Robocode Documentation

Schrijf een state diagram uit

Behavior Tree

The behavior tree is instantiated the moment the robot is constructed. Required values of the blackboard are set and 3 behavior trees are initialized: one for the scanner, one for the gun and one for the wheels.

Extra values of the blackboard are set in every OnBullet… method and in the OnScannedRobot method, as well as in the Run method where a few values are updated each frame.

The scanner behavior tree sequences different scanners, but if it works correctly it will only ever execute the first 2. In the event that it DOES reach the last state however, it is a state that should never fail. And the apocalyptic event that it DOES fail, it will just loop it.

The gun behavior tree is really simple. It just sequences through different states until it reaches the end state where it will stay endlessly.

The wheel behavior tree is surely the most elegant of these, as it doesn't use any decorators. Instead of sequencing, it selects the first possible appropriate state. The last one is always appropriate so it will never NOT do anything.

States

Scanner

The scanners states are all variations on either RoamScanning or ScanLock. RoamScanning is used when the robot is not detecting anything in its scanner, and the ScanLock is used when it is.

RoamScanning simply turns the scanner in 1 direction as fast as possible until it picks something up. This is used by RoamScanningToCenter and RoamScanningToOppositeDirection.

RoamScanningToCenter decides if the scanner should turn left or right to scan the center of the battlefield the fastest. This ensures that the largest possible area to find an enemy player is scanned as fast as possible.

RoamScanningToOppositeDirection flips the direction from what it was the last frame. This is used to change the direction of the scanner where a ScanLock to fail (which it never should unless you set its lockFactor value to something ridiculous).

ScanLock is used by ThinLock, NarrowLock and FixedLock. These states, however, change nothing about the ScanLocks execution; they simply have a set lockFactor that is practical for your robot to use.

What ScanLock does is, how to put it… wipe the scanner over the space the other robot was scanned. This makes it scan the other robot again every next frame, which ensures that it will never lose track of it. The lockFactor specifies the range of this wipe. 1 will give you a thin wipe that locks straight onto the other robot. 1.9 starts big and continually narrows down and 2 just wipes the possible area where the other robot could be next frame as well. You would still rather use 1 however, as 2 might skip a frame or 2 of scanning if the area angle is bigger than the maximum turn rate of the scanner.

Gun

All gun states inherit from a GunNodeBase class, which adds some generic functionality useful for all gun states. It has a Fire which handles everything that goes along with firing a bullet (some calculations are made to make sure that this bullet uses the optimal amount of energy) and a GunToEnemyAngle method, which returns the remaining angle to turn towards the enemy, which is essential for targeting.

TurnGunWithScanner is the guns state used when there is no scanned enemy. The gun simple rotates in the same direction as the scanner, as it is assumed the scanner turns in the optimal direction to find an enemy as soon as possible.

TurnGunToEnemy is used when an enemy is scanned but the gun is not within a certain angle margin to shooting this enemy. It simply turns towards the enemy as fast as possible.

FireAtEnemy is constantly turning the gun towards the enemy and firing as much as it can.

Wheels

All move states inherit from a MoveNodeBase class, which adds some generic functionality useful for all move states. It has a TooCloseToWall method, which checks if the robot is dangerously close to a wall.

The wheels first check if they should try to ram the opponent, which they will do if there is a certain energy difference between the robot and the opponent. It simply turns towards the enemy and just moves towards it as fast as it can.

If the Ram state isn't applicable, it will check if it should move away from the wall. It checks if it is close to one, and if it is it turns towards the center and moves towards there until it decides it is far enough from the wall.

If the AwayFromWall state isn't applicable, it will move into its final state: Spin. This… spins. It just moves forward and turns right constantly.