SG2016 - SWARMBOT ASSEMBLAGE

SIMULATION MANUAL

Petra Jennings, Nils Napp, Kirstin Petersen

SIM SETUP

The simulation is done in Processing using the Box2D physics engine. It is a 2D arena with a static boundary around it. Within this there are parts and robots. Parts are dead objects that can only moved when pushed, while robots are active objects with a force that makes them go forward.

You need to have Processing installed on your computer, with the Box2D library.

INSTALLATION

Install Processing:

https://processing.org/download/

Install Box2D:

Within Processing. Click the Sketch tab, then Import Library…, Add library… Then search for Box2D and Install.

FILE STRUCTURE

The Simulation folder contains the following files:

Boundary.pde
Parts.pde
Robot.pde
Sim.pde
Part1.json
Part2.json
Part3.json
Robot1.json
Robot2.json
SimInfo.json

The simulation contains a main tab called Sim which calls functions from the tabs Boundary, Parts, and Robots. To run the simulation just open Sim.pde in processing and hit the play button.

The settings that you need to alter during the workshop are foun d in the JSON files. To make changes to them, open them in Wordpad or Notepad++. Once saved they will automatically be read back into the Sim.pde file.

SimInfo.json holds the information about what parts and robots that should be in the simulation, while the files called Part.json and Robot.json describes the parts and the robots respectively.

SETTINGS

PARTS

File: Part.json

PARAMETER	CODE EXAMPLE	DESCRIPTION
Name	name: testpart,	Don’t change
Scale	scale: 8,	Size of the object
Table friction	table_friction: 0.4,	Affects the behaviour of the parts, how easily they get pushed and how quickly they stop.
Part friction	part_friction: 4,	Affects the friction between the part and other objects.
Density	density: 0.3,	How heavy the parts are
Restitution	restitution: 0.1,	Bounciness
Color	color: {r:255, g:255, b:255},	RGB value
Polygons	polygons: [{poly: [{x:-2,y:0}, {x:0,y:3}, {x:2,y:0}] }, {poly:[{x:-2 , y:2}, {x:0 , y:-1}, {x:2 , y:2},] }]	This is where you “draw” polygonal shapes. You can only draw convex shapes. More complex shapes and concave objects needs to be built up by an assembly of convex shapes. Assemblies can exist of both polygons and circles. The 0,0 coordinate is the centre of the shape. You specify the vertices x and y coordinates clockwise. Add as many vertices as you need. But make sure they all form a convex shape. Example draws 2 polygons.
Circles	circles:[{x:0,y:0,r:2.5},],	To draw a circle. Specify the centre point coordinates and the radius of the circle.

To make new parts:

1. Copy an existing Part.json file
2. Rename it to Part(NUMBER).json, ie Part2.json.
3. Make your changes
4. Make sure that the new part file is referenced in the simulation by adding it to the SimInfo.json file.

ROBOTS

File: Robot.json

PARAMETER	CODE EXAMPLE	DESCRIPTION
Name	name: robot,	Don’t change
Force	force: 100,	The force that makes the robot go forward
Noise	noise: 0.8,	Degree of noise/randomness in the robot's direction of movement
Scale	scale: 5,	The size of the robot
Table friction	table_friction: 0.4,	The friction between the robot and the table
Part friction	part_friction: 4,	The friction between the robot and other robots or parts
Density	density: 0.5,	The weight of the robot
Restitution	restitution: 0.1,	Bounciness
Colour	color: {r:200, g:30, b:30},	Colour, RGB-value
Polygons	polygons:[{poly: [{x:1,y:1}, {x:-1, y:1}, {x:-1, y:-4}, {x:1, y:-4}] }],	The robot's geometry is made of polygons or circles or assemblies of the two. Each polygon must be convex. To make concave shapes you need to assemble several polygons or circles. When making assemblies make sure to write the largest shape last. You “draw” by adding the vertices of the polygon. X:0, Y:0 is the center of the robot
Circle	circles:[{x:0,y:0,r:2.5},],	Circles are drawn by centerpoint and radius.

To make new robots:

1. Copy an existing Robot.json file
2. Rename it to Robot(NUMBER).json, ie Robot2.json.
3. Make your changes
4. Make sure that the new robt file is referenced in the simulation by adding it to the SimInfo.json file.

When all fails):

- Check that you got the right amount of curly brackets and commas.
- Call for Nils or Kirstin!

SIMINFO

File: SimInfo.json

This is where you set which parts and robots should be active in your simulation, and how many of each. For example, the code below specifies a simulation with 100 robots of type “robot1.json”, 100 parts of “part1.json”, and 30 parts of type “part2.json”.

Example code:

```
{
 name: SimInfo,
 robots:[{file: robot1.json,  count: 100},
          {file: robot2.json, count: 0}],
 parts: [{file: part1.json, count: 100},
           {file: part2.json, count: 30},
           {file: part3.json, count: 0}]
}
```

RESOURCES

http://natureofcode.com/book/chapter-5-physics-libraries/
http://www.box2d.org/manual.html
https://processing.org/reference/