.

Figure 7.1: Screenshot from the application showing two dolphins swimming near the user with Oculus touch hands.

References

[1] T. Back, 'The Interaction of Mutation Rate, Selection, and Self-Adaptation Within a Genetic Algorithm', in *Proc. 2nd Conference of Parallel Problem Solving from Nature, 1992*, Elsevier Science Publishers, 1992 (p. 69).

[2] J. E. Baker, 'Adaptive Selection Methods for Genetic Algorithms', in *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Hillsdale, New Jersey, 1985, pp. 101–111 (p. 64).

[3] 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 (p. 16).

[4] M. Billinghurst, H. Kato and I. Poupyrev, 'The Magicbook-Moving Seamlessly Between Reality and Virtuality', *IEEE Computer Graphics and Applications*, vol. 21, no. 3, pp. 6–8, 2001 (p. 24).

[5] F. Biocca and B. Delaney, 'Immersive Virtual Reality Technology', *Communication in the Age of Virtual Reality*, vol. 15, p. 32, 1995 (p. 23).

[6] J. Blascovich and J. Bailenson, *Infinite Reality: Avatars, Eternal Life, New Worlds, and the Dawn of the Virtual Revolution*. William Morrow & Co, 2011 (p. 22).

[7] W. R. Boot, D. P. Blakely and D. J. Simons, 'Do action video games improve perception and cognition?', *Frontiers in psychology*, vol. 2, 2011 (p. 61).

[8] A. Bruderlin and L. Williams, 'Motion signal processing', in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, 1995, pp. 97–104 (p. 15).

[9] T. Burrell, A. Montazeri, S. Monk and C. J. Taylor, 'Feedback Control-—Based Inverse Kinematics Solvers for a Nuclear Decommissioning Robot', *IFAC-PapersOnLine*, vol. 49, no. 21, pp. 177–184, 2016 (p. 16).

[10] S. R. Buss, 'Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares Methods', *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004 (p. 16).

[11] Q. Chao, J. Shen and X. Jin, 'Video-Based Personalized Traffic Learning', *Graphical Models*, vol. 75, no. 6, pp. 305–317, 2013 (p. 19).

[12] K.-J. Choi and H.-S. Ko, 'Online Motion Retargetting', *The Journal of Visualization and Computer Animation*, vol. 11, no. 5, pp. 223–235, 2000 (p. 16).

[13] A. J. Clark and J. M. Moore, 'A Web-Based Simulation Viewer for Sharing Evolutionary Robotics Results', in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ACM, 2018, pp. 1357–1362 (p. 20).

[14] S. Coros, P. Beaudoin and M. Van de Panne, 'Generalized Biped Walking Control', *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, p. 130, 2010 (p. 18).

[15] J. J. Cummings and J. N. Bailenson, 'How Immersive is Enough? A Meta-Analysis of the Effect of Immersive Technology on User Presence', *Media Psychology*, vol. 19, no. 2, pp. 272–309, 2016 (p. 24).

[16] L. Davis, 'Handbook of genetic algorithms', 1991 (p. 71).

[17] S. Davis, K. Nesbitt and E. Nalivaiko, 'A Systematic Review of Cybersickness', in *Proceedings of the 2014 Conference on Interactive Entertainment*, ACM, 2014, pp. 1–9 (p. 45).

[18] ——, 'Comparing the Onset of Cybersickness Using the Oculus Rift and Two Virtual Roller Coasters', in *Proceedings of the 11th Australasian Conference on Interactive Entertainment (IE 2015)*, vol. 27, 2015, p. 30 (p. 45).

[19] K. Deb and D. Deb, 'Analysing Mutation Schemes for Real-Parameter Genetic Algorithms', *International Journal of Artificial Intelligence and Soft Computing*, vol. 4, no. 1, pp. 1–28, 2014 (p. 69).

[20] D. H. Eberly, *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*. CRC Press, 2006 (p. 16).

[21] A. C. Fang and N. S. Pollard, 'Efficient Synthesis of Physically Valid Human Motion', in *ACM Transactions on Graphics (TOG)*, ACM, vol. 22, 2003, pp. 417–426 (p. 15).

[22] M. Fêdor, 'Application of Inverse Kinematics for Skeleton Manipulation in Real-Time', in *Proceedings of the 19th spring conference on Computer graphics*, ACM, 2003, pp. 203–212 (p. 16).

[23] T. C. Fogarty, 'Varying the Probability of Mutation in the Genetic Algorithm', in *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers Inc., 1989, pp. 104–109 (p. 69).

[24] T. Geijtenbeek, M. van de Panne and A. F. van der Stappen, 'Flexible Muscle-Based Locomotion for Bipedal Creatures', *ACM Transactions on Graphics*, vol. 32, no. 6, 2013 (p. 19).

[25] M. Gleicher, 'Retargetting Motion to New Characters', in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, 1998, pp. 33–42 (p. 14).

[26] F. Glover, 'Tabu Search—Part I', *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989 (p. 12).

[27] W. L. Goffe, G. D. Ferrier and J. Rogers, 'Global Optimization of Statistical Functions with Simulated Annealing', *Journal of econometrics*, vol. 60, no. 1-2, pp. 65–99, 1994 (p. 13).

[28] D. E. Goldberg, 'Genetic Algorithms in Search, Optimization, and Machine Learning, 1989', *Reading: Addison-Wesley*, 1989 (p. 69).

[29] K. Grochow, S. L. Martin, A. Hertzmann and Z. Popović, 'Style-Based Inverse Kinematics', in *ACM transactions on graphics (TOG)*, ACM, vol. 23, 2004, pp. 522–531 (p. 17).

[30] R. Grzeszczuk and D. Terzopoulos, '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, 1995, pp. 63–70, ISBN: 0-89791-701-4. DOI: 10.1145/218380.218411. [Online]. Available: http://doi.acm.org/10.1145/218380.218411 (p. 17).

[31] C. Harvey, K. Debattista, T. Bashford-Rogers and A. Chalmers, 'Multi-Modal Perception for Selective Rendering', in *Computer Graphics Forum*, Wiley Online Library, vol. 36, 2017, pp. 172–183 (p. 25).

[32] C. J. Headleand, G. Henshall, L. Ap Cenydd and W. J. Teahan, 'Randomised Multiconnected Environment Generator', Bangor University, 2014 (pp. 6, 7, 31).

[33] ——, 'Towards Real-Time Behavioral Evolution in Video Games', in *Artificial Life and Intelligent Agents Symposium*, Springer, 2014, pp. 3–16 (pp. 6, 7, 31).

[34] ——, 'The Influence of Virtual Reality on the Perception of Artificial Intelligent Characters in Games', in *Research and Development in Intelligent Systems XXXII*, Springer, 2015, pp. 345–357 (pp. 6, 7, 31).

[35] G. Henshall, C. Headleand, W. Teahan and L. ap Cenydd, 'Towards Crowd-Sourced Parameter Optimisation for Procedural Animation', Cyberworlds, 2015 (pp. 6, 7, 52, 113, 149).

[36] G. Henshall, W. Teahan and L. ap Cenydd, 'Crowd-Sourced Optimisation of Procedural Animation Systems', Artificial Evolution (EA), 2017 (pp. 6, 7, 52, 113, 150).

[37] ——, 'Crowd-Sourced Procedural Animation Optimisation: Comparing Desktop and VR Behaviour', Cyberworlds, 2017 (pp. 6–8, 83, 92, 113).

[38] ——, 'Towards Real-Time Animation Optimisation in VR', Computer Grahpcis & Visual Computing (CGVC), 2017 (pp. 6–8, 83, 113).

[39] ——, 'Virtual Reality's Effect On Parameter Optimisation for Crowd-Sourced Procedural Animation', The Visual Computer, Springer, 2018 (pp. 6–8, 83, 113).

[40] B. Herbelin, R. Salomon, A. Serino and O. Blanke, '5 Neural Mechanisms of Bodily Self-Consciousness and the Experience of Presence in Virtual Reality', *Human Computer Confluence Transforming Human Experience Through Symbiotic Technologies*, p. 80, 2016 (p. 23).

[41] J. Hesser and R. Männer, 'Towards an Optimal Mutation Probability for Genetic Algorithms', in *International Conference on Parallel Problem Solving from Nature*, Springer, 1990, pp. 23–32 (p. 69).

[42] L. J. Hettinger, K. S. Berbaum, R. S. Kennedy, W. P. Dunlap and M. D. Nolan, 'Vection and Simulator Sickness', *Military Psychology*, vol. 2, no. 3, p. 171, 1990 (p. 45).

[43] J. K. Hodgins and N. S. Pollard, '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., 1997, pp. 153–162 (p. 2).

[44] J. H. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975 (p. 62).

[45] ——, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT press, 1992 (pp. 11, 69).

[46] J. Hou, Y. Nam, W. Peng and K. M. Lee, 'Effects of Screen Size, Viewing Angle, And Players' Immersion Tendencies on Game Experience', *Computers in Human Behavior*, vol. 28, no. 2, pp. 617–623, 2012 (p. 110).

[47] S. Houser, D. Houser, T. Donovan and J. King, *Rockstar games website*, `https://www.rockstargames.com`, Accessed: 17/04/2018 (p. 19).

[48] J. Howe, 'The Rise of Crowdsourcing', *Wired magazine*, vol. 14, no. 6, pp. 1–4, 2006 (p. 99).

[49] W. IJsselsteijn and H. De Ridder, 'Measuring Temporal Variations in Presence', in *Presence in Shared Virtual Environments Workshop, University College, London*, 1998, pp. 10–11 (p. 28).

[50] Indie Bytes, *Vehicle Physics Toolkit*, [Online]: https://www.assetstore.unity3d.com/en/#!/content/14868, Sep. 2014 (p. 31).

[51] S. A. Jackson and M. Csikszentmihalyi, *Flow in Sports*. Human Kinetics, 1999 (p. 24).

[52] C. Jennett, A. L. Cox and P. Cairns, 'Being "in the Game"', 2008 (p. 24).

[53] R. S. Johansen, 'Automated Semi-Procedural Animation for Character Locomotion', *Aarhus Universitet, Institut for Informations Medievidenskab*, 2009 (p. 21).

[54] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, 'Optimization by Simulated Annealing', *science*, vol. 220, no. 4598, pp. 671–680, 1983 (p. 12).

[55] E. Kokkevis, D. Metaxas and N. I. Badler, 'User-Controlled Physics-Based Animation for Articulated Figures', in *Proceedings Computer Animation'96*, IEEE, 1996, pp. 16–26 (p. 17).

[56] L. Kovar, M. Gleicher and F. Pighin, 'Motion Graphs', in *ACM transactions on graphics (TOG)*, ACM, vol. 21, 2002, pp. 473–482 (p. 15).

[57] O. Kramer, *Genetic Algorithm Essentials*. Springer, 2017, vol. 679 (p. 11).

[58] S. H. Kweon, H. J. Kweon, S.-j. Kim, X. Li, X. Liu and H. L. Kweon, '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, 2017, pp. 194–203 (p. 28).

[59] F. D. Lab, *Procedural dungeon generation*, [Online]: http://www.futuredatalab.com/proceduraldungeon/, Oct. 2014 (p. 35).

[60] Laerd Statistics. (visited on 21.10.2017). Dynamic Motion Synthesis (2011), [Online]. Available: `https://statistics.laerd.com/spss-tutorials/two-way-anova-using-spss-statistics.php` (pp. 76, 107).

[61] E. Langer, M. Djikic, M. Pirson, A. Madenci and R. Donohue, 'Believing is Seeing: Using Mindlessness (Mindfully) to Improve Visual Acuity', *Psychological Science*, vol. 21, no. 5, pp. 661–666, 2010 (p. 61).

[62] J. Laszlo, M. van de Panne and E. Fiume, 'Interactive Control for Physically-Based Animation', in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 2000, pp. 201–208 (p. 17).

[63] J. M. Lee, 'Fast K-Nearest Neighbor Searching in Static Objects', *Wireless Personal Communications*, pp. 1–14, 2017 (p. 15).

[64] J. Lee and S. Y. Shin, 'A Hierarchical Approach to Interactive Motion Editing for Human-Like Figures', in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 1999, pp. 39–48 (p. 14).

[65] Y. Lee, K. Wampler, G. Bernstein, J. Popović and Z. Popović, 'Motion Fields for Interactive Character Locomotion', in *ACM Transactions on Graphics (TOG)*, ACM, vol. 29, 2010, p. 138 (p. 15).

[66] C. K. Liu, A. Hertzmann and Z. Popović, 'Learning Physics-Based Motion Style with Nonlinear Inverse Optimization', *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 1071–1081, 2005 (p. 18).

[67] M. Lombard and T. Ditton, 'At the Heart of it All: The Concept of Presence', *Journal of Computer-Mediated Communication*, vol. 3, no. 2, 1997 (p. 22).

[68] MathWorks, *What is Simulated Annealing?*, `https://uk.mathworks.com/help/gads/what-is-simulated-annealing.html`, Accessed: 2019-02-17 (p. 12).

[69] ——, *What is the Genetic Algorithm*, `https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html`, Accessed: 2019-02-17 (p. 10).

[70] W. S. McCulloch and W. Pitts, 'A Logical Calculus of the Ideas Immanent in Nervous Activity', *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943 (p. 13).

[71] 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 . . ., Tech. Rep., 2004 (p. 16).

[72] T. Merritt, K. McGee, T. L. Chuah and C. Ong, 'Choosing Human Team-Mates: Perceived Identity as a Moderator of Player Preference and Enjoyment', in *Proceedings of the 6th International Conference on Foundations of Digital Games*, ACM, 2011, pp. 196–203 (p. 45).

[73] L. Michailidis, E. Balaguer-Ballester and X. He, 'Flow and Immersion in Video Games: The Aftermath of a Conceptual Challenge', *Frontiers in Psychology*, vol. 9, 2018 (p. 24).

[74] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Science & Business Media, 2013 (p. 70).

[75] M. Mori, 'The Uncanny Valley', *Energy*, vol. 7, no. 4, pp. 33–35, 1970 (p. 26).

[76] N. Motion, *Dynamic natural motion (2011)*, `http://www.naturalmotion.com/middleware/euphoria/`, Accessed: (p. 19).

[77] K. Murias, K. Kwok, A. G. Castillejo, I. Liu and G. Iaria, 'The Effects of Video Game use on Performance in a Virtual Navigation Task', *Computers in Human Behavior*, vol. 58, pp. 398–406, 2016 (p. 61).

[78] M. Nakada, H. Chen and D. Terzopoulos, 'Deep Learning of Biomimetic Visual Perception for Virtual Humans', in *Proceedings of the 15th ACM Symposium on Applied Perception*, ACM, 2018, p. 20 (p. 20).

[79] K. L. Nowak and F. Biocca, 'The Effect of the Agency and Anthropomorphism on Users' Sense of Telepresence, Copresence, and Social Presence in Virtual Environments', *Presence: Teleoperators and Virtual Environments*, vol. 12, no. 5, pp. 481–494, 2003 (p. 30).

[80] Oculus, *VR Best Practices Guide*, [Online]: http://static.oculus.com/sdk-downloads/documents/Oculus_Best_Practices_Guide.pdf, Jan. 2015 (p. 45).

[81] ——, *Ocean rift*, `https://www.oculus.com/experiences/go/1249878741704255/`, Accessed: 28/09/2018 (p. 6).

[82] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh and J. Widom, 'Crowdscreen: Algorithms for Filtering Data with Humans', in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ACM, 2012, pp. 361–372 (p. 61).

[83] A. Parameswaran, S. Boyd, H. Garcia-Molina, A. Gupta, N. Polyzotis and J. Widom, 'Optimal Crowd-Powered Rating and Filtering Algorithms', *Proceedings of the VLDB Endowment*, vol. 7, no. 9, pp. 685–696, 2014 (p. 61).

[84] C. Regan, 'An Investigation Into Nausea and Other Side-Effects of Head-Coupled Immersive Virtual Reality', *Virtual Reality*, vol. 1, no. 1, pp. 17–31, 1995 (p. 33).

[85] J. Ren, X. Wang, X. Jin and D. Manocha, 'Simulating Flying Insects Using Dynamics and Data-Driven Noise Modelling to Generate Diverse Collective Behaviors', *PloS one*, vol. 11, no. 5, e0155698, 2016 (p. 20).

[86] C. W. Reynolds, 'Flocks, Herds and Schools: A Distributed Behavioral Model', *ACM SIGGRAPH computer graphics*, vol. 21, no. 4, pp. 25–34, 1987 (p. 2).

[87] ——, 'Steering Behaviors for Autonomous Characters', in *Game developers conference*, vol. 1999, 1999, pp. 763–782 (p. 84).

[88] G. Ridsdale, 'Connectionist Modelling of Skill Dynamics', *The Journal of Visualization and Computer Animation*, vol. 1, no. 2, pp. 66–72, 1990 (p. 18).

[89] M. J. Schuemie, P. Van Der Straaten, M. Krijn and C. A. Van Der Mast, 'Research on Presence in Virtual Reality: A Survey', *CyberPsychology & Behavior*, vol. 4, no. 2, pp. 183–201, 2001 (p. 27).

[90] R. Schwarz, *10 Creepy Examples of the Uncanny Valley*, `https://www.strangerdimensions.com/2013/11/25/10-creepy-examples-uncanny-valley/`, Accessed: 02/2019 (p. 27).

[91] O. Shaer, E. Hornecker *et al.*, 'Tangible User Interfaces: Past, Present, and Future Directions', *Foundations and Trends® in Human–Computer Interaction*, vol. 3, no. 1–2, pp. 4–137, 2010 (p. 25).

[92] K. Sims, 'Evolving Virtual Creatures', in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, 1994, pp. 15–22 (p. 18).

[93] M. Slater and A. Steed, 'A Virtual Presence Counter', *Presence: Teleoperators & Virtual Environments*, vol. 9, no. 5, pp. 413–434, 2000 (p. 28).

[94] M. Slater and M. Usoh, 'Representations Systems, Perceptual Position, and Presence in Immersive Virtual Environments', *Presence: Teleoperators & Virtual Environments*, vol. 2, no. 3, pp. 221–233, 1993 (p. 22).

[95] M. Slater, M. Usoh and A. Steed, 'Depth of Presence in Virtual Environments', *Presence: Teleoperators & Virtual Environments*, vol. 3, no. 2, pp. 130–144, 1994 (p. 26).

[96] M. Slater and S. Wilbur, 'A Framework for Immersive Virtual Environments (FIVE): Speculations on the Role of Presence in Virtual Environments', *Presence: Teleoperators & Virtual Environments*, vol. 6, no. 6, pp. 603–616, 1997 (p. 24).

[97] R. H. So, W. Lo and A. T. Ho, 'Effects of Navigation Speed on Motion Sickness Caused by an Immersive Virtual Environment', *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 43, no. 3, pp. 452–461, 2001 (p. 45).

[98] Steam VR, *Knuckles ev2: What's new*, `https://steamcommunity.com/sharedfiles/filedetails/?id=1411984190`, Accessed: 26/09/2018 (p. 25).

[99] T. A. Stoffregen, E. Faugloire, K. Yoshida, M. B. Flanagan and O. Merhi, 'Motion Sickness and Postural Sway in Console Video Games', *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 50, no. 2, pp. 322–331, 2008 (p. 45).

[100] I. E. Sutherland, 'The Ultimate Display', *Multimedia: From Wagner to virtual reality*, pp. 506–508, 1965 (p. 22).

[101] Unity Asset Store, *Fog volume 3*, `https://assetstore.unity.com/packages/tools/particles-effects/fog-volume-3-81802` (p. 88).

[102] Unity Technologies, *Car Tutorial*, [Online]: https://www.assetstore.unity3d.com/en/#!/content/10, Dec. 2012 (p. 31).

[103] ——, *Unity Documentation Trail Renderer*, `https://docs.unity3d.com/Manual/class-TrailRenderer.html`, Accessed: 2015-11-22 (p. 53).

[104] M. Uysal, 'A Comparison of Heuristic Search Algorithms for Predicting the Effort Component of Software Projects', in *2008 International Conference on Computational Intelligence for Modelling Control & Automation*, IEEE, 2008, pp. 92–97 (p. 14).

[105] G. Villarrubia, J. F. De Paz, P. Chamoso and F. De la Prieta, 'Artificial Neural Networks used in Optimization Problems', *Neurocomputing*, vol. 272, pp. 10–16, 2018 (p. 13).

[106] M. D. Vose, 'Modeling Simple Genetic Algorithms', in *Foundations of genetic algorithms*, vol. 2, Elsevier, 1993, pp. 63–73 (p. 11).

[107] J. M. Wang, D. J. Fleet and A. Hertzmann, 'Optimizing Walking Controllers', *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, p. 168, 2009 (p. 19).

[108] D. R. Westhead, D. E. Clark and C. W. Murray, 'A Comparison of Heuristic Search Algorithms for Molecular Docking', *Journal of Computer-Aided Molecular Design*, vol. 11, no. 3, pp. 209–228, 1997 (p. 13).

[109] D. Whitley, 'A Genetic Algorithm Tutorial', *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994 (p. 11).

[110] L. D. Whitley, A. E. Howe, S. Rana, J.-P. Watson and L. Barbulescu, 'Comparing Heuristic Search Methods and Genetic Algorithms for Warehouse Scheduling', in *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, IEEE, vol. 3, 1998, pp. 2430–2435 (p. 14).

[111] A. Witkin and Z. Popovic, 'Motion Warping', in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, 1995, pp. 105–108 (p. 14).

[112] B. G. Witmer and M. J. Singer, 'Measuring Presence in Virtual Environments: A Presence Questionnaire', *Presence*, vol. 7, no. 3, pp. 225–240, 1998 (p. 27).

[113] J.-c. Wu and Z. Popović, 'Realistic Modeling of Bird Flight Animations', in *ACM SIGGRAPH 2003 Papers*, ser. SIGGRAPH '03, San Diego, California: ACM, 2003, pp. 888–895, ISBN: 1-58113-709-5. DOI: 10.1145/1201775.882360. [Online]. Available: http://doi.acm.org/10.1145/1201775.882360 (p. 17).

[114] K. Yin, S. Coros, P. Beaudoin and M. van de Panne, 'Continuation methods for adapting simulated skills', in *ACM Transactions on Graphics (TOG)*, ACM, vol. 27, 2008, p. 81 (p. 21).

[115] K. Yin, K. Loken and M. Van de Panne, 'Simbicon: Simple Biped Locomotion Control', in *ACM Transactions on Graphics (TOG)*, ACM, vol. 26, 2007, p. 105 (p. 19).

[116] H. Zhang, S. Starke, T. Komura and J. Saito, 'Mode-Adaptive Neural Networks for Quadruped Motion Control', *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 145, 2018 (p. 20).

[117] C. Zhu, K. Muraoka, T. Kawabata, C. Cao, T. Fujimoto and N. Chiba, 'Real-time Animation of Bird Flight Based on Aerodynamics', *The Journal of the Society for Art and Science*, vol. 5, no. 1, pp. 1–10, 2006. DOI: 10.3756/artsci.5.1 (p. 18).

[118] K. Zibrek, E. Kokkinara and R. McDonnell, 'Evaluating the Response to Virtual Characters by Using a Social Behavioral Task', 2017 (p. 26).

[119] ——, 'The Effect of Realistic Appearance of Virtual Characters in Immersive Environments-Does the Character's Personality Play a Role?', *IEEE transactions on visualization and computer graphics*, vol. 24, no. 4, pp. 1681–1690, 2018 (p. 26).

Appendix A

Genetic Algorithm Source Code

```
import csv
import urllib
import urllib2
import random
import time
import datetime
import os, sys

def getKey(item):
  # ----- Must be position of Rating -----
  return item[34]

ts = time.time()
st = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d.%H-%M')

pUrl = "http://shadowraider.com/unity/parameters.100_3D.txt"
rUrl = "http://shadowraider.com/unity/nulistcsv.php?type=100_3D"

Parameters = "/Users/Gareth/Dropbox/University/PhD/TestResults/
                                    100_3D/Parameters/" + st + ".txt"
Ratings = "/Users/Gareth/Dropbox/University/PhD/TestResults/100_3D/
                                    SavedRatings/" + st + ".csv"

urllib.urlretrieve(pUrl, Parameters)
urllib.urlretrieve(rUrl, Ratings)

with open(Ratings, 'rb') as f:
  reader = csv.reader(f)
  lines = list(reader)
  newLines = list()
```

```
    # ----- Re-ordering the parameters so they are read correctly
                                          -----
  lineNo = [0, 21, 8, 12, 18, 19, 13, 28, 9, 14, 20, 7, 24, 22, 25,
                                    23, 29, 30, 31, 32, 11, 10, 26, 1,
                                     2, 3, 35, 34,  33, 5, 4, 15, 16,
                                    17, 6, 27]
  i = 0

  for line in lines:
    newLine = list()
    while i < len(lineNo):
      newLine.append(line[i])
      i += 1
    newLines.append(newLine)
  lines = newLines

with open(Parameters, 'rb') as f:
  reader = csv.reader(f)
  params = list(reader)

counter = int(params[-1][0]) + 1

k = 0
while k < len(lines):
  lines[k].pop(0)
  k += 1

# ----- How many results are needed to run the generation -----
if (k > 100):
  finalList = list()
  avg = 0
  while 0 < len(lines):
    count = 0
    j = 0
    currentLine = lines[0][0]
    while j < len(lines):
      if currentLine == lines[j][0]:
        temp = list()
        count += 1
        # ----- The position of the Ratings -----
```

```
        avg = avg + int(lines[j][34])
        temp.append(lines.pop(j))
      else:
        j += 1
    avg = avg / (count * 1.0)
    temp[0][34] = avg
    finalList.append(temp[0])
    avg = 0
finalList = sorted(finalList, key=getKey, reverse=True)

    # ----- Adding the fittest 25\% to the new generation -----
percentage = len(finalList)/float(100)*25
nextGen = list()
i = 0
while i < percentage:
    # ----- The position of the Ratings -----
    nextGen.append(finalList[i] [:34])
    i += 1

# ---------------------------------------------------------
i = 0
tFitness = 0
while i < len(finalList):
    # ----- The position of the Ratings -----
    tFitness += finalList[i][34]
    i += 1

ID = counter

while ID < counter + 100:
    P1 = list()
    def RouletteWheel1 (finalList):
      i = 0
      pick = random.uniform(0, tFitness)
      current = 0
      while i < len(finalList):
        # ----- The position of the Ratings -----
          current += finalList[i][34]
          if current > pick:
            # ----- The position of the Ratings -----
```

```
            P1.extend(finalList[i][1:34])
            break
        i += 1

P2 = list()
def RouletteWheel2 (finalList):
  i = 0
  pick = random.uniform(0, tFitness)
  current = 0
  while i < len(finalList):
    # ----- The position of the Ratings -----
      current += finalList[i][34]
      if current > pick:
        # ----- The position of the Ratings -----
          P2.extend(finalList[i][1:34])
          break
      i += 1

RouletteWheel1(finalList)
RouletteWheel2(finalList)

# ----- Change depending on number of Parameters -----
cp = random.randint(1, 34)

C1 = list()
C1.extend(P1[:cp])
C1.extend(P2[cp:])

C2 = list()
C2.extend(P2[:cp])
C2.extend(P1[cp:])

    # ----- Running a mutation function on each parameter for
                                        both children -----
i = 0
while i < len(C1):
  chance = random.randint(1, 100)
  if chance == 1:
    mutate = random.uniform(-0.01, 0.01)
    C1[i] = round(float(C2[i]) + mutate, 2)
```

```
      if (C1[i] < 0):
        C1[i] = 0
      i += 1
    else:
      i += 1
  i = 0
  while i < len(C2):
    chance = random.randint(1, 100)
    if chance == 1:
      mutate = random.uniform(-0.01, 0.01)
      C2[i] = round(float(C2[i]) + mutate, 2)
      if (C2[i] < 0):
        C2[i] = 0
      i += 1
    else:
      i += 1

  C1.insert(0, ID)
  ID += 1
  C2.insert(0, ID)
  ID += 1

  nextGen.append(C1)
  nextGen.append(C2)

upload = ""

i = 0

  # ----- Populating the new generation with children -----
while i < len(nextGen):
  j = 0
  while j < len(nextGen[i]):
    upload += str(nextGen[i][j]) + ", "
    j += 1
  upload += str(nextGen[i][j]) + "\n"
  i += 1

url = 'http://www.shadowraider.com/unity/change.php?type=100_3D'
values = {'data' : upload }
```

```
    data = urllib.urlencode(values)
    req = urllib2.Request(url, data)
    response = urllib2.urlopen(req)

    urllib2.urlopen('http://shadowraider.com/unity/nuclear.php?type=
                                        100_3D')
else:
    os.remove("/Users/Gaz/Dropbox/University/PhD/TestResults/100_3D/
                                        Parameters/" + st + ".txt")
    os.remove("/Users/Gaz/Dropbox/University/PhD/TestResults/100_3D/
                                        SavedRatings/" + st + ".csv")

print("Done")
```

Appendix B

A Dolphin Models Parameters

To aid the understanding of how each parameter effects the dolphins behaviour they have been broken down into five section: Body, Brain, Mouth, Tail.

B.1 Body

Barrel Roll Speed Range Min:

Range - 100 - 600

Granularity - 1

Type - float

Description - Minimum speed of a barrel roll (how it spins around the x axis). A random speed between min and max is chosen every time "change time" triggers.

Barrel Roll Speed Range Max:

Range - barrelRollSpeedRangeMin - 600

Granularity - 1

Description - Maximum speed of a barrel roll (how it spins around the x axis). A random speed between min and max is chosen every time "change time" triggers.

Default Rotate To Target Speed:

Range - 0.00 - 10.00

Granularity - 0.01

Description - Speed that dolphin rotates towards target in the water.

Default Speed:

Range - 0.00 - 5.00

Granularity - 0.01

Description - Base swim speed of the dolphin.

Default Turn Speed:

Range - 0.00 - 5.00

Granularity - 0.01

Description - Speed that dolphin locally rotates body (undulation of backbone, driven by frequency and amplitude).

Fastest Speed:

Range - defaultSpeed - 5.00

Granularity - 0.01

Description - Fastest swim speed of the dolphin (every time "change time" triggers, dolphin will accelerate to fastest speed, then slow down until default speed is reached).

Acceleration:

Range - 0.00 - 10.00

Granularity - 0.01

Description - How fast dolphin can accelerate to target swim speed.

Near Player Speed:

Range - 0.00 - defaultSpeed

Granularity - 0.01

Description - Dolphin’s target speed when near the player (usually much slower).

Swim Amplitude Min X:

Range - 0.00 - 100.00

Granularity - 0.01

Description - Local amplitude of dolphin as it swims through water (side to side).

Swim Amplitude Max X:

Range - swimAmplitudeMinX - 100.00

Granularity - 0.01

Description - Local amplitude of dolphin as it swims through water (side to side).

Swim Amplitude Min Y:

Range - 0.00 - 150.00

Granularity - 0.01

Description - Local amplitude of dolphin as it swims through water (up and down).

Swim Amplitude Max Y:

Range - swimAmplitudeMinY - 150.00

Granularity - 0.01

Description - Local amplitude of dolphin as it swims through water (up and down).

Swim Frequency Min X:

Range - 0.00 - 10.00

Granularity - 0.01

Description - Local frequency of dolphin as it swims through water (side to side).

Swim Frequency Max X:

Range - swimFrequencyMinX - 10.00

Granularity - 0.01

Description - Local frequency of dolphin as it swims through water (side to side).

Swim Frequency Min Y:

Range - 0.00 - 10.00

Granularity - 0.01

Description - Local frequency of dolphin as it swims through water (up and down).

Swim Frequency Max Y:

Range - swimFrequencyMinY - 10.00

Granularity - 0.01

Description - Local frequency of dolphin as it swims through water (up and down).

B.2 Brain

Barrel Role Chance:

Range - 100 - 1500

Granularity - 1

Description - Chance of triggering a barrel roll each frame.

Change Time:

Range - 1.00 - 10.00

Granularity - 0.01

Description - How often a dolphin changes all its range parameters (barrel roll speed, target position, chattering speed, mouth open time, local frequency and acceleration). Also causes it to kick its tail and accelerate to fastest speed.

Faithfulness:

Range - 0 - 1000

Granularity - 1

Description - Chance that dolphin will lose interest in the player, make random position in water the target.

Friendliness:

Range - 0 - 1000

Granularity - 1

Description - Chance that dolphin will take interest in the player, make player the target.

Near Player Distance:

Range - 0.00 - 8.00

Granularity - 0.01 **Description** - Distance to player before it considers itself close. Used for switching target speed from default to near player speed.

Near Player Min Distance:

Range - 0.00 - 3.00

Granularity - 0.01

Description - The closest the dolphin can get to player before repulse force is applied to push away.

Near Player Repulse Force:

Range - 0.00 - 1.00

Granularity - 0.01

Description - Force applied when dolphin is too close to it push away.

B.3 Mouth

Chattering Speed Min:

Range - 1.00 - 50.00

Granularity - 0.01

Description - Max speed that dolphin chatters mouth (opens and closes).

Chattering Speed Max:

Range - chatteringSpeedMin - 50.00

Granularity - 0.01

Description - Min speed that dolphin chatters mouth (opens and closes).

Mouth Open Chance:

Range - 0 - 1000

Granularity - 1

Description - Chance every frame that dolphin opens mouth.

Mouth Open Time Range Min:

Range - 0.00 - 2.00

Granularity - 0.01

Description - Min time dolphin chatters.

Mouth Open Time Range Max:

Range - mouthOpenTimeRangeMin - 2.00

Granularity - 0.01

Description - Max time dolphin chatters.

B.4 Tail

Tail Rest Amplitude:

Range - tailRestAmplitude - 120.00

Granularity - 0.01

Description - When "change time" triggers, dolphin will accelerate to max speed, with tail kicking at this amplitude (frequency is tied to overall swim frequency).

Tail Rest Amplitude:

Range - 0.00 - 120.00

Granularity - 0.01 **Description** - The amplitude that the tail kicks when at rest.

Tail Amplitude Decay:

Range - 0.00 - 50.00

Granularity - 0.01

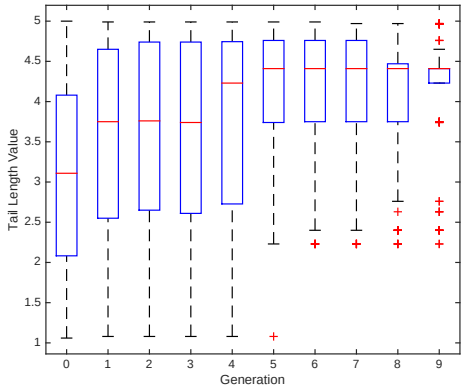
Description - Rate that tail decays from max amplitude to rest amplitude.

Appendix C

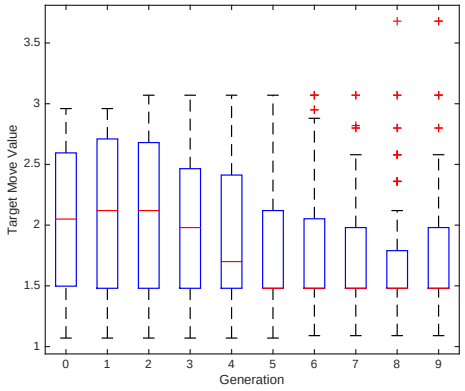
Snakes Experiment Graphs

This section includes box plots for all 7 parameters used within the snake system.

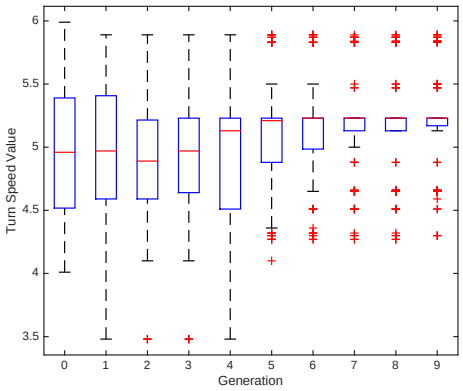
Figure C.1: All graphs for snake study



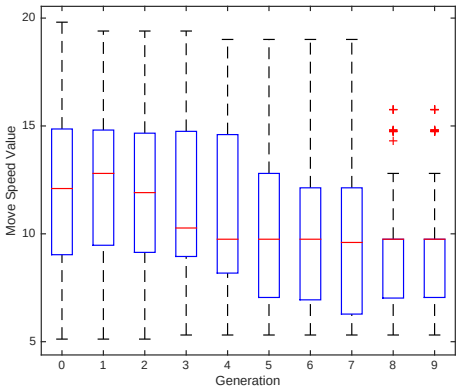
(a) Tail Length Trends



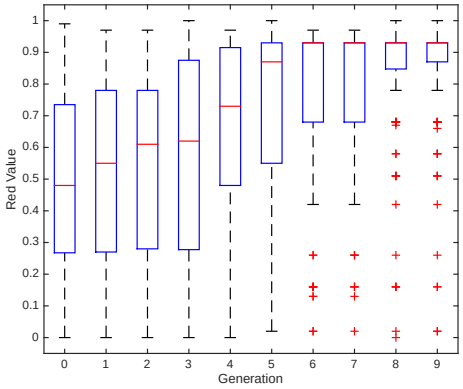
(b) Target Move Trends



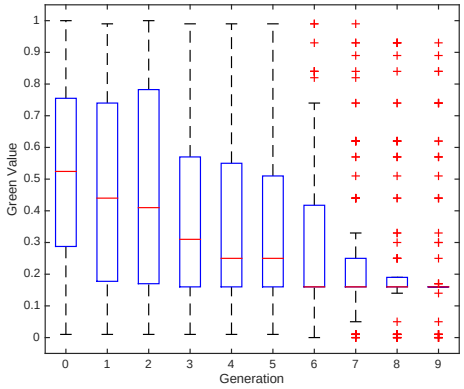
(c) Turn Speed Trends



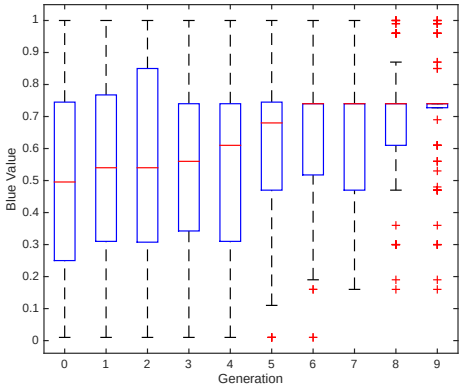
(d) Move Speed Trends



(e) Red 2D Trends



(f) Green VR Trends



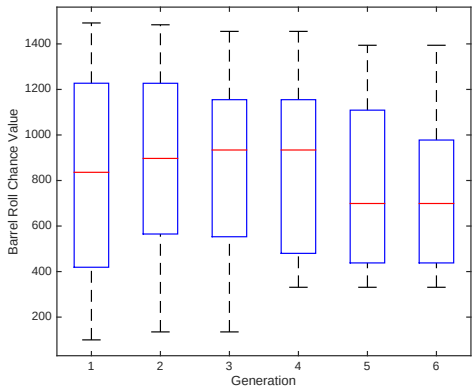
(g) Blue 2D Trends

Appendix D

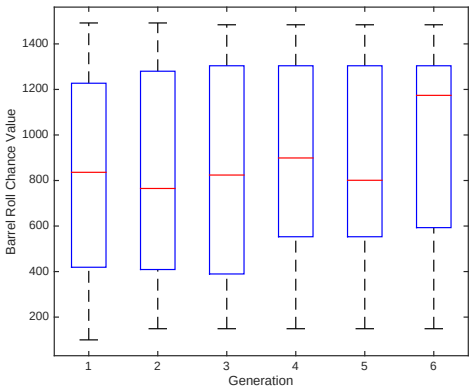
2D Vs. VR Experiment Graphs

This section includes box plots for all 33 parameters used within the Dolphin system. They are in alphabetical order with the 2D version on the left and VR on the right.

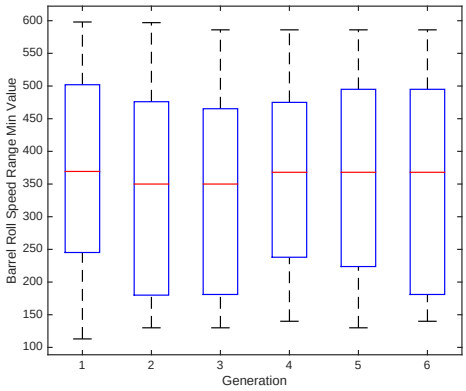
Figure D.1: All graphs for 2D Vs. VR study



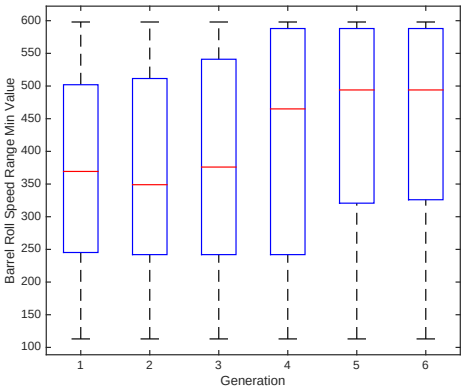
(a) Barrell Roll Chance 2D Trends



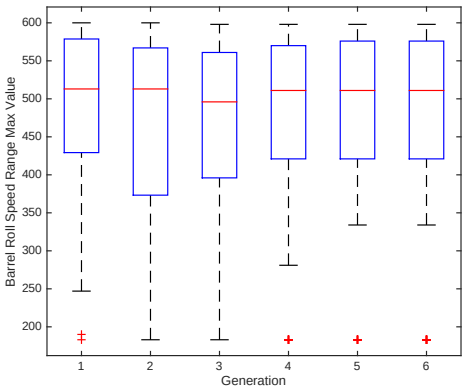
(b) Barrell Roll Chance VR Trends



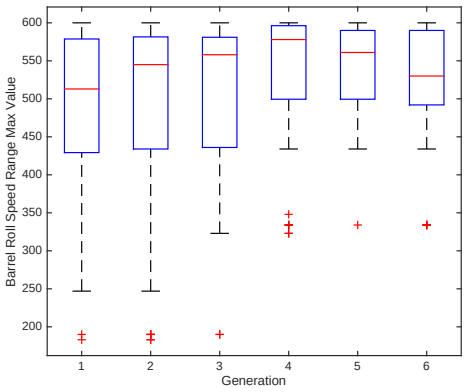
(c) Barrel Roll Speed Range Min 2D Trends



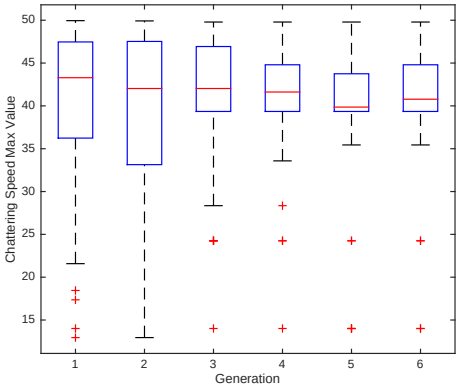
(d) Barrel Roll Speed Range Min VR Trends



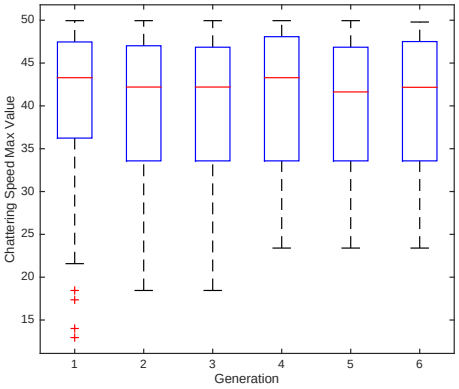
(e) Barrel Roll Speed Range Max 2D Trends



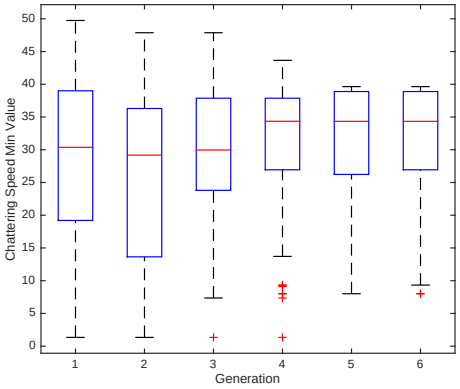
(f) Barrel Roll Speed Range Max VR Trends



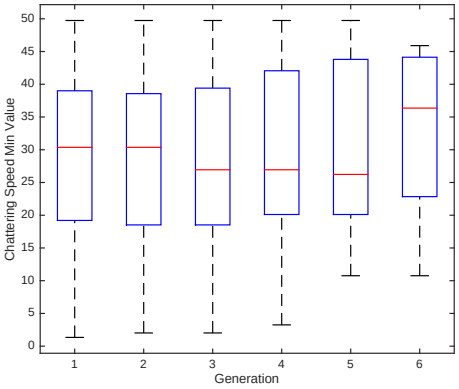
(g) Chattering Speed Max 2D Trends



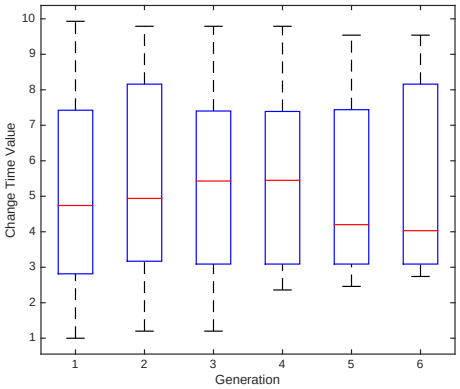
(h) Chattering Speed Max VR Trends



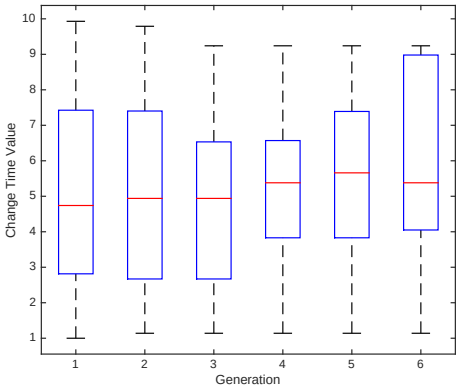
(i) Chattering Speed Min 2D Trends



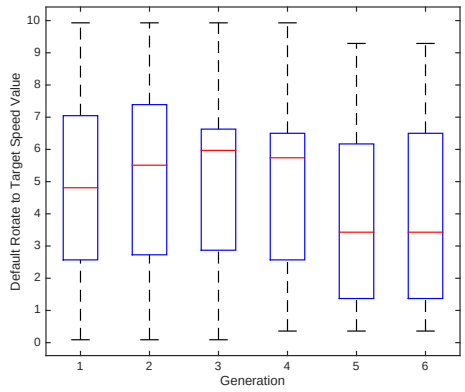
(j) Chattering Speed Min VR Trends



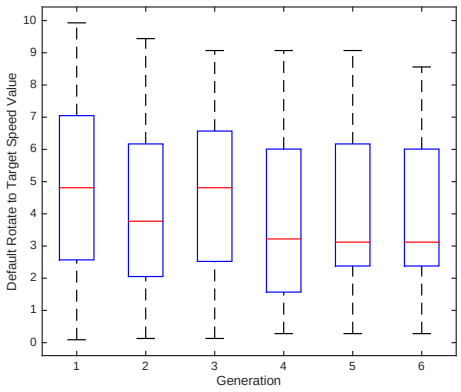
(k) Change Time 2D Trends



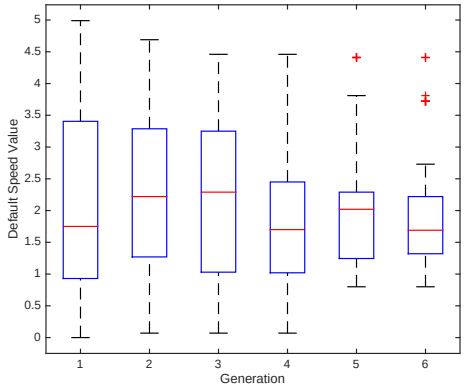
(l) Change Time VR Trends



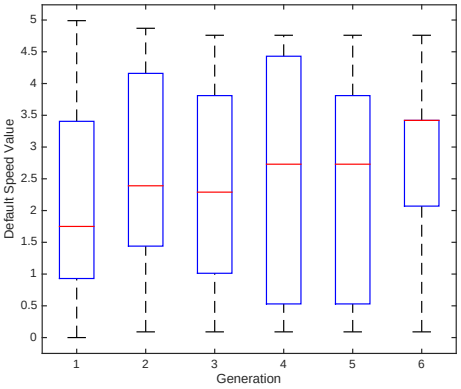
(m) Default Rotate to Target Speed 2D Trends



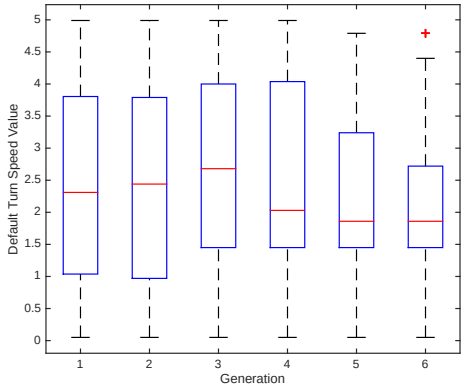
(n) Default Rotate to Target Speed VR Trends



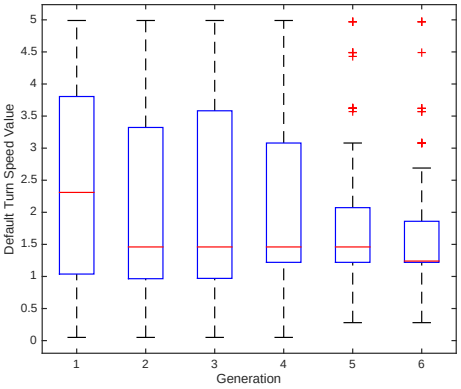
(o) Default Speed 2D Trends



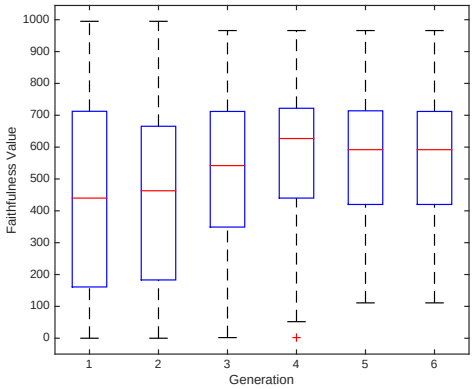
(p) Default Speed VR Trends



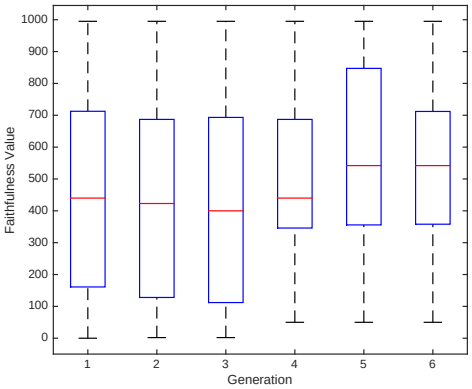
(q) Default Turn Speed 2D Trends



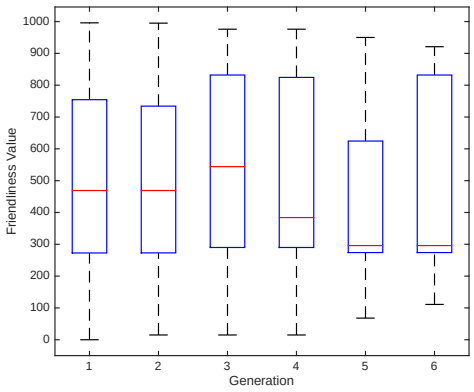
(r) Default Turn Speed VR Trends



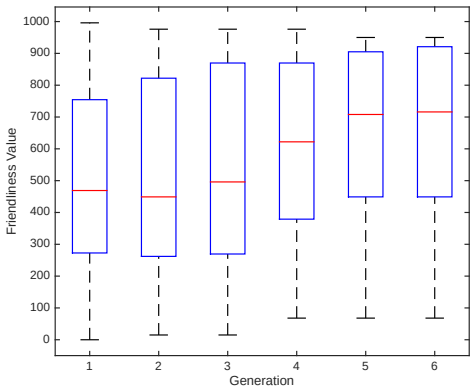
(s) Faithfullness 2D Trends



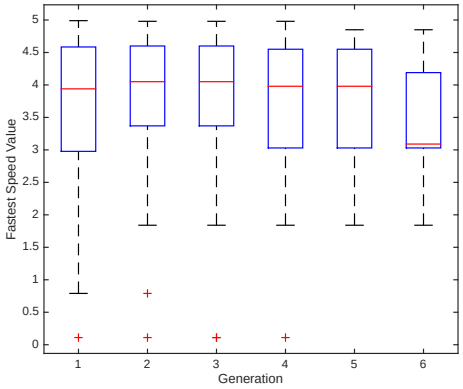
(t) Faithfulness VR Trends



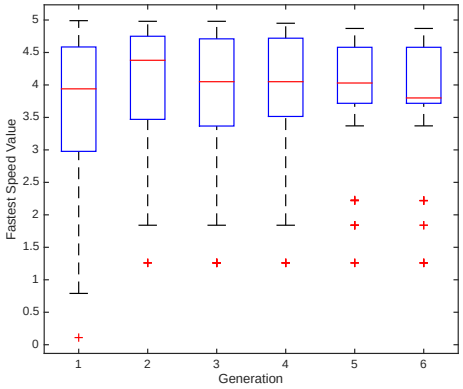
(u) Friendliness 2D Trends



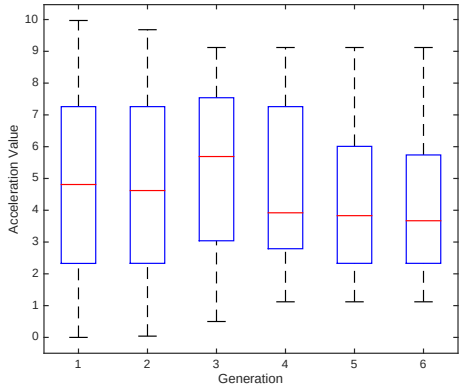
(v) Friendliness VR Trends



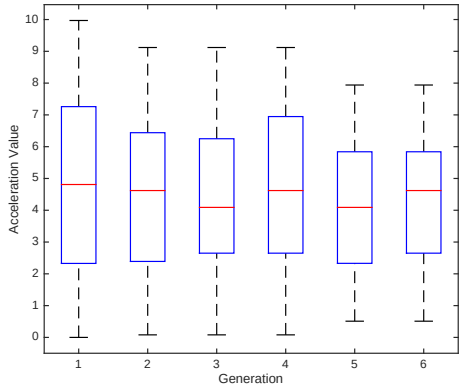
(w) Fastest Speed 2D Trends



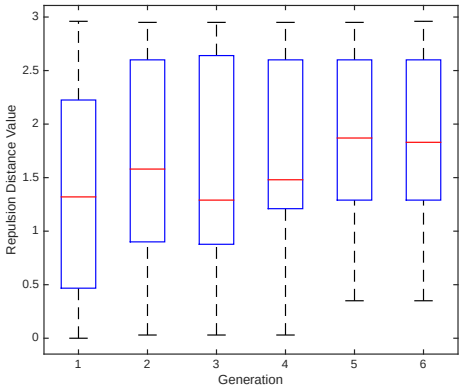
(x) Fastest Speed VR Trends



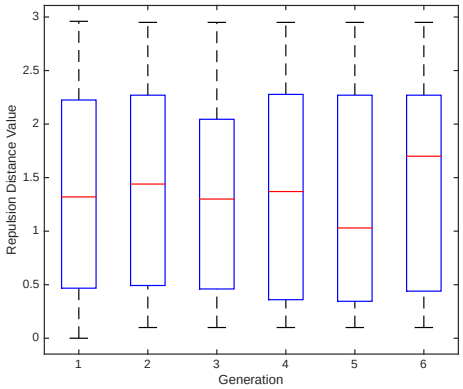
(y) Max Acceleration 2D Trends



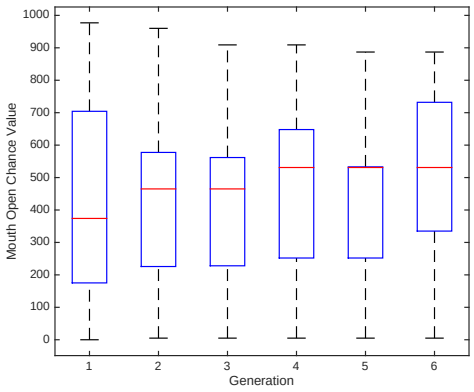
(z) Max Acceleration VR Trends



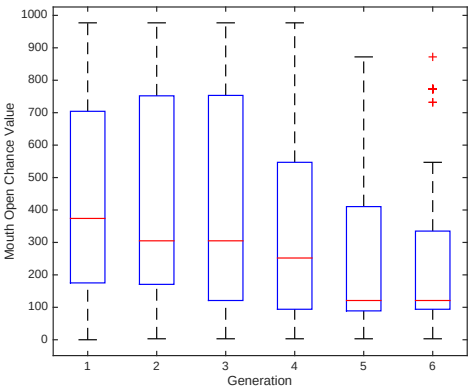
(aa) Min Distance



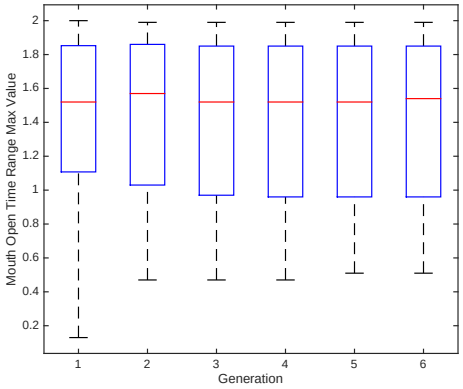
(ab) Min Distance VR Trends



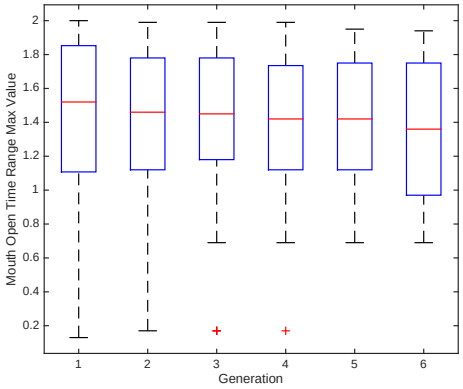
(ac) Mouth Open Chance 2D Trends



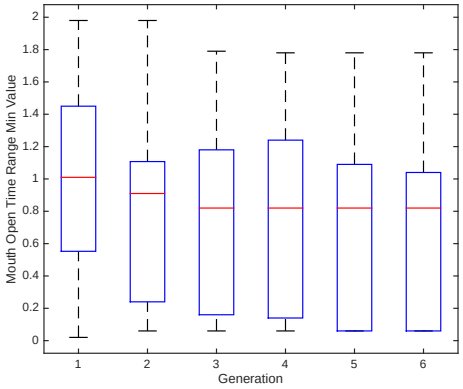
(ad) Mouth Open Chance VR Trends



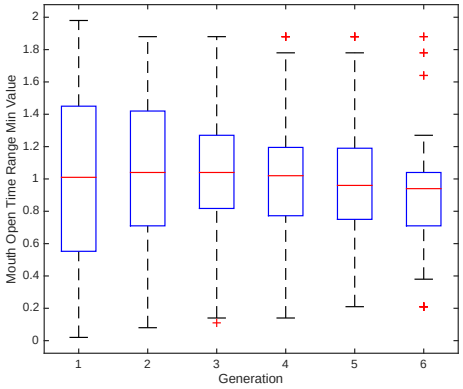
(ae) Mouth Open Time Range Max 2D Trends



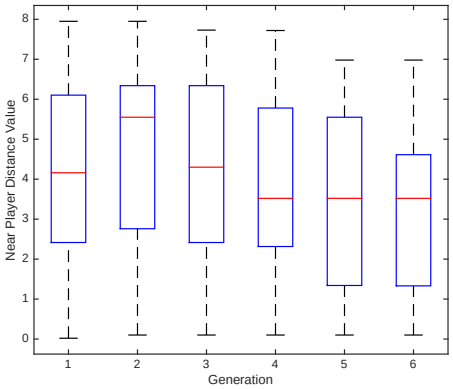
(af) Mouth Open Time Range Max VR Trends



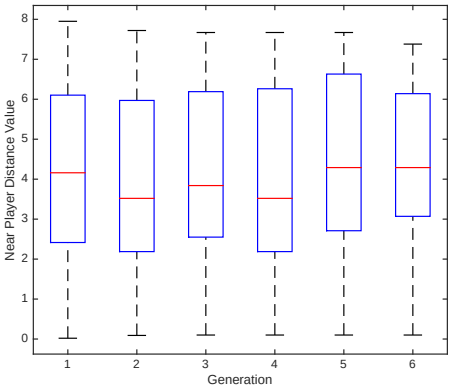
(ag) Mouth Open Time Range Min 2D Trends



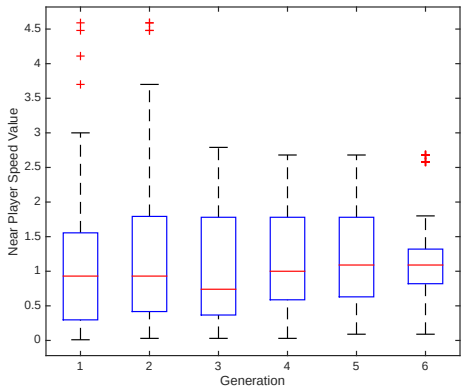
(ah) Mouth Open Time Range Min VR Trends



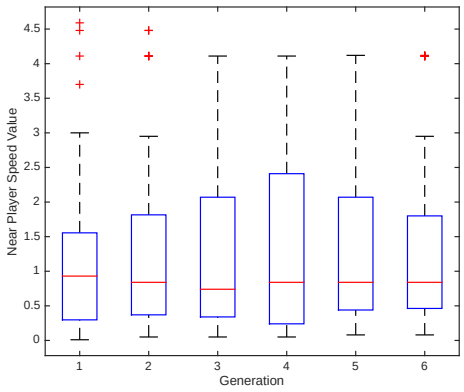
(ai) Near Player Distance 2D Trends



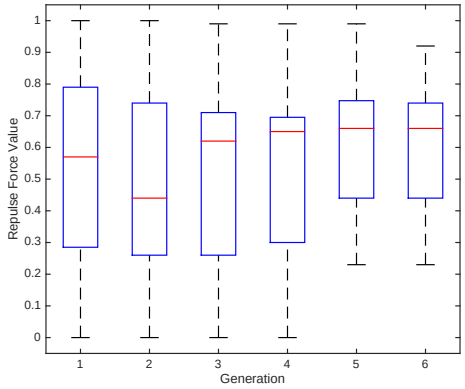
(aj) Near Player Distance VR Trends



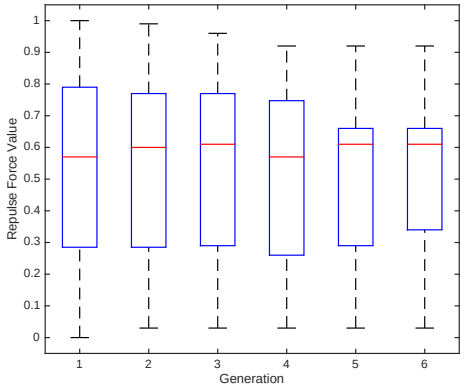
(ak) Near Player Speed 2D Trends



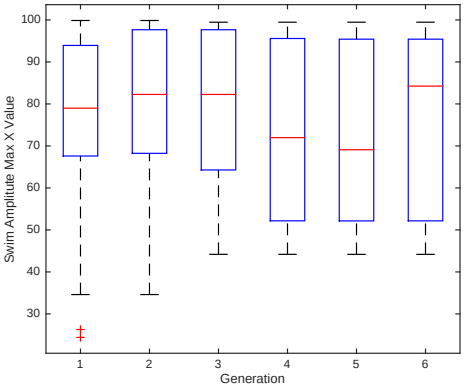
(al) Near Player Speed VR Trends



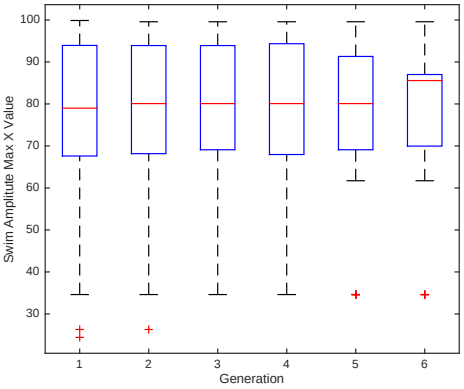
(am) Repulse Force 2D Trends



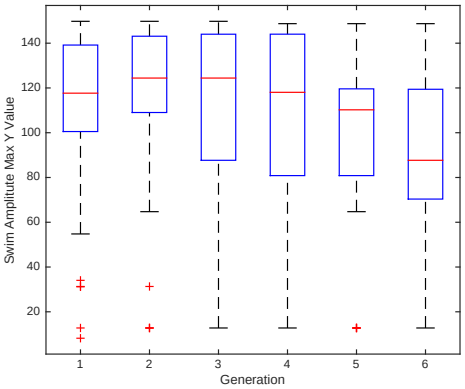
(an) Repulse Force VR Trends



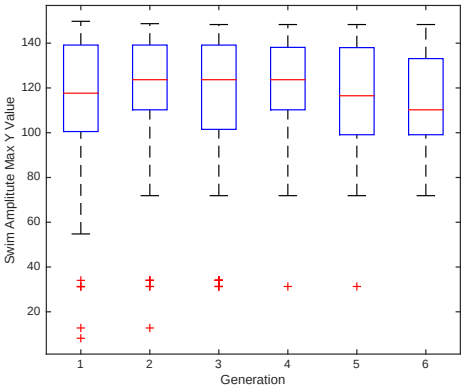
(ao) Swim Amplitude Max X 2D Trends



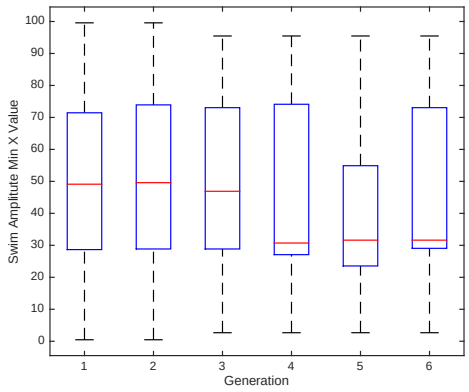
(ap) Swim Amplitude Max X VR Trends



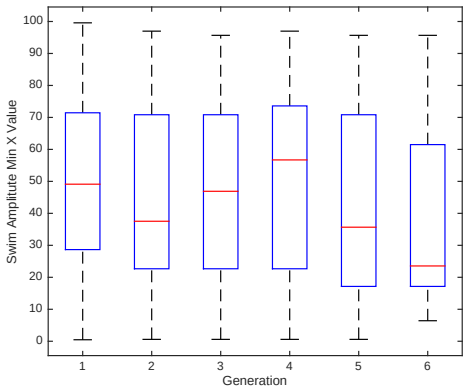
(aq) Swim Amplitude Max Y 2D Trends



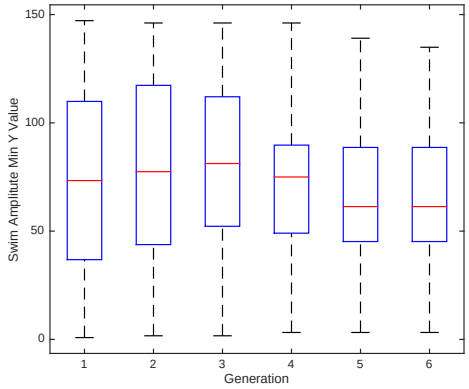
(ar) Swim Amplitude Max Y VR Trends



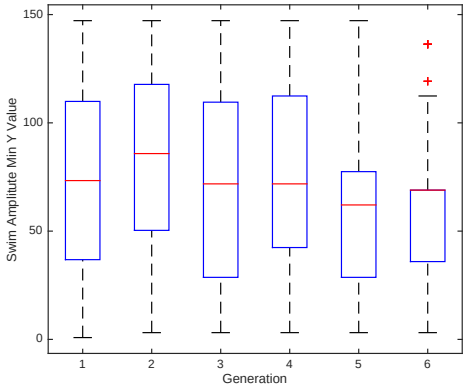
(as) Swim Amplitude Min X 2D Trends



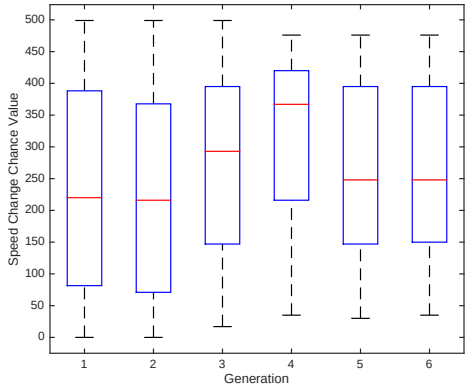
(at) Swim Amplitude Min X VR Trends



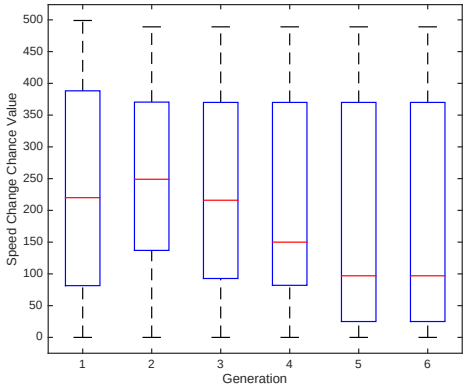
(au) Swim Amplitude Min Y 2D Trends



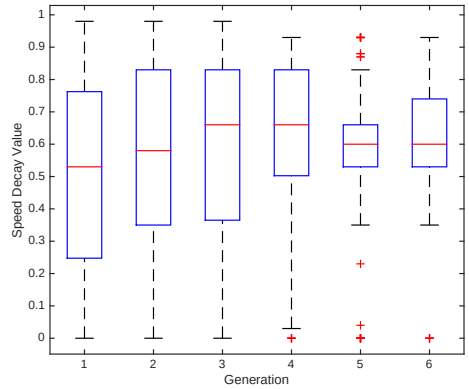
(av) Swim Amplitude Min Y VR Trends



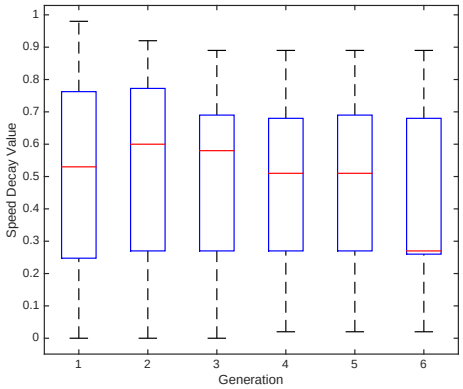
(aw) Speed Chance Chance 2D Trends



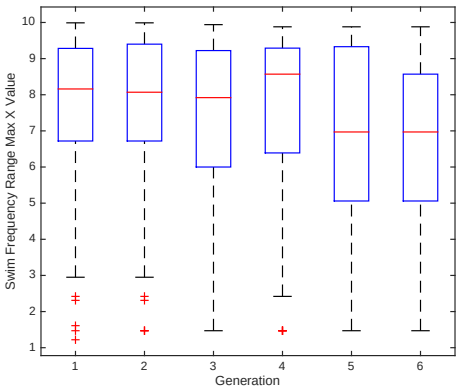
(ax) Speed Change Chance VR Trends



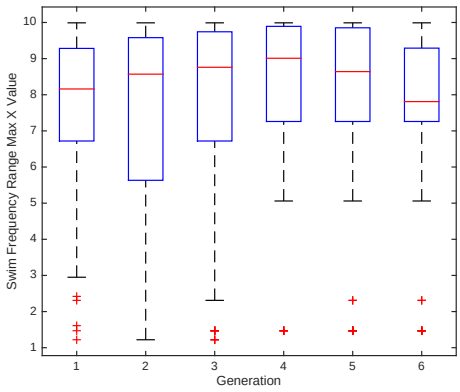
(ay) Speed Decay 2D Trends



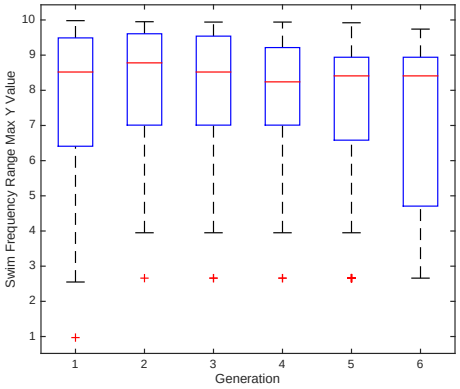
(az) Speed Decay VR Trends



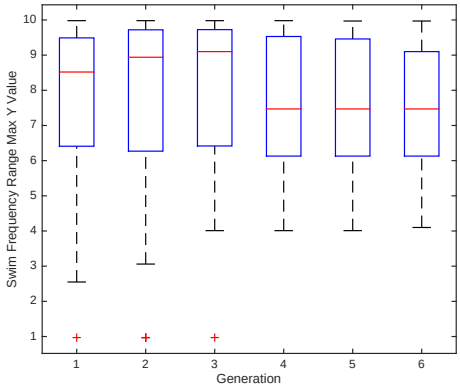
(ba) Swim Frequency Max X 2D Trends



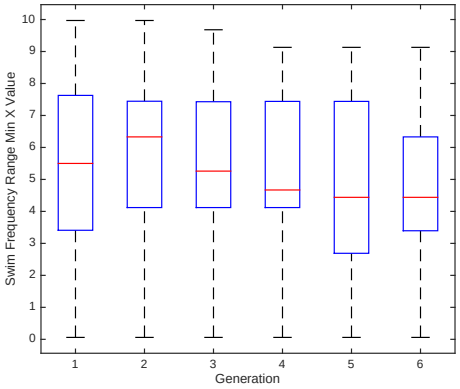
(bb) Swim Frequency Max X VR Trends



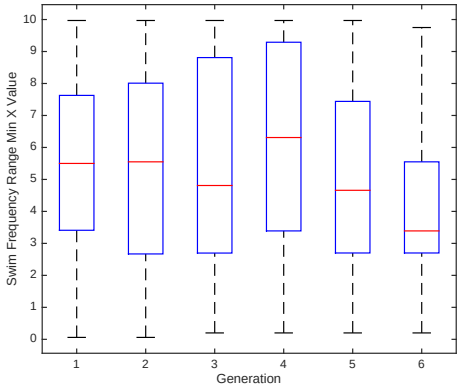
(bc) Swim Frequency Max Y 2D Trends



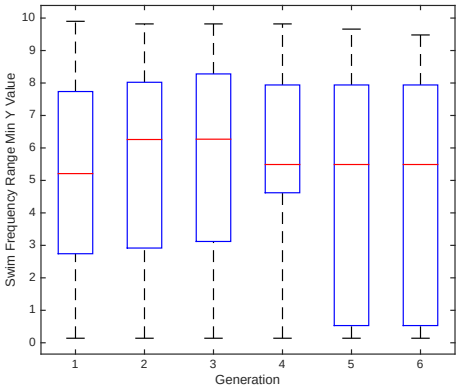
(bd) Swim Frequency Max Y VR Trends



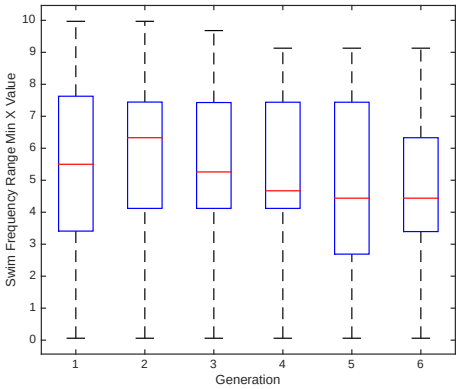
(be) Swim Frequency Min X 2D Trends



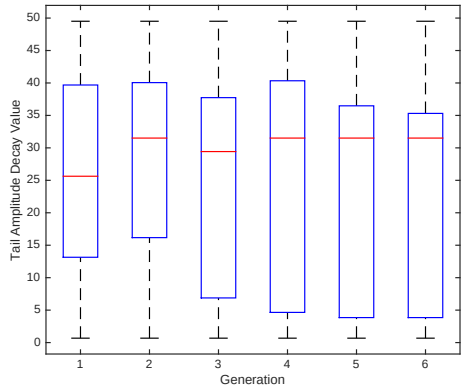
(bf) Swim Frequency Min X VR Trends



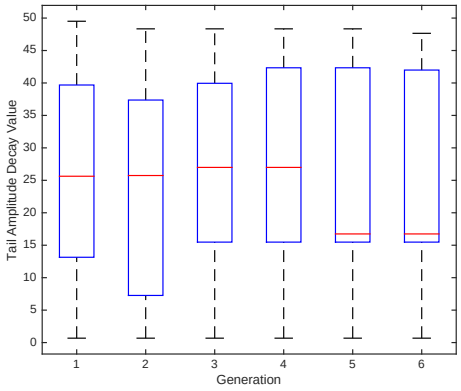
(bg) Swim Frequency Min X 2D Trends



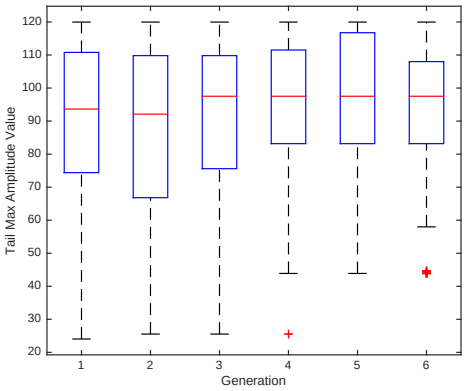
(bh) Swim Frequency Min X VR Trends



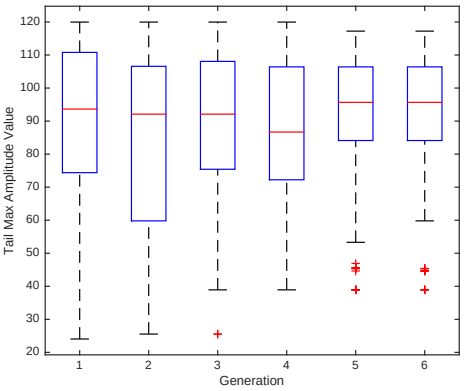
(bi) Tail Amplitude Decay 2D Trends



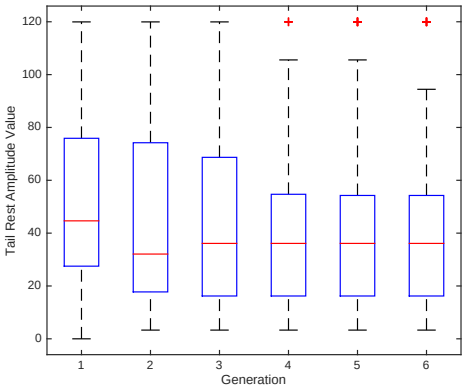
(bj) Tail Amplitude Decay VR Trends



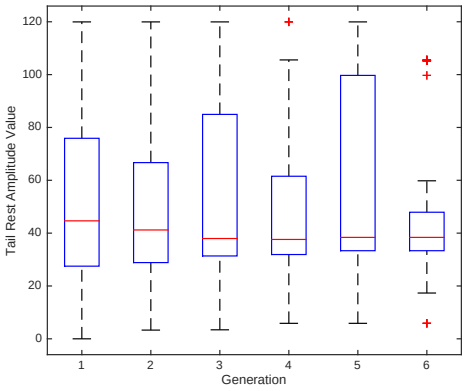
(bk) Tail Max Amplitude 2D Trends



(bl) Tail Max Amplitude VR Trends



(bm) Tail Rest Amplitude 2D Trends



(bn) Tail Rest Amplitude VR Trends

Appendix E

Cyberworlds 2015 Conference Poster [35]

Towards Crowd-Sourced Parameter Optimisation for Procedural Animation

Gareth I. Henshall, Christopher J. Headleand, William J. Teahan and Llyr Ap Cenydd
Bangor University, United Kingdom

Abstract

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.

Introduction

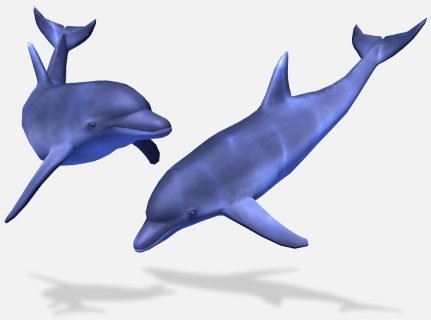
Procedural animation potentially allows for the automatic generation of limitless and situated organic animation in real-time. However, as procedural animation systems become more complex, the parameter space can very quickly become vast and difficult to optimise. We aim to crowd-source the tuning of the parameter space using an interactive web application.

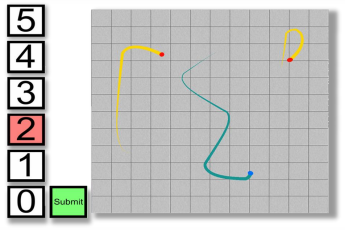
Initial Population

At the start of our simulation, an initial random population of n candidate snakes are generated. During each instance of the experiment, the client machine retrieves and checks out two candidate creatures. For the purpose of ranking, each snake is identified by a coloured marker.

Ratings System

The candidate creatures are ranked by the user based on a scale of 0 (low) to 5 (high). 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. At the end of the generation, all raw scores are averaged, resulting in an overall fitness for each candidate creature.





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. 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.

Conclusion

We intend to use our system to tune a much more complex creature animation system currently being developed in tandem. The system consists of over 70 parameters and is capable of producing life-like and diverse behaviour for a variety of marine animals including dolphins. A further goal of this research is to also test whether virtual reality affects the user's perception of a creatures' organicity and behavioural realism.

HPC WALES • CYMRU

For more details or to take part in future experiments please take a card and email me for more information.

PRIFYSGOL BANGOR UNIVERSITY

Appendix F

Artificial Evolution (EA) 2017

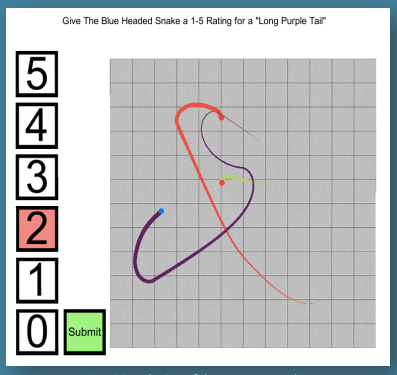
Conference Poster [36]

Crowd-Sourced Optimisation of Procedural Animation Systems

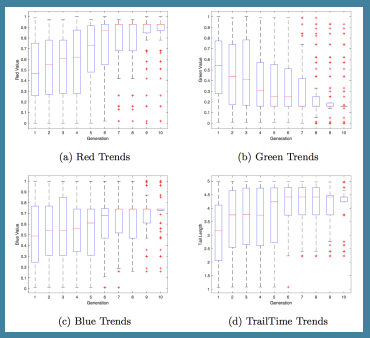
While rendering techniques allow for increasingly life-like visuals, the animation of the vast majority of complex characters is still reliant on pre-created data. Procedural animation systems on the other hand have the potential to synthesise life-like motion and behaviour automatically.

However as procedural animation systems become more complex, it can be difficult to find a desired or optimised behaviour within the vast parameter space. Automatic optimisation can also be notoriously difficult, due to potentially hundreds of interlinked parameters and the subjectivity of the results.

We describe a web-based simulation which allows a large number of anonymous users to interactively rate a population of snake-like creatures to a prescribed criteria, such as "long purple tail".



Participant's view of the prototype study



(a) Red Trends (b) Green Trends (c) Blue Trends (d) TrailTime Trends

RGB & TrailTime trends across generation

A wide demographic of participants were asked to rate an initial random population of snakes on a 0 (low) to 5 (high) scale.

A simple genetic algorithm then uses Roulette wheel selection to breed the next generation, based on the user ratings. The algorithm chooses the parents based on their fitness calculated from their average rating, combined with a small chance of mutation.

The graphs (right) show RGB colour and tail length trends across each generation. The results demonstrate that our system can successfully allow a crowd of people to tune animation systems to a required specification.

We are currently developing techniques that allow users to interactively rate much more complex animation systems, such as for virtual dolphins and sea lions. We are also investigating how our system can be used to optimise animation systems in and for Virtual Reality (VR), including the ability to tune parameters using natural motion controller interaction and by collaborative users.

Gareth I. Henshall, William J. Teahan and Llyr ap Cenydd
Email: g.i.henshall@bangor.ac.uk

PRIFYSGOL BANGOR UNIVERSITY