Version 1.1 Jun 2017

SWARMATHON 5

COMPETITION GUIDE

You are now ready to begin writing your code submission for the Swarmathon HS Division Competition! This document describes the competition structure and rules, and includes a checklist for you to complete prior to file submission.

1 COMPETITION OVERVIEW

1.1 GOAL

The goal of the NASA Swarmathon High School competition is to program a virtual swarm of 6 robots in NetLogo to search a square arena and find and retrieve as many simulated resources as possible in a fixed amount of time.

1.2 TOURNAMENT STRUCTURE

1.2.1 TECHNICAL DETAILS

Your code will be competed locally at UNM using NetLogo 5.2 and a custom Python script. The script will automate the runs and check for illegal commands and modifications to the required base code.

1.2.2 TIME LIMIT

Your code will be run for exactly 3600 NetLogo ticks in simulation time, which corresponds to a one-hour simulation in real time, with each tick representing one second.

1.2.3 SCORING

Your code will be run three times. Your total competition score will be the sum of the resources collected in each run. The team with the highest score will be declared the winner of the competition. In the event of a tie, an additional tiebreaker run will be performed. If it is the case that multiple teams collect all resources in the tiebreaker round, the team who collected all resources the quickest will be declared the winner.

1.2.4 ARENA

The arena is 101 x 101 pixels and uses the included parking lot image as a background. World wrapping will be turned off.

1.2.5 RESOURCE DISTRIBUTION

Robots on Mars need to be flexible. To test the flexibility of your code, we will choose three different distributions of 512 resources. Each file will be tested using these same three distributions. The chosen distributions will not be revealed in advance. However, ensuring that your code performs well across the built-in distributions in the Swarmathon 5 base code will prepare you well for success in the competition.

1.3 SUBMITTING YOUR FILE

Files must be submitted to your GitHub repo by midnight on the day of the submission deadline. Emailed submissions will not be accepted. Please see the User Guide for the competition schedule and for more information on using GitHub.

2 RULES

Read the following rules carefully before beginning your competition submission. Complete the checklist as you work, then use the final checklist at the end. **Be sure to double-check your code using the final checklist before submission. Submissions that violate any of the following rules will not be accepted.**

2.1 FILE SETUP

- ☐ You are using NetLogo 5.2.
- ☐ Your file is named *HighSchoolName_Sw17.nlogo*. Example: *DelNorte_Sw17.nlogo*.
- ☐ Your code includes the base code in *[SW5]competitionBaseCode.nlogo*, **unmodified.**
- ☐ Your file includes your name(s), the name of your high school, the name of your team mentor in comments at the top.
- ☐ Your code must include your own comments that explain what your code is doing. Submissions without comments or with comments copied directly from the Swarmathon modules will not be accepted.
- ☐ You should not copy-paste directly from any of the Swarmathon modules. Using code snippets is acceptable,

but you are strongly advised to type the code yourself or you will run into problems with illegal characters, etc.

2.2 WORLD SETUP

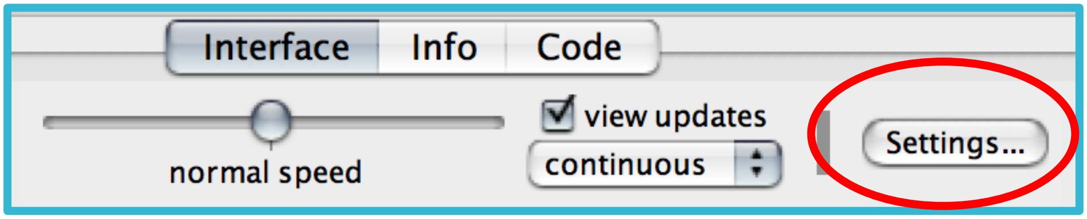
Your world has the following properties:

- ☐ The origin is located at the center.
- ☐ min-pxcor and min-pycor are set to -50.
- ☐ max-pxcor and min-pxcor are set to 50.
- ☐ Both horizontal and vertical world wrapping are unchecked.
- ☐ Patch size is 5.
- ☐ The tick counter is on (box is checked).
- ☐ The tick counter label is "ticks".

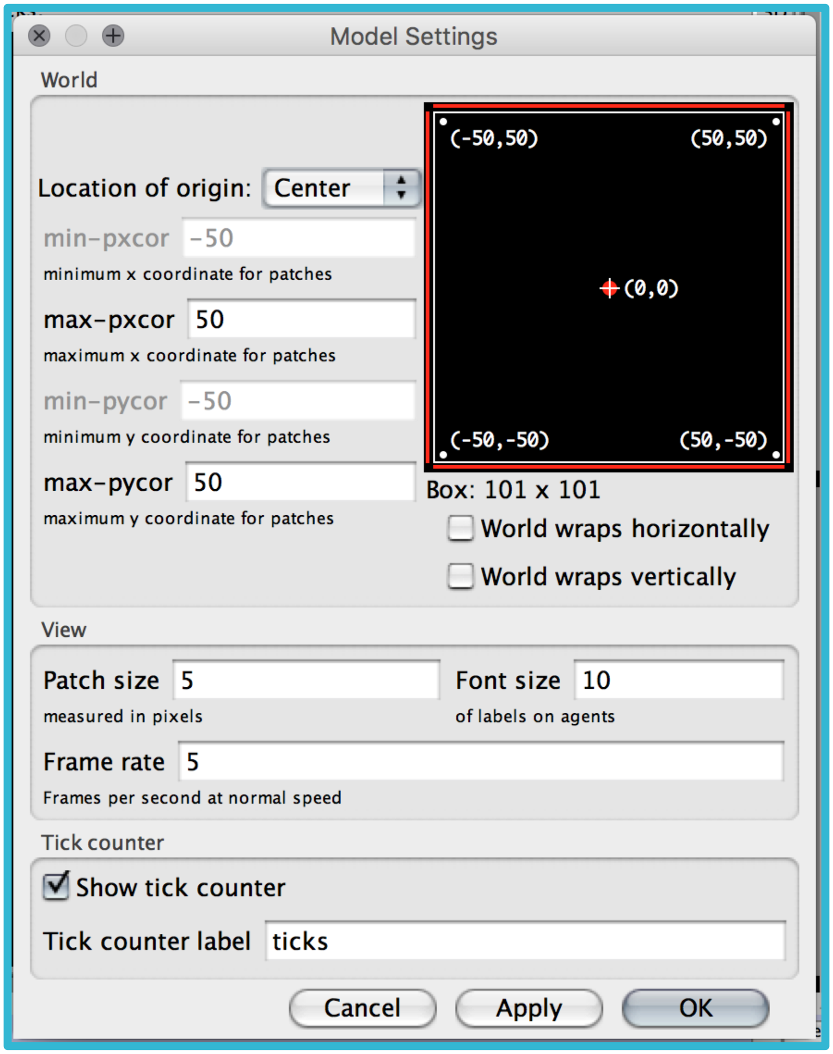
The base code for [Sw5] comes with these settings already. However, because each NetLogo install can have its own quirks and default settings, be sure to check these settings on your file before submission.

2.2.1 HOW DO I CHECK THE SETTINGS ON MY FILE?

Navigate to the interface tab and press the **Settings…** button:



A menu will pop up. Compare the picture below to your menu. If everything matches, you're good!



Model Settings

World

Location of origin: Center

min-pxcor -50
minimum x coordinate for patches

max-pxcor 50
maximum x coordinate for patches

min-pycor -50
minimum y coordinate for patches

max-pycor 50
maximum y coordinate for patches

Box: 101 x 101

□ World wraps horizontally

□ World wraps vertically

View

Patch size 5
measured in pixels

Font size 10
of labels on agents

Frame rate 5
Frames per second at normal speed

Tick counter

☑ Show tick counter

Tick counter label ticks

2.3 THE INTERFACE

- □ When the setup button is clicked, your file sets up and the required code is executed. When the robot-control button is clicked, your code runs. Only the robot-control button is a "forever" button. There should be no other buttons on the interface.
- □ You may add sliders, monitors etc. to the interface.
- □ The file must be turned in with any sliders, choosers, etc. that you create set to the desired settings. It will be competed with the settings it is turned in with.

2.4 GLOBAL VARIABLES, EXTENSIONS, AND BREEDS

- ☐ Global variables may not provide robots with information that they would not have access to locally.
- ☐ The base code file includes the bitmap extension. No other extensions are allowed.
- ☐ You may use different breeds of robots. The number of robots created in each breed must be fixed, not random, and the sum total of robots created across breeds is 6.

2.5 ROBOTS AND THEIR PROPERTIES

- ☐ You must create and setup **exactly six robots**. All robots must spawn at the origin/base (this is the default setting).
- ☐ These same six robots must be in play throughout the competition. (You may not use the commands **hatch**, **die**, or **sprout**, or any other commands that create or destroy robots after the initial setup.)
- ☐ Robots can move a maximum of 1 step on each tick. Be careful that you are not calling multiple procedures in one tick that include move commands! Robots may also remain stationary.
- ☐ Turning is not considered a move command. Example:

```
left 90

forward 1
```

  is allowed.

- ☐ Robots cannot teleport. Example: You may not use **setxy** or **move-to** to change a robot's coordinates.
- ☐ The robot must have the shape "robot" when not carrying a resource and the shape "robot with rock" when carrying a resource.

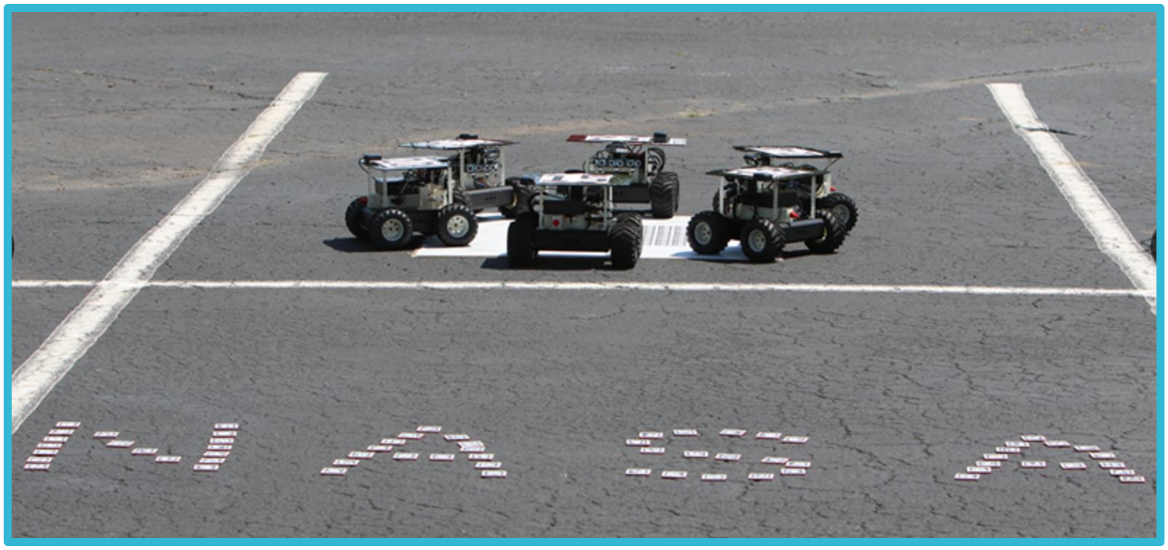
- ☐ Robots may have any labels you wish, or none.
- ☐ Robots do not have an unlimited vision distance—they can see a maximum of 2 patches around them. Examples: The commands **in-radius 1** and **in-radius 2** are allowed; **in-radius 3**, **in-radius 4**, … etc. are not. The commands **patch-ahead 1** and **patch-ahead 2** are allowed; **patch-ahead 3**, **patch-ahead 4**, …etc. are not. **Neighbors** and **neighbors 4** are allowed.
- ☐ Robots do not have **global** knowledge of the state of the arena. Example: They can't create a list of all resources at the beginning of the simulation and then pick them up.
- ☐ Robots have **local** knowledge based on what is in-radius 2 around them. As was introduced in Swarmathon 3, a robot can store a list of resources it has encountered **locally**.
- ☐ A robot can communicate **locally** with another robot in-radius 2 or less of itself.

2.6 PATCHES AND THEIR PROPERTIES

- ☐ Patches may change color or properties, as was introduced in Swarmathon 2, but those changes must occur as a result of contact with a robot and must respect the presence of resources.
- ☐ Patches must remove the resource (change color from yellow back to baseColor) when the robot has picked it up, or the score will not be counted for that resource.
- ☐ Patches may **only** remove a resource (change color back to the baseColor) when a robot has contacted that patch and picked up a resource.
- ☐ Patches may not **hatch** or **sprout** any agent or anything else.

2.7 PROGRAMMING

- ☐ You may not use recursive algorithms.
- ☐ You may not add additional calls to tick in procedures called by robot-control.
- ☐ All additional procedures you create must be called by either setup or robot-control.
- ☐ You may experiment with any NetLogo commands not explicitly outlawed here. If you are in doubt, ask! Be creative.



Good luck in the competition!

GREAT JOB! You completed SWARMATHON 5.

BUG REPORT? FEATURE REQUEST?

Email sherbet@unm.edu with the subject SW5 Report

FINAL CHECKLIST

FILE, WORLD, AND INTERFACE SETUP

- ☐ You are using NetLogo 5.2, your file is named *HighSchoolName_Sw17.nlogo*, your code includes the base code it came with, **unmodified**, and the names of team members, the team mentor and your high school are included in comments.
- ☐ Your code includes your own original comments that explain what your code is doing.
- ☐ Your world settings are identical to those in the picture in section **2.2.1.**, you didn't add any buttons to the interface, and your file is saved with the sliders, etc., set to the values you want to use for the competition.

CODE, ROBOT, AND PATCH PROPERTIES

- ☐ You didn't add any extensions, use recursion, or include additional calls to tick, and all of your procedures are called by either setup or robot-control.
- ☐ Global variables don't give robots access to knowledge that they wouldn't have locally.
- ☐ If you used breeds, the number of robots created in each breed is fixed.
- ☐ You have created and setup **exactly six robots** that spawn at the origin, only interact with and have knowledge of patches or robots in-radius 2 of themselves, change their shape to reflect holding or not holding a resource, do not die, do not teleport, and do not create other robots.
- ☐ Patches change color when a robot directly picks up a resource, any other color changes you implemented do not affect rocks, and patches do not create anything.