CHALMERS |  UNIVERSITY OF GOTHENBURG

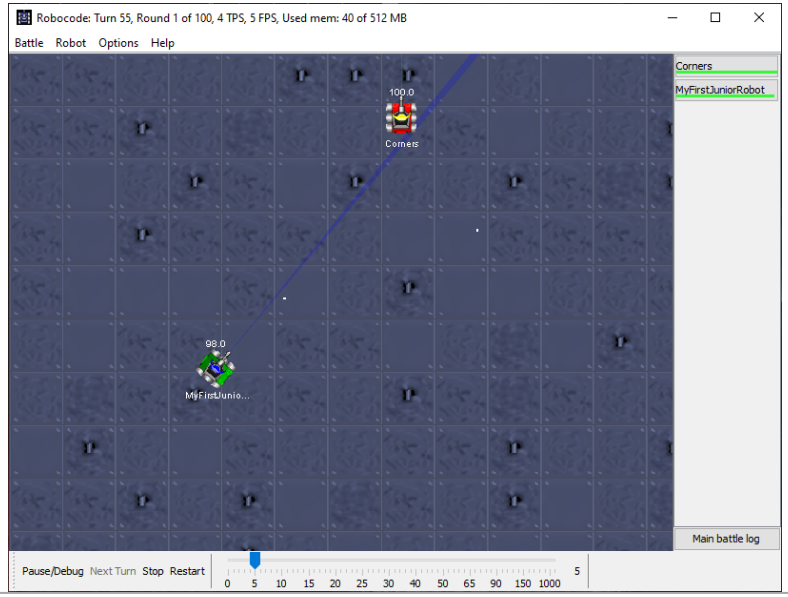
Software Engineering Principles for Complex Systems - Robocode introduction -

Tobias Schwarz, Mazen Mohamad, Thorsten Berger, Wardah Mahmood

Robocode

What is Robocode?

- Robocode is a programming game
  - Provides a game engine to simulate robot competitions (battles)
- You're not directly controlling the robot, but programming it to manage the battle by its own
- Implement one or many Java classes how the robot behaves and reacts to its environment
- Robot competitions takes places on a simulated battlefield



- Purpose of assignment:

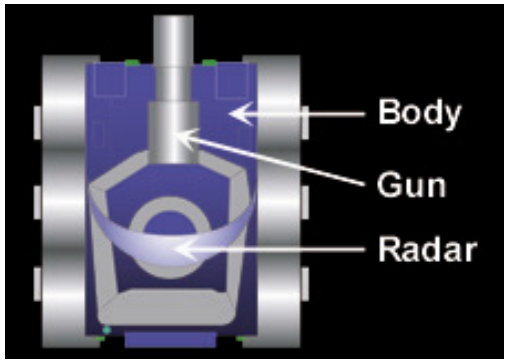
  Learn creating an SPL in a fun and interactive environment

Robocode Important Websites

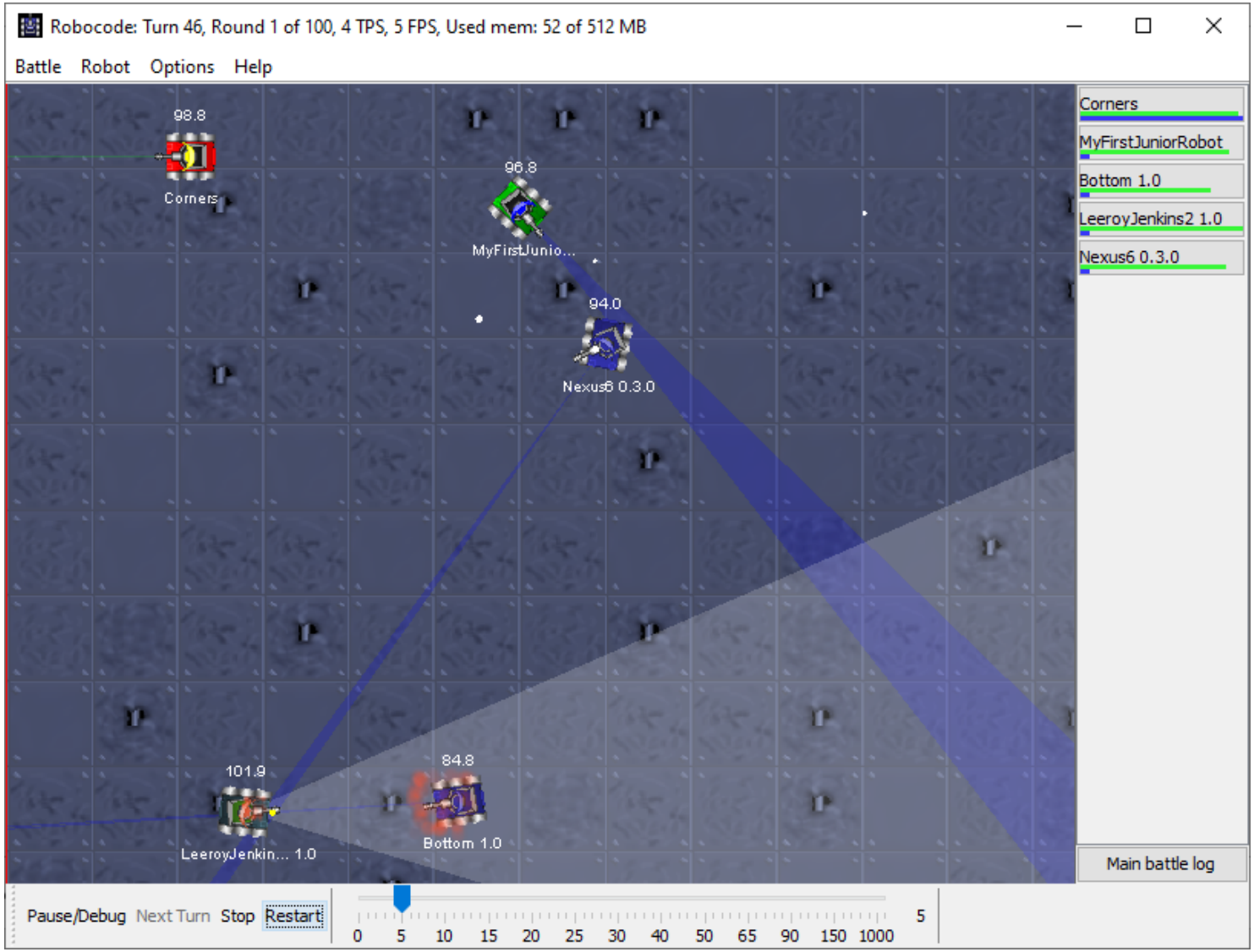
- RoboWiki
  - Main source of information
  - Over 200 OS robots available
- Robocode website
  - Download Robocode.
  - Robocode API.

Robot Anatomy

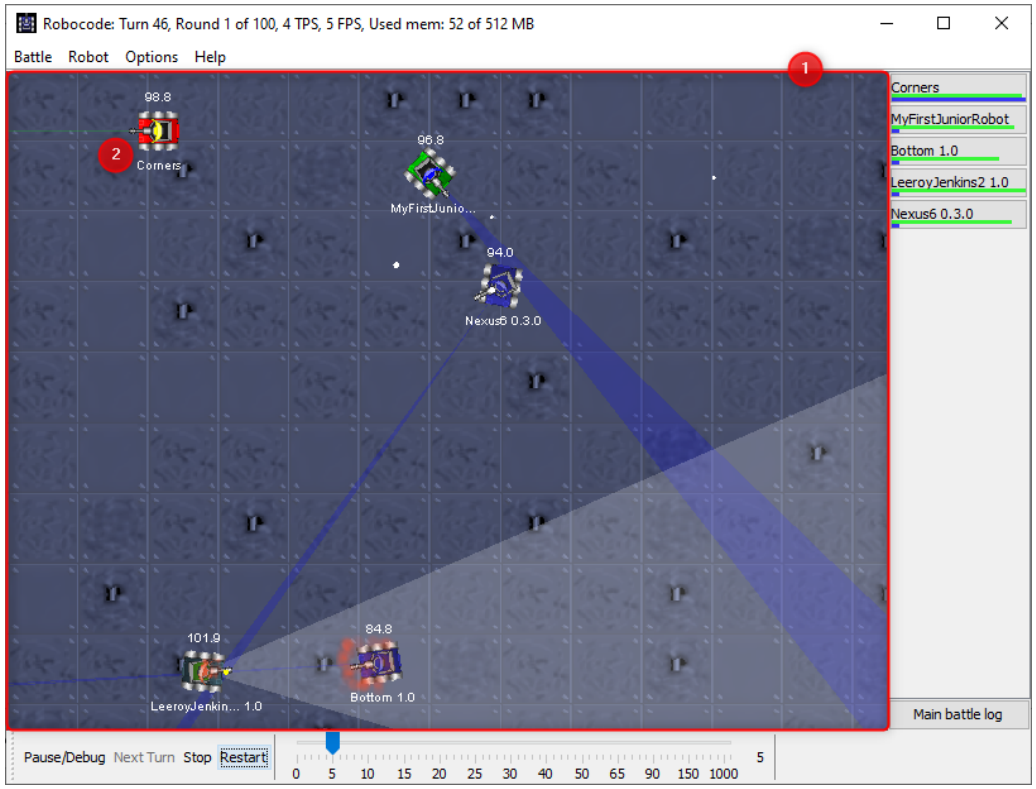
- **Body** – Carries the gun with the radar on top. The body is used for moving the robot ahead and back, as well as turning left or right.
- **Gun** – Mounted on the body and is used for firing energy bullets. The gun can turn left or right. Carries the radar on top.
- **Radar** – Mounted on the gun and is used to scan for other robots when moved. The radar can turn left or right.
- Each area contains a set of strategies



The battlefield

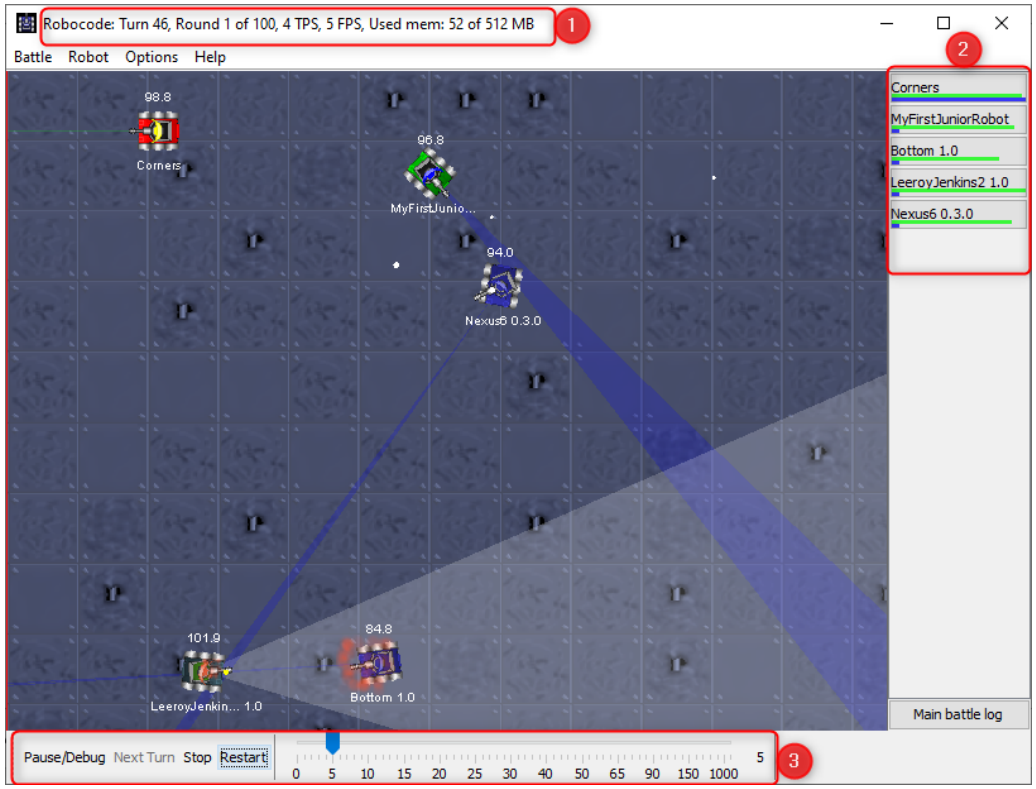


The battlefield



1. Simulated battle
   - Different robots
   - Radar
   - Bullets
   - Hit by bullet
2. Individual robot
   1. Health points (=98.8)
   2. Name (=Corners)

The battlefield



1. Simulator information
   - Turns/Ticks (time measure)
   - Round
   - FPS (1Tick per FPS)
   - Memory
2. Robot information
   - Name and health
3. Simulation settings
   - Pause/stop/restart
   - FPS setting

The battlefield

Results for 100 rounds

Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds	3rds
1st	apc.LeeroyJenkins2 1.0	46381 (45 %)	18850	3560	20789	3173	10	0	89	5	2
2nd	banshee.micro.Nexus6 0....	23288 (23 %)	7850	40	11329	1089	2837	143	3	10	44
3rd	sample.MyFirstJuniorRobot	17995 (18 %)	12200	320	5169	254	46	7	10	49	23
4th	sample.Corners	11427 (11 %)	7500	0	3747	169	11	0	0	30	19
5th	ad.last.Bottom 1.0	3502 (3 %)	3500	0	2	0	0	0	0	4	12

Save OK

- Rank
- Robot Name
- Total Score

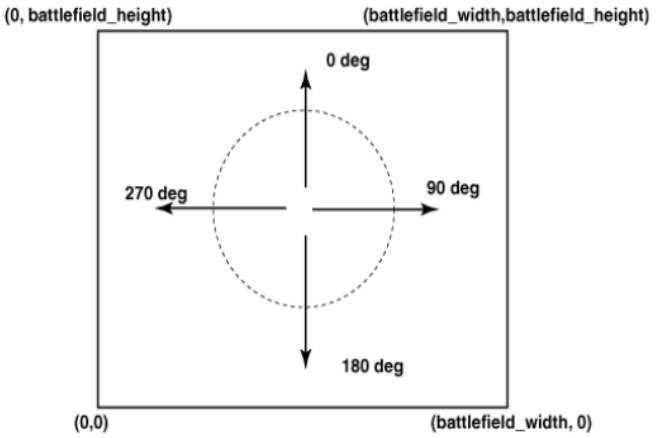
- Points for
  - Survival
  - Survival Bonus
  - Bullet Damage
  - Bullet Bonus
  - Ram Damage
  - Ram Bonus
- Times won on position X

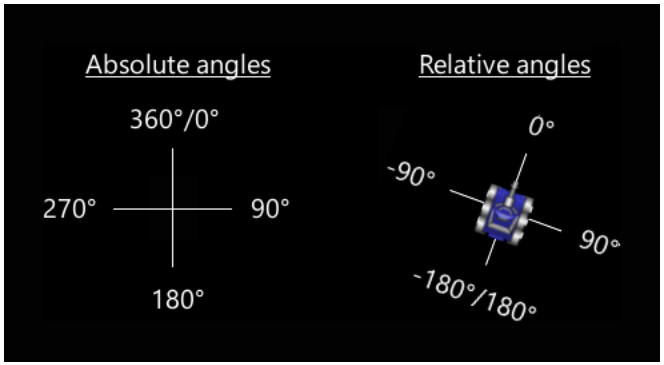
Constrains

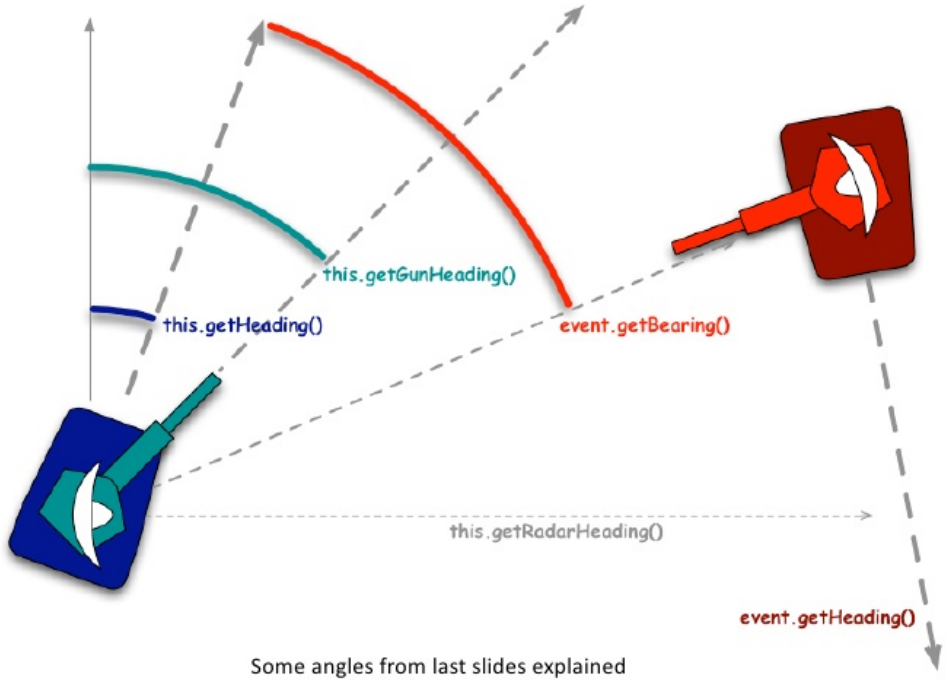
- Robots health (100points)
  - Lose health points for: Getting hit, shooting, hitting a wall or other robots
  - Gain health points for: Hitting other robot
- Actions per tick
  - E.g. gun turns max 20 degrees and radar turns max 45 degrees
  - Robot's velocity influences body turn rate
  - Bullet power and speed. Shooting cooldown phase
  - Body, gun, and radar influence each other

More details: Robowiki - Game physics

Battlefield and Robot Positioning







Trigonometric:
https://www.ibm.com/developerworks/java/library/j-robocode2/sidefile-robo2.html

Live Demo

- With robots
  - Sample.MyFirstRobot
  - Sample.Corners
  - Sample.RamFire
  - Sample.Crazy

MyFirstRobot - Code explained

```
1.  package pkg;
2.
3.  import robocode.*;
4.
5.  public class MyFirstRobot extends Robot {
6.      public void run() {
7.          while (true) {
8.              ahead(100);
9.              turnGunRight(360);
10.             back(100);
11.             turnGunRight(360);
12.         }
13.     }
14.
15.     public void
    onScannedRobot(ScannedRobotEvent e) {
16.         fire(1);
17.     }
18. }
19.
```

- **“Package”** for organizational purposes of robots in RoboCode Simulator
- **Import RoboCode library** to receive access to its functions

MyFirstRobot - Code explained

```
1.  package pkg;
2.
3.  import robocode.*;
4.
5.  public class MyFirstRobot extends Robot {
6.      public void run() {
7.          while (true) {
8.              ahead(100);
9.              turnGunRight(360);
10.             back(100);
11.             turnGunRight(360);
12.         }
13.     }
14.
15.     public void
    onScannedRobot(ScannedRobotEvent e) {
16.         fire(1);
17.     }
18. }
19.
```

- **"Main" class** of your robot
- **Extends Robot** / AdvancedRobot
  - Robot = blocking calls
  - Ad.Robot = non-blocking calls

MyFirstRobot - Code explained

```
1.  package pkg;
2.
3.  import robocode.*;
4.
5.  public class MyFirstRobot extends Robot {
6.      public void run() {
7.          while (true) {
8.              ahead(100);
9.              turnGunRight(360);
10.             back(100);
11.             turnGunRight(360);
12.         }
13.     }
14.
15.     public void
    onScannedRobot(ScannedRobotEvent e) {
16.         fire(1);
17.     }
18. }
19.
```

- **Run()** for robot configuration
- **While-loop** contains basic behavior; always executed when no on-events, e.g. onScannedRobot
- In this example:
  - Continue loop until health points run out or onScannedRobot is called

MyFirstRobot - Code explained

```
1.  package pkg;
2.
3.  import robocode.*;
4.
5.  public class MyFirstRobot extends Robot {
6.      public void run() {
7.          while (true) {
8.              ahead(100);
9.              turnGunRight(360);
10.             back(100);
11.             turnGunRight(360);
12.         }
13.     }
14.
15.     public void
    onScannedRobot(ScannedRobotEvent e) {
16.         fire(1);
17.     }
18. }
19.
```

- **Ahead** – Pixels to move
- **turnGunRight** – Degree to turn gun attached radar
- In this example:
  1. Move ahead 100 pixels.
  2. Turn the gun right by 360 degrees.
  3. Move back 100 pixels.
  4. Turn the gun right by 360 degrees again.

MyFirstRobot - Code explained

```
1.  package pkg;
2.
3.  import robocode.*;
4.
5.  public class MyFirstRobot extends Robot {
6.      public void run() {
7.          while (true) {
8.              ahead(100);
9.              turnGunRight(360);
10.             back(100);
11.             turnGunRight(360);
12.         }
13.     }
14.
15.     public void
    onScannedRobot(ScannedRobotEvent e) {
16.         fire(1);
17.     }
18. }
19.
```

- **Event handling code** on certain event and implementation about action to take
  - **onScannedRobot()**
  - **onHitByBullet()**
  - **onHitWall()**
  - **…**
- Callout contains information about event, such as scanned enemy robot

Next steps

- Download and install required software
- Build your first own robot
- Run a competition with your own robot

- Learn more about movement, targeting and firing

  Highly recommended:

  – https://www.ibm.com/developerworks/java/library/j-robocode/
  – https://robocode.sourceforge.io/docs/robocode/

  Recommended:

  – https://www.ibm.com/developerworks/java/library/j-robocode2/j-robocode2-pdf.pdf
  – http://robowiki.net/ (Chalmers Mirror) -> Radar, Targeting, Movement, Tutorials

Further reading

- RoboCode FAQ
- Basic knowledge in trigonometry (used to targeting, movement and avoid getting hit): https://www2.clarku.edu/faculty/djoyce/trig/
- Secrets from the Robocode masters
  https://robocode.sourceforge.io/developerWorks.php
  http://mark.random-article.com/robocode/
- Interests of research
  - Applying Machine Learning to Robocode
  - Deep Q-Learning for Robocode

If you have something to discuss or ask,
use CANVAS discussion section.