## Requirements Specification

## **Overall Requirements –**

* **World visualisation**
* **World generation**
* **Gameplay system**
* **Tournament system**
* **Syntax checker for ant brains and worlds**
* **Ant Brain implementation for competition and testing purposes**

## **Functional Requirements -**

#### **Ant Brain**

* **Representation:**
  + **FSM of x states exactly x lines long**
  + **Line number x = state**
  + **Each state contains command and arguments**
* **Functions:**
  + **Move – move forward one cell, change state**
  + **Turn – change direction left or right by one direction to face new direction, change state**
  + **Sense – change state based on condition of adjacent cell**
  + **Flip – generate random number to determine state change**
  + **Mark – set a mark of number i of ant’s colour in current cell, change state**
  + **UnMark – remove mark of number i of ant’s from current cell, change state**
  + **PickUp – pickup food from current cell, change state**
  + **Drop – drop food in current cell, change state**

#### **Ant Control**

* **Ant Properties:**
  + **Colour – red or black identifying team colour**
  + **ID – unique per ant per team**
  + **State – state of FSM representing brain**
  + **Resting – rest period before next action legal**
  + **Direction – forward facing of ant in game world**
  + **HasFood - whether ant is carrying or not carrying food**
* **Movement capabilities:**
  + **Within the hexagonal layout constraints of the world, 6 directions**
  + **Move North East, East, South East, South West, West, North West**
  + **Movement based on direction, turn to face the next direction sequentially based on left/right**
  + **Directions:**
    - **0 – East**
    - **1 – South East**
    - **2 – South West**
    - **3 – West**
    - **4 – North West**
    - **5 – North East**
  + **Determine adjacent cell using direction and current position:**
    - **E, SE, SW, W, NW, NE**
* **Sensing capabilities:**
  + **Current cell**
  + **Ahead – directly ahead of ant**
  + **Left or right ahead – ahead plus left/right turn**
  + **Friend/foe identification**
  + **Rock/impassable terrain**
  + **Food/ant(friend/foe) with food**
  + **Markers (friendly/enemy)**
  + **Home position (friend/foe)**
* **Actions:**
  + **Move – as above**
  + **Mark/Unmark position/cell**
  + **Pickup/Drop food**
  + **Turn – as above**
  + **Flip – change state based on random number x**
  + **Set chemical marker 0-6**
  + **Sense friendly marker and marker value**
  + **Sense enemy marker (not value, just presence)**
  + **Set state of ant to s**
  + **Set resting value to a value r**
  + **Set direction to direction d**
  + **Set whether ant has food to true**

World Generation

* **Hexagonal grid**
* **Co-ordinate system:**
  + **x,y – x = column, y = row**
  + **0,0 – Top left of world**
  + **Odd rows are offset by 1 to accommodate hexagonal grid shape**
* **Identifiers for:**
  + **Rocky cells**
  + **Clear cells**
  + **Anthills (red/black)**
  + **Food particle (number representing quantity in that cell)**
  + **Chemical markers per team**
  + **Ant is present**
* **Contest requirements:**
  + **150x150 cells**
  + **Rocky perimeters**
  + **2 anthills – sides length 7, 6-way symmetric (hexagonal)**
  + **14 rocks**
  + **11 cells of food – 5x5 rectangular layout each cell containing 5 food**
  + **Ants spawned in anthill cell facing direction 0 (East)**
* **Functions:**
  + **Is there an ant at cell p**
  + **Get ant at cell p**
  + **Set ant at cell p**
  + **Clear ant at cell p**
  + **Check ant is alive given id**
  + **Check ant position given id**
  + **Kill ant (clear ant at cell) give id**
  + **Check for food at cell p**
  + **Set food at cell p**
  + **Check for anthill at cell p and of colour c**

#### **World Visualisation**

* Plain text file
* Line 1 – x dimension
* Line 2 – y dimension
* Rest of file = y lines of x characters
  + # Rocky
  + . Clear
  + + Red Anthill
  + - Black Anthill
  + 1-9 Food Particles in cell

#### **Gameplay System**

* Load ant brains from ant brain definition
* Cells of anthills each populated by 1 ant facing direction 0 (East)
* Ants assigned IDs based on left-to-right, top-to-bottom scan of the world cells
* Execute 300,000 rounds
  + Per round iterate through ants
  + Execute ant brain instruction based on state of FSM and random number generation
  + Detect surrounded ants (5 or more adjacent enemy ants)
    - Kill ant if true
    - Turn into 3 food particles at cell
* Count food particles per anthill cell per team after round 300,000
  + Most total food particles = winning team

#### Tournament System

* Match ant brains against another ant brain
* Allow n number of ant brains to compete
* Execute gameplay system with those two ant brains
* Determine winner
* Tally wins per brain
* Result of tournament = ant brain with largest number of wins

#### Syntax Checker for Brains/World

* Analyse files representing world:
  + Check for dimensions in first two lines
  + Iterate through the rest of lines and check layout is correct (odd rows are offset +1)
  + Check world conditions meet specification
    - Dimensions
    - Number of elements
    - Bordering rocky cells
* Analyse files representing ant brains:
  + Check for erroneous tokens
  + Ensure tokens follow correct syntax (take correct number of arguments etc.)
  + File length <= 10000 lines
  + Comments laid out so ignored by game, using semicolon to denote comment

#### Random Number Generator

* Return random number between 0 and n-1 when given number n
* Use seed to determine random number calculation
* Match generator to specification

## **Non-Functional Requirements -**

* Capable of running for 300,000 rounds in q reasonable time
* Ease of construction of ant brains
* Well documented code and features
* Well tested for reasonable inputs and scenarios
* Maintainable code base that facilitates extension in the future (e.g. modularity, encapsulation)

## **Deliverables -**

* Project plan
  + Project overview/introduction
  + Conflict resolution plans
  + Project phase/timing plans
    - Gantt Chart
    - Deadlines
  + Organisation plans
  + Peer assessment criteria and planning
* Requirements specification
  + Analysis
  + Functional & non-functional requirements
  + UML where appropriate
* Acceptance criteria and testing plan
* System Design (pseudo code, UML)
  + High level overview
  + Specific detailed design
  + UML
    - Sequence diagrams
    - Class diagrams
    - Object diagram
    - Activity diagram
    - Use Case diagram
  + Code standards
  + Language to use (Java/C#)
* Implementation of code
  + Use of version control
  + Code repository
  + Collaboration on code
  + Documentation and commenting
* Testing
  + Plan
  + Procedures
  + Results
* User documentation
* Management documents:
  + Meeting minutes
  + Version control evidence
  + Code Standards
  + Documentation Standards
  + Logging of project status and progress
  + Project reviews
  + Deliverable reviews