Build the best, destroy the rest

Regis Pires Magalhães
regispiresmag@gmail.com

Motivação

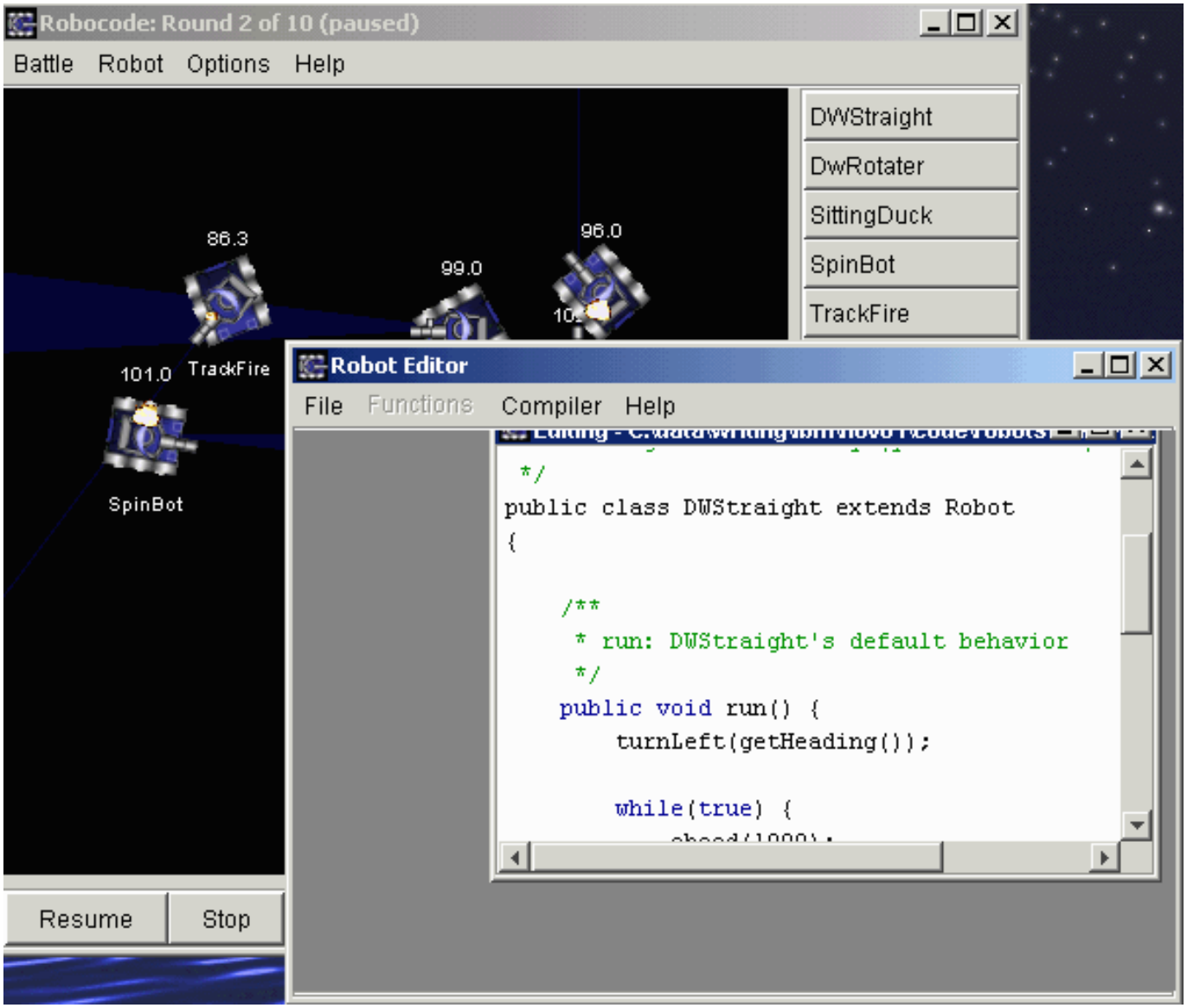
“Parte da motivação para escrever o Robocode foi provar ao mundo que as sentenças 'Java é lento' e 'Você não pode escrever jogos em Java' não são mais verdadeiras. Eu acho que consegui.”

Mathew Nelson – Criador do Robocode

Robocode IDE

Campo de Batalha

Editor de Robôs



Campo de Batalha

Anatomia de um Robô

O robô tem um canhão giratório e acima dele há um radar giratório.

O robô, o canhão e o radar podem girar independentemente a qualquer momento.

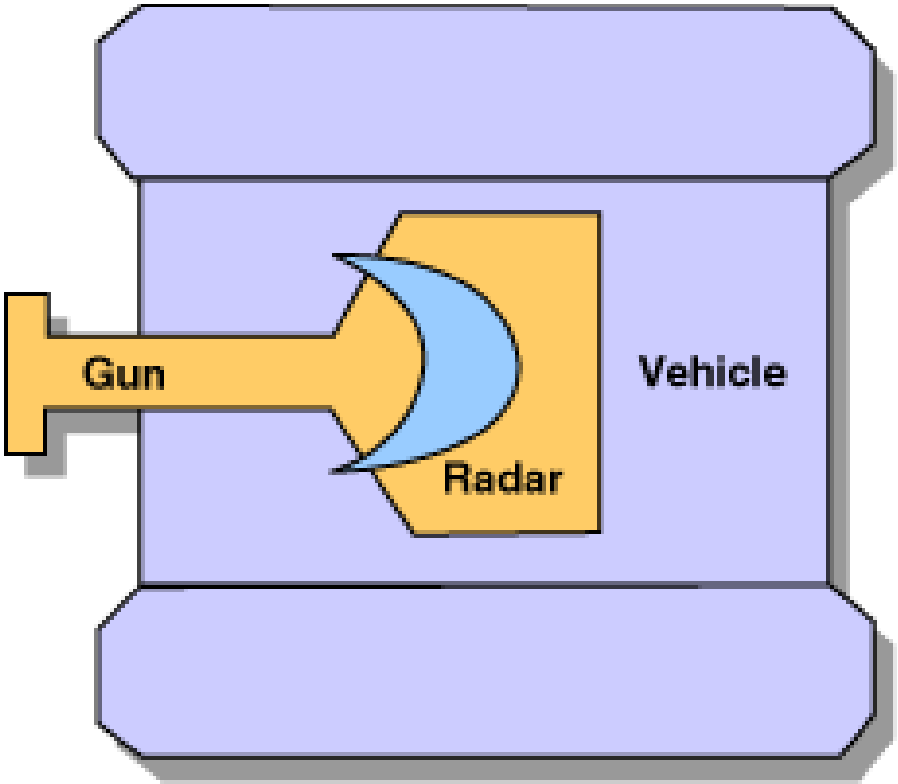
Por padrão, eles estão alinhados de acordo com a direção do movimento do robô.

Por padrão, o radar movimenta-se em conjunto com o canhão.

Veículo

Canhão

Radar



Restrições de um Robô

Energia

- Utilizada nas operações do Robô;
- No disparo, pode ser definida a quantidade de energia utilizada;
- É recuperada quando acerta-se outro robô.

Calor

- Um canhão só dispara quando seu calor estiver em zero. O calor gerado é proporcional à potência do disparo.

Comportamentos de um Robô

Vários métodos estão disponíveis na classe Robot.

Movimentação do Robô

`turnRight(double degree) / turnLeft(double degree)`

Giram o robô de acordo com o grau especificado.

`ahead(double distance) / back(double distance)`

Movem o robô de acordo com a distância especificada em pixels;

Finalizados quando o robô esbarra numa parede ou em outro robô.

`turnGunRight(double degree) / turnGunLeft(double degree)`

Giram o canhão, independente do veículo.

`turnRadarRight(double degree) / turnRadarLeft(double degree)`

Giram o radar, independente o canhão e do veículo.

Movimentação do Robô

Quando o robô gira, o canhão e o radar também giram, a não ser que um dos seguintes métodos seja usado:

`setAdjustGunForRobotTurn(boolean flag)`

Se flag for true, gira o canhão junto com o veículo.

`setAdjustRadarForRobotTurn(boolean flag)`

Se flag for true, gira o radar junto com o veículo.

`setAdjustRadarForGunTurn(boolean flag)`

Se flag for true, gira o radar junto com o canhão.

Informações sobre o Robô

`getX() / getY()`

Obtém a coordenada atual do robô.

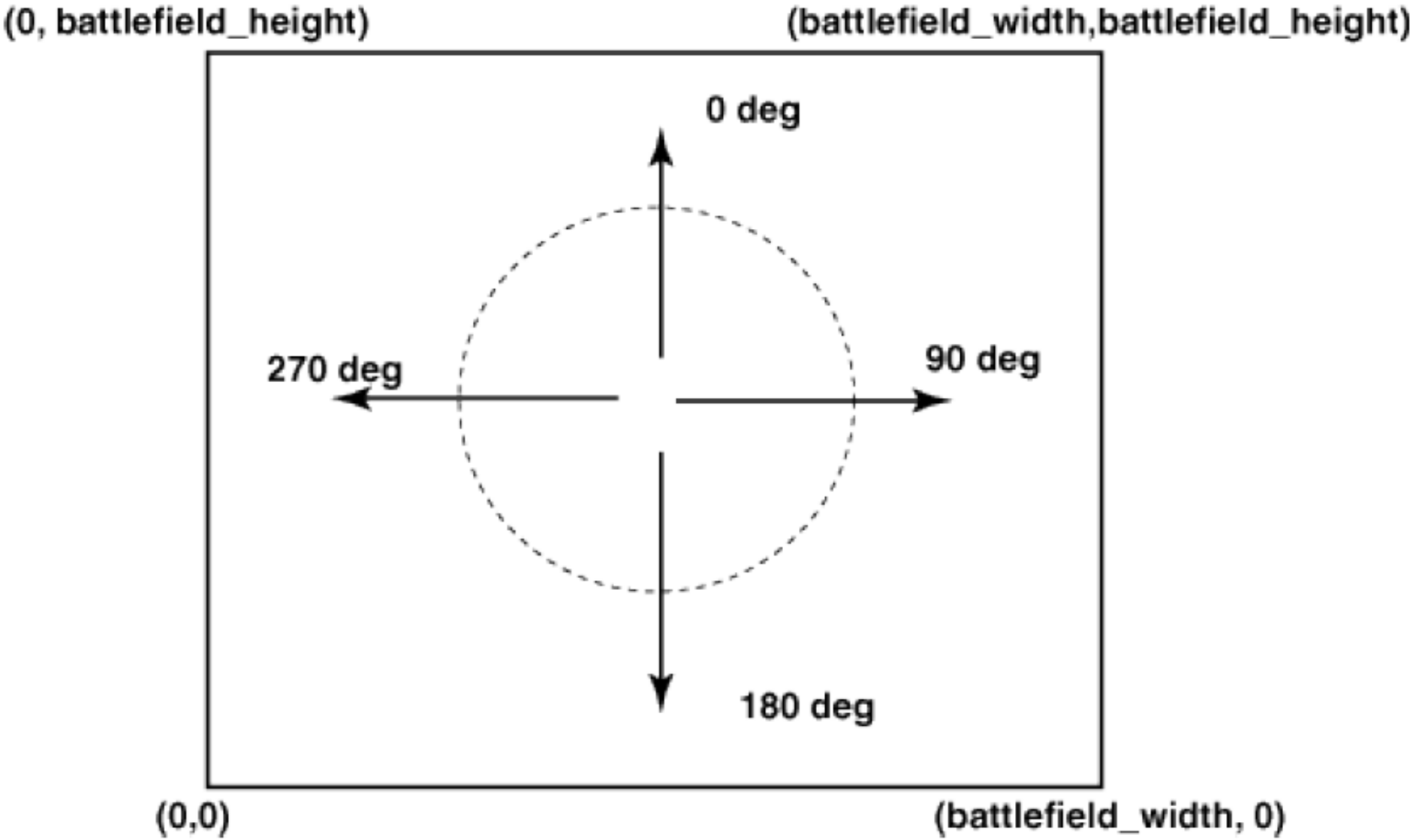
`getHeading() / getGunHeading() / getRadarHeading()`

Obtém a direção atual do robô, canhão ou radar em graus.

`getBattleFieldWidth() / getBattleFieldHeight()`

Obtém as dimensões do campo de batalha.

Navegação


(0, battlefield_height)
(battlefield_width,battlefield_height)
0 deg
270 deg
90 deg
180 deg
(0,0)
(battlefield_width, 0)

Disparo

Cada robô inicia com um nível de energia padrão e é destruído quando seu nível de energia atinge zero.

Ao fazer um disparo, o robô pode usar até 3 unidades de energia. Quando mais energia usada no disparo, mais danos causará ao robô atacado.

`fire(double power) / fireBullet(double power)`

- Usados para fazer um disparo com a energia especificada.
- O método fireBullet retorna uma referência ao objeto robocode.Bullet que pode ser usada em robôs avançados.

Eventos

O radar está sempre ativo e dispara um evento se detectar a presença algum robô dentro do seu limite de alcance.

O criador de um robô pode tratar vários eventos que podem ocorrer durante a batalha.

A classe Robot já tem tratamentos padrões para todos esses eventos. Entretanto, pode-se sobrescrevê-los.

Eventos

`onScannedRobot(ScannedRobotEvent e)`

Método chamado quando um robô for detectado pelo radar.

`onHitByBullet(HitByBulletEvent e)`

Método executado quando o robô é atingido por um disparo.

`onHitRobot(HitRobotEvent e)`

Método executado quando o robô colide com outro robô.

`onHitWall(HitWallEvent e)`

Método executado quando o robô colide com uma parede.

Mais documentação

Para mais documentação, acesse a documentação Javadoc da API do Robocode que pode ser acessada do menu Help do campo de batalha ou do menu Help do editor de robôs.

Exemplo de Robô

```
import robocode.*;

public class Asimov extends Robot {
  public void run() {
    while(true) {
      ahead(100);
      turnRight(90);
    }
  }
}
```

Exemplo de Robô

```
package dw;
import robocode.*;

public class DWStraight extends Robot {
   public void run() {
      turnLeft(getHeading());
      while(true) {
         ahead(1000);
         turnRight(90);
      }
   }
   public void onScannedRobot(ScannedRobotEvent e) {
      fire(1);
   }
   public void onHitByBullet(HitByBulletEvent e) {
      turnLeft(180);
   }

}
```

Robôs Avançados

Enquanto um robô comum faz apenas uma coisa de cada vez, no robô avançado você primeiro define as ações e depois pede para que sejam executadas.

Características:

- Múltiplos movimentos simultaneamente;
- Pode ser definida toda uma estratégia a cada tique de relógio;
- Podem ser definidos eventos personalizados;
- Pode ter um arquivo de dados.

Blocking x Non-Blocking

- turnRight() x setTurnRight()
- turnRight() x setTurnRight()
- ...

Exemplo de Robô

```
import robocode.*;

public class Asimov extends AdvancedRobot {
   public void run() {
      while(true) {
         setAhead(100);
         setTurnRight(90);
         execute();
      }
   }
}
```