# The rules of the simulation

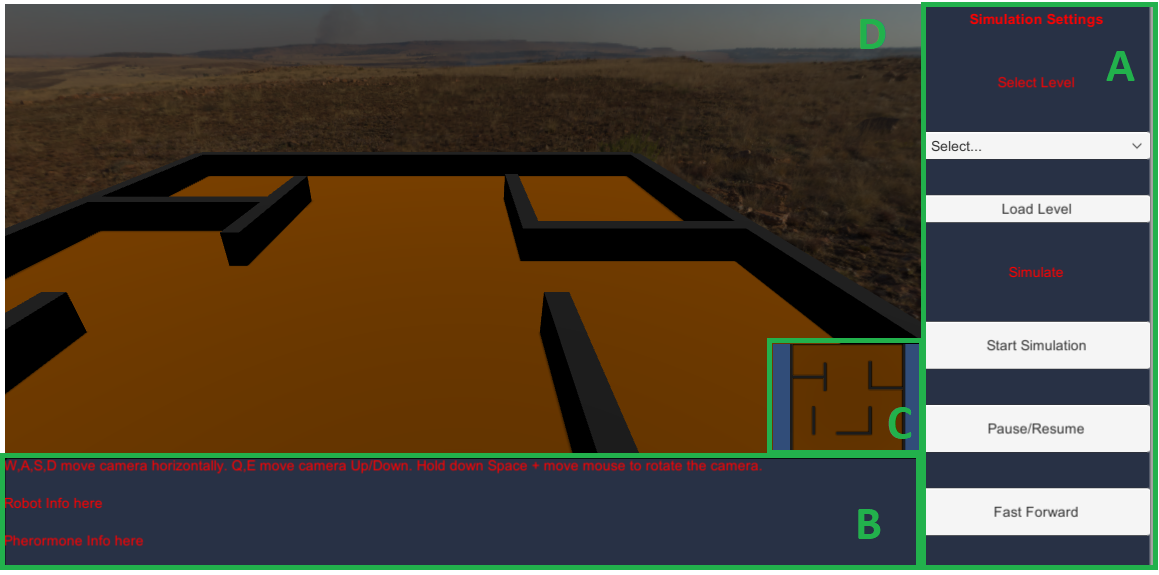
Based on the paper we attempted to set some ground rules about how the simulated world operates. By reading the paper we know:

* The robots are physical objects
* Each robot can spawn pheroromone at its position
* Pherormones have some intensity, which decays over time and when it reaches zero it disappears
* Each robot has sensors around it that can detect pherormones. The exact structure of these sensors in undefined, but they can detect the angle and intensity of all pherormones inside a range.
* A robot can identify if some pherormone is its own.

We also decided on the following rules, without loss of generality, which were not clarified in the paper:

* The game will be played in discrete rounds not continuous time.
* Each robot can do the following actions:
  + Move Forward any amount of distance
  + Rotate clockwise/anti-clockwise any number of degrees
  + Scan for Pherormones using its sensors
  + Scan for obstacles (walls or robots) using its sensors.
  + Spawn new pherormone
* Pherormones hold some extra parameters including
  + The ID of the robot that spawned it (so a robot can detect its own pherormone)
  + The type of pherormone it is (bee or ant)
  + The Hopcount and H parameters used by the bees
* Scanning actions are executed instantly, all other moves are executed at the same time after all the behaviors have been executed so we won’t have race conditions (one robot moving while another is scanning for it while executing its logic).
* We will not have a general terminal condition for our application and let it run infinitely. Our terminal condition may be different based on what we want to achieve so we decided to manually terminate the simulation for easier extendibility.

# The simulation interface

  
Figure 1: The simulation interface

The simulation interface consists of four different panels, as seen in Figure 1.

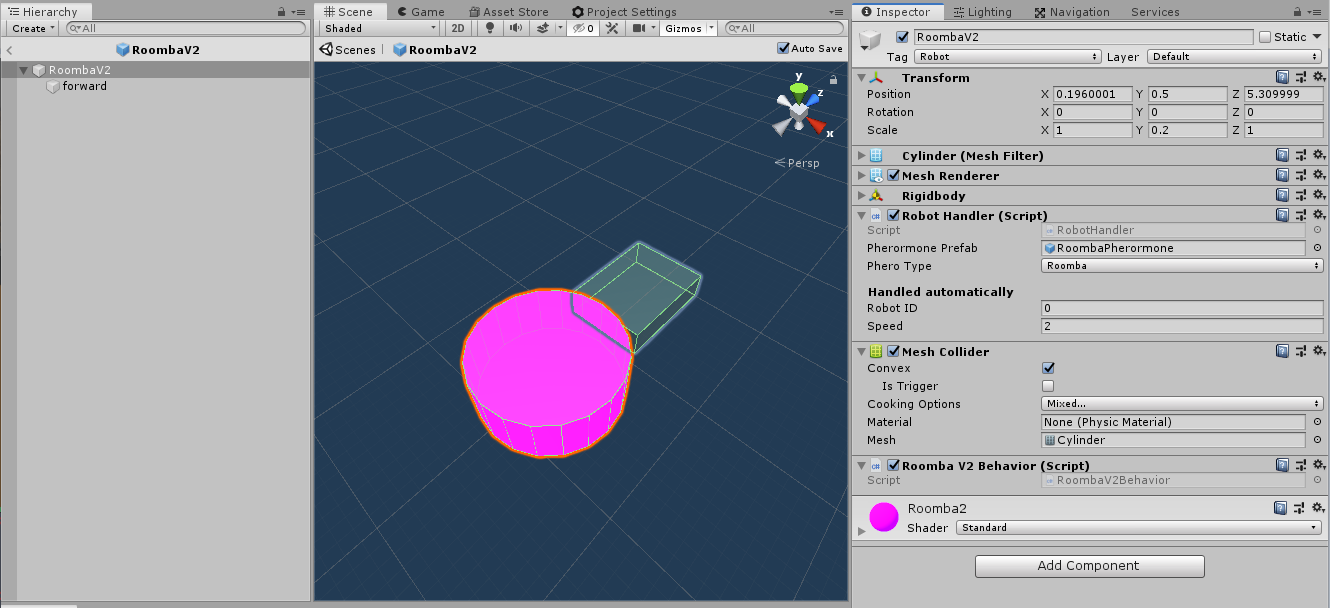
Panel A consists of a set of control buttons used for controlling the simulation. On the top part there is a level selection dropdown menu. By selecting a level in the dropdown and pressing load level will switch to that level. On the bottom half are three buttons. Start simulation is pressed once to start the simulation. Pause/Resume freezes the game clock or sets it to 1x speed while fast forward switches between 2x and 1x speed.

Panel B is an information panel. At the very top are the controls for moving the camera for Panel D. We can also click on a robot or pherormone in panel D while the simulation is running and we can see the robot and pherormone data (ex. the robot ID and type) in the appropriate text fields.

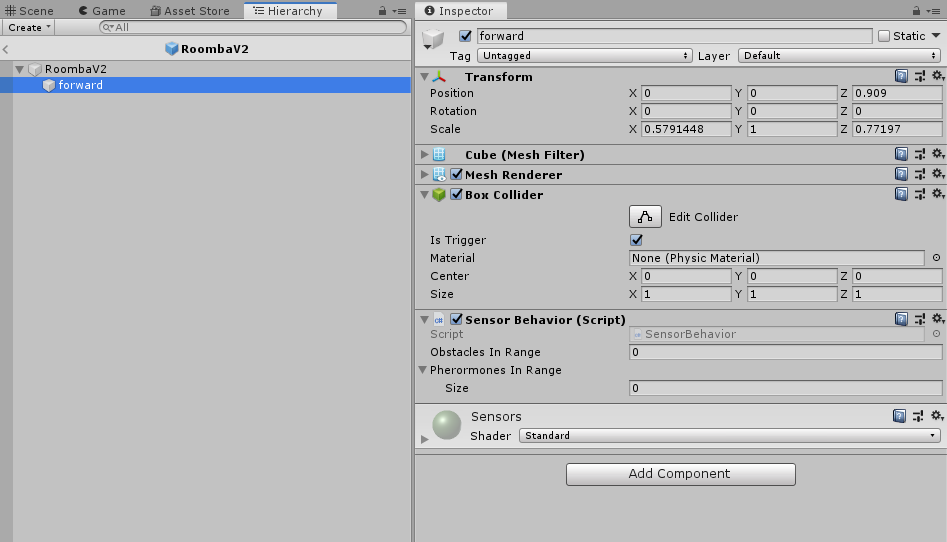
Panel C is an optional top-down view if the level. Using this we can easily see the coverage of our robots.

Finally, Panel D is the in-game camera of the simulation itself.

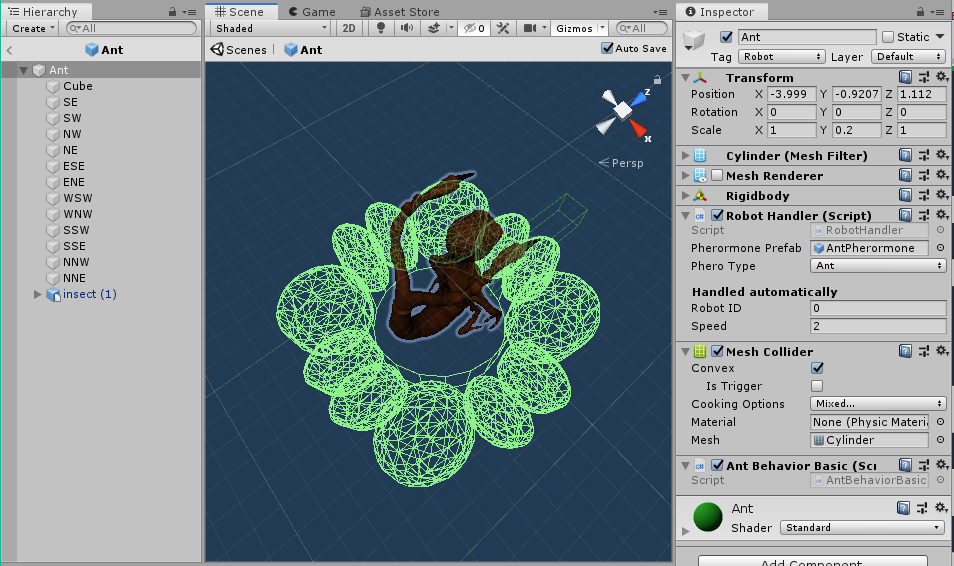
# Robot setup

Figure 2: The Roomba robot used as a generic template

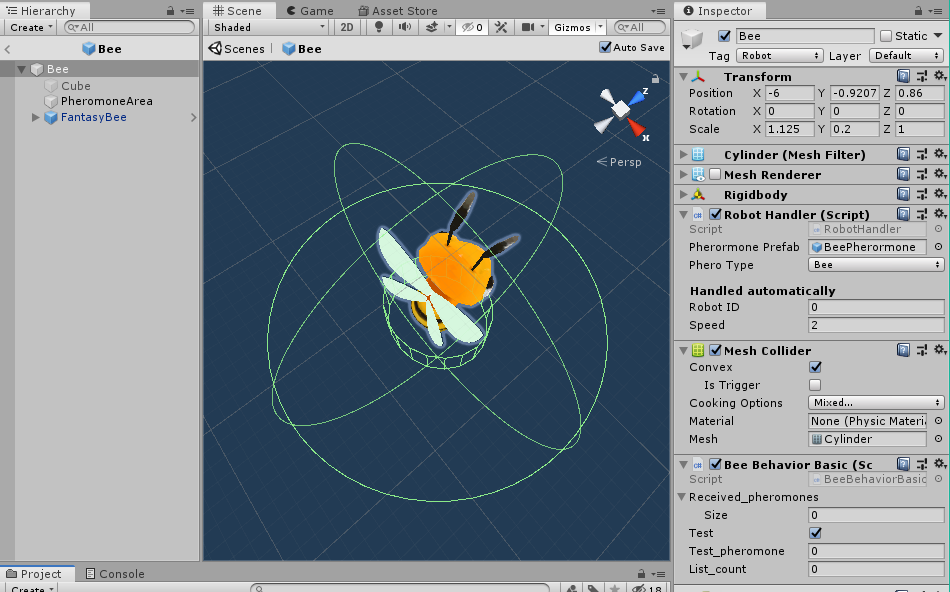
Each robot is a standalone entity in the Unity environment, known as a Prefab, as seen in Figure 2. As seen in the Hierarchy view (left window) a robot consists of a capsule object as the center, which has an even circular collider around it. As its children we have some trigger regions which act as the regions inside which a robot can scan, reffered to as sensors. Each sensor is invisible and works like a trigger (it can detect objects inside it but it does not collide with them unlike the body of the robot).

  
Figure 3: the parameters for a sensor

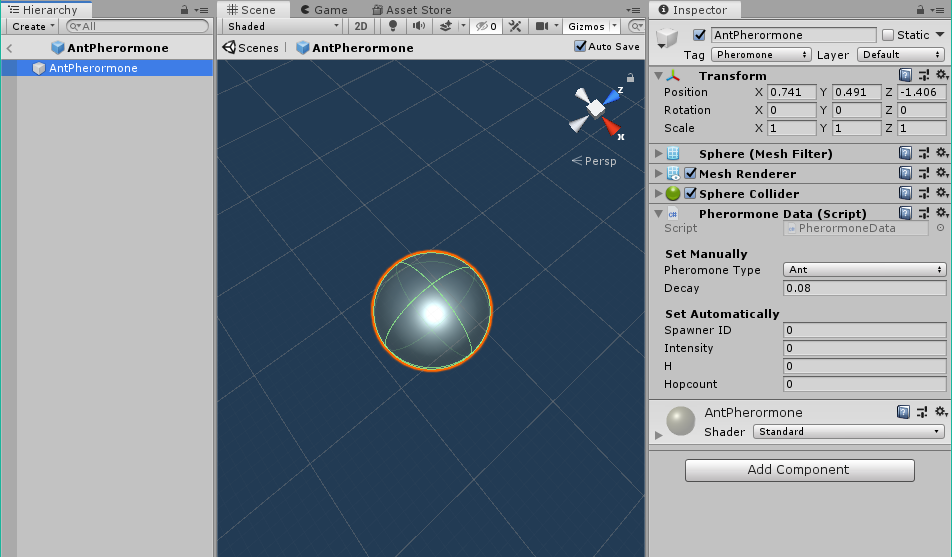
In the Inspector view we can see the components of the central robot object. Most components are not worth mentioning as they are used internaly by Unity. For our simulation what matters are two scripts: the Robot Handler and the <Robot name> Behavior. Similarly, each sensor has the Sensor Behavior script. The Robot Handler and Sensor Behavior are general use scripts used by all robots, only the <Robot name> Behavior script is specific to each robot with different logic.

Figure 4: The Ant robot

The Ant robot consists of the same central capsule as the body of the robot with the Ant Behavior Script and 12 sensors (SE-NNE). We also opted to change the visuals from a simple capsule to an animated freeware model for visual appeal.

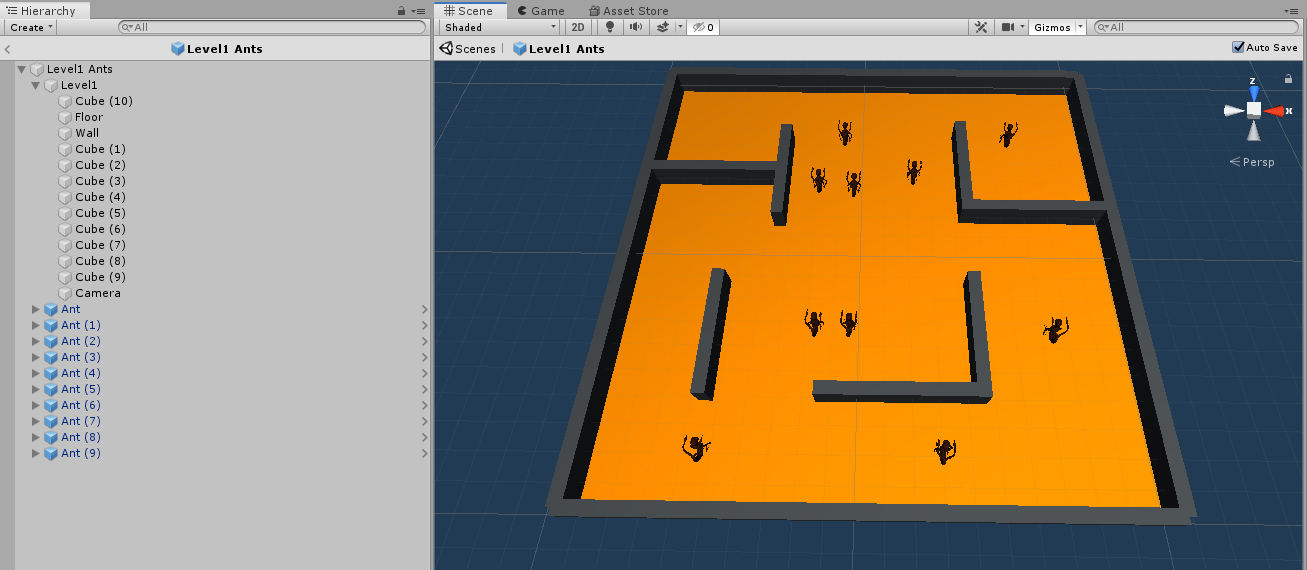
Figure 5: The bee robot

Similar to the ant, the Bee has the Bee behavior script with only one large sensor (Pherormone area).

Figure 6: The Ant Pherormone

Similarly to the robots, the pherormones are also created as Prefabs. Visually, they are semi-transparent circles of different color for each different type (bee or ant). In the Inspector window we can see the parameters a Pherormone has. When a pherormone is spawned we set the Spawner ID to be the same as the robot that spawned it and the initial intensity to 1 as well as set the scale of the object to make it as large as we want. The Decay factor determines how much intensity is subtracted every round.

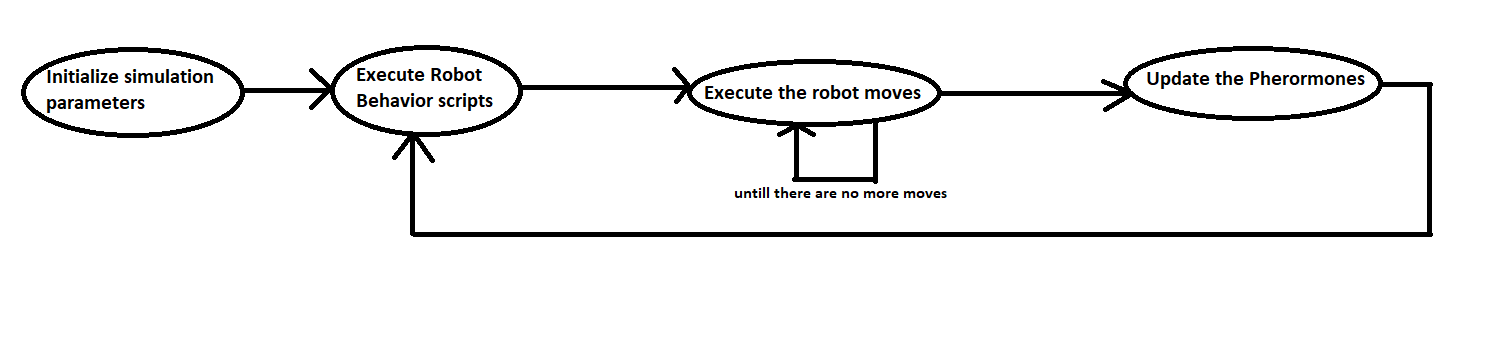
# Level setup

Figure 7: Level 1 with Ant robots

A level is a Prefab consisting of a floor, walls and robots. The floor and walls can easily be setup in any desirable way using Unity’s tools while the robots are duplicate instances of a Prefab with different starting position and rotation.

# Simulation flow

Figure 8: The Simulation’s flow



The flow of the simulation is handled by a central Game manager. This game manager holds a list of all robots and all pherormones present in the game and calls functions on each object at the correct times based on the game flow.

The first step is to perform all necessary initializations, like finding all the robots in the scene and giving them unique IDs.

At the start of a round we call the Behavior() function that is placed in the <Robot name> Behavior script, which executes the robot’s logic for one round and creates a list of moves to be executed for this round, for example: rotate 30 degrees, move forward 1 meter, spawn new pherormone.

After all behaviors have been executed we execute each robot’s moves in parallel until all robots have no more moves in their lists. This marks the end of the round so we update the intensity of all pherormones in the scene and restart the execution for the next round.