Robocode

2011-06-25, 2017-06-27

by

Dr. Günter Kolousek

Überblick

Was

- **Programmierspiel**
- Roboter werden programmiert
- …in Java
- …und kämpfen gegeneinander
- eigene Entwicklungsumgebung

Installation und Start

- Installation

  ```
  java -jar robocode-x.x.x.x-setup.jar
  ```

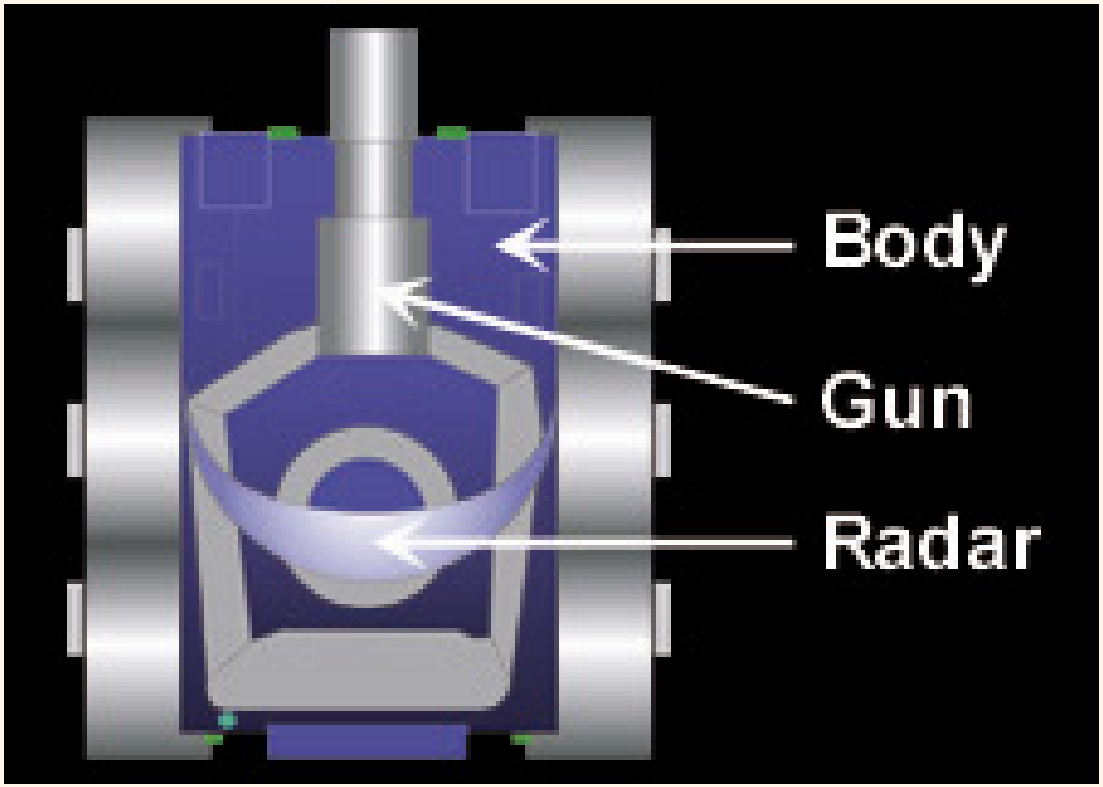
- Robocode ausführen

  ```
  ~/robocode/robocode.sh
  ```

Aufbau eines Roboters

- Unterbau (Body)
  - trägt die Kanone mit dem Radar
  - vorwärts bzw. rückwärts fahren
  - nach links bzw. rechts drehen
- Gun (Kanone)
  - ist am Body montiert
  - nach links bzw. rechts drehen
- Radar
  - montiert auf der Kanone!
  - nach links bzw. rechts drehen
  - generiert `onScannedRobot` Ereignisse (Event)

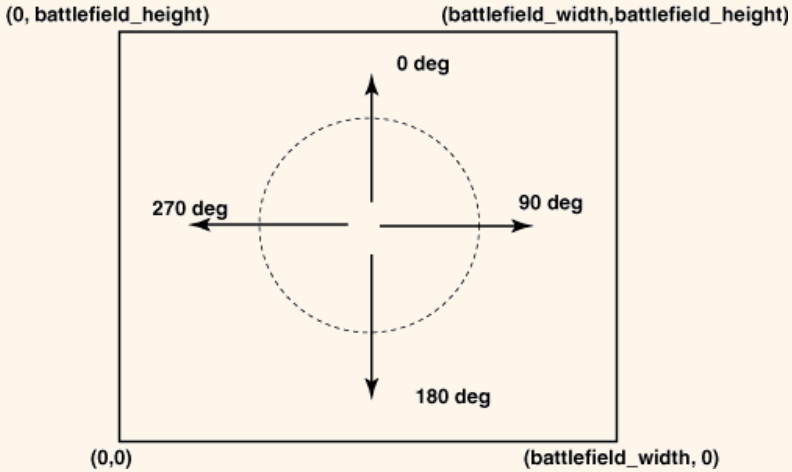
Aufbau eines Roboters – 2



Spielphysik

Koordinaten und Richtungen

- Kartesisches Koordinatensystem: `(0, 0)` links unten
- Einheit: Pixel, aber Gleitkommazahl!
- Roboter können Spielfeld nicht verlassen
- Richtungen
  - im Uhrzeigersinn
  - 0 deg (Grad) schaut nach Norden, 90 nach Osten,…
  - Richtungen im Überblick



Zeit, Geschwindigkeit, Beschl., Weg

time, velocity, acceleration, distance

- Zeit wird in *ticks* gemessen
- Jeder Roboter bekommt 1 tick je Runde (turn)
- 1 tick = 1 turn
- Beschleunigung: 1 pixel/turn/turn
- Verzögerung (Bremsen): 2 pixel/turn/turn
- Geschwindigkeit: $v = at$ (Maxium: 8 pixel/turn)
- Weg: $s = vt$

Max. Rotation

- … des Roboters: (10 - 0.75 * abs(velocity)) deg / turn
- … der Kanone: 20 deg / turn
  - zusätzlich zur Rotation des Roboters
- … des Radars: 45 deg / turn
  - zusätzlich zur Rotation der Kanone

Bullets

- Schaden: 4 * firepower
  - Wenn firepower > 1, dann:
    - zusätzlicher Schaden: 2 * (firepower - 1)
- Geschwindigkeit: 20 - 3 * firepower
- GunHeat: 1 + firepower/5
  - gunHeat > 0: kein Feuern möglich
  - gunHeat wird um 0.1 je Turn weniger
- gewonnene Power bei Treffer: 3 * firepower

Kollisionen

- mit anderen Robotern: jeder 0.6
- mit der Wand (nur bei AdvancedRobots): $0.5 \cdot \text{abs}(\text{velocity}) - 1$
  - aber nicht kleiner als 0)

Ablauf

1. Ansicht wird neu gezeichnet
2. Jeder Roboter führt seinen Programmcode aus (bis Aktion, dann Pause)
3. Zeit wird aktualisiert (time = time + 1)
4. Alle Bullets bewegen sich und es wird auf Kollisionen überprüft
5. Jeder Roboter bewegt sich (gun, radar, heading, acceleration, velocity, distance)
6. Jeder Roboter führt Scans aus
7. Jeder Roboter führt Aktion aus
8. Jeder Roboter fragt Ereignisse ab (event queue)

Programmierung

Neuer Roboter

1. Robocode: Robot $\rightarrow$ Editor
2. Robot Editor: File $\rightarrow$ New Robot
3. Name beginnt mit Großbuchstaben, z.B.: SimpleRobo
4. Paketname beginnt üblicherweise mit Kleinbuchstabe, z.B.: ko

Sourcecode

```
package ko;
import robocode.*;

public class SimpleRobo extends JuniorRobot {
    public void run() {
        // Initialization should be put here:
        // body, gun, radar, bullet, scan_arc
        setColors(orange,blue,white,yellow,black);

        while(true) { // replace the next 4 lines!
            ahead(100);
            turnGunRight(360);
            back(100);
            turnGunRight(360);
        }
    }
```

Sourcecode 2

```
        public void onScannedRobot() {
            // Power zwischen 0.1 (min) und 3 (max)
            // -> scannedDistance, scannedAngle,
            //    scannedBearing, scannedEnergy
            fire(1);
        }

        public void onHitByBullet() {
            // modified instanc variables -> see doc!
            back(10);
        }

        public void onHitWall() {
           back(20);
        }
    }
```

JuniorRobot

- einfachster Roboter
- Instanzvariablen zum Zugriff auf Eigenschaften
  - z.B.: `scannedBearing` … Winkel zum nähesten Roboter relativ zum Body
    - bearing … Peilung
- Viele Befehle blockieren bis abgeschlossen
- …und benötigen einen turn
- …aber: andere wie setzen der Farben geht sofort und benötigt keine Zeit

Grundbefehle

- `turnRight(double degree)`, `turnLeft(double degree)`
- `ahead(double distance)`,`back(double distance)`
- `turnGunRight(double degree)`, `turnGunLeft(double degree)`
- `turnRadarRight(double degree)`, `turnRadarLeft(double degree)`

blockieren!

Drehungen...

- `setAdjustGunForRobotTurn(boolean flag)` ... wenn `true`, dann bleibt Gun in der selben Richtung, wenn sich Roboter dreht
- `setAdjustRadarForRobotTurn(boolean flag)` ... wenn `true`, dann bleibt Radar in der selben Richtung, wenn sich Robotor (und Gun) dreht
- `setAdjustRadarForGunTurn(boolean flag)` ... wenn `true`, dann bleibt Radar in der selben Richtung, wenn sich Gun dreht

Informationsabfrage

- `getX()` und `getY()`
- `getHeading()`, `getGunHeading()`, `getRadarHeading()`
- `getBattleFieldWidth()`, `getBattleFieldHeight()`

Feuern und Events

- Feuern
  - `fire(double)` bzw. `fireBullet(double)` (gleiches Verhalten in JuniorRobot)
    - bis zu Energieeinheiten können beim Feuern verwendet werden
- Events
  - Radar ist beim Drehen und Bewegen immer aktiv
  - `onScannedRobot()` … Radar hat Roboter erkannt
  - `onHitByBullet()` … getroffen!
  - `onHitRobot()` … Zusammenstoß mit anderem Roboter
  - `onHitWall()` … Zusammenstoß mit Wand

Übersetzen und ausführen

- Javaprogramm muss in Zwischencode übersetzt werden
- dann Battle erstellen
- Vorgang:
  1. Robot Editor: Compiler $\rightarrow$ Compile
  2. Robocode: Battle $\rightarrow$ New
  3. Roboter hinzufügen
  4. $\rightarrow$ Start Battle
  5. Wenn fertig, dann Resultate ansehen…
- Neue (fremde) Roboter in das folgende Verzeichnis kopieren: `~/robocode/robots/<package>`

Start

- Los geht's!
- Weitere Doku: `~/robocode/javadoc/index.html`