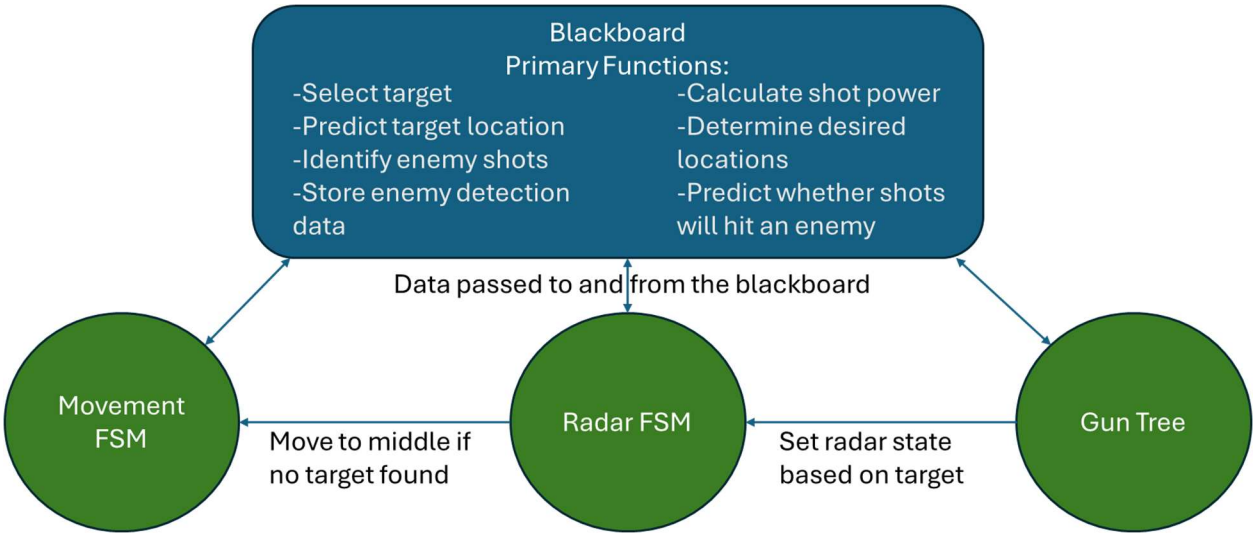
RoboCode: Harvester

Overall Vision:

Simplicity was the primary design philosophy for this bot, though this was not always achieved, and some areas experienced significant scope creep. Overall, this bot has four main structures/behaviors, two finite state machines controlling the movement and the radar, a simple decision tree to control the gun, and a blackboard data structure to store data about the game world. While not a part of the initial design plan, the blackboard also performs most of the calculations that each of the structures required.



Strategy and Development:

The initial goal for this bot was to focus on accurate, long-range fire, and a lot of time and code was put into the various target prediction algorithms. However, the targeting algorithm did not progress in a way that would allow this to be successful. Thankfully, the shot dodging algorithm in the movement FSM was far more successful than anticipated. The development drove a redesign in how the bot was positioned on the battlefield, pivoting to it attempting to maintain a certain desired distance from enemies as opposed to maximizing its safety. As a result, the bot now seems to function best as a medium to short-range "rouge" type bot, agilely dodging shots and getting damage in where it can.

Blackboard:

The blackboard's primary purpose is to store data collected about the world. Some of the functions are shown in the figure above, but there are two primary functions that need to be delved into.

The first is the target prediction algorithm. To begin with, data from ScannedRobotEvents is parsed and stored for each opponent bot, and this data persists from round to round. When there have been less than three detections, a linear prediction algorithm is used. When there have been many detections, a pattern matching algorithm is utilized. The algorithm searches through all past detections and finds the set of detections that most closely resemble the target's recent actions. Then it predicts that the target will follow the same pattern as it did after this window. Finally, if the number of detections is more than three, but less than the threshold needed to trigger this pattern matching, a circular prediction algorithm is utilized.

The second primary algorithm is used to determine a desired position on the board. This algorithm uses a modified intensity map. To accomplish this, it does two things. First, it breaks the arena into 16 tiles. Second, it keeps track of a desired distance that should be maintained from each enemy bot. This value is set to an initial value on first detection, then is modified based on different events. For example, getting hit by an enemy results in the desired distance being increased, and missing the enemy when shooting at it reduces it. Then, in order to limit long duration travel, the algorithm searches only the current and neighboring tiles to determine which tile is the closest match for the desired distances of each of the enemy bots.

Pattern Matching

Recent activity

dV	1	2	1	0	0
dΘ	-10	0	-10	0	-10

Stored detections

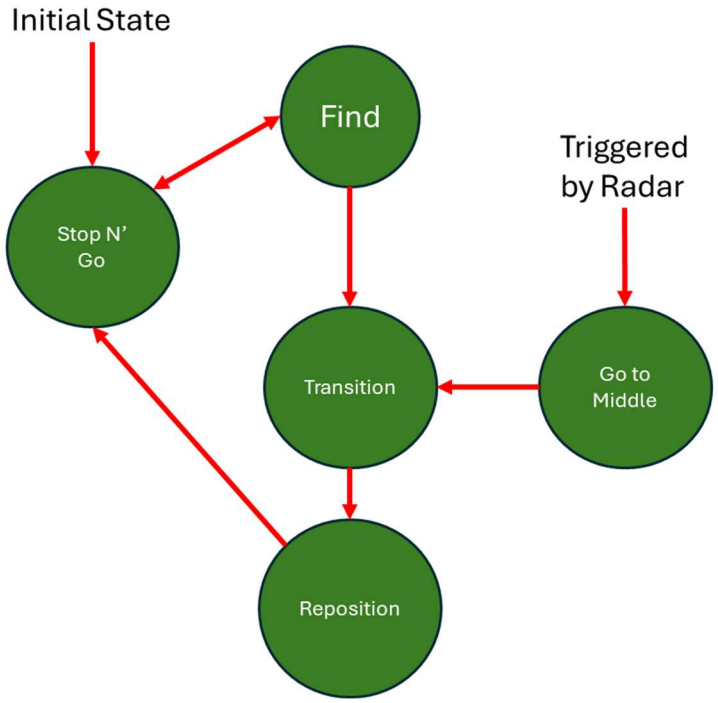
dV	2	2	-2	1	2	1	0	0	-2	-2	0	0	2
dΘ	15	5	0	-10	0	-10	0	-10	0	0	10	15	5

Best match

Future predictions

Movement FSM:

Overall, the FSM that controls bot movement operates under a hub and spoke strategy, with the hub portion implementing a variant of an influence map, and the spoke portion implementing a stop and go bullet dodging strategy once in place. In the find state, which is triggered by a counter reaching 0, the bot asks the blackboard for a new target coordinate, based on the desired position algorithm described above. If this location matches the current target location, the state reverts to the stop and go state, otherwise, a turn towards the point is started and the state is changed to a transition state. The transition state simply waits for the turn to complete before initiating the bot's forward movement, this is done to simplify calculations and avoid running into walls. Once the forward motion is initialized, the reposition state is entered. This state simply wiggles the bot back and forth with a relatively small turn every tick until the target point is reached. At this point the bot enters the stop and go state. This state utilizes the blackboard to track shots from all opponents being detected. It then continuously turns perpendicular to the location of the bot that fired the next incoming shot, and waits until the shot is predicted to be a certain number of ticks away. At this point, it then moves forward or backwards (determined randomly) a short random distance. This minimizes the



amount that the bot moves while attempting to minimize damage received. The final state is the go to middle state. This is triggered when the radar has not detected any opponent for a certain number of ticks. It simply sets the target location to the middle of the arena, turns towards it, and changes to the transition state.

Gun Decision Tree:

The tree used for gun control is relatively simple. It starts by asking the blackboard for a target. If no target is found, the radar is sent into a search state, otherwise the radar is sent a track while scan state. Next, it checks whether the bot can shoot and if a shot at the current gun heading is projected to hit the current target, and if both these conditions are met, a shot is taken. Next, it asks the blackboard to determine an angle to turn the gun to in order to fire at the current target during the next tick and turns towards it.

Radar FSM:

The FSM controlling the radar is also relatively simple. It has two states. A search state, which simply rotates the radar in a circle and a track while scan state. This state is used during target engagement. During the update phase, this state will turn the radar past the predicted target location by half of the maximum radar turn rate. This results in continuous detection on the target as well as maintaining a large radar sweep during every tick, maximizing the amount of information that the blackboard receives.