**Useful references and links**:

1. <https://robowiki.net/wiki/Radar>
   1. Has some tips and info about how the radars work.

**IMPORTANT**:

* NOTE FOR LATER: Add a check velocity to methods to modify the search direction, as, according to the wiki, if a robot rotates its radar in the direction it is turning, you can move it up to 65 degrees a tick verses 45 degrees if you don’t. Also, this could be used to set which way to search if the robot’s current target moves out of sight.

**Total targeting strategy behaviors**: 4

1. WeakRobot – the robot focuses on attacking a robot with low energy.
2. FocusAttack – once the robot targets a robot, it attacks it until it is destroyed.
3. Defense – the robot targets robots that attacked it.
4. NearestRobot **–** targets the closest robot.

**Additional genetic parameters**:

1. K
2. L

**Not Genetic, but targeting-related parameters**:

1. target – stores the name, energy level, and distance of the robot that our robot is trying to attack.
2. ticksSinceSeen – the number of ticks it has been since target has been detected.
3. targetLost – threshold for target to be considered dead or lost to the robot and for it to begin searching for a new target.
4. searchAngle – the amount to rotate the scanner. Primarily used when following target’s movements.
5. numOpponents – the number of other robots on the field.
6. scanCount – the number of robots that have been scanned.

**Targeting()**

Narrative: Conducts the general search for another robot.

Pre-condition: None

Post-condition: The robot has selected a target.

If target is null

Set numOpponents equal to the number of remaining opponents

Turn the radar clockwise until a robot is detected

Else

Turn the radar by searchAngle

Increment ticksSinceSeen

**OnScannedRobot(ScannedRobotEvent e)**

Narrative: The robot has detected another robot and will now select the appropriate behavior.

Pre-condition: None

Post-condition: The robot has performed its action on its target.

Increment scanCount

If target is null

Set target.name equal to e.name

Set target.energy equal to e.energy

Set target.distance equal to e.distance

Set scanCount equal to 1

Set ticksSinceSeen equal to zero

Else if target.name is equal to e.name

Set target.energy equal to e.energy

Set target.distance equal to e.distance

Set ticksSinceSeen equal to zero

If targeting strategy is weak:

WeakRobot(e)

Else If targeting strategy is focused:

FocusedAttack(e)

Else If targeting strategy is defense:

Defense(e)

Else If targeting strategy is nearest:

NearestRobot(e)

Else

Report that there is an invalid targeting strategy

If ticksSinceSeen is greater than or equal to targetLost

Set target equal to null

**WeakRobot(ScannedRobotEvent e)**

Narrative: The robot targets the weakest (lowest energy) robot it has encountered.

Pre-condition: None

Post-condition: The robot targets the weakest robot.

If target.name is not equal to e.name

If e.energy is less than target.energy

Set target.name equal to e.name

Set target.energy equal to e.energy

Set target.distance equal to e.distance

Set ticksSinceSeen equal to zero

Else

Return

If scanCount is greater than or equal to numOpponent

Call CalcAngle(e)

Call BulletStrategy(e)

Else

Return

**FocusedAttack(ScannedRobotEvent e)**

Narrative: Once the robot has found another robot, it will target that robot until it is dead or cannot be found.

Pre-condition: None

Post-condition: Robot attacks its target.

If target.name is not equal to e.name

Return

Call CalcAngle(e)

Call BulletStrategy(e)

**Defense(ScannedRobotEvent e)**

Narrative: The robot targets any robot that attacked it.

Pre-condition: This method would require a HitByBullet Event to direct the scanner.

Post-condition: The robot will attack its attacker.

Call BulletStrategy(e)

**NearestRobot(ScannedRobotEvent e)**

Narrative: The robot targets whatever robot is closest to it.

Pre-condition: None

Post-condition: The robot should have fired at e if it is a valid robot.

If target.name is not equal to e.name

If e.distance is less than target.distance

Set target.name equal to e.name

Set target.energy equal to e.energy

Set target.distance equal to e.distance

Set ticksSinceSeen equal to zero

Else

Return

If scanCount is greater than or equal to numOpponent

Call CalcAngle(e)

Call BulletStrategy(e)

Else

Return

**CalcAngle(ScannedRobotEvent e)**

Narrative: Determines the exact location of another robot *e* and calculates the angle between *e* and the robot’s gun and radar.

Pre-condition: None

Post-condition: searchAngle has been set to the angle between the robot’s gun and radar and *e*.

//See TrackFire robot for a possible implementation.

Calculate location of e

Set searchAngle equal to angle between gun (and radar) and e

**OnBulletHit(BulletHitEvent e)**

Narrative: Called whenever the robot is struck by a bullet. Tries to determine where the bullet came from and find the attacker.

Pre-condition: None

Post-condition: The robot will have calculated where the bullet came from and attempt to locate the robot that fired it.

If targeting strategy is defense

Calculate angle of bullet

Set searchAngle equal to the angle between the gun and radar of robot and bullet