AI

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

Overview

Current AI uses SMACH package that is in ROS. SMACH is a task-level architecture for rapidly creating complex robot behavior. At its core, SMACH is a ROS-independent Python library to build hierarchical state machines.

Advantages of SMACH are:

1. rapid development, ability to create complex state machines;
2. ability to quickly change state machines without big code changes
3. explicitly define outcomes of every state thus covering most or all possible situations.

Current AI

Our current AI was re-written using SMACH. There are several utility files such as:

1. gate_util.py - all states that are used by gate AI, they are generic.
2. util.py - contains utility functions for vision to filter labels, get N most probably, normalize coordinates from vision, or wrap yaw. **Note that vision will be changed in future, some of the function will no longer be useful.**
3. basic_states.py - contains all of the states for roulette and dice AI.
4. control_wrapper.py - wrapper made to ease communication with control system, making it easy to send basic commands such as dive, yaw, pitch, roll, move forward.
5. start_switch.py - every high-level state machine **must** have start_switch as their first state. It is a state that waits for ros message to be sent over topic /start_switch to be true at least 3 times.
6. blind_movement.py - contains move_forward state that moves forward with ***x*** speed for ***y*** number of seconds.
7. SubscribeState.py - a state that was made which accept also topic to which you want to subscribe. It is also modified to pass over any input/output keys. **In future this file will also contain SynchronousSubscribeState that subscribes to two topics and moves once it has two**

There is a useful tool to see state machine and transitions of it called **smach_viewer**. To run it, run

```
rosrun smach_viewer smach_viewer.py
```

An example of what our AI looks like in smach_viewer is this screenshot below

Things to know when developing AI

- When inheriting from **SubscribeState** instead of **SmachState**, you need to use **self.exit("outcome")** instead of **return "outcome"**
- Every time you create control_wrapper instance, you need to set depth value again.
- If your state is using control wrapper to move, right before final outcome make sure to set changed yaw/roll/pitch/forward to 0.
- Control wrapper forward and strafe do not use relative same way as yaw and pitch, use instead **strafeLeftError()** and **forwardError()**
- Some current AI files use parameters from roscore server for different configuration of values. They will crash if they do not load parameters. To load parameters run

```
rosparam load [param_file_name].yaml
```

- Every time you restart roscore you need to reload parameters
- Our vision detection requires undistortion to be running.