Simulating Metamorphism in a Swarm Robotics System

Final Report for CS39440 Major Project

Author: Mr. Rhydian Jenkins (rlj10@aber.ac.uk)

Supervisor: Dr. Elio Tuci (elt7@aber.ac.uk)

4th May 2016

Version 0.1 (Plan)

This report is submitted as partial fulfilment of a BSc degree in Computer Science (G400)

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Wales, UK

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name ……

Date ……

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name ……

Date ……

Acknowledgements

Elio

Neil

Computer Science Department

Abstract

Swarm Robotics is a new, scalable, and rugged solution to getting one robot to work with other robots. In a system that requires potentially vast amounts of agents to work together and complete a task too large or too complex for any single agent, it is essential that every bot is able to think for itself. An overall manager could be used to individually feed instructions to each bot and micro-manage the entire task however this method falls flat as more and more bots are added or as the task becomes more and more complex. No single machine can keep up with thousands of agents all needing detailed instructions at once.

Swarm robotics takes this problem and moves the responsibility of control over to every bot. With each bot thinking for itself, the system now has no limit to the amount of bots that can be added. Not only does the system become more scalable, but it also is less prone to error. If any of the bots malfunction or if any unexpected conditions arise, the bots will have the ability to adapt and decide for themselves based on their immediate environment what to do.

This project aims at a specific problem within Swarm Robotics – getting multiple bot to metamorphose. This document will tackle all the problems faced with simulating S-Bots as they organise themselves into a specific shape. They will be doing this through the use of LEDs situated on the bot itself. The bots will position themselves so that they will connect to the neighbouring bot to form a rigid shape.

Contents

1. Background, Analysis, and Process

1.1. Background

Wanted a project that was visual, high potential, seemed fun and interesting, as well as challenging and something I haven't done before.

Swarm Robotics stood out to me as a potential project as it seemed like a new and exciting future technology. I read many papers about possible applications and how this kind of multi-robot system is better than traditional multi-robot systems which sparked my curiosity further.

I found that this kind of methodology is closer to the real world examples of nature where sometimes vast numbers of agents (ants, bees, birds) work together. There is no leading project manager in an ants nest instructing each worker any on how to do their job, neither is there any king starling that instructs every other starling on where to fly to form the amazing flocks that we see at sunset. How could there be? How could a single agent instruct all those others.

After finding out what the essence of what Swarm Robotics is, I began to read papers on the current state of robotics and discovered that the subject has a long way to go before it is truly relevant in every-day life. The theory of "a thousand robots are better than one" is a solid idea however it seems that the implementation of such systems have yet to be fully achieved. Of course, there are certain systems such as AutoStore™ that are starting to creep into every day goings-on, however the real potential such as disaster recovery or automatic building construction is yet to be implemented.

Before very little research it was clear that the potential for such systems was huge, however only a small section of that potential has been implemented. This is the main reason I chose Swarm Robotics as my Dissertation project, as I see it as a technology that holds a large place in multi-robot systems of the future.

With more research the reasons for the lack of implementation became more clear. Getting a machine to decide on what to do with unknown conditions is always hard, and these bots will need to do exactly that. Given a high level task instruction – "Build this", "Search for survivors", "Clean this structure" etc – the bots will have to decide how they are going to help complete this task using only the data on their environment that the sensors they are equipped with gather.

Finally, I found a topic I would later invest the focus of my dissertation into – Metamorphism. While looking through various articles and videos I found a paper discussing the need for bots to connect themselves together and form certain shapes in order to complete tasks that are too physically difficult for any single bot – such as situations where a gap in the terrain needs to be crossed without the use of a bridge, or where an object that is too heavy to move by a single bot is blocking the way.

Even though there has already been some development on this topic, it sounded like a fun and interesting topic that will allow me to learn and understand the fundamental principles of Swarm Robotics.

1.2. Analysis

Swarm Robotics has many open tasks that need to be better understood and advanced before it can truly be integrated, such as … (common problems AND REFERENCES)

Through my research, I found that Swarm Robotics has many areas that need advancement before a lot of the potential is realised. There was plenty of articles about ideas of Robots automatically building sky-scrapers, exploring Mars, or scavenging for earthquake survivors. However most of these ideas are in very early stages of development and still have a long way to go before they're realised. For now, the study of Swarm Robotics is limited to a relatively small group of robots doing relatively simple and mundane tasks. That is not to say that there are not systems creeping into practice. For example, BLA BLA BLA [REFERENCES]

My problem was simple enough to understand - get the bots to form a shape. This task would have been a lot easier if the system did not comply with the traditional rules and had an over-all controller class telling each bot what to do. However, this would have defeated the point of Swarm Robotics. As we have already discussed, this system would not be scalable in the real world and will not be able to have a go at any and every environment it was placed in. Also, the bots will probably be small, cheap, and relatively simple mechanically. This means that assuming every single bot in the system (potentially thousands) will need to work correctly. We cannot have 999 bots waiting for the 1000th to make a move before they all continue.

So, a system that has no leader is needed. Every bot will need to be exactly the same, and every bot will need to be observant to its immediate environment and nothing else. If done correctly, an external input that has not been foreseen or pre-calculated could move the bots in a random direction (maybe wind blowing the bots off course in the real world) and it will not make any difference. Each stage of the simulation the bots will act on what it immediately sees and that alone, calculating what to do on data that it has just gathered about its environment.

Ideally, the bots will do this without fail and be ready to act on literally any environment they are placed in without the need of calibration. Such a system will not be prone to any faults in the environment (i.e. a blockage in the planned path) or in any other bots (i.e. if a bot breaks down). Such a system will also be easily scalable. If the system has too many bots in, some can be taken away without worrying on their effect with the rest of the system. If there are not enough, bots can be added without the need for the others to even acknowledge the change in the amount of bots in the overall system.

With all this taken into account, my system should have by the end of it…

Ideally, the bots will not know everything in the environment, but the way I did it with the sensor class is…

Originally, I planned to have the bots form a circle around an object, just like THIS ARTICLE. However, after seeing that not many people has managed to do that I decided to simplify it…

Therefore, the final list stands as…

I feel these features are doable in the time frame I have because…

1.3. Current Systems

Many of these will have been mentioned in 1.2.

How are these systems different? Reference

1.4. Process

I knew that the process of the assignment was going to be important as I have never tackled anything close to this size before…

Before planning any of the structure or any code, or producing any UI designs, a full feature list was written. This feature list was a comprehensive outline containing all of the *stuff* the finished project will be able to do. After the list was completed, each feature was broken down into individual tasks. For example, one of the simulators features could be that the bots could be dragged and dropped by the mouse in order to move the bots around. This was considered quite an important feature and was to be completed as soon as possible. For this to be completed, tasks such as "produce a working GUI", and "Allow *something* to be drawn on the canvas" needed to be created.

After careful planning the tasks for each feature, the system could be chronologically planned out. Some features were given priority over others as some could not be started without completing others. With a full list of features, versions were planned. A version was to be a milestone for testing and backups, meaning if something during the development was to go wrong, an earlier version could be picked so and started from that point.

Each version of the simulator was saved into one directory (regularly backed up!) with a text document describing what was done and what was completed with each version, as well as when it was started and how long it took. Any bugs found and fixed were also listed here, as well as any changes to the system that was made.

At this point, the system was split up into versions, with each version having a list of features, and each feature having a list of tasks. Coding the project could now start at the beginning of the list. New tasks could be added as more content was thought of By doing it this way, it was easy to see progress and you never felt that you were lost with what to do next.

This way is a mixture between version control and feature driven…

A possible other methodology that I thought of was… because…

I am happy with the way I tackled the project, however if I were to do it again I would change…

2. Design

The design was intended so that once I started the code it would simply be a case of grinding out function and not having to thing about the system too hard. This was done by…

However, this wasn't strictly the case as I found due to… as I will discuss in a section below.

Overall, the design went well. I planned out the basic functionality and no major changes were needed. The UMLs turned out to be completely unefficient and ended up changing a lot…

Below is a section breakdown on what I did… notice this and think about that…

2.1. Overall Architecture

An early UML diagram was written, however this was later changed. As I started writing code, I found that the class had to change. Originally the bots had… and was later changed to … because…

The engine works by keeping everything ticking at a certain rate, while running at the highest framerate it can…

This approach of ticks can be used to describe a step system, as each bot gets one step each. The order of the steps depend on…

A custom listener was used to listen for everything, and attached to the GUIManager class. Instead of… Because…

I am happy with the overall architecture, however I feel the bot class turned out a little too big. This was because… To prevent this I could have…

2.2. Detailed Design

2.2.1. Original Design UML

Picture of original UML

2.2.2. Justification for Original UML

Talk about original UML

2.2.3. Final Design UML

Picture of the final UML used

2.2.4. Justification for Final UML

Talk about final UML, what changed and why.

2.3. User Interface Design

The graphical user interface is simple enough. During development there was a panel at the bottom containing buttons for testing. These buttons included functionality to move the bots (centre all, scatter all…), functionality to spawn in new bots, and functionality to remove bots already on the canvas.

The aspect ratio is… and was chosen because… Also, the size in scale is… (256x155? Or something…)

The window will need to not be resizable as the edges of the canvas will act as the boundaries of the bots. To resize, a scale variable was implemented that was multiplied with the aspect ratio when the program was compiled. This means changing the size of the window is left to the developers and the functionality will not be available to the users.

After most of the programs functionality was added, the button panel was removed and most of the functionality was transferred to the mouse. Text was superimposed to the top left of the canvas instructing the user on how to use the system which seemed a lot cleaner.

The colour scheme was chosen because….

Even though the colour of the actual bots does not matter they still show colours because… Justify the bots design and talk about

As the user interface panel was blocking the bottom of the canvas, there was an original issue where the bots would disappear…

Overall I am happy with the simplicity of the user interface…

2.4. Choice of Language and Libraries

Why did I use them? Why java? How would the project have been different with other Languages and Libraries?

3. Implementation

The position of the LEDs changed to be outside. The bots now go there and face the LEDs to make the lines straight

Edges of canvas, bots simply faced central.

Bots can only move forward and turn at specified rates at any one time. Why?

3.1. The Constructors

3.1.1. A Class' Constructor

3.1.2. Another Class' Constructor

3.1.3. And Another Class' Constructor…

3.2. The Functions

3.2.1. Virtual Force Field (VFF)

3.2.2. Directions

3.2.2.1. The Global Direction System

3.2.2.2. Directions to the Bot

3.2.2.3. Average Direction Finding

3.2.2.4. Turning to Desired Direction

3.2.2.5. Moving in Direction Facing

3.2.2.6. Looking at a [X, Y] Position

3.2.3. LEDs

3.2.3.1. Meaning of the LEDs

3.2.3.2. LED Charges

3.2.3.3. Updating the LEDs

3.2.3.4. Closing the Open Connections

3.2.4. Rendering

3.2.4.1. Bots

3.2.4.2. LEDs

3.2.5. Sensors

3.2.5.1. Getting Direction to [X, Y] Position

3.2.5.2. Getting Distance to [X, Y] Position

3.2.5.3. Finding Close Bots

3.3. The Behaviour of the Bots

Blab la bla…

3.4. The Mouse and User Actions

Blab la bla…

3.5. Comparing with the Specification

Blab la bla…

4. Testing

4.1. Overall Approach to Testing

Testing for this kind of project was mainly visual. Most of the time a project such as this would be correct if it appeared correct…

Testing tended to occur as the code was written…

Overall, I feel the continuous testing was a good thing and worked well because…

If I were to do the project again I would do … differently.

4.2. Automated Testing

Some stuff here.

4.3. User Interface Testing

As the user interface was so simple, it was almost completed before I asked some of my housemates and friends to use the system. None of them had a problem and all seemed to like it.

4.4. Stress Testing

Stress testing was a huge part of the development of the system…

After completing a large section of code, many bots were added to ensure that the framerate could keep up adequetelly (whats adequate?) at hugher tick rates.

I would spawn bots on top of each other and see the spread

When two bots cannot decide on which one them should go to the same LED

Two bots on one LED happened, so I changed…

Corner of the canvas is a problem because…

4.5. VFF Testing

Talk about the values tried, what happened?

Talk about values used, why did they work?

4.6. Acceptance Testing

After each version was completed, the project was compiled into a .jar and tested to see if it would run correctly, and that the spec for that section was met…

5. Critical Evaluation

5.1. General Evaluation

In retrospect, java was good enough for the complexity of the project. If I were to use C++ for example

5.2. Specification Evaluation

The requirements I gave myself needed to be changed as the project went on….

As I was learning whilst working on the project, I was not sure how I would do the end while doing the beginning… this lead to …

The minimum the project should have done was to form a simple shape, in this case a line. More shapes were eventually added… The way I wrote the code meant that adding the other shapes was easy as bots only connect and request a bot behind… more can be done (block shapes)

5.3. Process Evaluation

What did I do?

I am happy with the process I followed because… advantages and disadvantages

5.4. Product Evaluation

Although I am more than happy with what I was able to achieve in my project, there is always something that could be improved…

While the VFF system worked well when one LED was on and seemed to manage a lot of the problems I would have had face, it was not perfect… (more than one LED, tweaking values)

If I was forced to start the project again I would change…

5.5. Conclusion

To summarise, I have learned a lot on structure or large projects, time management, working to a product owner, coding, and of course Swarm Robotics (anything else?)

6. Appendices

6.1. Third-Party Code and Libraries

Engine was adapted from a games engine

StackOverflow was used for a lot of the maths

The VFF part was adapted from this website…

Certain maths was taken from forums and adapted…

Eclipse

6.2. Ethics Submission

6.3. Code Samples

7. Annotated Bibliography