

# Swarm Intelligence H-414

## Swarm robotics

Antoine Ligot and David Garzón Ramos

IRIDIA-CoDE — Université Libre de Bruxelles, Belgium

February 19, 2019

# Today

- ▶ Part 1
  - ▶ Introduction
    - ▶ Reminder on swarm robotics.
    - ▶ What is a swarm control software?
    - ▶ How to program a swarm control software?
    - ▶ How to use the ARGoS simulator?
- ▶ Part 2
  - ▶ Exercise: Obstacle avoidance

## Reminder on swarm robotics

In swarm robotics, we want a large group of robots to accomplish a task/mission (eg. flocking, foraging, patrolling, search and rescue).

The robots

- ▶ are relatively simple → they must cooperate.
- ▶ are autonomous → there is no leader or orchestrator, no global information.
- ▶ have local sensing and communication capabilities.

Robot swarms are meant to be

- ▶ Robust (ability to cope with the loss of individual)
- ▶ Scalable (ability to perform well with different group sizes)
- ▶ Flexible (ability to cope with different environments)

## What is a swarm control software?

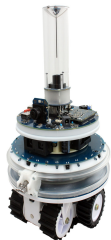
We don't program the swarm, but the individuals, based on the **local interactions**.



### Swarm control software

- ▶ the code that controls a single robot.
- ▶ each robot in the swarm executes the same control software.

# How to program a swarm control software?

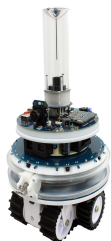


## Footbot

- ▶ *Sensors* to gather information from the environment.
  - ▶ Proximity, ground, range-and-bearing, camera.
- ▶ *Actuators* to “act” on the environment
  - ▶ Wheels, LEDs, range-and-bearing.

## How to program a swarm control software? (2)

Program in a sense-think-act loop.



- ▶ **Sense** ... the environment, the other robots.
- ▶ **Think** ... process the information sensed.
- ▶ **Act** ... do something based on your thinking.
- ▶ **Repeat**

The less sensors/actuators you use, the better.

# How to use the ARGoS simulator?

In a terminal:

```
argos3 -c experiment.argos
```

The screenshot shows the ARGoS simulator interface. On the left, a top-down view of a rectangular arena with a grid floor and teal walls. Several blue robot-like agents are scattered across the arena. On the right, a code editor window displays the ARGoS configuration file (experiment.argos) with various function definitions and comments.

```

1 -- Use Shift + Click to select a robot
2 -- When a robot is selected, its variables appear in this editor
3
4 -- Use Ctrl + Click (Cmd + Click on Mac) to move a selected robot to a different location
5
6
7
8 -- Put your global variables here
9
10
11
12 --[[ This function is executed every time you press the 'execute' button ]]
13 function init()
14   -- put your code here
15 end
16
17
18
19 --[[ This function is executed at each time step
20     It must contain the logic of your controller ]]
21 function step()
22   -- put your code here
23 end
24
25
26
27 --[[ This function is executed every time you press the 'reset'
28     button in the GUI. It is supposed to restore the state
29     of the controller to whatever it was right after init() was
30     called. The state of sensors and actuators is reset
31     automatically by ARGoS. ]]
32 function reset()
33   -- put your code here
34 end
35
36
37
38 --[[ This function is executed only once, when the robot is removed
39     from the simulation ]]
40 function destroy()
41   -- put your code here
42 end
43

```

## Exercise: Obstacle avoidance.

### Goal:

The robots must explore the environment while avoiding collisions (with objects and other robots).

### Material:

<https://iridia.ulb.ac.be/courses/2019/h-414/>

### VUB students:

Login with username/password: *fsa-vub-students*