Robocode

PROP 2022-23 Q1

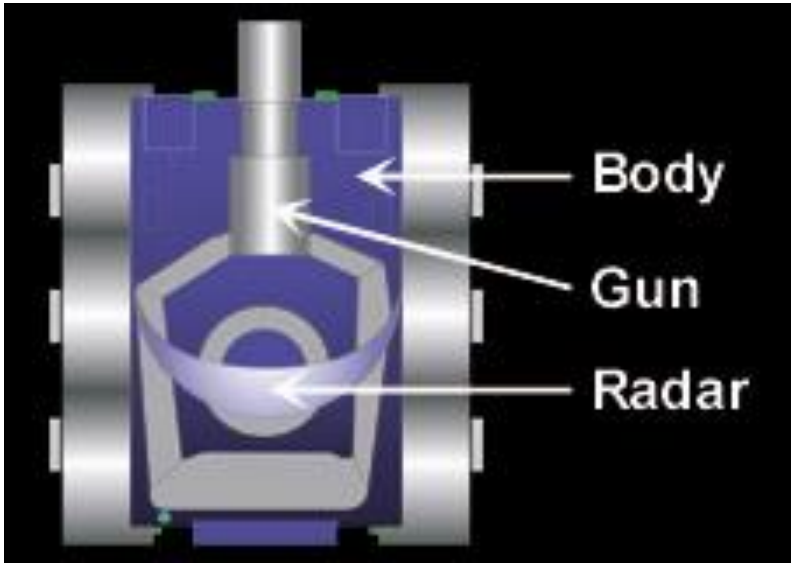
Mario Martin – Ignasi Gómez Sebastià – Bernat Orellana

Aprendre amb Robocode

- Aprenetatge de programació. Alguns exemples:
  - Programació Orientada a Objectes
  - Extendre classes
  - Herencia
  - Polimorfisme
  - Fer servir Java docs
  - Cridar al codi d'una API
  - Gestió d'events

Objectiu

- Robot (estil tanc) propi competeix amb d'altres robots.
- Objectiu: sobreviure a tots els altres
- Anatomia del robot:
  - Cos del vehicle
  - Radar
  - Canó (Gun)



Programació bàsica

- Estendre la classe robot amb la definició (millor redefinició o sobrecàrrega dels mètodes) del teu robot, especialment el mètode **run()**

```
import robocode.*;
public class MyFirstRobot extends Robot{
    public void run(){
        ....
    }
}
```

Accions

- Mètodes ja definits a la classe robot, útils i que no cal redefinir:
  - **turnRight(double degree), turnLeft(double degree)**
  - **ahead(double distance), back(double distance)** [Mouen el robot la quantitat de pixels donada. Paren després del moviment o quan es xoca amb alguna cosa]
  - **turnGunRight(double degree) ,turnGunLeft(double degree)**.
  - **turnRadarRight(double degree), turnRadarLeft(double degree)**

Notes sobre els girs

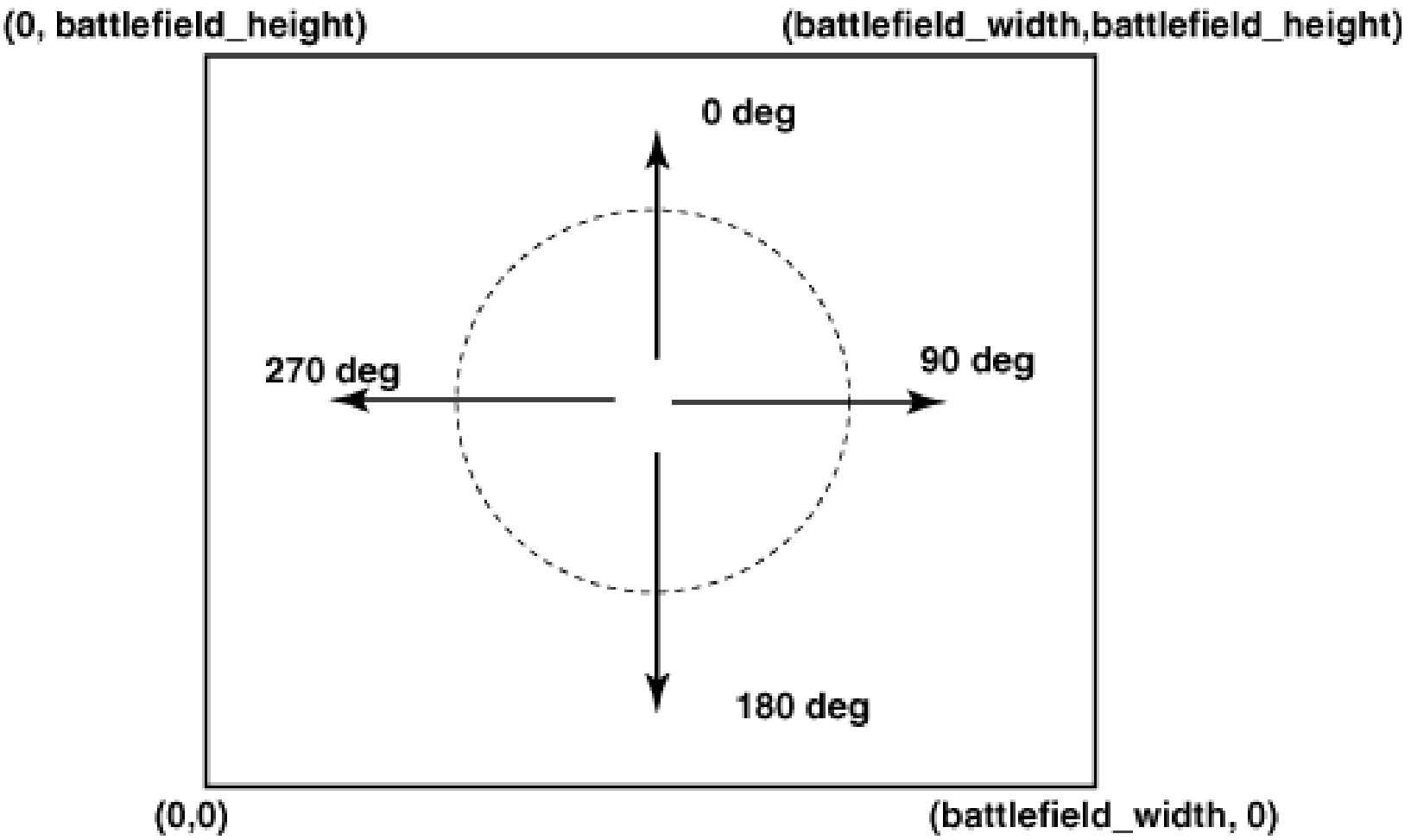
- Quan el robot gira, ho fa de forma conjunta excepte que posem els flags següents:
  - **setAdjustGunForRobotTurn(boolean flag)**: If the flag is set to true, the gun will remain in the same direction while the vehicle turns.
  - **setAdjustRadarForRobotTurn(boolean flag)**: If the flag is set to true, the radar will remain in the same direction while the vehicle (and the gun) turns.
  - **setAdjustRadarForGunTurn(boolean flag)**: If the flag is set to true, the radar will remain in the same direction while the gun turns. It will also act as if setAdjustRadarForRobotTurn(true) has been called.

Accions (2)

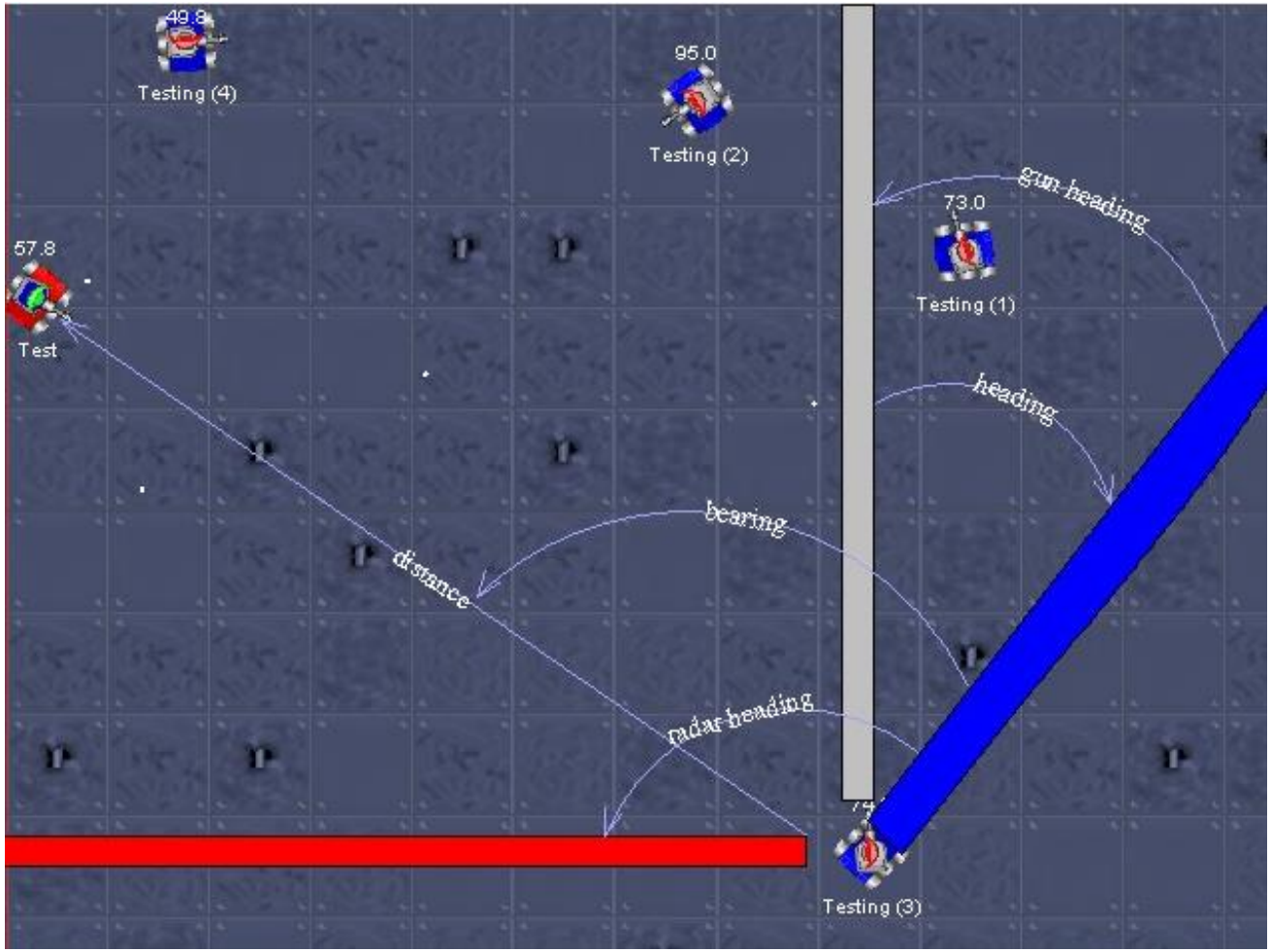
- **getX() , getY()** Obté les coordenades actuals del robot
- **getHeading(), getGunHeading(), getRadarHeading()**

  Obtenen direcció en graus (absoluts) del vehicle, canó i radar
- **getBattleFieldWidth(), getBattleFieldHeight()** Obtenen dimensions del mon

Coordenades



Angles


49.8
Testing (4)
95.0
Testing (2)
73.0
Testing (1)
57.8
Test
Testing (3)
gun heading
heading
bearing
distance
radar heading

Accions: Disparar

- Dues accions per disparar: **fire(double power)** i **fireBullet(double power)**
  - Cada dispar redueix energia del robot
  - Cada robot comença amb energia per defecte i es destrueix quan arriba a zero.
  - Dispara amb paràmetre d'energia que està en un rang de [0.1..3]. Com més energia més mal a l'altre (i més energia perduda)
  - Podeu usar les constants físques de la classe **Rules**. Per exemple, per la potència del projectil, el rang és:

    [Rules.MIN_BULLET_POWER ... Rules.MAX_BULLET_POWER]

  - **fireBullet()** retorna objecte de la classe Bullet que ens dona informació sobre com hem disparat.

Accions

- **getEnergy()**

  Obté l'energia que li queda al robot.

- Més informació al JavaDoc de la classe ***Robot***.

Energia

- El robot perd energia al
  - Disparar
  - Xocar
  - Ser impactat per un dispar.
- El robot guanya energia
  - Impactant un dispar sobre un altre
  - Amb el temps

Events

- Interrupció del **run** davant de possibles events
  - ScannedRobotEvent
  - HitByBulletEvent
  - BulletHit
  - HitRobotEvent
  - HitWallEvent
- Programació de la gestió dels events
  - ScannedRobotEvent → onScannedRobot(ScannedRobotEvent e)
  - HitByBulletEvent → onHitByBullet(HitByBulletEvent e)
  - BulletHit → onBulletHit(BulletHitEvent e)
  - HitRobotEvent → onHitRobot(HitRobotEvent e)
  - HitWallEvent → onHitWall(HitWallEvent e)
- Métodes implementat per defecte "donothing" sobrecarregar els que es creguin necessaris.

Altres events

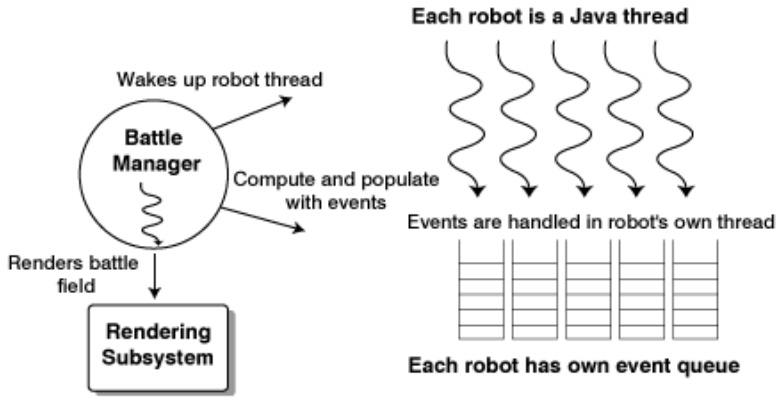
- Altres events
  - onBulletMissed
  - onBulletHitBullet
- No cal programar-los tots.

Events

- Cada event té informació associada per facilitar la programació.
- Exemple: ScannedRobotEvent
  - **getHeading() [getBearing**()] Retorna orientació del robot [respecte vehicle] en graus
  - **getDistance**() Retorna distancia en pixels al robot detectat
  - **getVelocity**()
  - **getEnergy()**

- Veure info per cada event a JavaDocs

Torns



Each robot is represented as a thread, with a master battle simulator thread, which operates as follows:

```
while (round is not over) do
   call the rendering subsystem to draw robots, bullets, explosions
   for  each robot do
       wake up the robot
       wait for it to make a blocking call, up to a max time interval
   end for
   clear all robot event queue
   move bullets, and generate event into robots' event queue if applicable
   move robots, and generate event into robots' event queue if applicable
   do battle housekeeping and generate event into robots' event queue
         if applicable
   delay for frame rate if necessary
end do
```

Instal.lació

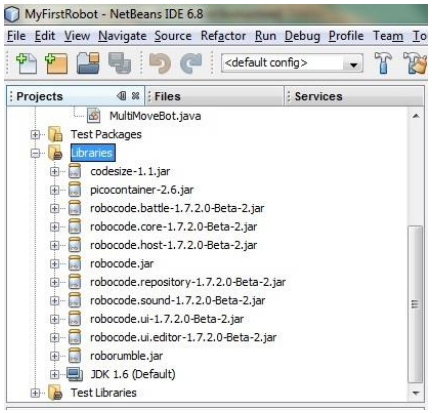
- Descàrrega del Robocode

  http://sourceforge.net/projects/robocode/files/robocode/1.9.2.5/

  - API: http://robocode.sourceforge.net/docs/robocode/
- Executar per auto-instal.lar (donar path local si no hi ha privilegi d'administrador ni root)

```
java –jar robocode-setup.jar
```

- [Opcional: Afegir llibreries al Netbeans]



Exemple de robot simple

```
package meurobot;
import robocode.*;

public class RobotPatrulla 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);
    }
}
```

Advanced robot

- En el robot standard, les accions es fan una darrera de l'altre, dona lloc a moviment robòtic.
- En l'advanced robot pots fer moviments complexes.

Advanced robot vs. Robot

Robot class:

- `turnRight()`
- `turnLeft()`
- `turnGunRight()`
- `turnGunLeft()`
- `turnRadarRight()`
- `turnRadarLeft()`
- `ahead()`
- `back()`

AdvancedRobot class:

- `setTurnRight()`
- `setTurnLeft()`
- `setTurnGunRight()`
- `setTurnGunLeft()`
- `setTurnRadarRight()`
- `setTurnRadarLeft()`
- `setAhead()`
- `setback()`

Per a que s'executin els set's

- Execute()

En advanced robot

```
package meurobot;
import robocode.*;
public class RobotPatrullaProgramat extends AdvancedRobot
{
    public void run() {
        turnLeft(getHeading());
        while(true) {
            setTurnRight(10000);
            setTurnGunRight(90);
            setAhead(2000);
            execute();
        }
    }
    public void onScannedRobot(ScannedRobotEvent e) {
        fire(1);
    }
    public void onHitByBullet(HitByBulletEvent e) {
        turnLeft(180);
    }
}
```

Dummy robot

```java
package PROPRobot;

import robocode.HitByBulletEvent;
import robocode.Robot;
import robocode.ScannedRobotEvent;

/**
 *
 * @author Ignasi GómeSebastià
 */
public class DummyRobot extends Robot{
    public void run() {
        turnLeft(getHeading());
        while (true){
            ahead(500);
            turnRight(90);
        }
    }
    public void onScannedRobot(ScannedRobotEvent e) {
        fire(1);
    }

    public void onHitByBullet(HitByBulletEvent e) {
        //turnLeft(180);
    }
}
```

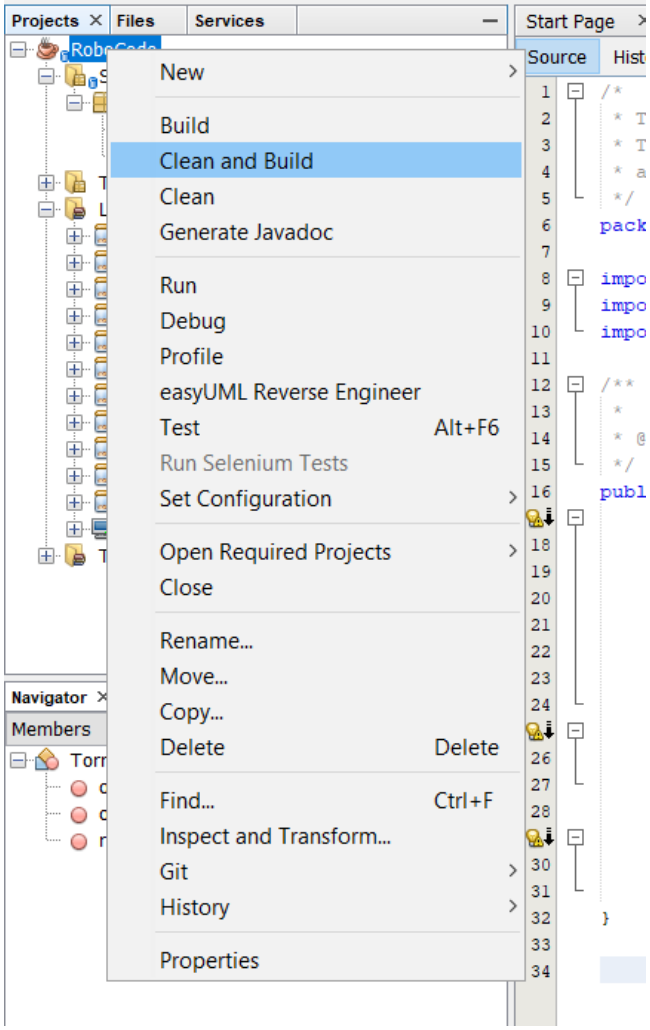
Torre robot

```
public class TorreRobot extends Robot{

    private static double bearingThreshold = 5;

    public void run() {
        turnLeft(getHeading());
        while (true){
            turnGunLeft(90);
            turnRadarLeft(90);
            //turnRight(90);
        }
    }

    double normalizeBearing(double bearing) {
        while (bearing >  180) bearing -= 360;
        while (bearing < -180) bearing += 360;
        return bearing;
}

    public void onScannedRobot(ScannedRobotEvent e) {

        if (normalizeBearing(e.getBearing()) < bearingThreshold){
            fire(1);
        }
    }

    public void onHitByBullet(HitByBulletEvent e) {
        //turnLeft(180);
    }
}
```

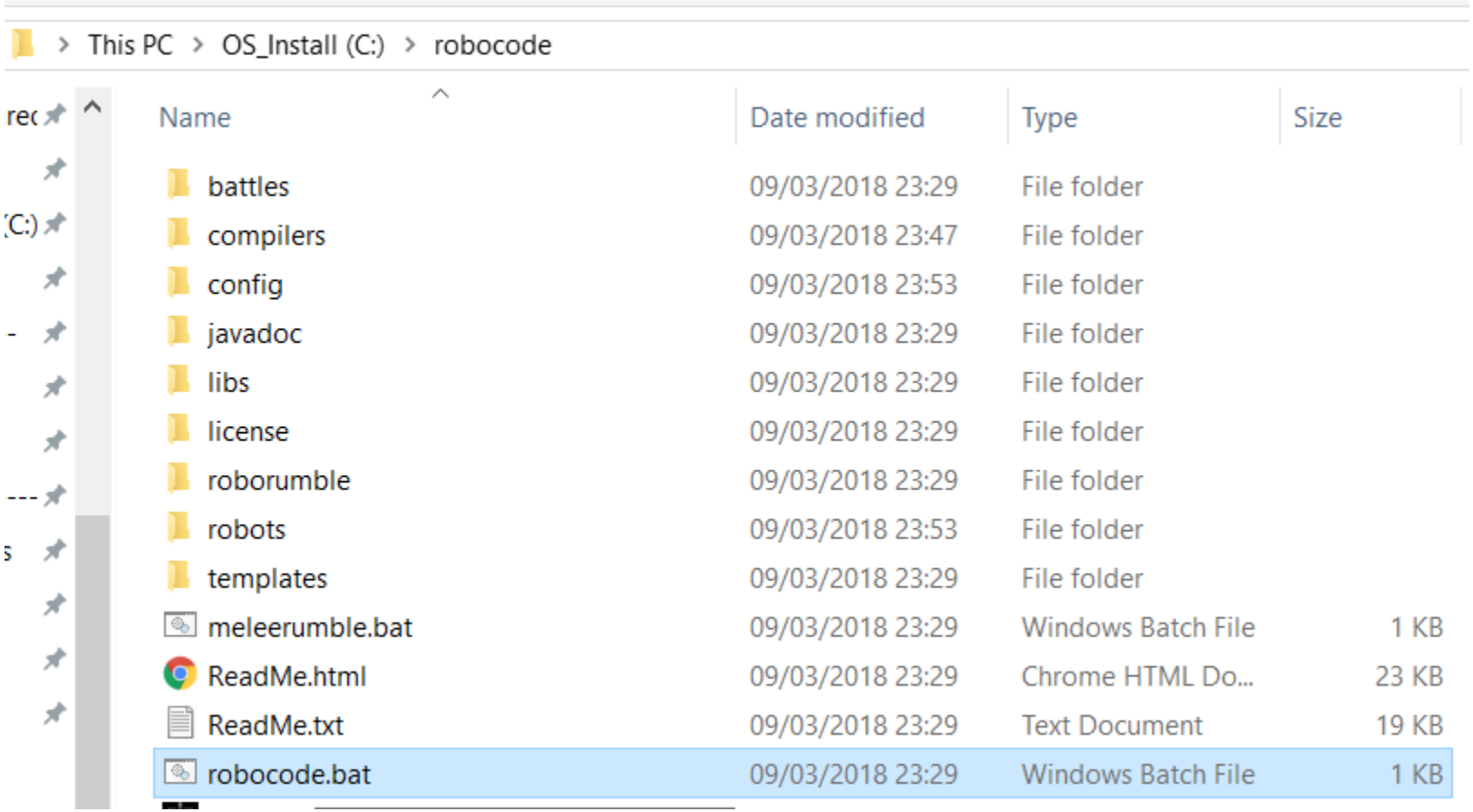
TeamRobot

- Classe base per fer equips col·laboratius de robots que s'enfrontin a d'altres equips.
- Dos modalitats: tots els robots iguals o un lider i "bots".
- API de *AdvancedRobot* , afegint:
  - broadcastMessage(Serializable message)
  - getTeammates(): String[]
  - isTeammate(String name)
  - onMessageReceived(MessageEvent e)
  - sendMessage(String name, Serializable message)

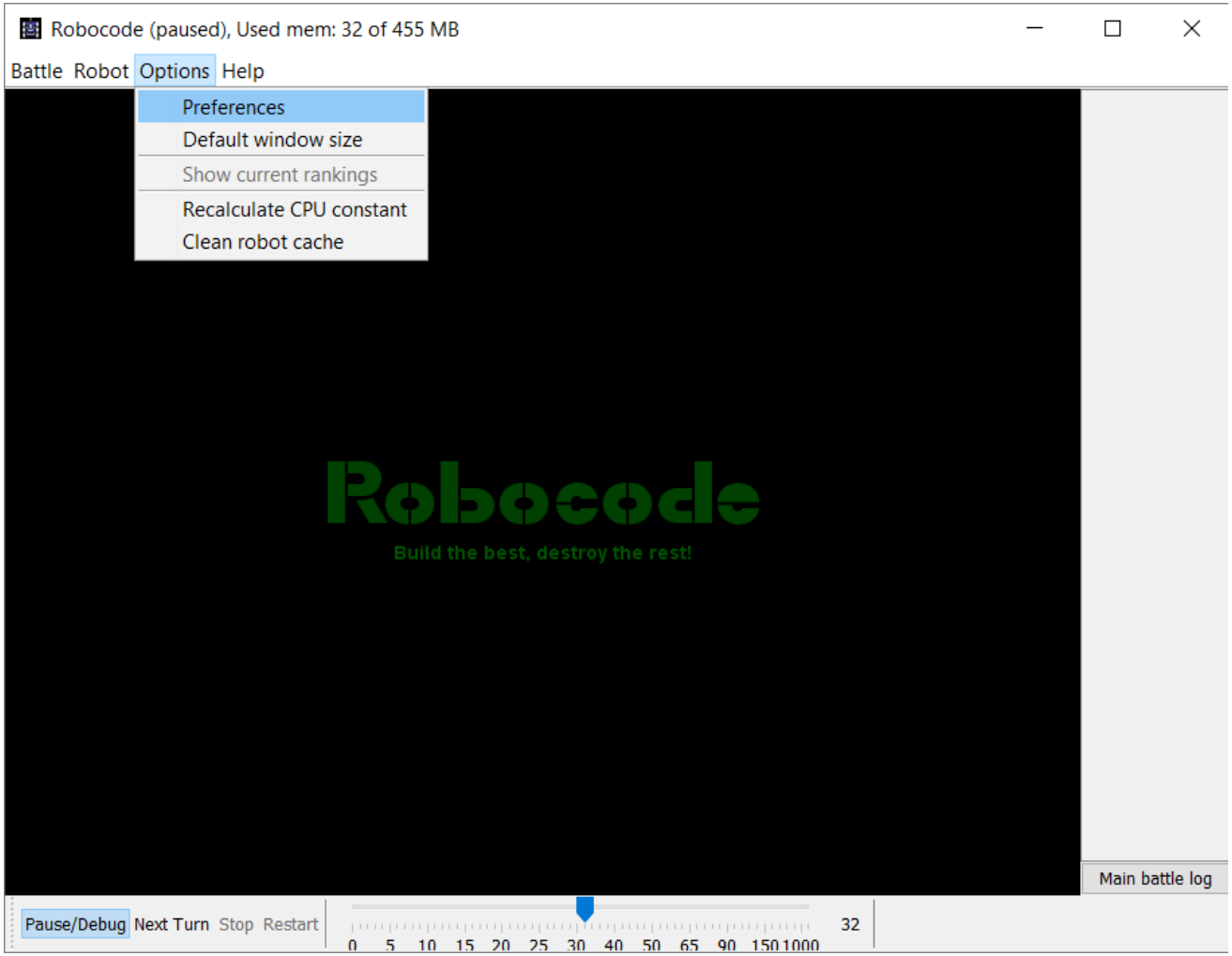
Carregant robots



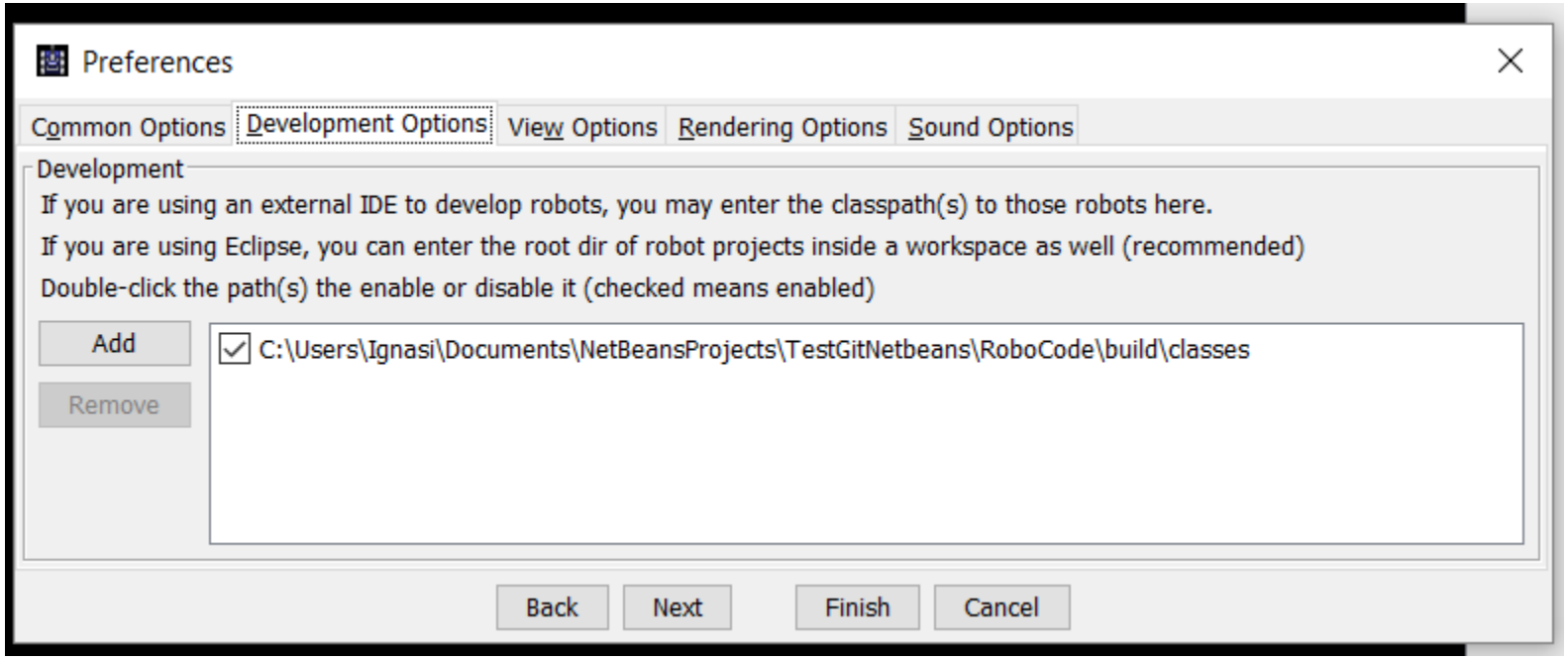
Carregant robots



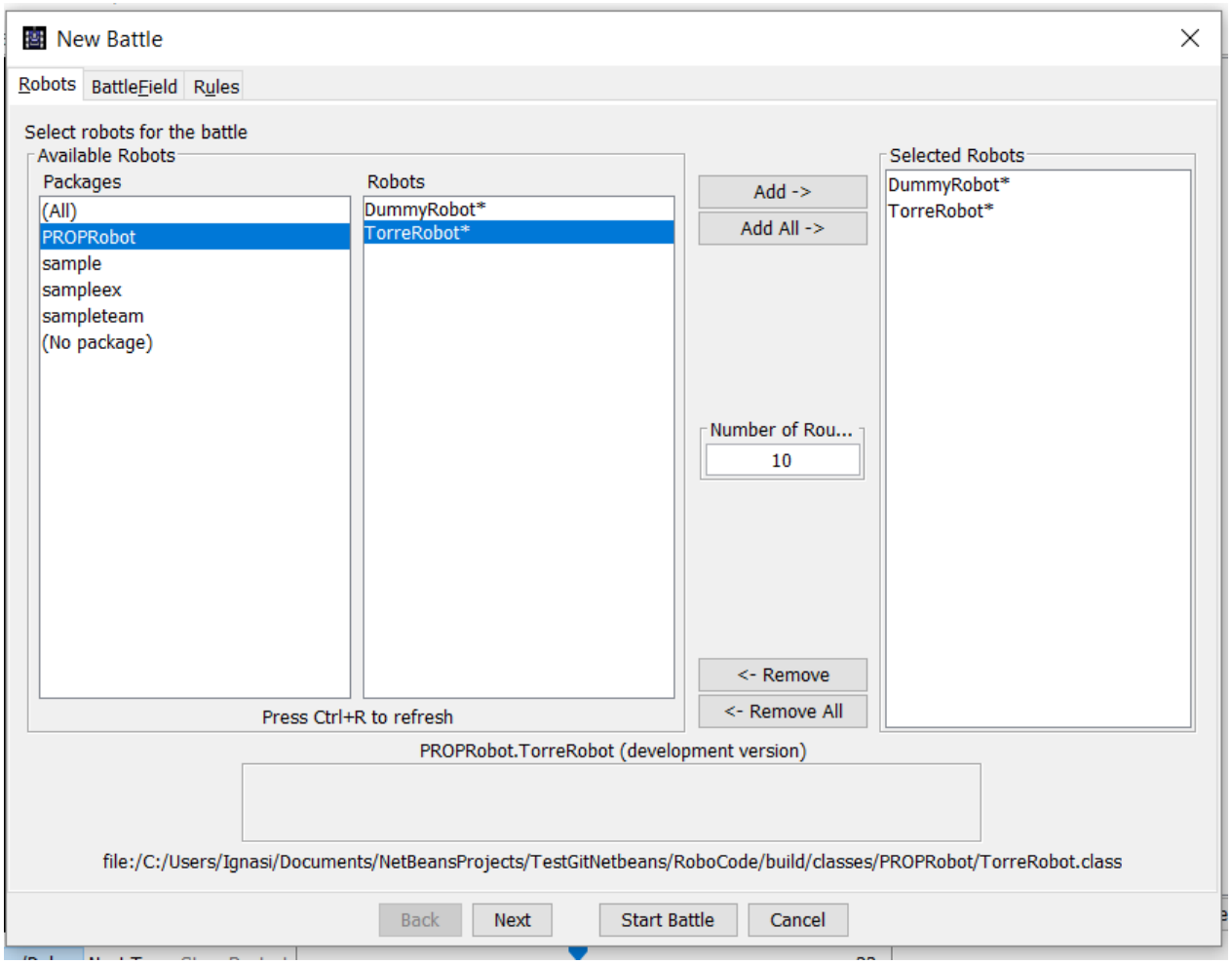
Carregant robots



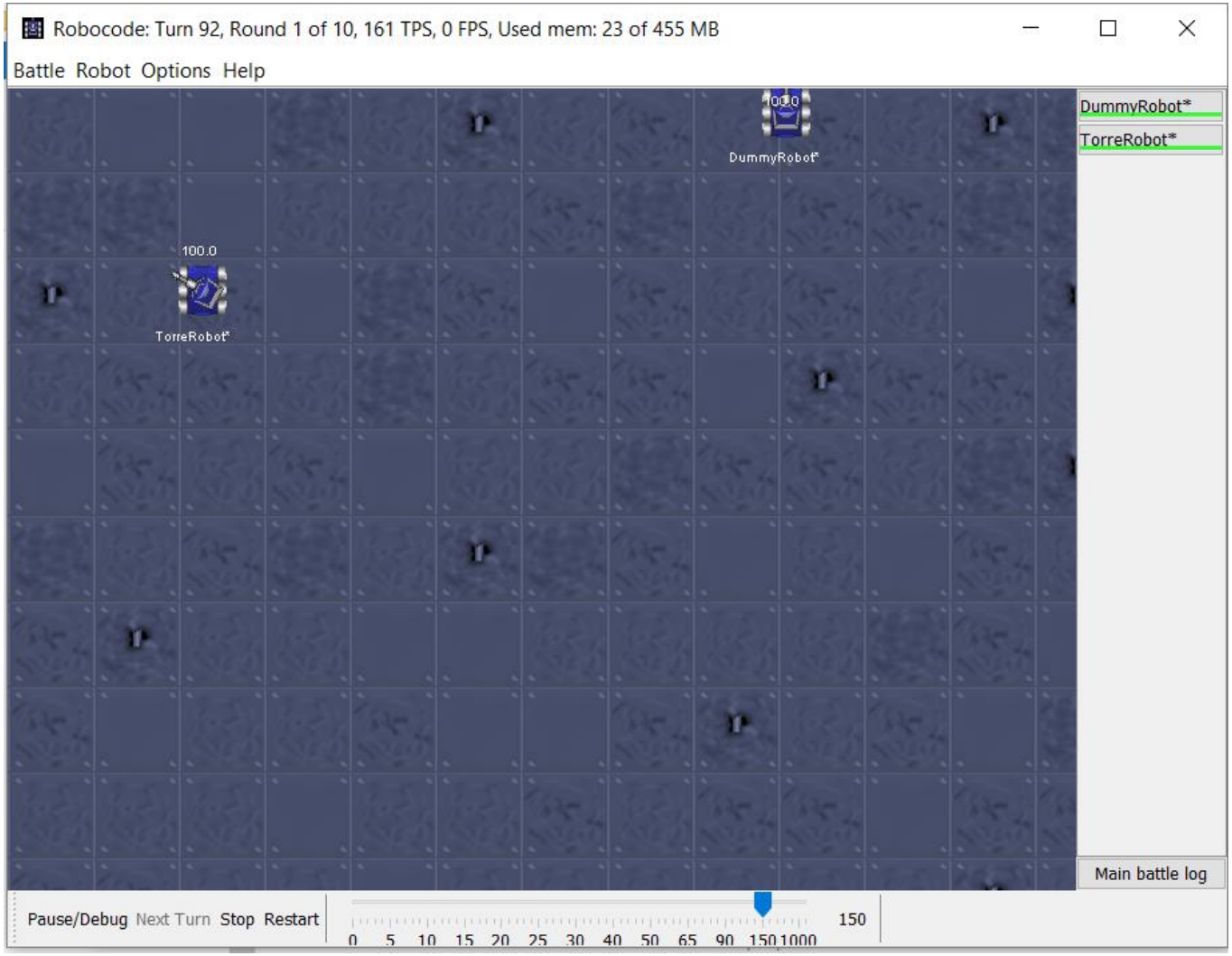
Carregant robots



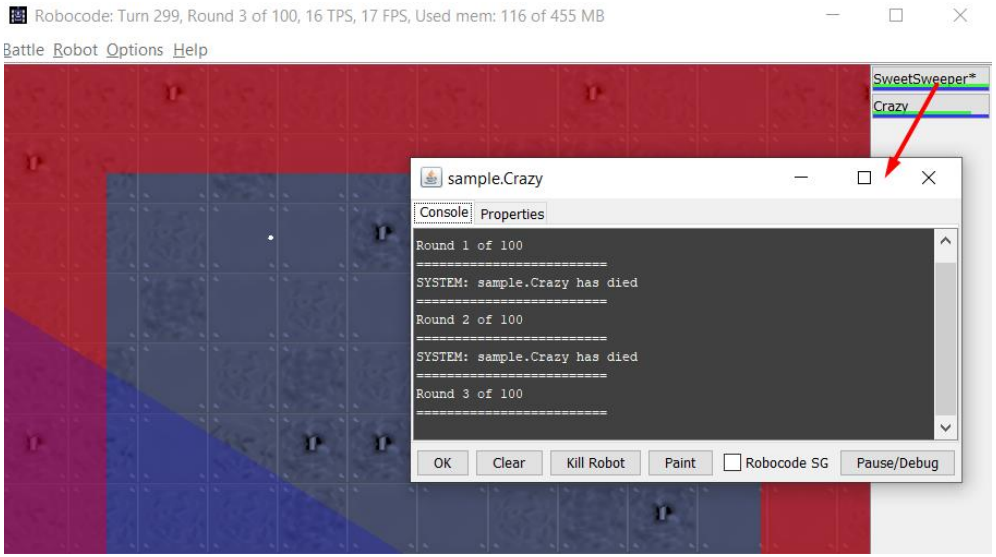
Carregant robots

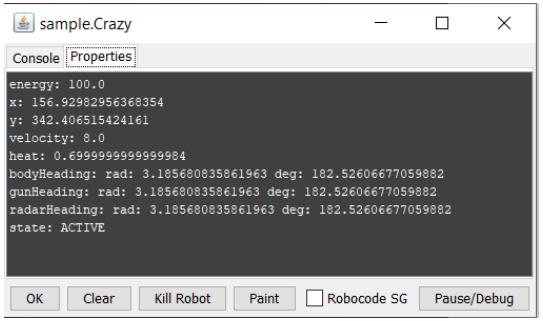


Carregant robots



Depuració (text)

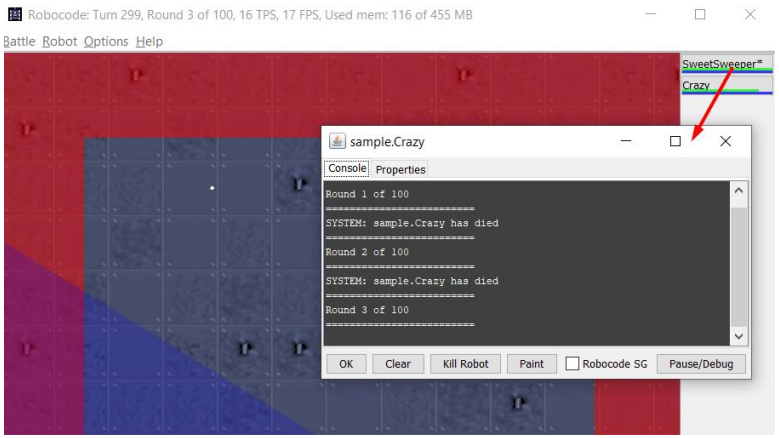




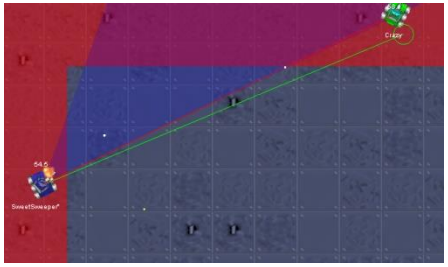

```
setDebugProperty("TIME", "" + getTime());
```

Depuració (gràfica)

- Cal activar-la amb el botó “Paint”

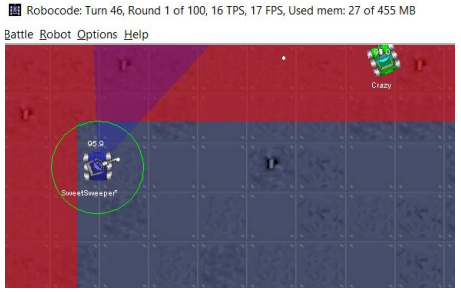


- MOLT
- ÚTIL!



- Es programa sobreescribint el mètode Paint del Robot.

```
@Override
public void onPaint(Graphics2D g) {
    int r = 60;
    g.setColor(Color.green);
    g.drawOval((int)getX()-r, (int) getY()-r, 2 * r, 2 * r);
}
```



Activitat

- Entrega a Atenea el 10 d'octubre.
- Grups de 2
- Zip amb
  - Projecte de Netbeans amb el codi dels robots
  - Document PDF amb documentació
  - Fitxer de text amb noms i DNIs dels alumnes
- El nom del zip està format dels dnis dels alumnes del grup :

[NIF1]_[NIF2].zip

Exercici 1: CornerTeam

En aquest primer exercici desenvolupareu un equip de 5 robots, anomenat "**CornerTeam**" amb un comportament prefixat.

- A l'inici de la partida els robots avaluen la seva posició i la comparteixen amb els companys. L'objectiu és tenir un robot a cada cantonada del camp de batalla, i un altre robot "kamikaze" que vagi per lliure atacant a qui li sembli.

- En la fase inicial de la partida, els robots consensuaran a quina cantonada va cadascun, prioritzant anar a la cantonada que tenen més propera, i deixant un d'ells com a robot "kamikaze".
- Un cop decidida la cantonada de destí, els robots s'hi dirigeixen immediatament, amb el radar mirant cap endavant i disparant a tota metxa si tenen enemics que s'interposin al seu camí cap a la cantonada.
- Quan arriba a la cantonada, el robot activa el seu radar per a que giri contínuament, i dispara a l'enemic més proper.
- Cada cert temps el robot farà un moviment "sentinella" per evitar ser un blanc estàtic, el moviment és d'anada i tornada, i pot ser en horitzontal o en vertical, resseguint el contorn del camp de batalla.

- El robot "kamizake" activarà el seu radar per girar contínuament, triarà el robot més proper, fixarà el radar sobre seu i el perseguirà fins destruir-lo o ser destruït.

- **Millores**

  - **Millora 1**: Friendly-fire: eviteu que els companys disparin al pobre "kamikaze" si es troba enmig d'un objectiu.
  - **Millora 2**: Feu que tots els robots disparin sobre el mateix objectiu. Penseu un criteri raonable per seleccionar l'objectiu a batre.

Exercici 2

- Competició per equips
- Programeu un equip de robots pensat per una competició contra els vostres companys de classe.
  - Cal que dissenyeu acuradament una estratègia d'equip, i que refineu el comportament individual dels robots.
- La mida del camp de batalla serà 1000x800

Activitat

- Representa un 10% de la nota total de PROP
- Nota
  - 70% Codi
    - Exercici 1: 40% Desenvolupament del CornerTeam (30% base, 10% millores)
    - Exercici 2: 15% Desenvolupament d'un equip de competició. L'equip guanya amb solvència al *MyFirstTeam* i a *CornerTeam*.
    - 15% Puntuació a la competició entre grups (15% guanyador, 12% 2n, 9% semis, 6% quarts, 3% octaus.
  - 30% Documentació i nivell d'estratègia (!)

Activitat

- Condicions
  - Els robots han de ser un treball original vostre (!). Si useu idees o estratègies de tercers cal que:
    - Citeu totes les fonts
    - En el cas de trobar codi, no es copiarà sinó que caldrà que l'integreu fent-ne una reescriptura pròpia.
  - El codi del robot ha de ser correcte
    - Netbeans no reporta errors
    - El projecte compila
    - El robot carrega al robocode
  - Entregar document
    - Documentació codi (JavaDoc).
    - **Estratègia** seguida per desenvolupar robot.
    - Detalls de la implementació de la estratègia, incloent l'explicació dels càlculs (p.ex. per situar a l'enemic).