AI

This page is **stale** and needs to be reviewed. It may be deleted or radically changed in the near future.

Overview

- The AI system is the top level of abstraction for the sub, mission elements, and states. Most, if not all decision making should be done at this level. The AI system for the sub is a python scripting environment interfaced with the standard C++ modules via asynchronous messaging.
- The AI system is designed around a behavior tree implementation in order to take advantage of the very flexible and fast behavior creation using primitive nodes. Behavior trees have been widely used in game engines for AI for their simplicity and power. Since the AUV missions are known beforehand, a behavior tree perfectly fits in this use case.
- A quick and dirty behavior tree guide is here. Behavior Trees

Setup

- The AI system uses python and pyzmq for interfacing with the other modules. To setup, run the install script that came with the repo to get both components. You can run the AI using the command module bash interface or directly using the python interpreter.

Making Missions

- The core goals of the AI system is code re-usability and extensibility. When designing new missions for the submarine always try to call upon or extend previous code. This not only speeds up development, but also reduces the assumptions you have to make as previous code is usually tested and proven. If you must implement something new, always start at the highest level and break tasks down into the smallest primitives you can. From there, assemble a tree using composite nodes in order to make a traversal that is suited for the particular functionality.
- Note, that the AI behavior tree implementation calls the tree until SUCCESS or FAILURE is returned. This means that the tree returns if any nodes within the tree return RUNNING. This is so that time and iteration constraints can be tested during each loop and to prevent the tree from blocking other calls the AI may want to make.