Software Requirements Specification

for

Travelling SalesBee
(Nature Inspired Algorithm Visualisation)

Version 1.0 approved

Prepared by:
Christopher Lane
Melvyn Mathews
Bradley Rowe
Neil Farrington
Todd Waugh Ambridge

29 January 2016

Table of Contents

Revision History

Name	Date	Reason For Changes	V.
Initial Proposal	29/01/2016	An initial outline of the specification for our proposed program. The proposed program being a visualisation of graph algorithms.	1.0

1. Introduction

1.1 Purpose

The aim of our project is to build a program that can successfully represent a nature-inspired algorithmic solution to the travelling salesman problem (TSP). In nature, bees 'solve' the TSP while pollenating flowers. They track their routes by doing something along the following lines:

- **Stage 1 -- Naivety** Upon leaving the hive they follow a naive path by going to the closest flower. After checking if they have checked all the flowers, the bee will continue to fly to the next closest flower until it has reached them all. It will then return to the hive.
- **Stage 2 -- Experimentation** The bee then begins a number of 'experimental' runs of the algorithm where random paths are chosen and compared with the current best path. When a better path (shorter distance while visiting all flowers) is found, it is replaced in their mind as the best path. The average number of experimental runs in nature is 26.
- **Stage 3 -- Settling** After experimenting, the bee will 'settle' on their best path and continue to follow that one in the future.

To be able to represent this we will build an application that is split up into sections. These sections being a toolbox, a grid-map and a settings window (these sections will be explained in more detail later on). This product will be primarily aimed at a younger audience and will hopefully make it easier for them to get a grasp at fundamental graph algorithms used within computer science.

1.2 Document Conventions

Formatting:
Font: Times New Roman

Header 1	– Size 18
Header 2	– Size 14
Header 3	– Size 12, partially indented
Normal	– Size 12, indented
Header and Footer	– Size 10, bold and italicised

Keywords or terms:
See glossary.

1.3 Intended Audience and Reading Suggestions

When first learning about graph traversal algorithms, it can be overwhelming and difficult. Manually stepping through each part of an algorithm with real life examples is a good approach to understanding and learning these algorithms and we believe this program will achieve a similar effect. The specified target audience therefore will most likely be a younger crowd studying computer science/algorithms, with hopes to get a better understanding.

1.4 Product Scope

The purpose of our project is to build an application which acts as a teaching aid for learning the TSP (for a more detailed purpose, see 1.1). This project will take place over the next 10 weeks (ending week 11), since there is relatively large amount of work to do per week we have organised ourselves as follows:

- Each week we shall meet up on Mondays, Tuesdays and Thursdays.
  - Monday: 11:00 - 14:00
  - Tuesday: 11:00 - 14:00
  - Thursday: 12:00 - 14:30
- Each Thursday we shall also have a tutor meeting from 14:30-15:00.
- At these meetings we will discuss progress on the project and assign tasks to individuals or small groups, or work together as a whole team when necessary.

1.5 References

We shall reference using the following format:
Title, Author, Version Number, Date, Source/Location

How bumblebees find efficient routes without a GPS: Bees use trial and error to select optimal route; Queen Mary University of London; Date: 20/09/2016; http://www.sciencedaily.com/releases/2012/09/120920194612.htm (Accessed 27/01/2016)

2. Overall Description

2.1 Product Perspective

The product we are creating is a completely new, self-contained product that visualises a possible solution to the TSP. The algorithm we are visualising will be developed by us and inspired by nature; i.e. how bees solve the TSP.

2.2 Product Functions

The product must:

- Visualise the nature-inspired solution to the TSP.
- Allow the user to alter the state of the gridmap; i.e. adding bee objects and adding/removing nodes at chosen positions.
- Change the number of bees in the visualisation.
- Allow the user to change the speed of the visualisation.
- Allow the user to zoom in/out and scroll around the gridmap.
- View a table of all bee objects, showing their current best path and its cost.

2.3 User Classes and Characteristics

Our product could be used by one of two user classes: an **Inexperienced User** or an **Experienced User**.

User Class	**Description**	**Requirements**
Inexperienced /Young	The product is aimed primarily at children and people wishing to learn about graph search algorithms.	We must make sure that our system is simple enough for children and inexperienced users to use. We must make sure that the system is clear in what is happening at all times; showing appropriate information to the user.
Experienced	Most likely teachers of the aforementioned school-children, who wish to demonstrate the program to them. Could also be Computer Scientists who wish to see the TSP visualised.	We must make sure that our visualisation has enough tools to allow teachers to teach students about the TSP to the maximum extent. For example, we must make sure the visualisation can hold a lot of objects at one time.

2.4 Operating Environment

The system will be programmed in Java, meaning it can be run on all common operating systems. For our system, we do not expect there to be any additional hardware or software requirements other than a late Java Runtime Environment (JRE 8+).

2.5 Design and Implementation Constraints

Project Time Constraints: The deadline for completing our product is the week commencing 21st March.

Memory Constraints: We must make sure that we can have a high amount of bee objects and nodes on the gridmap, all operating at one time, with a low amount of memory. This is because we wish the program to run on low end systems, such as the ones that schools would have.

2.6 User Documentation

The users will be able to access helpful information and tutorials through the internal documentation, which will be accessible via the home menu. The documentation will be written for a younger audience and try to use as little jargon as possible (any jargon used will be defined).

2.7 Assumptions and Dependencies

We will assume that:

- The user will have basic knowledge of how to use a computer (keyboard and mouse).
- The computer being used will run one of the three common operating systems: Windows, Mac or Linux.
- The computer being used will be able to run Java and have Java already installed.
- The computer being used will have a reasonable amount of memory (2048 MB+).

3. External Interface Requirements

3.1 User Interfaces

'The Travelling SalesBee' software is an interactive TSP algorithm visualiser. The software should allow for the user to be able to navigate windows and adjust various parameters using the keyboard and mouse.

3.1.1 Main Menu View

The 'main menu' is the screen that the software will start on. The main menu will simple be a screen with the name of the software (Travelling SalesBee), some icons as decoration for example a SalesBee and navigation links.
The navigation links will allow the user to go to any screen in the program. The options in the menu will be **Algorithm Visualiser**, **Help** and **About**. The menu items will be clickable and will also be selectable via the keyboard arrow keys or representative letter keys e.g. 'H' for Help.

3.1.2 Algorithm Visualiser View

The algorithm visualiser view is the most complex since it contains the most elements and many of these must actually be interactive.
The main part of this view is the grid map, this is a grid that can have objects placed on it to be used in travelling salesman algorithms. The grid will have numbers along the top and left hand side to indicate the coordinates of each cell. Items placed on the grid will centre in the cell that they are placed in. The grid map will allow users to position and reposition items in cells. The grid will show a line that is representative of a given path. Since the grid map is the main element of this view, it will be the largest element and positioned in the centre of the screen.

The drag and drop menu will be positioned on the left hand side of the screen. The drag and drop menu will be able to list objects that can be placed on the grid map such as flowers or a hive as well as options to remove items. The menu will be scrollable for if not all the objects fit in the pane. Users will be able to click and hold on an item and then drag the item to where they want to place it on the grid map.

The 'Bees' pane will keep track of the bees that have been traversing the grid. The pane will be scrollable and will display the efficiency of each bee's most recent path when compared to an optimal path. Each bee entry in the pane will have a bee icon.

The options pane will allow the user to change various settings for the visualiser such as the speed of the algorithm visualisation or the zoom on the grid map. The options pane will likely contain many different control methods for the options such as sliders and drop down menus. The options pane should have a reset button to reset the visualiser to its original state.

Some things in the visualiser could be adjusted using shortcuts on a keyboard. For parts of the visualiser that can be controlled using the keyboard, their labels will have a letter underlined that is related to the shortcut key e.g. for adjusting the speed of the algorithm visualisation we could use S and + or - and have these shortcuts explained in full when hovering over the setting's label.

3.1.3 Help View

The 'help view' will display brief information on how to use the software. Help will be ordered by each view, for example all of the visualisation details will be grouped together. Keyboard shortcuts will be accompanied by an icon of each key to improve readability.

3.1.4 About View

The 'about view' will contain some short information about the software as well as a list of creators.

3.1.5 Main Menu Button

Each different view in the software will have a ‘Main Menu’ button at the bottom of the window, this button when clicked will take the user back to the main menu.
This button will be seen to be present on all views except the main menu.

3.1.6 Scaling

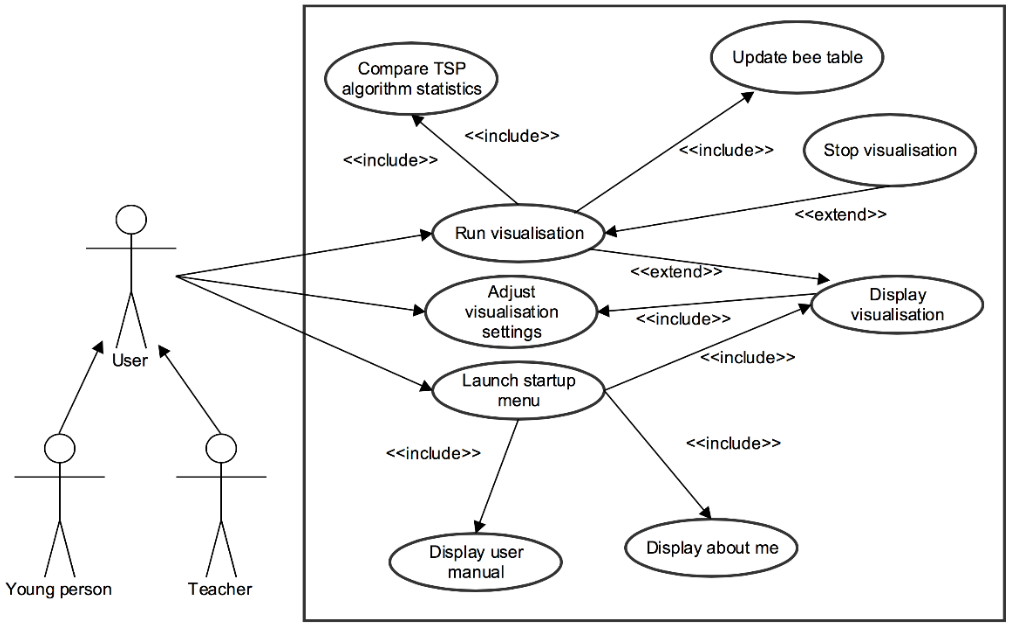
Window elements will inherit the scaling of the operating system, this ensures that the UI will match the rest of the operating system and look correct on all systems. This will prevent the issue of users with HiDPI displays having very small UI.

3.2 Hardware and Software Interfaces

Users should be able to control the program with both the keyboard and mouse where practical. No other hardware interfaces are required and the program will not interface with other software.

4. System Features

This section describes the features that are planned to be implemented in the software. Each feature listed has a section containing a short description of the feature as well a priority. The priorities for the features can be either High, Medium or Low. Features with a “**High**” priority are essential for the release of the software and should be implemented before features of lower priority. Features with “**Medium**” priority are desired but not necessary for the release of the software, these features should be implemented before features of low priority. Features listed as “**Low**” priority are not required by the program for release, these features will not be missed if not implemented but would add small improvements to functionality of the program if they are.



4.1 Nature Inspired Algorithm - High Priority

4.1.1 Description and Priority

High Priority

An algorithm must be created that can closely replicate the behaviour of bees when finding a route between flowers as an approach to solving the TSP.

4.1.2 Stimulus/Response Sequences

- The user requests for the TSP problem to be visualised with the given grid map and visualiser settings
  - The system should lock the grid map, preventing the user to edit the grid map during the current run of the algorithm visualisation
  - The system will then begin to instantiate bee GUI components onscreen that will each then begin to solve the TSP problem through generating routes and following them within the given grid map. The bees will independently find their own solution and new bee's will be created up until the number of bee's onscreen is equal to the selected amount prior to the running of the current visualisation. Eventually each bee should come to the same conclusion on the optimal route for the given grid map.
  - The system displays the final route onscreen along with other statistics.

4.1.3 Functional Requirements

REQ-1: The system must provide a solution for the TSP with a finite number of nodes that is similar to bee path finding.

4.2 Toolbox and Grid-map - High Priority

4.2.1 Description and Priority

High Priority

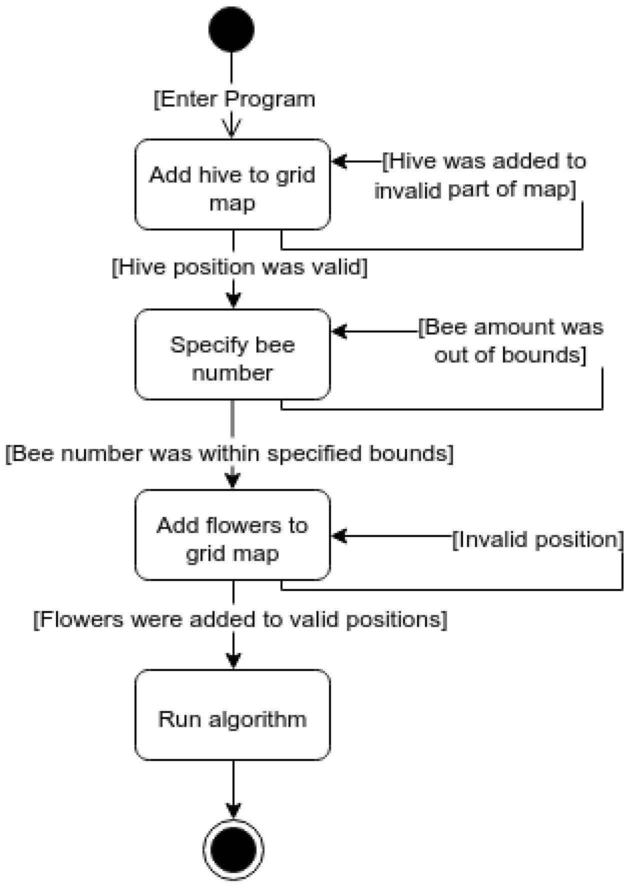
The visualiser must have a toolbox pane and a grid map pane. The grid map is a container for various objects that can be used for calculating paths with our algorithm. For example the grid map will be containing flower objects that will then be represented in a graph. The grid map must be able to show paths using lines and also support items being dragged and dropped on to it.
The toolbox is the container of all objects that can be placed on the grid map. Users should be able to scroll through items and drag and drop these items on to the grid. This toolbox pane should also contain controls for deleting items on the grid.

4.2.2 Stimulus/Response Sequences

- The user clicks an item in the toolbox, proceeds to drag it over to the grid map and drops it.

- The item is then placed accordingly on the grid map

- The user clicks an already placed object in the grid map, proceeds to drag it to another position and then drops it.
    - The item is then placed accordingly in it's new position in the grid map.


[Enter Program
Add hive to grid map
[Hive was added to invalid part of map]
[Hive position was valid]
Specify bee number
[Bee amount was out of bounds]
[Bee number was within specified bounds]
Add flowers to grid map
[Invalid position]
[Flowers were added to valid positions]
Run algorithm

4.2.3 Functional Requirements

REQ-1: The system must allow users to select tools/items from the toolbox.
REQ-2:.The system must allow for tools/items to be dragged and dropped on to the map.

4.3 Adjustable Visualiser Settings - High Priority

4.3.1 Description and Priority

High Priority

In order to allow for the visualisation to be interactive, the program will include the feature of adjusting the visualisation through a number of different GUI components. Some of these are yet TBD however the program will include components such as sliders to increase speed and zoom view of the grid map, a dropdown menu to increase/decrease the number of bees as well as other miscellaneous components such as radio buttons etc. to perform actions within the program. After a user settles on the grid map of nodes they want the algorithm to act on, the adjustable visualiser settings will come into play allowing users to continue to interact with the system even if they wish to re-run the same grid map. For instance a user can gain a better grasp of the algorithm if they can visualise it in different speeds or view routes more/less up-close. Therefore the varying of these visualiser settings by user input to adjustable GUI components allows for the interactivity to increase.

4.3.2 Stimulus/Response Sequences

- The user clicks the bee number drop down, proceeds to select a number and clicks that number
    - The number of bees in the next displayed visualisation should be equal to the new value selected within the number of bee's dropdown menu.

- The user clicks the speed slider toggle, proceeds to drag it and then lets go of it
    - The speed of the next displayed visualisation should be scaled by the scale factor relevant to the new position of the draggable toggle within the speed slider.

- The user clicks the zoom slider toggle, proceeds to drag it and then lets go of it
    - The zoom of the grid map within the next displayed visualisation should be scaled by the scale factor relevant to the new position of the draggable toggle within the zoom slider.

4.3.3 Functional Requirements

REQ-1: The system must allow for the speed of the visualiser to be changed. REQ-2: The system must allow for the window zoom to be adjusted. REQ-3: The system must allow for the number of bee's within the visualisation to be adjusted

4.4 Start Menu - Medium Priority

4.4.1 Description and Priority

Medium Priority

An interactive start menu is needed to enable the program to be well rounded. The start menu should include user access to; the visualisation itself, a user manual for help on using the program and an about me section including author details.

4.4.2 Stimulus/Response Sequences

- The user launches the executable program.
    - The system displays the splash screen.
    - The system will load the start menu onto the program window.
    - The system should then initiate an event loop to handle further user actions with the start menu GUI components as events.

- The user clicks an item within the start menu.
    - The system should handle the clicking event through exiting the start menu by displaying the clicked sub item's GUI window on screen.

4.4.3 Functional Requirements

REQ-1: The system upon launch should always load the start menu.
REQ-2: The system must allow users to use either the mouse or keyboard to interact with the menu options.
REQ-3: The system must provide a menu option to load the algorithm visualisation onto the screen.
REQ-4: The system must provide a menu option to load an about me page onto the screen.
REQ-5: The system must provide a menu option to load an embedded user manual onto the screen.

4.5 Table of Bee Objects - Medium Priority

4.5.1 Description and Priority

Medium Priority

There must be a table of bee objects in the visualiser view that is able to list details about bees such as the efficiency of their current path.

4.5.2 Stimulus/Response Sequences

- The user launches the visualisation.
  - The System updates the table with the number of rows equal to the number of bees in the current visualisation.
  - The System then continues to update column data for each bee.

4.5.3 Functional Requirements

REQ-1: The system must calculate and display the efficiency of each bee's path.
REQ-2: The system must highlight the row within the Bee table that currently has the most optimal path.

4.6 Comparison of Multiple TSP Algorithms - Low Priority

4.6.1 Description and Priority

Low Priority

The ability to compare other TSP algorithms will give a better picture in seeing the efficiency of the bee algorithm as a solution to TSP. This feature will therefore provide statistics to the user, specifying information which could be useful to identifying elements of TSP, such as the overall costs and the time taken to execute each algorithm.

4.6.2 Stimulus/Response Sequences

- The user launches the visualisation.
  - The System completes the visualisation
  - The System then displays the statistics of other TSP algorithms to compare efficiency with the bee inspired algorithm.

4.6.3 Functional Requirements

REQ-1: The system will allow you to select a different TSP algorithm for visualisation.
REQ-2: The system will allow you to compare the efficiency of different TSP algorithms.

5. Other Non-functional Requirements

5.1 Performance Requirements

Algorithm calculation and processing must be as efficient as possible, in order to reduce any waiting time before a simulation/GUI transition takes place. If high computation is necessary, and increases wait times (in excess of 0.5 seconds), a suitable interface and message must be shown to the user.

User-constructed algorithm models must scale reasonably in performance to a size which achieves the training and simulation aims of the software. Models may be restricted to this limit, thus achieving the training and simulation aims of the software without allowing users to instruct the software to compute a model beyond its processing abilities. The manner in which this restriction is implemented may vary, for example, there may be a limit to the amount of nodes that can be placed, or a limit to the amount of paths that may lead from/to a specific node. The exact restrictions and their values will be calculated during development and testing, however there should be a hard limit of not allowing a model of such a size that it cannot fit within the given display area of a standard screen size (1024px by 768px).

Software must transition naturally and seemingly instantaneously (in less than 0.1 second) between core menus and display screens, utilising multi-threading and background processing where necessary to achieve this.

5.2 Security Requirements

No personal data should be collected by the software.

5.3 Software Quality Attributes

The software must be able to run on common PC operating systems (Windows, OS X and common Linux distributions) without requiring a differently programmed/compiled package for each platform. Java is specifically designed to achieve this purpose.

The software must be able to run on systems with reasonable modern hardware specifications.

The software must handle any user errors that may occur, including incorrect/invalid input and unexpected actions, as well as any system runtime errors that could occur.

The software must provide an easy to use interface and package for the user, with minimal technical knowledge required to run and use the software.

The software must be rigorously tested utilizing both automatic and manual testing techniques. Despite rigorous testing, the software must also provide the user with an easy way of reporting any unexpected issues and exceptions that occur.

The software must not rely on any resources external to the original package in order to run, including any resource that must be obtained over a network connection. This will ensure 100% availability after the package is successfully downloaded/installed.

The software must be able to scale to the display it is being used on. If the window is resized, the contents must fill the space appropriately.

Basic configuration data for the software may be saved in an appropriate configuration file for the software. Any other data, including saved models or images, must be separate from the software, allowing multiple users to use the same installation interchangeably without hindrance.

The software must be able to run for long periods of time without being restarted. Large amounts of data and memory must not be used. Additional data, such as models, may be saved to the file system, rather than remaining in the memory.

The software must not require continuous maintenance and updates from the developer. If updates are required, the software should be easily replaceable with the updated version, and not involve a complicated upgrade process.

The user must agree to the End User License Agreement (if present) and any other applicable legislative material before being able to use the software.

The software must follow applicable software and computer usage legislation, including the Data Protection Act 1998 and the Computer Misuse Act 1990.

In order to easily comply with the Data Protection Act 1998, the software must not collect/process any personal user data.

6. Risk Analysis Document

The following is our team's risk analysis report for the project. We use the following ratings:

Probability - Low, Medium, High
Effect - Tolerable, Serious, Catastrophic
Type of Strategy - Contingency Plan, Minimisation Strategy, Avoidance Strategy

Risk	**Probability**	**Effect**	**Type of Strategy**	**Strategy**
Computer Malfunction	Low	Catastrophic	Contingency Plan	We shall use the lab computers instead.
Loss of power while working	Low	Catastrophic	Contingency Plan	We will make sure that we continually save, backup and commit our work to minimise loss of data.
Struggling to meet deadlines	Medium	Serious	Avoidance Strategy	We will make sure that we continually update each other with where we are in terms of our workload. If necessary to meet a deadline, we will also cut back our workload.
SVN Issues - Merging	Medium	Serious	Minimisation Strategy	We will make sure that we make regular, meaningful commits. We will also make sure that everyone knows which files they should be working on so that multiple people are not working on the same file at once.
Team Member Illness	Medium	Serious	Contingency Plan	We will split the work up into smaller more-manageable chunks and spread this work throughout the rest of the team.
Connection Issues while at Campus	High	Tolerable	Contingency Plan	Ensure work is backed up and make an organised commit plan in the team so that we don't have any merging problems later on.
SVN Issues - Connection	High	Tolerable	Contingency Plan	We will back up work to Google Docs or Dropbox and when possible, re-connected and attempt to re-commit to SVN.

7. Project Schedule

Identifier	**Task**	**Start Date**	**End Date**	**% Done**
1	Complete specification	24/01/16	29/01/16	100%
1.1	Risk analysis	24/01/16	28/01/16	100%
1.2	Project schedule	24/01/16	28/01/16	100%
2	Algorithm classes	01/02/16	06/02/16	0%
2.1	Bee methodology	01/02/16	03/02/16	0%
2.2	Additional algorithm methodologies	03/02/16	06/02/16	0%
3	Main visualisation GUI	01/02/16	06/02/16	0%
4	GUI functionality	09/02/16	21/02/16	0%
4.1	Interactive elements (drag & drop)	09/02/16	14/02/16	0%
4.2	Interface with main worker class	14/02/16	21/02/16	0%
5	Model generation	21/02/16	01/03/16	0%
6	Start menu	01/03/16	07/03/16	0%
7	Additional visualiser functionality	07/03/16	11/03/16	0%
8	Final testing and validation	11/03/16	18/03/16	0%
9	Final report	18/03/16	25/03/16	0%
10	Demo	18/03/16	25/03/16	0%

Appendix A: Glossary

Name	**Definition**
TBD	To Be Decided
GUI	Graphical User Interface
TSP	TSP
Flower	Node/Item that bees with search for/fly to.
Hive	Start/End Node.
Total Path Cost	The total amount of metres that it took for the bee to fully search the map and visit every node.
Grid Map	The map which shall take up the main part of our application window. The map shall be made up of a grid (hence the name). The grid-map will allow for tools to be dragged onto the window.
Solution to TSP	We acknowledge that there is no optimal algorithm for solving this problem, we intend for this to refer to what we are using to solve this problem within our application.

Appendix B: To Be Determined List

1. Adjustable settings for the visualiser.
2. Details shown for bees in the bee pane.
3. Instructions to be included in the ‘Help’ menu