Use case description

System

* User Initialises **tournament** and **gameplay** via a GUI.
* The user initialises world via a GUI.
* User loads **ant brain** to the system via a GUI.

Ant brain

* System checks **ant brain** is syntactically correct.
* System gets and executes **ant brain** instruction.

World

* System analysis **world** to check it is valid.
* System calls **get adjacent cell** function which returns **current cell** position.
* System calls **find ant ID** function and returns the ant with the corresponding **ant ID**.
* System calls the **do something to sell** function
  + Returns true of false if cell **is rocky**.
  + Returns true or false if cell **is anthill**.
  + If **food** is in cell, then set food to current cell.
  + If ant is in cell is **ant alive/dead**?
  + **Kill ant** and **clear ant** from cell.
  + Set **ant ID** at current cell.
  + Returns **marker** type if cell **is marked**.

Game play

* System loads new **ant world** and checks it is syntactically correct.
* System checks red & black **ant brains** are syntactically correct and loads them to the **ant** **world**.
* System starts **new game**.
* System gets and executes **ant brain** instruction
* System activates the **step** function
* Game loop is executed.
* **Game ends** and system declares a winner after 300000 steps.

Tournament

* System adds **competitors** to **tournament** and loads **ant brains**.
* System generates and **matches competitors.**
* System calls **play match in tournament** which initialises **gameplay**.
* System declares **winner** of **tournament**.