Software Engineering Requirements

Team 4

May 3, 2016

Contents

1 Introduction

The game proposed by the customer is a two player competitive strategy game. Both players design an ant-brain (a finite-state machine). The brains are uploaded into an ant world with two ant colonies, one for each player. In a match, the two player's ants are placed in a random world containing two anthills, some food sources, and several obstacles. They must explore the world, find food, and bring it back to their own ant hill. Ants can communicate or leave trails by means of chemical markers. Each species of ant can sense, but not modify, the markers of the other species. Ants can attack the ants of another species by surrounding them. Ants that dies as a result of an attack become food. The match is won by the species with the most food at its anthill at the end of 300000 rounds.

Broadly, the requirements of the system are:

- The program shall check that that a player-supplied ant-brain is well formed.
- The program shall check that a supplied ant-world is well formed, and meets the requirements for ant worlds used in tournaments.
- A program that can visualise an ant world.
- A program that allows the generation random but well formed ant worlds.
- The program shall allow two players to play, i.e. enable two players to upload their ant-brains and choose an ant-world, and then run the game in that ant-world.
- provided for the competition at the end of the course between all the teams.

2 Glossary

Ant Brains

The ant brain is defined by a finite state machine. The specification of the state machine follows:

state machine here

Geometry

The ant-world on which a match is played is a hexagonal grid. Ants may move to the six adjacent cells.

Biology

There are two colors of ants: Red and Black. These represent the two species of the opposing player's ant armies.

Geography

Each cell in the world is either clear or rocky. Rocky cells are not very interesting, they just impede movement. At any given moment, a clear cell may contain:

- At most one ant.
- Some non-negative number of food particles.
- One set of markers for each of the two ant colors.

Another important feature of the world's geography is anthills. Some set of clear cells is designated as comprising the anthill of one player. The anthills of the two players must be disjoint.

Cartography

Worlds files have the following format:

- The first line contains a single integer representing the size of the world in the *x* dimension.
- The second line contains a single integer representing the size of the world in the *y* dimension.
- The rest of the file contains *y* lines, each containing *x* one character cell specifiers.

The possible cell specifiers are:

`#`	rocky cell
`x`	clear cell (containing nothing interesting)
`+`	red anthill cell
`-`	black anthill cell
`1` to `9`	clear cell containing the given number of food particles

During initialization, each anthill cell will be populated with an ant of the appropriate color.

Chemistry

Each ant can place and sense six different kinds of chemical markers, numbered 0 through 5. The markers for the two colors of ants are completely separate. Markers persist until they are explicitly cleared. All markers in all cells are initially clear.

Phenomenology

The *Sense* instruction in the ant control language can be used to check an number of conditions:

Cell Condition	**Description**
Friend	Cell contains ant of the same color
Foe	Cell contains ant of the other color
Friend With Food	Cell contains ant of the same color carrying food
Foe With Food	Cell contains ant of the other color carrying food
Food	Cell contains food (not being carried by an ant)
Rock	Cell is rocky
Marker	Cell is marked with a marker of this ant's color
Foe Marker	Cell is marked with some marker of the other color
Home	Cell belongs to this ant's anthill
Foe Home	Cell belongs to the other team's anthill

Note that the *Marker* instruction is parameterized by the kind of chemical marker to be sensed; an ant can test for the presense of just one of its own markers in a single instruction.

Neurology

The brain of each ant consists of a finite state machine. States are numbered beginning from 0, up to a possible maximum of 9999. In each state, the next action to be taken and the next state to enter after executing the action are determined by one of the following instructions:

⟨*instruction*⟩	::=	Sense ⟨*sensedir*⟩ ⟨*st*⟩ ⟨*st*⟩ ⟨*cond*⟩
	|	Mark ⟨*i*⟩ ⟨*st*⟩
	|	Unmark ⟨*i*⟩ ⟨*st*⟩
	|	PickUp ⟨*st*⟩ ⟨*st*⟩
	|	Drop ⟨*st*⟩
	|	Turn ⟨*lr*⟩ ⟨*st*⟩
	|	Move ⟨*st*⟩ ⟨*st*⟩
	|	Flip ⟨*p*⟩ ⟨*st*⟩ ⟨*st*⟩

⟨*sensedir*⟩	::=	Here
	|	Ahead
	|	LeftAhead
	|	RightAhead

```
⟨cond⟩              ::=  Friend
                      |  Foe
                      |  FriendWithFood
                      |  FoeWithFood
                      |  Food
                      |  Rock
                      |  Marker ⟨i⟩
                      |  FoeMarker
                      |  Home
                      |  FoeHome

⟨lr⟩                ::=  Left | Right

⟨st⟩                ::=  0 | 1 | 2 | . . . | 9999

⟨i⟩                 ::=  0 | 1 | 2 | 3 | 4 | 5

⟨p⟩                 ::=  1 | 2 | 3 | . . .
```

Note the syntax of the *Marker* condition.

Ant Combat

Besides gathering food, ants may engage in combat with other ants. The rules are simple:

- If ant ever finds itself adjacent to 5 (or 6) ants of the other species, it dies.
- When an ant dies, it turns into 4 particles of food.

Random Number Generation

Kinetics

This section describes what happens when an ant actually executes an instruction.

Game Play and Scoring

A complete *game* is played as follows:

- The world, and the brains of the two ant species are loaded from the files described above.
- Each cell in both the team anthills is populated by an ant of the respective color. All ants start in state 0, facing east (direction 0), and not carrying any food.
- All ants (regardless of colors) are assigned their identities in top-to-bottom, left-to-right order– i.e. the topmost-leftmost ant gets identity 0; the next ant to its right gets identity 1, and so on.
- The game is played for 300000 complete rounds. Each ant therefore gets to execute 300000 instructions.
- At the end, the amount of food particles in each team's anthill is counted. The team with the most food particles in the anthill wins. Note that if an ant is carrying food and standing on an anthill then that particle of food does *not* count towards the total amount of food in that anthill.

Contest Mechanics and Contest Worlds

The worlds used for judging shall be randomly generated, but according to the following rules:

- The dimensions of the world are always 150 by 150 cells.
- The cells on the perimeter are always rocky.

- Every world contains the same elements, of particular shapes: 2 anthills, 14 rocks, and 11 blobs of food. The anthills are hexagons with sides on length 7. A blob of food is always a 5 by 5 rectangle, with each cell containing 5 food particles.
- The positions and orientations of the elements are chosen randomly, subject to the constraint that there is always at least one empty cell between non-food elements. Also, no elements overlap.

Judging

During the tournament, each pair of submissions shall be pitted against each other twice on each of the contest worlds– once with the first submission playing black and the second black, and once with the first playing black and the second playing re. A submission gains 2 points for each game it wins, and one point for each draw. The submission with the most points wins the tournament. The number of worlds used in the tournament is unspecified, but will be large enough to determine a clear winner. If there is nevertheless no clear winner, the tournament is repeated with a certain number of finalist submissions.

3 User Requirements

A broad outline of the requirements that the system shall provide is outlined below:

3.1 Functional Requirements

This section describes the services that the system shall provide and how the system will react to particular inputs, and how the system will react in particular situations.

3.1.1 Ant Brain File Validation

The program shall check that an ant brain supplied by the player is well formed according to the rules laid out in section 2.

Importance	High
Inputs	A text file describing a finite state machine according to the grammar in section 2.
Source	The user of the program.
Outputs	If the format of the file is valid, then an ant brain will be constructed according to the rules in the file. If the format is not valid, then a parsing error will be reported allowing the correction of the ant-brain file.
Prerequisites	The program has been initialised.
Post Condition	A valid ant brain is constructed or the user is prompted to correct the format of the supplied ant brain file.

3.1.2 World File Validation

The program shall validate a submitted ant-world file. The format of the world is described in section 2. The world described in the file must also conform to the contest world specifications in section 2.

Importance	Moderate
Inputs	A text file that describes an ant-world according to the rules laid out above.
Source	The user of the program.
Outputs	If the format of the file is valid and the world described adheres to the format and rules laid out above then an ant-world will be constructed that can be used in the game. If the file describes a world which is malformed then the user will be prompted to correct mistake in the ant-world file.
Prerequisites	The program has been initialised.
Post Condition	The ant-world is constructed and can be used in tournaments, or the user is prompted to fix the error in the ant-world and re-validate and no world is constructed.

3.1.3 Ant World Visualisation

The program shall provide a visualisation of the ant-world.

The game progress should be displayed as well as a level of interactivity (pause the game, etc).

Importance	High
Inputs	The current state of the ant game in progress.
Source	None.
Outputs	An interactive visual representation of the ant-game in progress. The visualisation should represent all the cell types as well as each individual ant of both colors. There should be some way for the user to pause or interrupt the game in progress.
Prerequisites	The program must be initalised and both players must have uploaded their valid ant-brains. A valid world file must have also been uploaded.
Post Condition	A visualisation of the game in progress is provided.

3.1.4 Ant World Generation

The program shall generated valid ant worlds.

Importance	Low
Inputs	None.
Source	None.
Outputs	A random valid ant-world that conforms to the specification of the contestworlds (section 2).
Prerequisites	The program has been initialised.
Post Condition	A random and valid ant-world is generated that may be used in competitions.

3.1.5 Playing the Game

The program shall allow for two players to compete against each other.

Importance	Essential
Inputs	None.
Source	Two users.
Outputs	The winner of the game according to the rules laid out in section 2.
Prerequisites	Two players must upload valid ant-brains and a valid world must be provided.
Post Condition	The users of the program are prompted with the winning player.

3.1.6 Playing a Tournament

The program shall allow an arbitrary number of players to upload ant-brains and complete. Players shall be paired and the winner of most games will win the tournament.

Importance	Essential
Inputs	An arbitrary number of ant-brains supplied by the user.
Source	Users of the program.
Outputs	The winner of the tournament according to the rules laid out in section 2.
Prerequisites	An arbitrary number of valid ant-brains must be provided and a number of contest worlds must be provided.
Post Condition	The users of the program are prompted with the player that has won the tournament.

3.1.7 Ant Brain

A "novel" ant-brain must be provided for the competition between the teams.

Importance	Low
Inputs	An valid ant brain file.
Source	The creators of the system.
Outputs	An ant-brain that can play competitively in a tournament.
Prerequisites	None.
Post Condition	An ant-brain is created that shall play in the tournament.

4 System Requirements

4.1 Functional Requirements

4.1.1 Uploading an Ant-Brain

The system shall prompt the user to upload two appropriate ant-brain files. The user shall then supply the path of their ant-brain files. If the ant-brain files are well formatted, then the program shall proceed. If they are not valid, then the users shall be alerted with the location of the error and prompted to re-upload the files.

Importance	High
Inputs	Paths to two ant-brain files.
Source	The user of the system.
Outputs	The two ant-brains shall be created based on the uploaded files.
Prerequisites	Two users to the program.
Post Condition	The two ant-brains shall be created based on the uploaded files.

4.1.2 Ant-World Generation

The system shall generate a valid ant-world according to section 2. The option to generate a world shall be provided when a user has uploaded ant-brain files. Alternatively, the user may upload their own ant-brain file.

Importance	High
Inputs	None.
Source	The ant-world generation program.
Outputs	An ant-brain world.
Prerequisites	An appropriate number of ant-brains must have been provided. For a pair match, this number is 2.
Post Condition	An ant-world is provided.

4.1.3 Ant-World Population

Once the ant-brains are uploaded, the world may be populated with ants. Each cell in the red anthill will be populated with a red and each cell in the black anthill will be populated with a black ant. All ants initially face east.

Ants are assigned top to bottom, left to right. The top-most left ant gets identity 0. All ants start in state 0, facing east (direction 0).

Importance	Essential
Inputs	None.
Source	The program.
Outputs	None.
Prerequisites	Two ant brains must have been uploaded and the world file must have been provided.
Post Condition	The world that is represented by the provided file is populated with ants that have the respective ant-brains that have been provided.

4.1.4 Gameplay

Once the ant-brains and ant-world have been provided then the game may proceed according to the rules laid out in section 2. The game will proceed for 300000 rounds. After this, the winner will be declared and announced to the user.

Importance	Essential
Inputs	None.
Source	The program.
Outputs	None.
Prerequisites	Two ant brains must have been uploaded and the world file must have been provided. The world must have been populated with ants.
Post Condition	The game is played and the winner is announced.

4.2 Non-Functional Requirements

Broadly, this section covers requirements that pertain to what the system has to *be*, rather than what it has to *do*.

4.2.1 Visualisation

A visualization of the Ant Game shall be provided. The visualization shall represent the game world at specific intervals definied by the user. The user shall be able to modify the intervals through the user interface of the system.

4.2.2 Platform and Availability

In order to make the program as accessible as possible it shall be provided as a web application. The application shall run on the server and provide a representation to the client which shall visualize the data that the server has sent back about the game. The web application server will be hosted using Amazon's EC2 service.

4.2.3 Tools and Technologies

Implementation Language
: Python shall be used as the implementation language for the game logic and server. Specifically, Python 2.7.10 should be used initially. For the client, the visualization shall use Javascript and HTML5 canvas.

Documentation
: Documentation shall be provided by the Sphinx Documentation Generator, and be available in both PDF and HTML format.

Testing
: Pytest shall be used for both unit testing, and later systems testing.

Deployment
: The Ant Game server will be deployed to an AWS EC2 instance running Ubuntu. The web browser client will connect to the server and game data shall communicated using websockets.

5 Process Requirements

5.1 Testing

Testing shall be carried out on all of the game logic methods of the system. Units tests shall be written in conjunction with the parts of the system that they test. All tests shall be written in the `pytest` framework in a folder that corresponds to the module that the functions that they test are in.

5.2 Documentation

Documentation shall be provided for all code. For all python code Sphinx will be used to generate documentation.

6 Acceptance Criteria and Testing

Several levels of acceptance criteria are specified. These correspond to some of the functional requirements. The finished system should be pass the tests laid out in this section, as they are essential to the customer's specified requirements of the system.

6.1 Valid Gameplay

Prerequisites	Two valid ant worlds, and an ant brain for each team.
Section of System being Tested	Game Logic and Ant Game Correctness
Passing Criteria	The ant game plays 300000 rounds as expected, without crashing or getting stuck.

6.2 User Uploading of Ant Worlds

Prerequisites	The user must be able to upload an ant world, than will be checked for validity by the system and be useable in gameplay.
Section of System being Tested	User interface
Passing Criteria	The user uploaded ant world is useable in gameplay.

6.3 Visualization of Ant Game

The Ant Game must be visualized, including results of the tournament.

Prerequisites	A running Ant Game
Section of System being Tested	GUI
Passing Criteria	A visualization of the ant game, including different all the different states an individual cell can be in.

6.4 User interaction with Ant Game

A user shall be able to connect to the Ant Game Server using a web browser. The ant game shall be able to be played remotely from any internet connection.

Prerequisites	A running Ant Game, and a running User Interface
Section of System being Tested	Ant Game Server and GUI
Passing Criteria	The user is able to play an ant game in a web browser.