Lukas Schabler, BSc

Tagorithms

Master Project

to achieve the university degree of

Master of Science

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Dr. Johanna Pirker

Institute for Softwaretechnology
Head: Univ.-Prof. Dipl-Ing. Dr.techn. Stefanie Lindstaedt

Graz, December 2017

This document is set in Palatino, compiled with pdfLaTeX2e and Biber.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Abstract

In computer games, designers frequently leverage biologically inspired algorithms such as flocking, particle swarm optimization, and firefly algorithms to give players the perception of intelligent behaviour of groups of non-player characters (NPCs). While extensive effort has been expended designing these algorithms, a comparison of biological algorithms to naive directional algorithms (travel towards the opponent) has yet to be completed. In the tagorithm paper, we compare the biological algorithms listed above against a naive control algorithm to assess the effect that these algorithms have on measures of player experience. The results reveal that the Swarming algorithm, followed closely by Flocking provide the best gaming experience. However, players noted that the firefly algorithm was most salient. An understanding of the strengths of different behavioural algorithms for NPCs will contribute to the design of algorithms that depict more intelligent crowd behaviour in gaming and computer simulations.

Contents

Abstract iii

1 Introduction 1

2 Algorithms 3
- 2.1 Control Algorithm . . . 3
- 2.2 Flocking Algorithm . . . 4
- 2.3 Particle Swarm Optimization Algorithm . . . 5
- 2.4 Firefly Algorithm . . . 5

3 Game Structure 9

4 Study Results 11

Bibliography 13

List of Figures

2.1 Control algorithm 4
2.2 Flocking rules 5
2.3 PSO motion vector 6
2.4 Firefly algorithm 7

3.1 Software architecture 10

1 Introduction

Non-player characters (NPCs) are a vital part of most computer games. To bring NPCs to life is a major task for developers. Game designers work with predetermined or responsive algorithms up to true artificial intelligence to model realistic behaviour. One of the primary aspects of NPC behaviour involves the characters movement. The creation of realistic movements is necessary both for individual NPCs and for NPCs groups. In this work, I focused on the character group movement with the associated algorithms during the game. While there is a lot of research in designing these algorithms a comparison of industry standard biological algorithms and directional algorithms has not yet been completed. An understanding of the strength of the different behavioural algorithms is vital for developers to select the proper algorithm for their application. The evaluation of the algorithms is the main contribution in tagorithms.[1]

In this report I want to go more into detail about the implementation, design and technology in the tarorithms study. For the study, we invited the participants to play a game where a group of enemies is controlled by different algorithms. In the first chapter I characterize the used algorithms in more detail. The goal of all these algorithms is to enhance realism, player engagement and enjoyment. The chapter describes the practical implementation in tagorithms since every algorithm had to be modified based on the specifics of the algorithm to fit our application. The second chapter focuses on the general structure of the game, data storage and web services. I give important insights about the implementation and the used technology. The last chapter presents the results of the study game metrics, questionnaire data and interview data. I go over the essential differences in gameplay as reported by players.

[1] Wehbe et al., 2017.

2 Algorithms

For our study, we control NPC groups with four different algorithms: Flocking, Particle Swarm Optimization, and Firefly Algorithms, against a simple Control Algorithm. Each algorithm requires significantly altered development approaches. In this chapter I will go into detail about the basic algorithm principles, implementation and modifications for every algorithm.

2.1 Control Algorithm

For the study we compare biological algorithms with a control algorithm. Every flock moves directly to the target as seen in Figure 2.1.

Figure 2.1: Control algorithm: Every flock moves directly to the target.

2.2 Flocking Algorithm

Displaying the complex motion of a flock of birds naturally in a computer simulation is very challenging. Reynolds (1987) developed an algorithm based on scripting the paths for each bird individually. The Flocking model is a generalization of a particle system. Each simulated bird is implemented as an independent actor that navigates according to its local perception of the dynamic environment.

Natural flocks consist of two opposing behaviors: a desire to stay close to the flock and a desire to avoid collisions within the flock. The simulated flock models the behavior with three simple rules as displayed in Figure 2.2. The rules are ordered with decreasing priority.

1. Collision Avoidance: avoid collisions with nearby flocks
2. Velocity Matching: attempt to match velocity with nearby flocks
3. Flock Centering: attempt to stay close to nearby flocks

Collision avoidance and dynamic velocity matching ensures that the members of a simulated flock are free to fly within a tight group. Flock centering makes a bird want to be near the center of the flock. The rules control heading and speed. An individual flock pays attention to all flocks within a certain distance as shown in Figure 2.2.

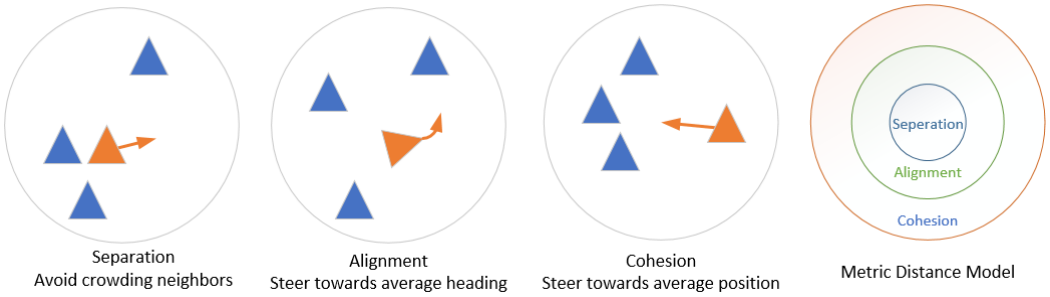

Figure 2.2: Three simple flocking rules, creating complex motion and interaction.

2.3 Particle Swarm Optimization Algorithm

Particle swarm optimization (PSO) was developed by Kennedy and Eberhart (1995), based on the swarm theory in particular tied to fish schooling. It is a genetic algorithm implemented using evolutionary programming. The natural behavior of animal schools is founded on specific rules. The individuals try to stay within the school at all times. The particle swarm optimization algorithm models the swarm behavior by solving an optimization problem.

The first part of the implementation initializes each particle with a random velocity. Than iteratively each particle's movement is influenced by its own local best known position, the global best position, which is found by other particles and the inertia of the current motion as seen in Figure 2.3. This is expected to improve a candidate solution with regard to a given measure of quality.

2.4 Firefly Algorithm

The Firefly algorithm is a special variation of PSO, based on the behavior of fireflies. Fireflies produce short rhythmic flashes to attract partners and prey. Since the light intensity decreases with distance to the light source other fireflies are only visible in a limited distance, usually several hundred meters at night.

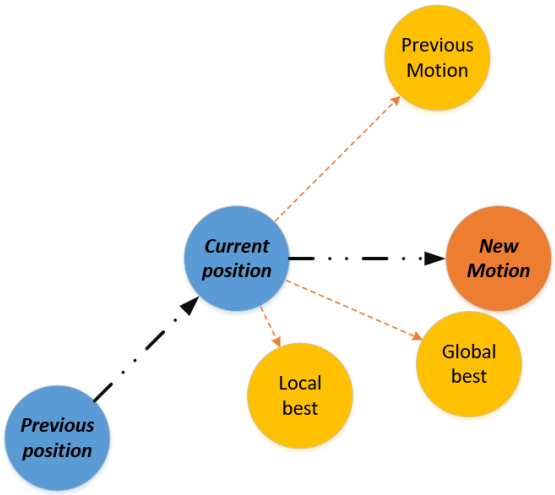

Figure 2.3: PSO algorithm: calculate new motion vector.

To simplify implementation by Yang (2009) the flashing characteristics are idealized. The algorithm is based on three rules (See Figure 2.4). All fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex. The attractiveness is proportional to their brightness, thus a firefly will always move towards the brightest in range. If there is no brighter one than a particular firefly, it will move randomly. The brightness of a firefly is calculated by a given measure of quality and the distance.

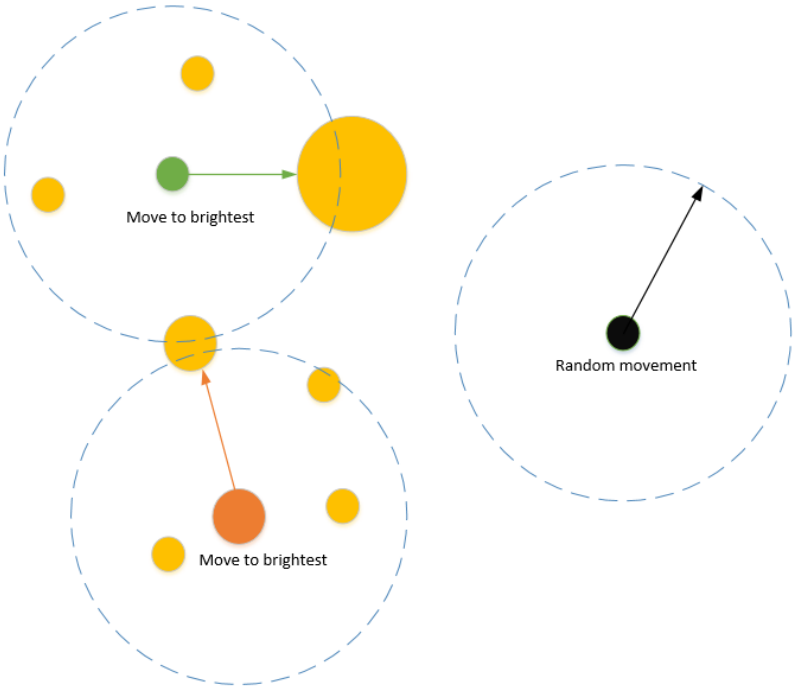

Figure 2.4: Firefly searches for a local maximum.

3 Game Structure

In Tagorithms developed by Wehbe et al. (2017) consists of fours levels, each of which has its own distinct enemy behavior based on a biological algorithm. The levels contain a consistent amount of 20 flocks, or rigid-body objects with applied steering algorithms. Every level consists of a minute of game play in which the user evades twenty enemy characters by moving their player character about the screen with the mouse. Each time the player is hit by an enemy, the hit counter is incremented by one and the enemy that made contact with the player respawns at a random location on the screen's perimeter. If an enemy moves off the side of the screen, it reappears on the opposite side at the same relative height it disappeared. The player character cannot move off screen and should the user move their cursor off their character at any point, the character will remain stationary until the cursor retouches the character. Enemies in each level all move at the same speed.

The main development tools are Unity 5.6 and Adobe Photoshop to design the in-game objects. As first step we developed a small demo on a tablet as proof of concept. The design is very simple with primitive geometries as seen in Figure 3.1. We implemented local data storage to save movements and statistics. The users had trouble with controlling the character movements using the touch input. We changed the input movement for the next iterations.

For the desktop version we improved the graphics with high quality assets. Using the mouse input improves the usability. We integrated the questionnaire for the study directly in the game. The data is collectively stored on an online SQL-server (seen in Figure 3.1).

For the second study we focused only on the two best algorithms from the first study: Flocking and PSO. We ported the game to FTP-Server using

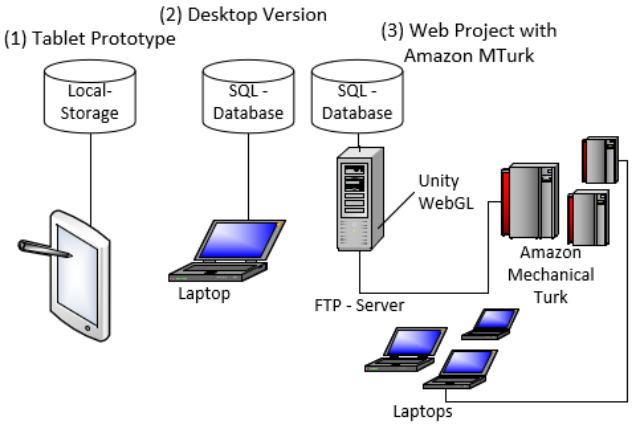

Figure 3.1: Program architecture: (1) Concept Prototype (2) Local desktop version with online SQL-Server (3) Online Version using Amazon MTurk Webservice.

the Web Graphics Library. The participants are recruited with Amazon Mechanical Turk. This Amazon Web Service reaches more participants and simplifies the evaluation.

4 Study Results

The interviews reveal some insight into the player experience associated with each algorithm tested. Players were separated by their feedback.

Participants reported liking the PSO enemies the best, followed closely by the Firefly enemies. Enemies liked least were almost equally divided. When asked about the hardest enemy, participants reported that the Flocking and Firefly enemies were the most difficult. PSO was reported to be the easiest to defeat, followed by the enemies of the Control algorithm.

Although participants felt that the Firefly algorithm was salient, the general consensus did not indicate that the effect on player experience would be positive.

Bibliography

Kennedy, James and Russell C. Eberhart (1995). "Particle swarm optimization." In: *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948 (cit. on p. 5).

Reynolds, Craig W. (1987). "Flocks, Herds and Schools: A Distributed Behavioral Model." In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '87. New York, NY, USA: ACM, pp. 25–34. ISBN: 0-89791-227-6. DOI: 10.1145/37401.37406. URL: http://doi.acm.org/10.1145/37401.37406 (cit. on p. 4).

Wehbe, Rina et al. (2017). "Biologically Inspired Gameplay: Movement Algorithms for Artificially Intelligent Non-Player Characters." In: Tokyo Japan: ACM (cit. on pp. 1, 9).

Yang, Xin-She (2009). "Firefly Algorithms for Multimodal Optimization." In: *Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications*. SAGA'09. Sapporo, Japan: Springer-Verlag, pp. 169–178. ISBN: 3-642-04943-5, 978-3-642-04943-9. URL: http://dl.acm.org/citation.cfm?id=1814087.1814105 (cit. on p. 6).