

Software Engineering

# Requirements Specification

74120

105957

108069

108252

109195



University of Sussex

## Table of Contents

1	Preface .....	3
2	Introduction .....	3
2.1	Purpose .....	3
2.2	Scope .....	3
2.3	Overview .....	4
3	Glossary .....	5
4	User Requirements Definition .....	6
5	System Architecture .....	8
5.1	Product Perspective .....	8
5.2	User Characteristics .....	10
5.3	General Constraints.....	10
6	System Requirements Specification .....	11
6.1	External Interface Requirements .....	11
6.1.1	User Interface .....	11
6.2	Functional Requirements.....	17
7	System Model .....	21
8	System Evolution .....	23
8.1	Fundamental Assumptions.....	23
8.2	Anticipated Changes Due To Hardware Evolution .....	23
8.3	Changes User Needs and Optional Further Improvements .....	24
9	Appendix.....	25
9.1	Customer's Proposal .....	25
10	Index.....	26

## 1 Preface

This document reports the *Software Requirements Specification* (SRS) for the "Ant Game". The aim of this project is to provide an implementation of the "Ant Game", such that the implementation matches the customer specifications for the project as accurately as possible.

This document has been prepared in accordance with the IEEE Std 830-1998 standard, IEEE Recommended Practice for Software Requirements Specifications [IEEE 1998].

The intended audience of the following document are the stakeholders for the "Ant Game" project. This includes, but is not restricted to our customer, software developers, software testers, project managers, and all users.

## 2 Introduction

### 2.1 Purpose

The purpose of this document is to provide the complete software requirements for the "Ant Game" documenting the design and architecture decisions encountered, each with their corresponding description. The main motivation for this *Software Requirements Specification* (SRS) is for it to be used as a development guide for the initial version of the game and to serve as a reference to ensure that the designed system will meet the client's expectations.

### 2.2 Scope

The system proposed by the client is to be a competitive strategy game. The software will provide the back-end mechanics of the game along with a visual representation of the game's state and a graphical interface for user interaction.

A main obligation of the system is to provide matches between players. In a match each player of the game will provide a file (ant-brain) which will be used by the game to control their team of ants. For a match, two player's ants colonies are played against each other, the ants in each team will follow the instructions of the ant-brain until the match has ended. The team that has obtained the most points (from collecting food particles) at the end of the match is declared the winner.

Other main obligations of the system are to provide a tournament system between players, random generation of correctly formed ant-worlds and checking ant-brain and ant-world files for correctness.

Hence, there are seven main components to the system, the ant-brain file (provided by the user), the ant-brain parser, the ant-world generator, the ant-world file (provided by the system), the ant-world parser, the game's core (dealing with matches and tournaments) and a graphical interface representing the game's state.

## 2.3 Overview

Throughout the rest of the document the User Requirements Definition, System Architecture, System Requirements Specification, the System Model and the System Evolution will be covered. In the User Requirements Definition, information about the services the game provides for the user are described, as well as the non-functional system requirements. All of these are derived from what has been given by the customer, and any ambiguities have been resolved during further communications with the customer.

In the following section of the document a high level description of the system architecture is provided, along with an enumeration of the system modules and their respective functions. This includes an abstraction of the entire system showing general user interaction possibilities.

In section 6.1.2 the functional and non-functional requirements are described in a higher level of specificity. All requirements will be inferred from any information that has been gained from more detailed documentation provided by the customer, and will be focusing on implementation choices from an abstracted view.

In section 5 the relationship between system components, and the system with its environment will be described. This will be achieved using graphical models to support understanding.

Section 8 covers the system evolution, looking at any fundamental assumptions system plans have been based upon. Some of these are looking at what the customer may desire for the system in the future. This section will also cover how changing hardware will affect the system. Any optional/extra functionality that could be added to the system during development will also be covered here.

### 3 Glossary

Term	Definition
Ant	A controlled object which has its movements determined by the ant brain
Ant Brain	A simple finite state machine which represents what actions the ant will take on each step of the game. Has a maximum of 1000 states.
Ant Game	A game in which two Ant colonies (teams), each with their own Ant Brain, are simulated in an Ant World. A game consists of 300,000 rounds.
Ant World	A hexagonal grid of tiles in which the ants exist and interact.
FSA	Finite State Automata
FSM	Finite State Machine - a mathematical model of computation used to design both computer programs and sequential logic circuits. It is conceived as an abstract machine that can be in one of a finite number of states.
GUI	Graphical User Interface
IEEE	Institute of Electrical and Electronics Engineers
Player	A human who submits an Ant Brain
Round	A single <i>round</i> of the game consists of executing the next instruction for every ant - i.e. calling the <i>step</i> function for each ant in numerical order, from 0 up to the maximum <i>id</i> .
SRS	Software Requirement Specification
Tile	A tile is an atomic (indivisible) part of the Ant world which can either contain rocks, an anthill, food, markers, or be clear.

Table 1 Glossary Table

## 4 User Requirements Definition

For each match in the game, two players shall be played against each other. [appendix 1.a]

Each player shall provide an ant-brain in the form of a finite-state machine. [appendix 1.b]

The game shall support the uploading of ant-brain files into the game. Each player's ant-brain file corresponds to a single ant colony. [appendix 1.c]

Each individual ant's behaviour in the game shall be regulated by its respective team's uploaded ant-brain. [appendix 1.d]

In each tournament match, the ant colonies belonging to each player are placed into a ant-world, which was randomly generated for the tournament. [appendix 1.e]

Each world shall always contains two anthills, zero or more food sources, and three or more obstacles. [appendix 1.e]

To obtain points, a team's ant colony must find food contained within the ant-world and bring it back to their respective team's anthill. [appendix 1.f]

The team that has obtained the most points at the end of the match is declared the winner; in the case of equal points both players tie for that match. [appendix 1.k]

A match shall end when 300,000 rounds have been completed. [appendix 1.k]

A single round consists of every ant, in both teams, performing a single action.

The ants shall be able to make, remove or identify chemical markers of their own species. [appendix 1.g]

An ant must be able to identify, but not modify, chemical markers from other ant colonies. [appendix 1.h]

It shall be possible for ants of one colony to kill an ant of another colony by blocking at least five out of six of that ant's possible movements. [appendix 1.i]

An ant shall be converted into three pieces of food if it is the recipient of an attack. [appendix 1.j]

The game shall check if an uploaded ant-brain file is syntactically correct. [appendix 1.l]

The game shall check if an ant-world file is syntactically correct. [appendix 1.m]

The game shall also provide functionality that checks an ant-world against the rules of ant-world creation [appendix 1.m]

The game shall provide a graphical user interface (GUI). [appendix 1.n]

The GUI shall be able to display a match being played on a ant-world. [appendix 1.n]

The game shall contain functionality which randomly generates a valid ant-world. [appendix 1.o]

When playing a friendly match, the game shall allow players to choose an ant-world to play their match on, including the random generation option. [appendix 1.p]

The game shall simulate tournaments with two or more players [appendix 1.q]

In a tournament, each player plays every other player twice on every ant-world, once as the red team and once as the black team.[appendix 1.q]

A tournament will randomly generate multiple maps before starting, and those maps will be used for the duration of the tournament. [appendix 1.r]

The player with the highest number points after the conclusion of the tournament shall be declared the winner. [appendix 1.t]

For each match won in a tournament, the player gains two points, for each match drawn, the player gains one point, and for each match lost, the player gains zero points. [appendix 1.s]

The player with the highest number points after the conclusion of the tournament shall be declared the winner. [appendix 1.t]

If the tournament ends with a draw, drawing finalist will be put into another tournament to determine a clear winner. [appendix 1.u]

## 5 System Architecture

In this section, a general overview of the entire system will be presented. The way in which the system interacts with various other system modules will be introduced, as well as its basic functions. All of the assumptions/constraints made whilst writing this section will be described.

### 5.1 Product Perspective

The various components of the "Ant Game" can be represented as follows:

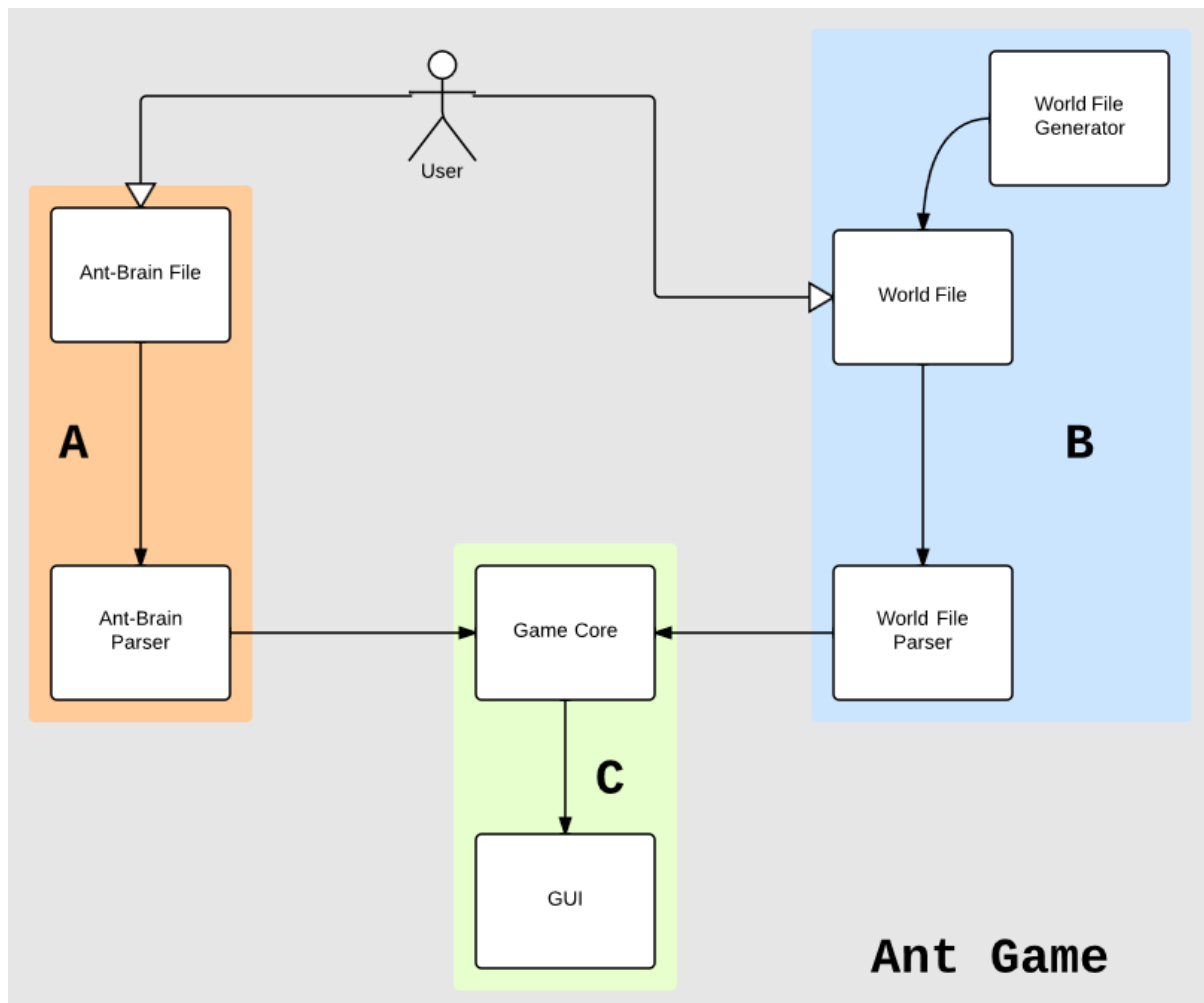


Figure 1 Context diagram of the Ant Game



As one can conclude from Figure 1, the various system modules are the:

- **A) Ant Brain Components:**
  - **Ant-Brain File:** The ant-brain file is simple Finite State Machine (FSM) that controls each ant in a team. The states are numbered from 0 to a maximum of 9999. Each line represents a state, the first line represents the first state, the second line represents the second state and so on; the file can therefore have a maximum of 10000 lines. In each state the next action to be taken and the next state to enter after executing the action is determined by an instruction.
  - **Ant-Brain Parser:** The ant-brain parser ensures that the ant-brain file supplied by the user is syntactically and semantically correct (the exact meaning of this will be explained in a later section). The parser also converts the instructions supplied in the file into a data type that can be used within the game.
- **B) World File Components:**
  - **World File:** A world file contains the specifications of a world on which a match will be played. A world has dimensions x & y, where x is the width and y is the height. The world file contains descriptions of the cartography and the geography (a cell can be either *clear* or *rocky*) of the match map.
  - **World File Generator:** The world-file generator has two main purposes, to generate ant-world files for use in tournament matches, and to generate ant-world files for use in non-tournament matches. When generating tournament worlds, the generator must create correctly formatted worlds that comply with the specified tournament rules. When generating non-tournament worlds, players can have control over some aspects of the map (number of food sources, anthill sizes, map size, etc.).
  - **World File Parser:** The world file parser will ensure that the supplied ant-world file is syntactically and semantically correct (the exact meaning of this will be explained in a later section). The parser also converts the specifications supplied in the file into a data type that can be used within the game.
- **C) Game:**
  - **Game-Core:** The Game Core component provides the underlying game mechanics for the entire Ant Game. The game core uses data provided by Ant-Brain Parser and the World File Parser to simulate the match being played. This component also handles matchmaking in tournaments and keeping track of player statistics.
  - **GUI:** The GUI provides a graphical representation of the current state of the game. For an ongoing match the interface will show the present state of the map and both ant teams, as well as relevant match statistics. The GUI also provides user interaction with the game's core allowing players to achieve tasks such as uploading their ant-brains, choosing ant-worlds and changing various settings.

## 5.2 User Characteristics

The Ant Game has only one main type of user, which is the player. The possible user-system interactions are as follows:

1. Uploading an ant-brain file
2. Uploading an ant-world file
3. Select the mode of play (tournament or non-tournament match)
4. Entering a name for their team
5. Altering game settings
6. Selecting a world to play non-tournament matches in

## 5.3 General Constraints

In order for the system to function correctly, given type of events should not be allowed to occur. The constraints that follow, describe operations that should not be permitted to happen during the execution of the game:

- A match should not be able to start until at least two players have uploaded their ant-brains into the game.
- In a non-tournament match, the game should not start until an ant-world has been chosen, independent of whether of that ant-world is randomly generated, pre-existing or user uploaded.
- A tournament can only have a single winner, if there were to be a draw, then the drawing players will be put into another tournament, this is repeated until a single winner is declared.
- For an ant-brain to be accepted into the game, all the instructions contained in the ant-brain files must conform to the specified language

## 6 System Requirements Specification

### 6.1 External Interface Requirements

This section provides a detailed description of all inputs into and outputs from the game, it also provides basic prototypes of the graphical user interface.

#### 6.1.1 User Interface

When the program is executed, the user should be presented with the window shown in figure 2 below. This screen provides the user with the options of either playing in tournament mode, or playing a non-tournament match.

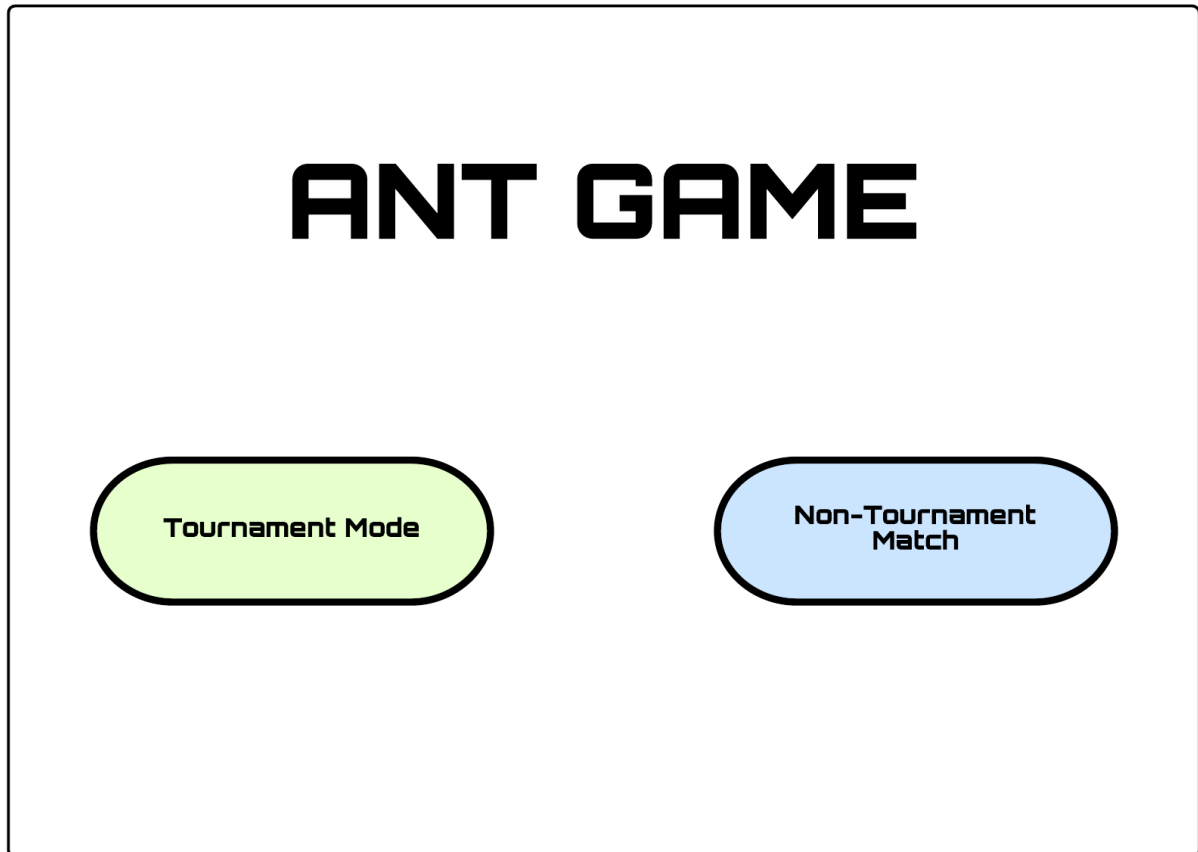


Figure 2 Welcome screen for the ant game

If the user clicks the "Tournament Mode" button, then the current interface will change to the interface shown in figure 3. If the user clicks the "Non-Tournament Match" button, then the current interface will change to the interface shown in figure 4.

**Tournament - Brain Selection**

Nickname:  Upload Ant-Brain +

Player 1	
Player 2	
Player 3	
Player 4	
Player 5	
Player 6	

Remove Selected Player START TOURNAMENT

Figure 3 Player and brain selection window for tournament matches.

Figure 3 above shows the screen for selecting the players to be included in a tournament. This interface provides the functionality to upload a player's ant-brain, give the players nicknames, remove a player from the list, and once satisfied with the players, start the tournament with the chosen players; these actions are described in more detail below:

1. When the "Upload Ant-Brain" button is clicked, the default file browser of the Operating System the user is using will open; this will allow the user to choose the intended ant-brain file.
2. When the "+" button is clicked, the validity of the nickname and ant-brain file will be checked, if both are valid, the player will be added to the list of players in the tournament.
3. To remove a player from the list, the user will first select a player from the list (figure 4 above shows "Player 4" selected), and then click the "Remove Selected Player".

Player 1	Player 2
Enter Nickname:	Enter Nickname:
<input type="text"/>	<input type="text"/>
<input type="button" value="Upload Ant Brain"/>	<input type="button" value="Upload Ant Brain"/>
<input type="button" value="GO"/>	

Figure 4 Initial window for a non-tournament match.

Presented in figure 4 above is the screen the user is prompted when "The non-tournament" match button is clicked on the initial screen. When the "Upload Ant-Brain" button is clicked, the default file browser of the Operating System the user is using will open; this will allow the user to choose the intended ant-brain file. When the "GO" button is clicked, the match will be stated; however it will only be responsive if both users have chosen a nickname and have uploaded their respective ant-brain file.

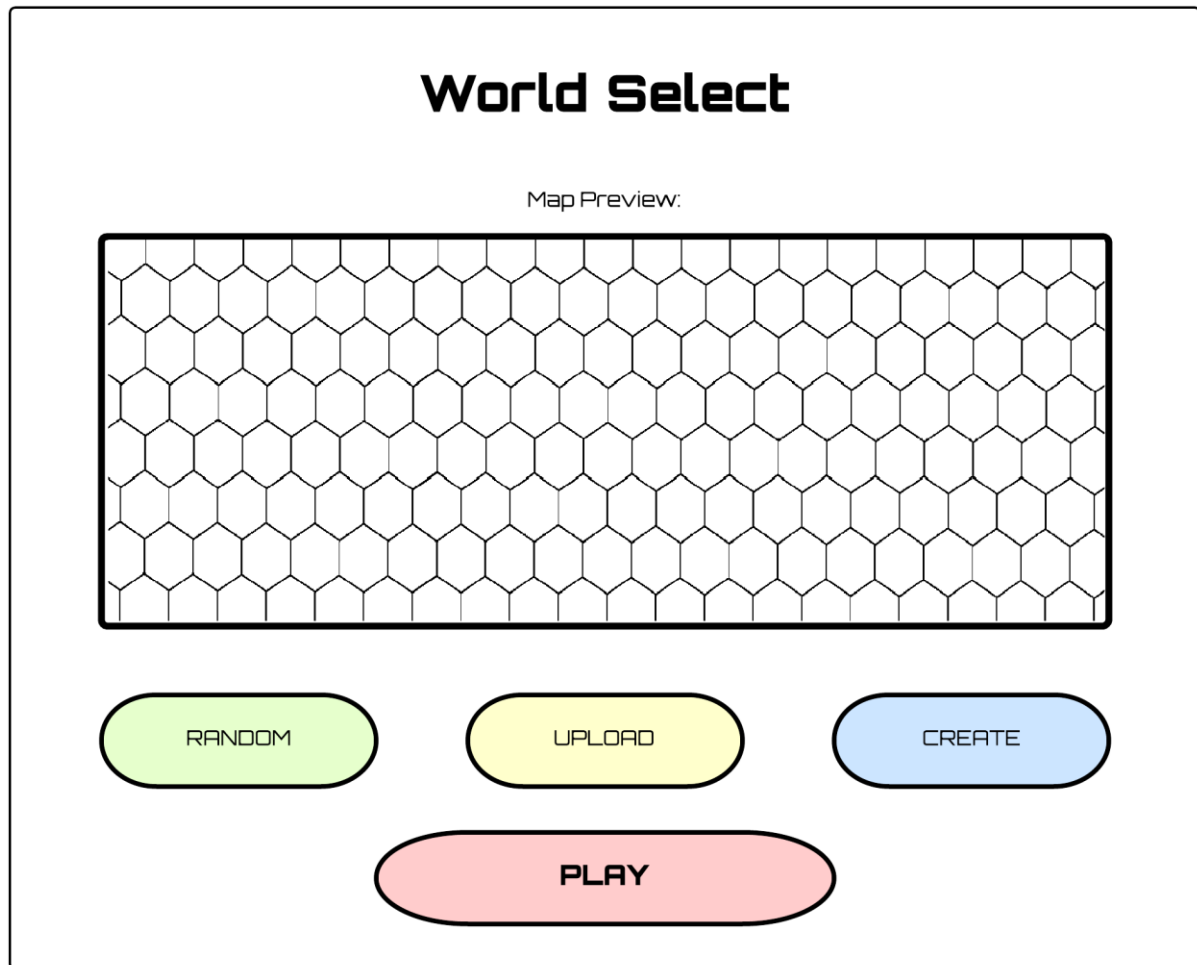




Figure 5 World selection screen for non-tournament matches.


For non-tournament matches, a choice of an ant-world is allowed. The screen in figure 5 shows the options provided to the user when selecting an ant-world to play on, these options include: randomly generating an ant-world, uploading an ant-world from a file (which will be checked for correctness), or creating an ant-world by tweaking various parameters. The screen which allows the creation of an ant world is shown in figure 6 below.

After a user has selected an ant-world, a preview of the world will be presented; this allows the user to ensure the ant-world is really the world they wanted before continuing to the match.

## Map Settings

Amount of Food:  Small Medium High

Number of Rocks:  Small Medium High

Size of Anthill:  Small Medium High

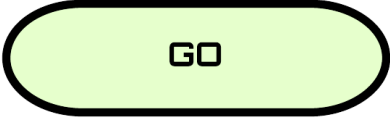


Figure 6 Settings window for a non-tournament match.

The game settings shown in Figure 6 relate to non-tournament matches, in which a variety of parameters for the ant-world are adjustable. This screen does not apply in tournament mode, as all ant-world used in a tournament will be randomly generated with a fixed anthill size, and fixed world size dimensions.

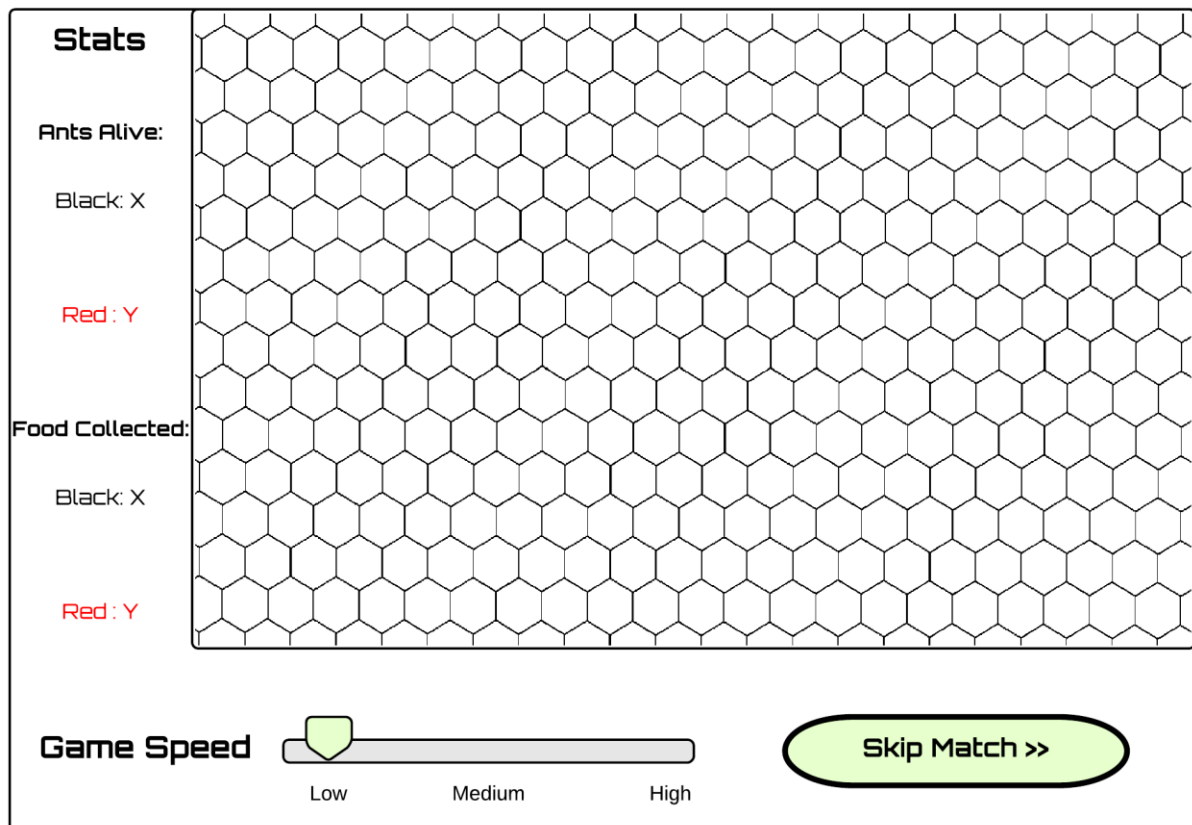


Figure 7 The screen of the current match being played, for both tournament and non-tournament matches.

Once either a tournament match or a non-tournament match is in play, the screen in figure 7 is displayed. The screen represents the current match in play, and provides a 'live' action simulation of the two ant colonies and their interactions. Although the players of the game are not permitted to physically control the ants, they do have the functionality to speed up or slow down the gameplay, or skip the gameplay of the match altogether; skipping the match will produce the same outcome as if the match would have been watched the whole way through.



## 6.2 Functional Requirements

This section includes the functional requirements that specify all the fundamental actions of the system.

### **Key:**

*REQ#: <The number of the requirement>*

*TITLE: <The overall function of the requirement>*

*INPUT: <An action that triggers the need for the requirement>*

*SOURCE: <How the INPUT will be given to the system>*

*OUTPUT: <The result the functional requirement will produce>*

REQ#: 1

TITLE: Convert Ant-Brain

INPUT: An ant-brain file

SOURCE: a user would upload the file through a graphical interface.

OUTPUT: The game's core now has access to the ant-brain file

REQ#: 2

TITLE: Ant-Brain Parsing

INPUT: An ant-brain file

SOURCE: The ant-brain would be obtained through the input of REQ #1

OUTPUT: A Boolean value, true if the uploaded ant-brain is a valid one, false otherwise

REQ#: 3

TITLE: Upload Ant-World

INPUT: A world file

SOURCE: a user would upload the file through a graphical interface.

OUTPUT: The game's core now has access to the world file

REQ#: 4

TITLE: Ant-World Parsing

INPUT: Ant world file

SOURCE: The world file would be obtained through the input of REQ#3

OUTPUT: A Boolean value or representation, true if the uploaded world file is a valid one, false otherwise

REQ#: 4.1

TITLE: World parsing, border check

INPUT: Ant world file

SOURCE: The world file would be obtained through the input of REQ#3

OUTPUT: Internal Boolean value to signify if the given world file has a perimeter of only rocky files, this REQ will be used as part of REQ#4

REQ#: 4.2

TITLE: World parsing, check for ant hills

INPUT: Ant world file

SOURCE: The world file would be obtained through random generation or REQ#3

OUTPUT: Internal Boolean value to signify if the given world file has 2 ant hills, and if it is a tournament map, then it checks that the ant hills are hexagons with sides of length 7.

REQ#: 4.3

TITLE: World parsing, size check for tournaments

INPUT: Ant world file

SOURCE: The world file would be obtained through random generation

OUTPUT: Internal Boolean value to signify if the given world file has a size of 150\*150 cells.

REQ#: 4.4

TITLE: World parsing, amount of rock tiles for tournaments

INPUT: Ant world file

SOURCE: The world file would be obtained through random generation

OUTPUT: Internal Boolean value to signify if the given world file has exactly 14 rocky cells.

REQ#: 4.5

TITLE: World parsing, food check for tournaments

INPUT: Ant world file

SOURCE: The world file would be obtained through random generation

OUTPUT: Internal Boolean value to signify if the given world file has exactly 11 blobs of food, each of which is a 5 by 5 rectangle, with each cell containing 5 food particles.

REQ#: 5

TITLE: Choice of match type

INPUT: Choice of whether Tournament mode or Non-Tournament mode

SOURCE: The user would make such a choice through the use of the provided GUI

OUTPUT: Depending on the choice made, the user will be prompted either one of the two GUIs previously presented. For Tournament mode, Figure 3 would be prompted, for non-tournament match Figure 4 would be prompted.

REQ#: 6

TITLE: Add players (Tournament Mode)

INPUT: A new player with corresponding ant-brain and nickname

SOURCE: The user would do such an action through the use of the provided GUI

OUTPUT: The game's core now has access to the created player, tournaments should theoretically allow any amount of players, but due to hardware limitations this may have to be capped.

REQ#: 7

TITLE: Add players (Non-Tournament mode)

INPUT: A new player with corresponding ant-brain and nickname

SOURCE: The user would do such an action through the use of the provided GUI

OUTPUT: The game's core now has access to the created player

REQ#: 8

TITLE: Deleting players (Tournament Mode)

INPUT: Choice of player to delete

SOURCE: The user would make such a choice through the use of the provided GUI

OUTPUT: The chosen player is removed from the tournament

REQ#: 9

TITLE: Random generation of world file (Non-Tournament mode)

INPUT: None

SOURCE: The game model (random world generator)

OUTPUT: After the user has uploaded the players to the game, figure 5 will appear to upload a world. At this point, the user should be able choose to randomly generate a world rather than uploading one. Figure 6 will then be shown to allow the user to have some input into how the world should be generated.

REQ#: 10

TITLE: Random generation of world files (Tournament Mode)

INPUT: A number of worlds that will be used in the tournament being set-up

SOURCE: The user

OUTPUT: Depending on the number chosen, n, an error message can be given for illegal arguments (if n is a negative number etc.), or the tournament can proceed, by generating the n random maps, which will all be well-formed, and correct to the specified parameters of a tournament world.

REQ#: 11

TITLE: Visualisation of ant-worlds

INPUT: Either an uploaded world file, or a randomly generated one from REQ#9 or REQ#10

SOURCE: The user/game core.

OUTPUT: A visual representation of the supplied world, using a hexagon grid system, and a suitable colour scheme to be able to identify features on the map.

REQ#: 12

TITLE: Tournament play format

INPUT: None

SOURCE: None

OUTPUT: When a tournament has been set-up, and the game has been told to proceed, every possible pairing of ant-brains should be played on every map, twice. The winning player gets one point and the loser gets 0. The scores are tallied to the scoreboard after every match.

REQ#: 13

TITLE: Rounds per match

INPUT: None

SOURCE: None

OUTPUT: For every match played, regardless of whether it is part of a tournament or a single match, there has to be 300000 rounds played.

REQ#: 14

TITLE: Results (Non-tournament)

INPUT: Two ant-brains and an ant-world

SOURCE: The user

OUTPUT: The results of the game, showing which player won. This can also include other stats such as how each of the brains performed in terms of food collected, ants killed etc. but the main focus is telling the players who won.

REQ#: 15

TITLE: Results (Tournament)

INPUT: Two ant brains from the tournament, and a map to be played on

SOURCE: Game core, but originally brains will be from the user

OUTPUT: A result screen as described in REQ# 14, but instead of showing an overall winner, the game will display a winner for the match being played, and will then continue on to the next match.

REQ#: 16

TITLE: Final results (Tournament)

INPUT: The set-up of a tournament

SOURCE: The user

OUTPUT: A final scoreboard showing scores for each player, and an overall winner.

## 7 System Model

This section contains diagrams representing the different ways in which the system can be utilised by the user. The first diagram represents the initial use cases; these are the possible actions that can be performed once the game has been started. The initial choice for the user is either to play a single match or a tournament; these both result in the user being required to make additional choices such as which ant brain to upload etc.

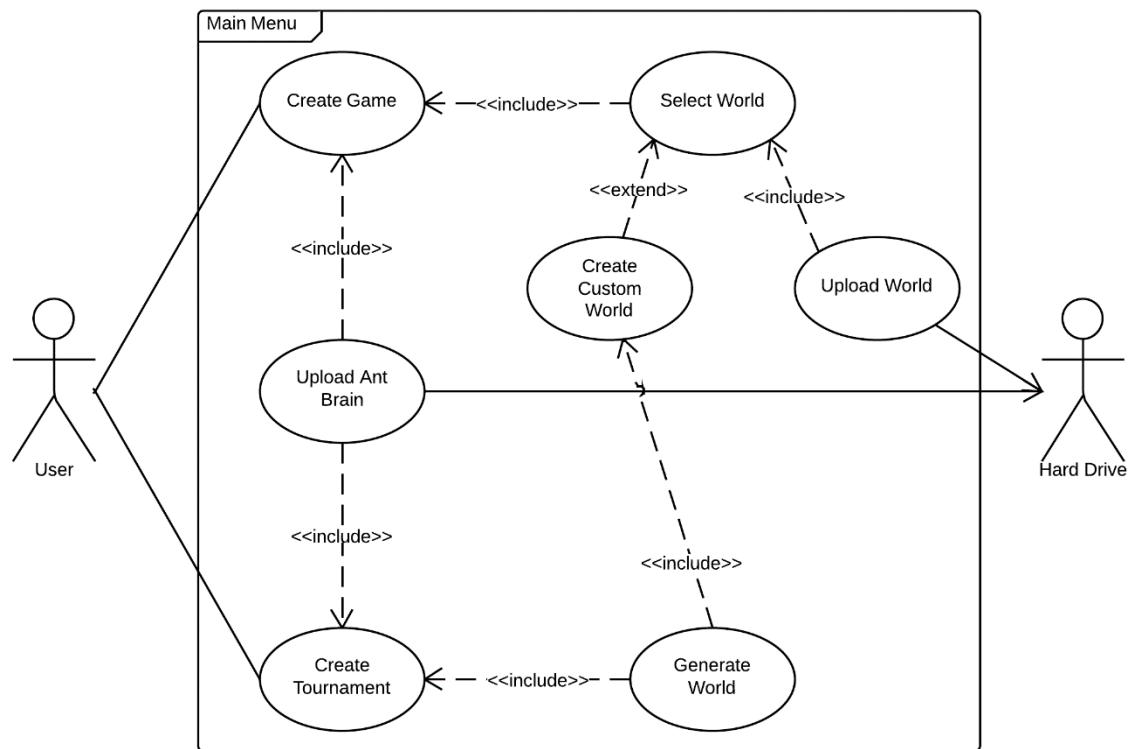


Figure 8 The initial use case diagram for the ant game.

Use case: Create Game.  
Actors: User, Hard Drive.  
Description: The user may create a single game with two ant brains and a single world.  
Data: One world file; either generated or loaded from file and two ant brains, one for each player. These brains will be loaded from file on the Hard drive.  
Stimulus: User interacted with the appropriate button on the GUI.  
Response: GUI displaying the Match Screen once the ant brain and world have been chosen.  
Comments: The user may customise world generation or upload their own world

Use case: Create Tournament.  
Actors: User, Hard Drive.  
Description: The user may create a tournament following the rules set out in the customer requirements.  
Data: One uploaded ant brain per player. These brains will be loaded from file on the Hard drive.  
Stimulus: User interacted with the appropriate button on the GUI.  
Response: GUI displaying the Match Screen once the ant brains and world have been paired to create matches.  
Comments: The user can decide both how many ant brains and worlds are used.

Once the user has chosen either of the options on the previous screen, they will inevitably reach a screen displaying a match. In this screen, the user will be able to change the speed at which the game simulates the match, zoom in and out to view the match at a different resolution. The user should also be able to skip to the end of a match as a long simulation time would make tournaments very lengthy.

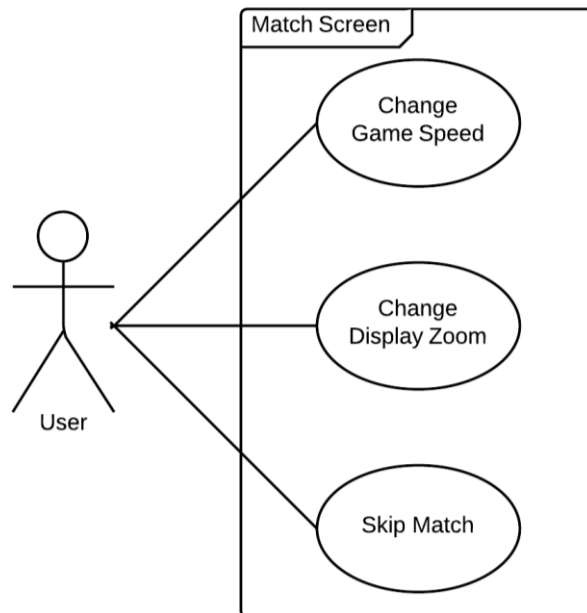


Figure 9 The use case diagram for when a match is in progress.

Use case: Change Game Speed.

Actors: User.

Description: The User may change the speed of the current game.

Stimulus: User interacted with the appropriate button on the HUD.

Response: Match Screen simulates the game at a faster rate.

Use case: Change Display Zoom.

Actors: User.

Description: The user may zoom in and out of the current match.

Stimulus: User interacted with the appropriate button on the HUD.

Response: Match Screen displays the tiles at a larger resolution.

Use case: Skip Match.

Actors: User.

Description: The User may skip to the end of a match to quickly display the results.

Stimulus: User interacted with the appropriate button on the HUD.

Response: Match Screen finishes the game and skips to the Results Screen.

## 8 System Evolution

This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.

### 8.1 Fundamental Assumptions

- The user's workstation will be running Java 7 or above.
- The user's workstation will have at least the following specification:
  - 1GB RAM
  - 50MB Free hard drive drive
  - 1GHz Processor
- The user's workstation monitor will have a resolution of 640 \* 400 or similar.

### 8.2 Anticipated Changes Due To Hardware Evolution

In 'changing user needs' (X.3.5), the customer may request the option for games to be played over a network; In this case, the system would require a server running all separate games through it and connecting clients. The decision of where the game will be physically processed is an important decision; If the processing is carried out server-side, the customer would require equipment to be able to host many games simultaneously, and depending on the number of games being played, expensive equipment may be required. A smarter alternative is for the server to just handle the transfer of ant-brains from one user to another, tell the receiving user's computer to run the game, and then broadcast the results back to the users. This option is very viable because the game is not time critical, both users don't need to watch it in synced real-time, unlike many other modern games.

### 8.3 Changes User Needs and Optional Further Improvements

1. The customer may find that creating an ant-brain is too syntactically challenging for the users, and therefore may require a graphical interface for creating and editing them. This 'text editor' GUI may highlight the syntax, alert the user to syntactic errors and guide a user through the process of creating an ant brain.
2. The customer may want a feature that allows users to create ant-worlds through a GUI to give users the ability to have greater visualisation for their desired world; this could mean that an option is available in the game which allows for a user to create an ant-world with on screen assistance.
3. The customer may require the game to have multiple ant-brains battling each other at one time on a single ant world. In this case the game would need to be restructured to accommodate this, and the system specifications may require a higher benchmark. In addition to this, if the customer wants this change to follow through into the tournament mode, the format of tournaments would need to be altered to rebalance the fairness of the game.
4. The customer may want the game to feature multiple ant-brains on each team. The format of the game could be unchanged, but when an ant is generated, it will be randomly assigned a brain to follow, from its team's available brains. The user may decide to add a ratio of roughly how many ants should be generated using each of their ant-brains. This allows for situations where the user can write 'fighter' brains and 'gatherer' brains etc.
5. The customer may want the game to be played through a network. This could be implemented through then running the game on a website, and having a server doing the work, or having the connection made through the original GUI for the game. For this change to be implemented, additional hardware described section 8.2 above will need to be added.
6. The customer may want the ability to pause a running game, whilst it's in play. This would give the players greater control over the game, and could be very beneficial to the final user. It could be implemented through a simple button or switch on the match screen on the GUI. How difficult it would be to implement into the workings of the game will have to be assessed in design documentation.
7. The customer may want the game for the speed of a match to be adjustable. The players may want to see very small incremental steps on the screen, or a very fast modelling of the game to just see the results. This would give the players greater control over a match, and allow the players to battle at a pace they want. This could be implemented through a slider/toggles on the GUI, to dynamically alter the speed, or a parameter before the match begins, to statically set a speed.



## 9 Appendix

### 9.1 Customer's Proposal

**(a)** The game proposed by the customer is a competitive two-player strategy game. **(b)** Both players design an ant-brain (a simple finite-state machine). **(c)** To run the game, the brains are uploaded into an ant world with two ant colonies, one for each player. **(d)** The ant world will then simulate the behaviour of both kinds of ants, using the ant-brains supplied by the players. **(e)** More precisely, in a match, the two species of ants are placed in a random world containing two anthills, some food sources, and several obstacles. **(f)** They must explore the world, find food and bring it back to their anthill. **(g)** Ants can communicate or leave trails by means of chemical markers. **(h)** Each species of ants can sense (with limited capabilities), but not modify, the markers of the other species. **(i)** Ants can also attack the ants of the other species by surrounding them. **(j)** Ants that die as a result of an attack become food. **(k)** The match is won by the species with the most food in its anthill at the end of 300000 rounds.

Your task is to supply the following:

- **(l)** A program that checks if an ant-brain supplied by a player is syntactically well-formed.
- **(m)** A program that checks if a given description of an ant world is syntactically well-formed and meets the requirements for ant worlds used in tournaments.
- **(n)** A program that can visualise a given ant world.
- **(o)** A program that allows the generation of random but well-formed ant worlds.
- **(p)** A program that allows two players to play: i.e. enables two players to upload their ant-brains and choose an ant-world, and then runs the game in the ant world, taking statistics and determines the winner of the game.
- **(q)** A program that allows to play tournaments, where an arbitrary number of players can upload ant-brains, who are all paired up to play against each other. The overall tournament winner is the ant brain that wins the most individual games.

**(r)** During the tournament, each pair of submissions is pitted against each other twice on each of the contest worlds - once with the first submission playing red and the second black, and once with the first playing black and the second red. **(s)** A submission gains 2 points for each game it wins, and 1 point for each draw. **(t)** The submission with the most points wins the tournament. The number of the worlds used during the tournament is unspecified, but will be large enough for determining a clear winner. **(u)** If there is nevertheless no clear winner, the tournament is repeated with a certain number of finalist submissions.

## 10 Index

### A

Ant .....	5
Ant Brain .....	5
Ant Brain Components:.....	10
ant brain instructions .....	11
Ant Game .....	3, 5, 9, 10, 11
Ant World.....	5
ant-brain.....	3
Ant-Brain File .....	10
ant-brain parser.....	3, 10
Ant-Brain Parser .....	10
anthill.....	7
Anticipated Changes .....	21
ant-world generator .....	3
ant-world parameters .....	16
ant-world parser .....	3

### C

chemical markers .....	7
client .....	3
colony.....	7
current match screen .....	17
customer .....	21

### E

External Interface Requirements .....	12
---------------------------------------	----

### F

file browser .....	14
FSA.....	5
FSM .....	5
Functional Requirements .....	17
fundamental assumptions .....	21
Fundamental Assumptions .....	21
Further Improvements .....	22

### G

game speed.....	17
Game-Core .....	10
General Constraints.....	11
Glossary .....	5
Graphical user interface.....	10
GUI.....	5
GUI prototypes .....	12

### I

IEEE .....	5
IEEE Standard.....	3
Introduction .....	3

### M

map settings .....	16
match .....	7
multiple ant-brains .....	22

### N

network gameplay .....	21
non-tournament .....	11

### P

player .....	11
Player .....	5
Player and brain selection window.....	13
Product Perspective .....	9
program execution.....	12
Project Overview .....	4
Purpose .....	3

### R

remove player .....	13
Round.....	5
rounds .....	7

### S

Scope .....	3
simulation.....	17
skip gameplay .....	17
Software Requirements Specification.....	3
SRS .....	3, 5
syntax .....	22
System Architecture.....	9
system designers.....	21
System Evolution.....	21
System Model .....	20
system module interaction .....	9
System Requirements Specification .....	12

### T

Tile .....	6
tournament .3, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20, 22	

### U

upload ant brain .....	13
Use case diagrams: .....	20
User Characteristics .....	11
User Interface.....	12
User Requirements Definition.....	7

### W

Welcome screen .....	12
window for non-tournament .....	14
winner.....	7
World File .....	10
World File Components.....	10
World File Generator.....	10
World File Parser.....	10
World selection .....	15
world size dimensions.....	16