Comportamentos a considerar

* Disparar
  + Stationary. Shoot at the current location of the target robot. This is the simplest and least effective targeting strategy, as any movement of the robot causes the bullets to miss.
  + Linear. Shoot at where the target will be, assuming that it moves at a constant velocity in a straight line. This targeting method is very effective, especially when the target is relatively close. All robot motion can be approximated by a straight line for a short time. However, this approximation becomes unreliable fairly quickly.
  + Circular. Shoot at where the target will be, assuming that it moves at a constant velocity and a constant angular velocity. A little better than linear targeting. This targeting can be avoided by changing velocity quickly, such as stopping and starting and moving back and forth quickly.
  + Oscillatory. This strategy assumes that the target robot is moving back and forth continually. This targeting is very effective against robots with this kind of movement, as only a small number of robots use this kind of targeting.
  + Movement pattern matching. This approach records the past movements of the robot and assumes that the robot will move in a repeatable pattern. Shoot at the predicted future position. This kind of targeting has the greatest potential of any of the previous methods. Potential drawbacks are the processing time that it takes for an exhaustive search.
* Descobrir inimigos
  + Simple. Just point the radar in the same direction as the gun.
  + Circular. Keep the radar continually turning.
  + Tracking. Keep the radar locked onto a single target.
  + Optimal. Move the radar in such a way as to scan all of the robots as quickly as possible.
* Deslocar/Desviar
  + Stay still. Easy to hit. Generally a bad idea.
  + Move in a straight line. Avoids being hit by Stationary targeting.
  + Move in a circular curve. Avoids being hit by Stationary and Linear targeting.
  + Move backwards and forwards in an oscillating motion. Difficult to hit using Linear and Circular targeting, but Stationary targeting can work quite well here.
  + Move in a random direction. Can be effective in reducing hits by all levels of targeting, but usually difficult to do in such a way as to not be hit.
  + Advanced movement. Use all kinds of data about the other robots to choose the best place to move to. This is by far the biggest and least understood area of robot strategy. It is one of the main differentiating factors between the best robots and the not-so-good robots.

Dicas

[Anti-gravity movement](http://www.ibm.com/developerworks/library/j-antigrav/index.html)

Anti-gravity movement, in its many modified forms, is the movement type of choice for most expert Robocoders. With it you can define points on the map to avoid, easily create movement patterns, and dodge enemy bullets. Alisdair Owens shows you how to implement this helpful technique.

[Predictive targeting](http://www.ibm.com/developerworks/library/j-pred-targeting/index.html)

All successful targeting and shooting of enemy robots requires an algorithm to fire bullets at the place where you predict that an opponent will be at a future point in time. In this tip, Simon Parker describes an algorithm that can be used for linear, circular, and oscillating predictive targeting.

[Tracking your opponents' movement](http://www.ibm.com/developerworks/java/library/j-movement/index.html)

Every targeting algorithm has limitations because there will always be some movement pattern that it has difficulty predicting. Learn how to make your bot select the best movement algorithm for each opponent.

[Circular targeting](http://www.ibm.com/developerworks/library/j-circular/index.html)

Circular targeting is the next step after you've mastered linear targeting. Using slightly more advanced mathematics, this system allows you to hit robots that travel in circles with perfect accuracy, all the while retaining effectiveness against those that travel in straight lines. Alisdair Owens shows you how to implement this technique and provides an example bot to take out for a test drive.

[Dodge bullets](http://www.ibm.com/developerworks/library/j-dodge/index.html)

To make a winning Robocode robot, you must be able to hit your opponent more than your opponent hits you. Making your robot target your opponent is a fine art, but what if your opponent couldn't target you? Using a simple trick and some guesswork, DodgeBot, listed here, shows you how to dodge bullets.

[Radar sweeps](http://www.ibm.com/developerworks/library/j-radar.html)

In this tip, learn how to make the most efficient use of your robot's radar, to get the most up-to-date information on your opponents' location.

[Polymorphic enemy cache](http://www.ibm.com/developerworks/library/j-tippoly/)

Successful robots maintain a store of information, accessible at any time for making crucial decisions while in battle. This is useful for a variety of reasons, from enemy movement pattern analysis to determining whom to attack based on proximity and strength. This tip explains how you can implement an effective, fast enemy cache while having the convenience of an always up-to-date object using polymorphism.

5 Robots

1. 2 Seekers
   1. Aplicar Radar Sweep para encontrar o máximo de informação sobre o terreno e os outros robots;
   2. Mandar ordens aos “Destroyers” sobre como se deslocar;
   3. Evitar estar parado e evitar disparar (os destroyers é que devem disparar) de forma a prolongar o seu período de vida, aumentando assim as hipóteses de vitória;
   4. Este robot deve usar os seguintes algoritmos para ordenar aos restantes robots
      1. Anti-gravity Movement
      2. Predictive Targeting/Targeting opponent movement
   5. Se todos os destroyers morrerem, este deve assumir o comportamento de destroyer ou evitar ser atingido, consoante a energia que os robots inimigos ainda tenham
2. 3 Destroyers
   1. Este robot não irá ter radar, portanto depende do robot seeker para lhe informar o que fazer;
   2. Primeiramente irá tentar andar em forma de 8 ou algo do gênero para evitar ser atingido enquanto o seeker recolhe informação sobre o jogo, até receber ordens do seeker irá continuar com este comportamento;
   3. Assim que tiver ordens do seeker irá seguir as ordens do seeker, estas podem podem:
      1. Roda <Graus>
      2. Anda <paços>
      3. Dispara <potência>
      4. Ou uma combinação das ordens acima ordenadas
   4. Além disso o seeker pode lhe informar de áreas a não frequentar, que este irá guardar numa lista, para se quando este se move, se encontrar alguma dessas áreas pelo caminho, tentar se desviar;
   5. No caso de ambos os seekers morrerem, ou de não estarem a enviar ordens a estes robots, eles devem assumir um comportamento que aumente a probabilidade de este conseguir ganhar aos restantes, sem ter informações sobre eles, ou seja, evitar as áreas perigosas que já foram informadas pelo seeker, e tentar percorrer o máximo de caminho possível para evitar ser atingido por alguma bala