Secrets from the Robocode masters: Dodge bullets

Jae Marsh

May, 2002

This tip was published in the "Cloak and turret: Learn secrets from the Robocode masters" article in the May 2002 issue of the IBM developerWorks journal.

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.

Before I show you the trick, you must know some Robocode basics. First, Robocode uses a single energy store for both offense and defense. This leads to interesting game play decisions: you have to decide when to use your energy to fire at the opponent and when to keep it in reserve to withstand potential damage.

Second, your robot has very limited information about the world around it. It can see the distance, bearing, heading, velocity, and energy level of other robots. No, it cannot see bullets, but maybe from these clues, you can guess how to detect when they are fired.

DodgeBot stands still and continuously tracks the previous energy level of the opponent. When it drops by a certain range, DodgeBot assumes the opponent fired and it moves left or right. Surprisingly, this will baffle most robots' targeting methods. Either they will fire directly at their target, or they will try to project your position based on your speed and direction. Since your robot is not moving, both algorithms will fire right at it. And your robot, guessing this, will make a small hop to the side and let the bullet continue on its way. Figure 1 shows DodgeBot in action.
Figure 1. DodgeBot versus Tracker (Gotcha!)

Listing 1 shows the DodgeBot code. The main block of code is executed every time the radar senses an enemy. First, DodgeBot maintains itself at right angles plus a 30-degree incline toward the opponent. Staying at 90 degrees gives a robot the greatest ability to move out of the way of bullets. The extra 30-degree incline causes the robot to be a bit aggressive and move closer to its target over time. Next is the key piece of code: if the robot senses an energy drop between 0.1 and 3.0 (the range of firing power), the robot will immediately switch directions and move to the left or right. Simple. It will switch the direction of its gun and radar sweep, assuming that if it saw a robot in its last sweep, it will find it again if it sweeps over the same area. Then the robot will fire. Because I left the gun and the radar linked and because the scanner will be called at the exact moment it is facing the opponent, the gun will fire directly at the opponent. Finally, I record the opponent's energy for the next time around.

Listing 1. DodgeBot code

```java
import robocode.*;

public class DodgeBot extends AdvancedRobot {
    double previousEnergy = 100;
    int movementDirection = 1;
    int gunDirection = 1;
    public void run() {
        setTurnGunRight(99999);
    }
    public void onScannedRobot(ScannedRobotEvent e) {
        // Stay at right angles to the opponent
        setTurnRight(e.getBearing()+90-30*movementDirection);

        // If the bot has small energy drop,
        // assume it fired
        double changeInEnergy = previousEnergy-e.getEnergy();
        if (changeInEnergy>0 && changeInEnergy<=3) {
            // Dodge!
            movementDirection = -movementDirection;
            setAhead((e.getDistance()/4+25)movementDirection);
        }

        // When a bot is spotted,
        // sweep the gun and radar
        gunDirection = -gunDirection;
    }
}
```
The robot that I used this trick on, Wolverine, uses some additional sensor information to make the sensing more accurate. When an opponent hits my robot, the opponent will be recharged. When my robot hits an opponent, the opponent's energy level will drop. A robot can sense these two events, and Wolverine uses the information to cancel out the corresponding energy fluctuations.

There are problems with this trick. The robot moves as soon as a bullet is fired, so that it may end up moving back into the line of fire. It might be better to move when the bullet is expected to arrive.

An even bigger problem is that even though you can confuse common targeting, side-to-side stepwise movement is, in fact, quite predictable. The best way to use this trick is, perhaps, to let the information guide your movements, but not control them.

You may be thinking that this trick is so simple that you could have thought of it yourself. Good. That is how this game is played, and that is exactly why it is so addictive. Robocode is like a game of chess where each new move is a new idea.

© Copyright IBM Corporation 2002
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)