RoboCode

Funktionalitäten des Roboters

Christian Riess

Eva Eibenberger

Pattern Recognition Lab (Computer Science Dep. 5)

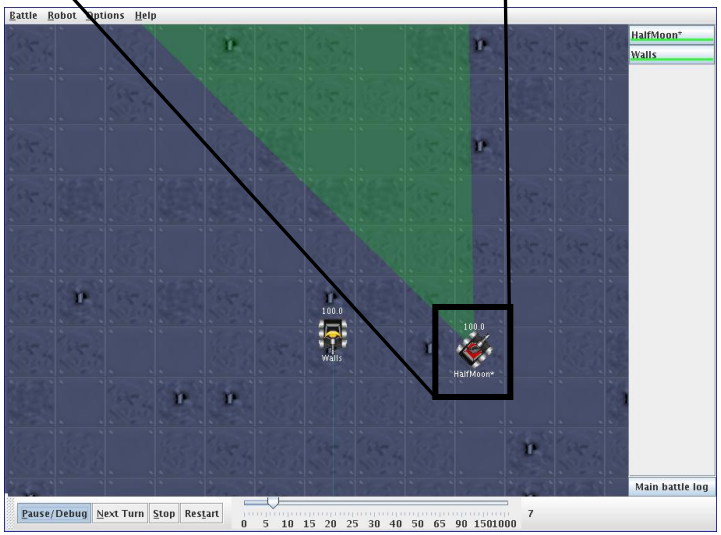
Friedrich-Alexander-University Erlangen-Nuremberg

Übersicht

- Funktionalitäten des Roboters
  - Aufbau eines Roboters
  - Energie des Roboters
  - Steuern des Roboters
  - Mögliche Ereignisse
  - Gegnerische Roboter
- Mögliche Strategien
  - Ausrichten der Kanone
  - Kollision mit der Wand
  - Energie beim Schießen
  - Bewegung
  - Bewegung nachdem Roboter getroffen wurde

Aufbau eines Roboters

- Komponenten des Roboters
  - Fahrgestell
    - Drehen nach links und rechts (langsam)
    - Vorwärts- und Rückwärts-Bewegung
  - Kanone
    - Drehen nach links und rechts (etwas schneller)
    - Schießen
  - Radar
    - Drehen nach links und rechts (schnell)
    - Finden des Gegners
- Beachte:
  - Unterschiedliche Ausrichtung der Komponenten zueinander möglich
  - Strategien für Bewegung, Anvisieren des Gegners und Radarbewegung sollten dies und die verschiedenen Fahr-/ Drehgeschwindigkeiten berücksichtigen



Energie eines Roboters

- Anfangsenergie: 100 Lebenspunkte
- Energieverbrauch
  - Schießen mit Schussenergie `pow`
    - Energieverbrauch ist so groß wie `pow`
  - Getroffen werden:
    - Falls Schussenergie pow > 1: `- 4 x pow - 2 x(pow-1)`
    - Andernfalls: `- 4 x pow`
  - Rammen (Gegner oder Wand)
- Energiegewinn:
  - Erfolgreich auf gegnerischen Roboter schießen: `+ 3 x Schussenergie`
- Tipp:
  - Um mehr Feedback zu bekommen: Setze die die Roboterfarbe entsprechend der verbleibenden Energie

Energie eines Roboters

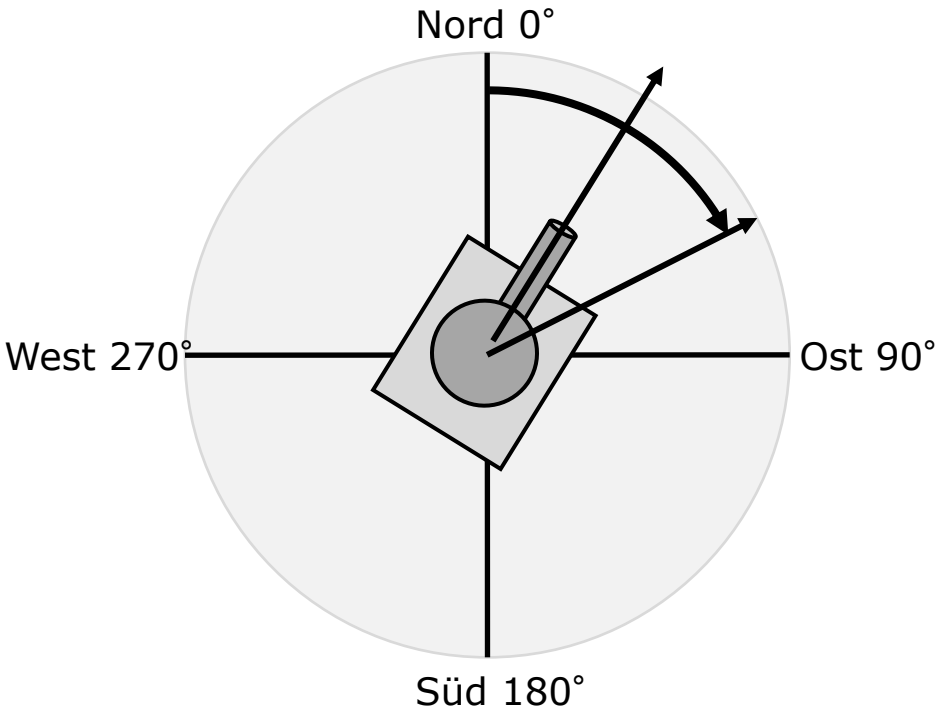
- Beispiel:

  Um visuelles Feedback über den Energiezustand des Roboter zu bekommen:
  Setze die die Roboterfarbe entsprechend der verbleibenden Energie

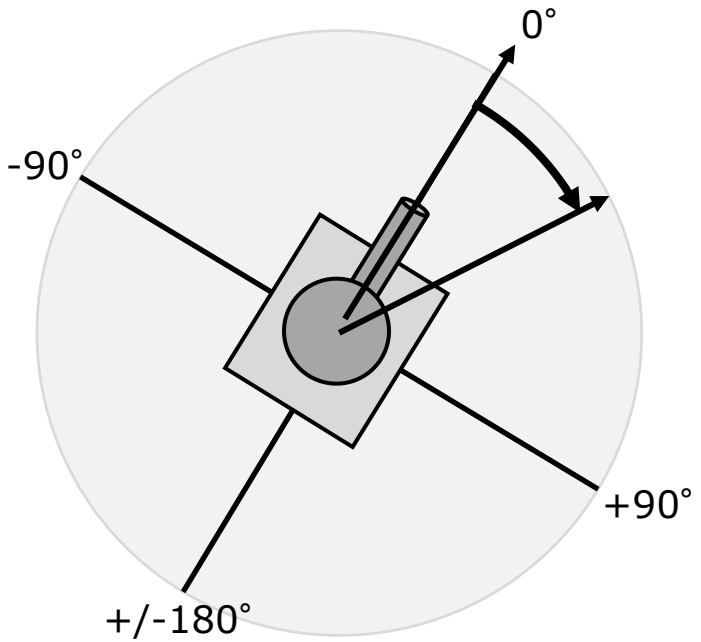
Erlangen von Informationen …

- Heading vs. Bearing
  - Heading: absolute Richtung (0° bis 360°)
  - Bearing:
    relative Richtung (relativ zur Orientierung des Roboters; -180° bis 180°)

Heading



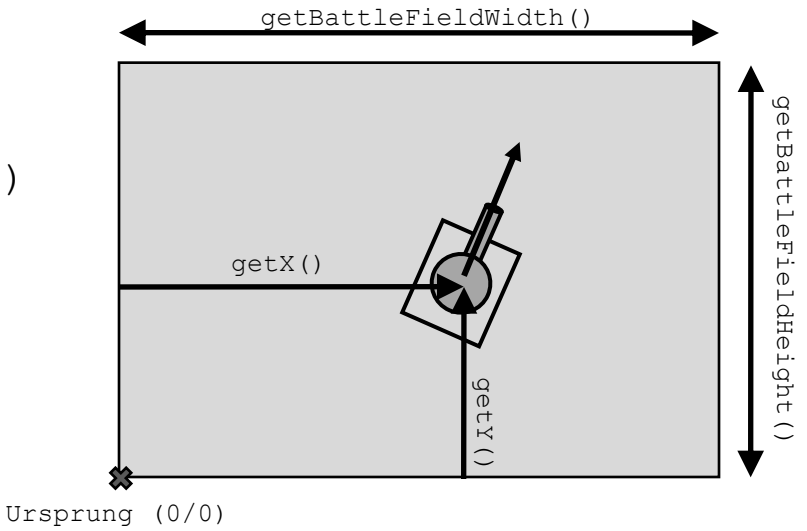
Bearing



Erlangen von Informationen …

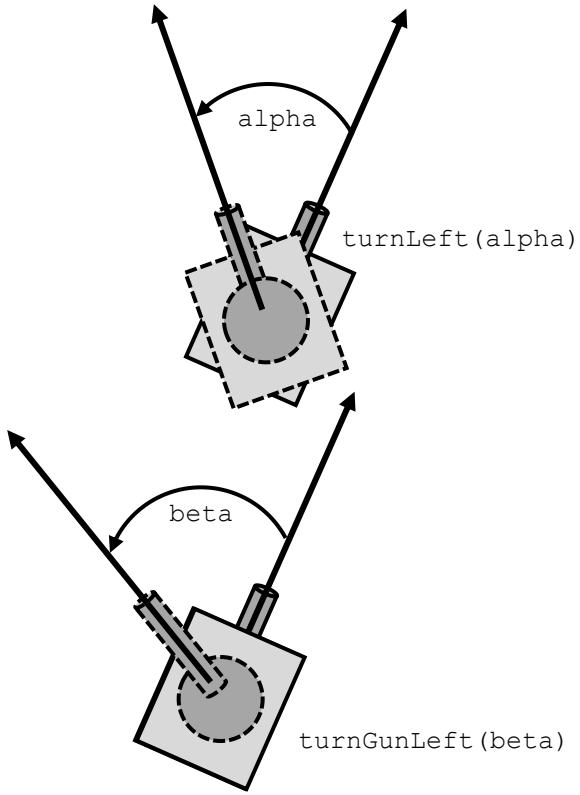
- … über den Roboter selbst
  - Absolute Orientierung des Roboterkörpers: `double getHeading()`
  - Absolute Orientierung des Gewehrs: `double getGunHeading()`
  - Absolute Orientierung des Radars: `double getRadarHeading()`
  - Energie des Roboters: `double getEnergy()`
  - X- und Y-Position: `double getX() / getY()`
  - Höhe und Breite des Roboters: `double getHeight() / getWidth()`
  - Geschwindigkeit des Roboters: `double getVelocity()`

- … über das Spiel/Spielfeld
  - Höhe und Breite des Spielfelds:
    `double getBattleFieldHeight()`
    `double getBattleFieldWidth()`
  - Anzahl der Gegner auf dem Feld:
    `double getOthers()`



Steuern des Roboters

- Steuern des Roboters:
  - Bewegt den Roboter um `dist` Pixel vorwärts/rückwärts:
    ```
    void ahead(double dist)
    void back(double dist)
    ```
  - Dreht den Roboter-Körper um `deg` Grad nach links/rechts:
    ```
    void turnLeft(double deg)
    void turnRight(double deg)
    ```
  - Dreht die Kanone um `deg` Grad nach rechts/links:
    ```
    void turnGunLeft(double deg)
    void turnGunRight(double deg)
    ```
  - Dreht den Radar um `deg` Grad nach rechts/links:
    ```
    void turnRadarLeft(double deg)
    void turnRadarRight(double deg)
    ```



  - Feuert eine Kugel mit der Stärke `power` (der Roboter verliert `power` Energiepunkte):
    ```
    void fire(double power)
    ```
  - Hält den Roboter an und lässt ihn wieder weiter machen:
    ```
    void stop()
    void resume()
    ```

Beispiele für Ereignisbehandlungen (1/3)

- Diese Methode wird aufgerufen, wenn …
  - … eine eigene Kugel den Gegner trifft:
    `void onBulletHit(BulletHitEvent e)`
  - … der eigene Roboter von einer Kugel getroffen wird:
    `void onHitByBullet(HitByBulletEvent e)`
  - … eine eigene Kugel den Gegner verfehlt hat:
    `void onBulletMissed(BulletMissedEvent e)`
  - … der eigene Roboter die Wand gerammt hat:
    `void onHitWall(HitWallEvent e)`
  - … der eigene Roboter einen gegnerischen Roboter gerammt hat:
    `void onHitRobot(HitRobotEvent e)`
  - … der eigene Radar einen gegnerischen Roboter gefunden hat:
    `void onScannedRobot(ScannedRobotEvent e)`
  - … der eigene Roboter die Runde gewonnen hat:
    `void onWin(WinEvent e)`

Beispiele für Ereignisbehandlungen (1/3)

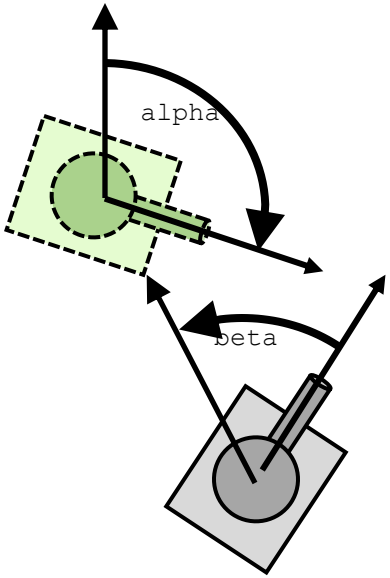
- Diese Methode wird aufgerufen, wenn …
  - … eine eigene Kugel den Gegner trifft:
    `void onBulletHit(BulletHitEvent e)`
  - … der eigene Roboter von einer Kugel getroffen wird:
    `void onHitByBullet(HitByBulletEvent e)`
  - … eine eigene Kugel den Gegner verfehlt hat:
    `void onBulletMissed(BulletMissedEvent e)`
  - … der eigene Roboter die Wand gerammt hat:
    `void onHitWall(HitWallEvent e)`
  - … der eigene Roboter einen gegnerischen Roboter gerammt hat:
    `void onHitRobot(HitRobotEvent e)`
  - … der eigene Radar einen gegnerischen Roboter gefunden hat:
    `void onScannedRobot(ScannedRobotEvent e)`
  - … der eigene Roboter die Runde gewonnen hat:
    `void onWin(WinEvent e)`

Die verschiedenen Ereignisse bieten einen guten Anhaltspunkt, welche Probleme berücksichtigt/verbessert werden können.

Beispiele für Ereignisbehandlungen (2/3)

- Innerhalb der Ereignisbehandlung können weitere Informationen zum jeweiligen Ereignis abgefragt werden

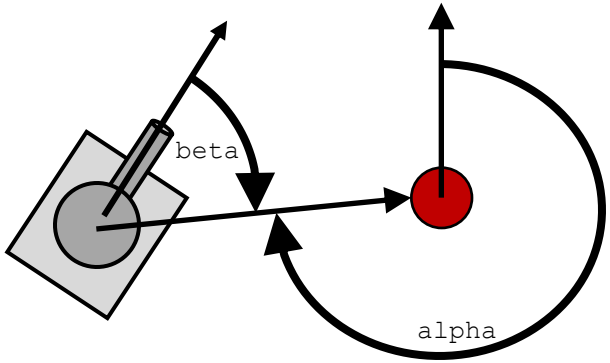
- Beispiel: `void onScannedRobot(ScannedRobotEvent e)`
  - Name des Gegners: `String name = e.getName()`
  - Energie des Gegners: `double eng = e.getEnergy()`
  - Abstand zum Gegner (Mittelpunkt zu Mittelpunkt): `double dist = e.getDistance()`
  - Absolute Orientierung des Gegners (0° bis 360°): `double deg = e.getHeading()`
  - Orientierung des Gegners relativ zur eigenen (-180° bis +180°): `double deg = e.getBearing()`




```
double alpha = e.getHeading();
double beta = e.getBearing();
```

Beispiele für Ereignisbehandlungen (2/3)

- Beispiel: `void onHitByBullet(HitByBulletEvent e)`
  - Stärke des Kugel: `double pow = e.getPower()`
  - Geschwindigkeit der Kugel: `double v = e.getVelocity()`
  - Orientierung der Kugel relativ zum eigenen Roboterausrichtung (-180° bis +180°): `double deg = e.getBearing()`
  - Orientierung der Kugel (0° bis 360°): `double deg = e.getHeading()`




```
double alpha = e.getHeading();
double beta = e.getBearing();
```

- Weitere Möglichkeiten in der RoboCode-Doku:
  - http://robocode.sourceforge.net/docs/robocode/ und Suche nach den Ereignisnamen (z.B. `BulletHitEvent`, `HitWallEvent`, …)

Beispiel-Überlegungen

Überlegungen: Wie verhält sich der Roboter am besten wenn…

- … er sich ganz allgemein fortbewegt?
  - Wie schaut eine gute Bewegungsbahn aus?
    Stehen bleiben oder stetig bewegen?
  - Wie werden Radar und Kanone ausgerichtet?
- … ein Gegner gefunden wurde?
  - Feuert er direkt eine Kugel ab?
  - In welche Richtung feuert er den Schuss am besten?
  - Mit welcher Schussenergie sollte er schießen?
  - Soll Radarbewegung eingeschränkt werden, um Gegner zu folgen?
  - Soll die Bewegungsrichtung des Roboters angepasst werden?
- … er von einer gegnerischen Kugel getroffen wurde?
  - In welche Richtung flüchtet er am besten?
    Einfach nach hinten? Senkrecht zur Kugelbahn?
- … er einen Gegner mit einer Kugel trifft?
  - Soll die Radarbewegung eingeschränkt werden?
- … es zu einer Kollision mit der Wand kommt?
  - In welche Richtung fährt er am besten weiter?
  - Kann die Kollision generell vermieden werden?

Beispiel-Überlegungen

Überlegungen: Wie verhält sich der Roboter am besten wenn…

- … er sich ganz allgemein fortbewegt?
  - Wie schaut eine gute Bewegungsbahn aus?
    Stehen bleiben oder stetig bewegen?
  - Wie werden Radar und Kanone ausgerichtet?
- … ein Gegner gefunden wurde?
  - Feuert er direkt eine Kugel ab?
  - In welche Richtung feuert er den Schuss am besten?
  - Mit welcher Schussenergie sollte er schießen?
  - Soll Radarbewegung eingeschränkt werden, um Gegner zu folgen?
  - Soll die Bewegungsrichtung des Roboters angepasst werden?
- … er von einer gegnerischen Kugel getroffen wurde?
  - In welche Richtung flüchtet er am besten?
    Einfach nach hinten? Senkrecht zur Kugelbahn?
- … er einen Gegner mit einer Kugel trifft?
  - Soll die Radarbewegung eingeschränkt werden?
- … es zu einer Kollision mit der Wand kommt?
  - In welche Richtung fährt er am besten weiter?
  - Kann die Kollision generell vermieden werden?

Ein paar Details in den anderen Folien zu: