How To Build a ROS Robot

SDSMT Robotics Team

Ian Carlson

July 19, 2015

Contents

List of Figures

1

Introduction

1.1 Purpose

Building a sophisticated robot can be a complex and potentially daunting task. While a single afternoon can be sufficient to strap a few servos and an arduino to a chassis, doing anything interesting required an investment of both time and resources. As the cost of a robot increases, it becomes more important to have a solid plan to follow. This document attempts to outline the major phases of robotics development, as well as providing the basic information required to build a basic robot featuring ROS. It should be noted that this document is entirely a product of my own experiences - mostly the failures - during my time on the Robotics team, in the hope that others will not repeat them. It is a certain truth that failure is a better teacher than success, but experience is the best teacher of all - preferably someone else's.

The major sections are as follows:

- Mechanical Design - Basic mobile robot kinematics and design principles. This area is not my specialty, so take it with a grain of salt.
- Electrical Design - What you need to know to not start your robot on fire. Probably.
- Setting Up The Odroid - How to go from a fresh Odroid to one running Fedora 22 and ROS
- ROS Architecture - The basic design for a remote controlled robot in ROS

This guide should be delivered alongside its companion document - Robot Paper Template - and potentially the two examples I created using that template - Mecanum Design Document and SMP Design Document. All of the code referenced in this document is available on the SDSMT Robotics Team GitHub pages - SMD-ROS-DEVEL and SMD-ROBOTICS.

1.2 Implementation

We chose to use C++ and OpenGL with Freeglut to run and visualize our simulation.

1.3 The Simulation

The basic setup is that there is a simulated fish tank with fixed boundaries. Within the tank there are 1 or more breeding populations of creatures we meant to find a cool name for, but eventually just ended up calling "Critters". Every generation, a number of individuals from each population are randomly placed on the game field. The Critters start with some amount of health and life points. Life points go down at a fixed rate based on the metabolic cost of the Critter's traits. Health goes down when two Critters battle. A battle is triggered when two Critters touch, and the result depends on both Critters' attack and defense power. Critters can devour one another by landing the killing blow, which restores life and health points.

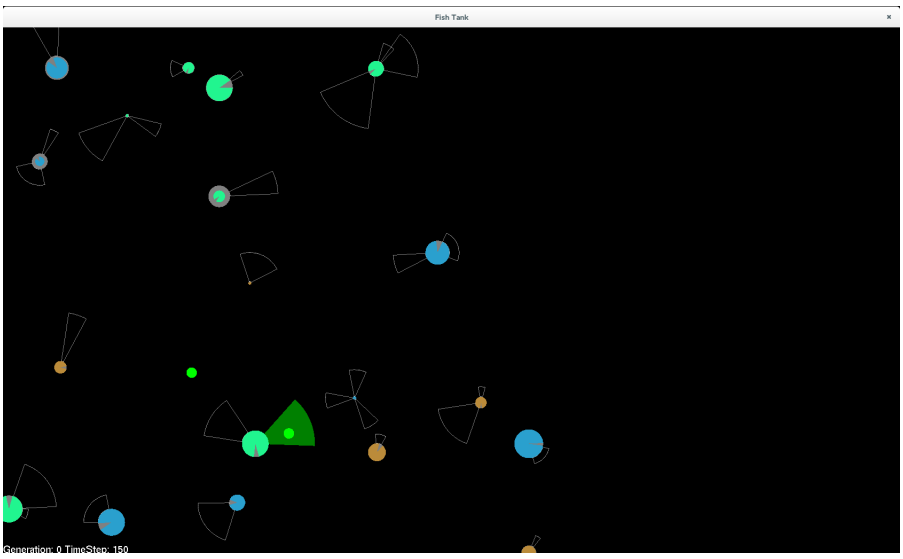

Figure 1.1: The Simulation

Over time, food pellets are randomly dropped on the field. Consuming a food pellet restores health and life points. However, the rate at which food pellets are drops decays over time, preventing infinite length generations. One generation continues until all but one Critter is left on the field. This Critter is then declared the winner and the generation ends with that Critter being ranked as the most fit of its population.

1.4 The Characteristics

Each Critter has the following set of basic characteristics that define its appearance and how it interacts with the environment.

- Max Health
- Size
- Attack
- Defense
- Speed
- Metabolic Cost

Max health determines the amount of health the Critter starts with. The Critter cannot go above this amount of health by eating food pellets or other Critters.

Size doesn't directly affect any of the simulation mechanics, but it does change the size of the critter on the screen. This indirectly also affects collisions with other Critters and food pellets.

Attack determines how much damage the Critter does to other Critters in battle.

Defense determines how resistant to damage the Critter is.

Speed determines the maximum rate at which the Critter can translate and rotate.

Metabolic cost is not an evolved characteristic, but a derived one. Each of the other characteristics has a cost associated with it. The metabolic cost is the sum of those costs, and determines the rate at which the Critter's life points decreases.

In addition to the basic characteristics listed above, each Critter also has some number of eyes. The number of eyes is an evolvable parameter.Each eyes also has three characteristics.

- Position

- Range
- Field of View

Position is just the position of the center of the eye on the Critter's circumference. This characteristic does not have a cost.

Range is the distance that the individual eye can see.

Field of View is the width of the angle an individual eye can see.

The minimum and maximum allowable value for each characteristic(except eye position) and the cost for each "point" in that characteristic are configurable in the SimConfig.conf file. There are also several other configurable options such as the number of populations, the number of individuals in a population, the mutation rate of each population, and food pellet distribution rate. A users manual detailing the entries in SimConfig.conf is available in

Bibliography

[1] Paul Oliver, *Guppies - Evolving neural networks (w.i.p.)*, https://youtu.be/tCPzYM7B338, 2012.