Robocode

PROP 2024-25 Q1

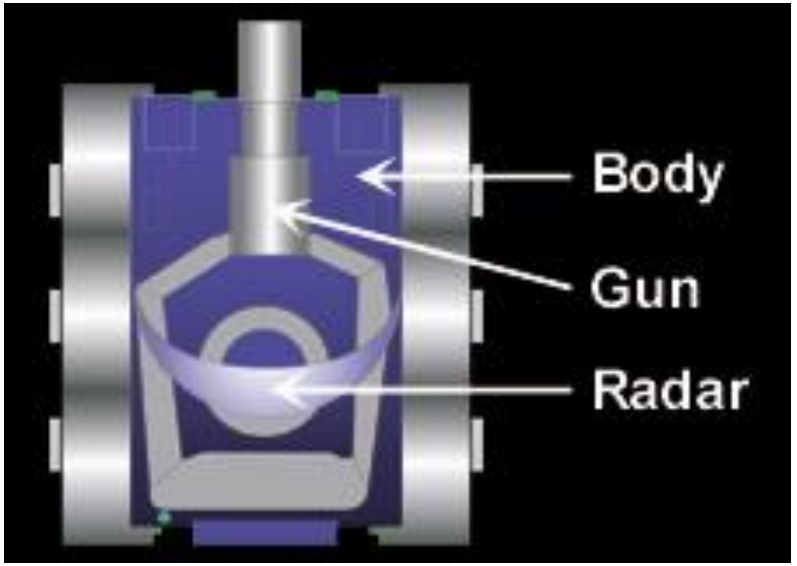
Bernat Orellana

Aprendre amb Robocode

- Aprenetatge de programació. Alguns exemples:
  - Programació Orientada a Objectes
  - Extendre classes
  - Herència
  - 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

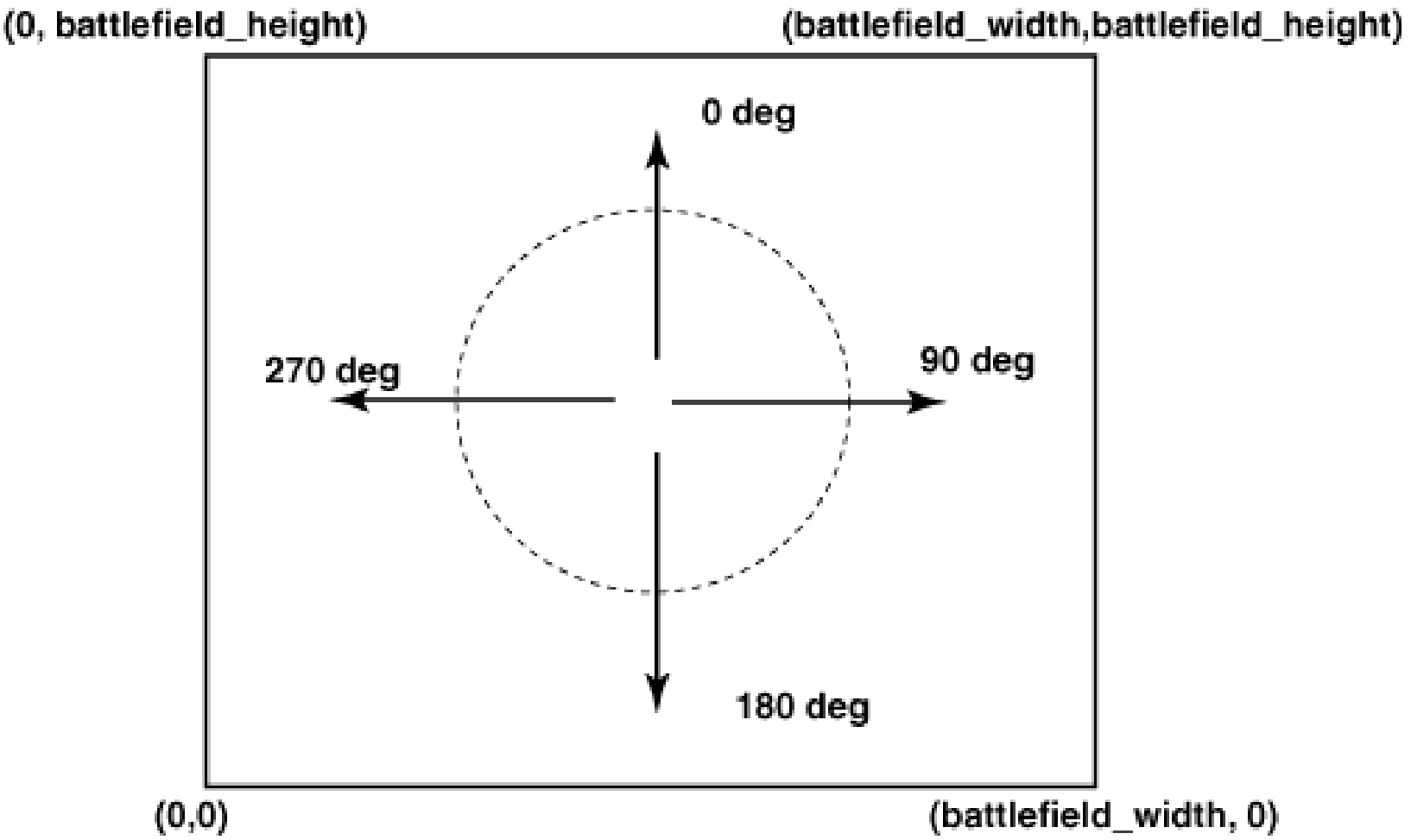
- Quan el robot gira, ho fa de forma conjunta excepte que posem els flags següents:
  - **setAdjustGunForRobotTurn(boolean flag)**: If the flag is set to true, the gun will remain in the same direction while the vehicle turns.
  - **setAdjustRadarForRobotTurn(boolean flag)**: If the flag is set to true, 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)

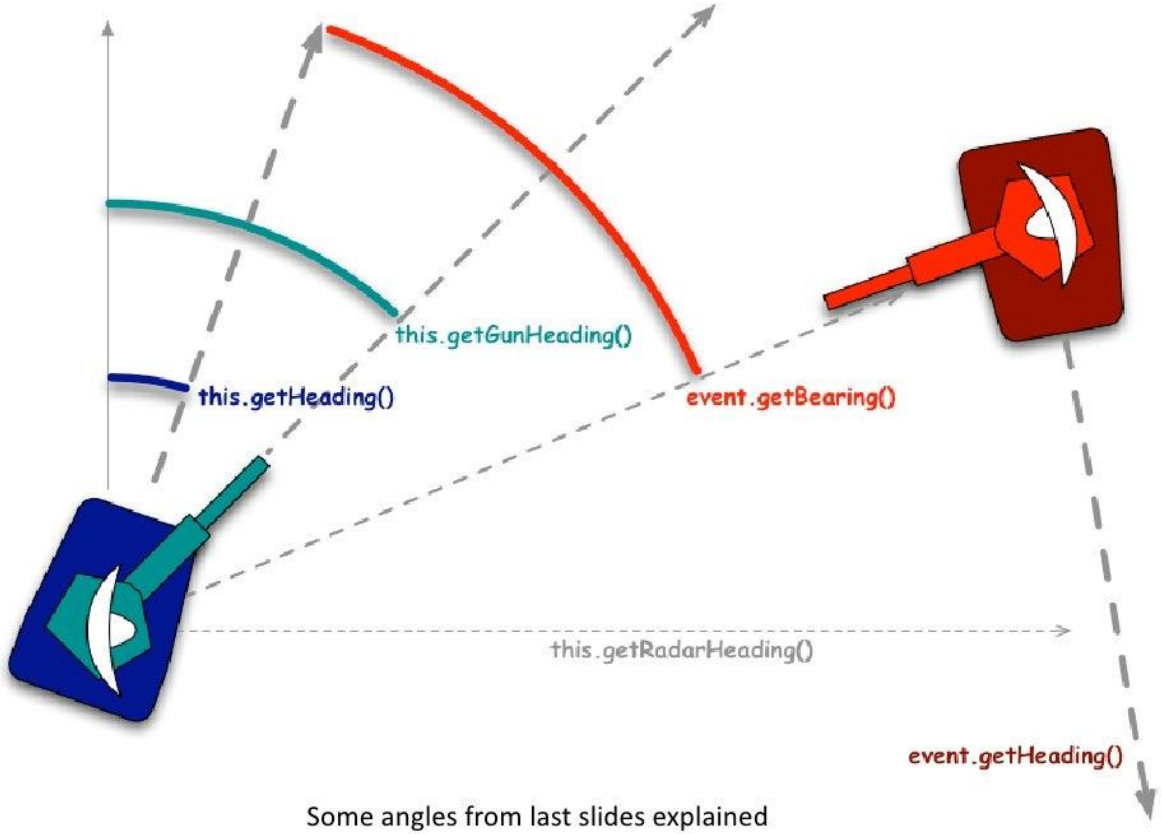
- **getX() , getY()** Obté les coordenades actuals del robot
- **getHeading(), getGunHeading(), getRadarHeading()**

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

Coordenades



Angles



Some angles from last slides explained

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 robot
  - 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

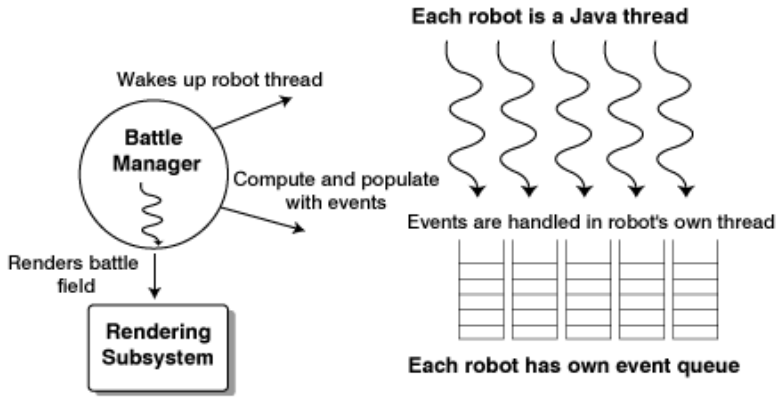
- Altres events
  - onBulletMissed
  - onBulletHitBullet
- No cal programar-los tots.

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

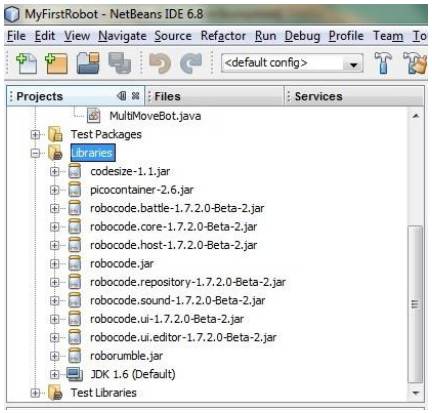
- Descàrrega del Robocode

  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

- Un cop donades les ordres del torn fem:
  - 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);
    }
}
```

Custom Events

- **Adding a Custom Event**
  http://mark.random-article.com/robocode/improved_movement.html

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)

TeamRobot

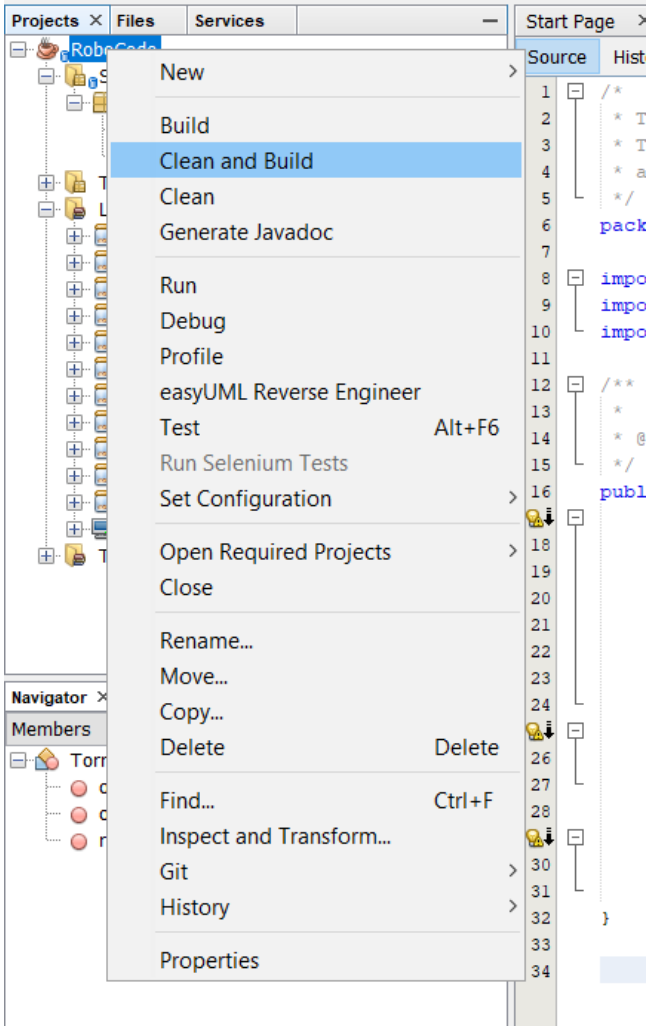
- Per definir un equip de robots, en el mateix paquet on estan les classes del robot cal crear un arxiu :
  - *[nom_equip].*team

  El contingut de l'arxiu cal posar-lo com es mostra a continuació:

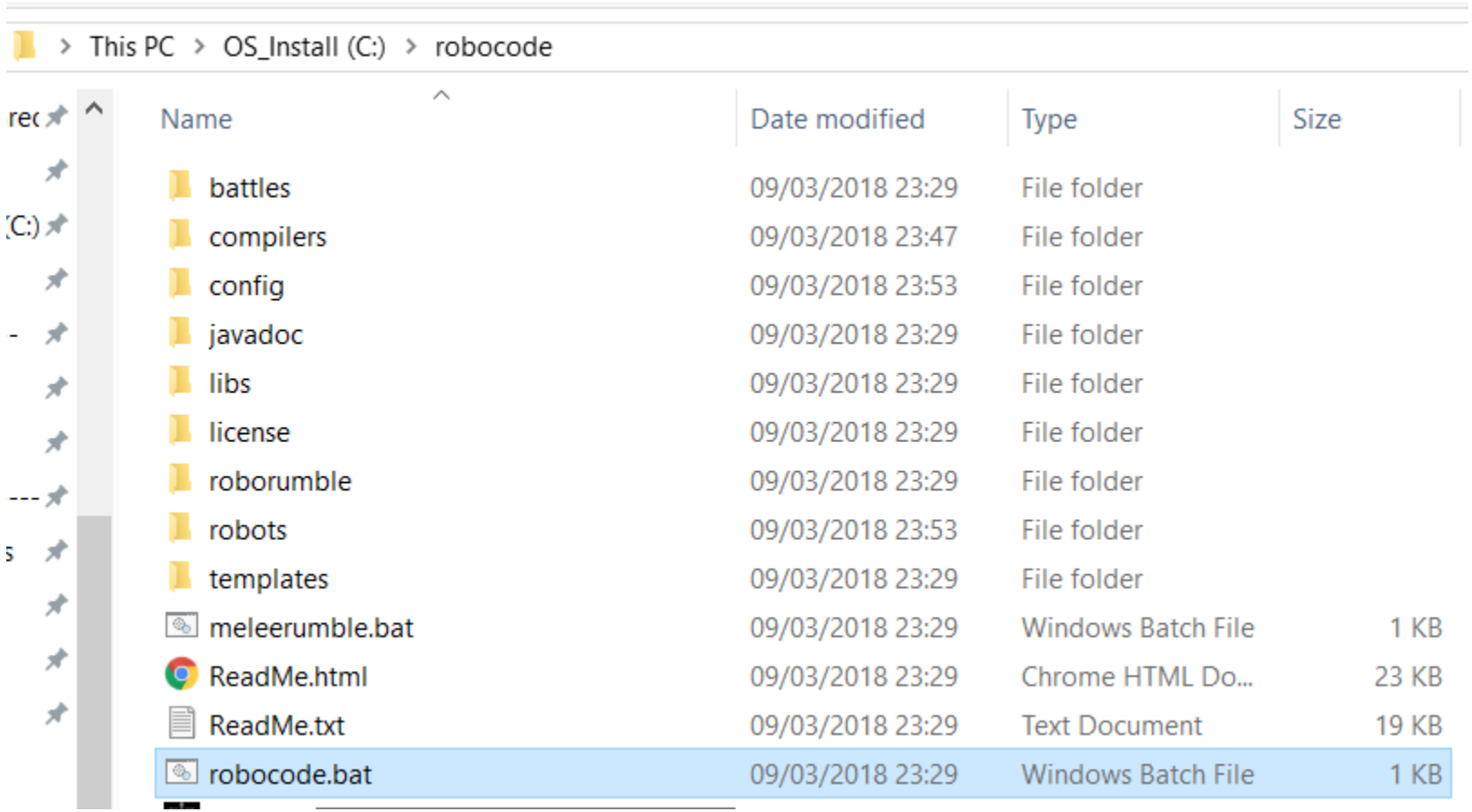
```
team.members=edu.upc.prop.teams.antteam.AntBot,edu.upc.prop.teams.antteam.AntBotDroid,edu.upc.prop.teams.antteam.AntBotDroid
```

  Podeu combinar 5 robots iguals o anar fent variacions. El robots tots han d'estendre de la classe *TeamRobot*, i podem usar fins a 5 robots de classes diferents.

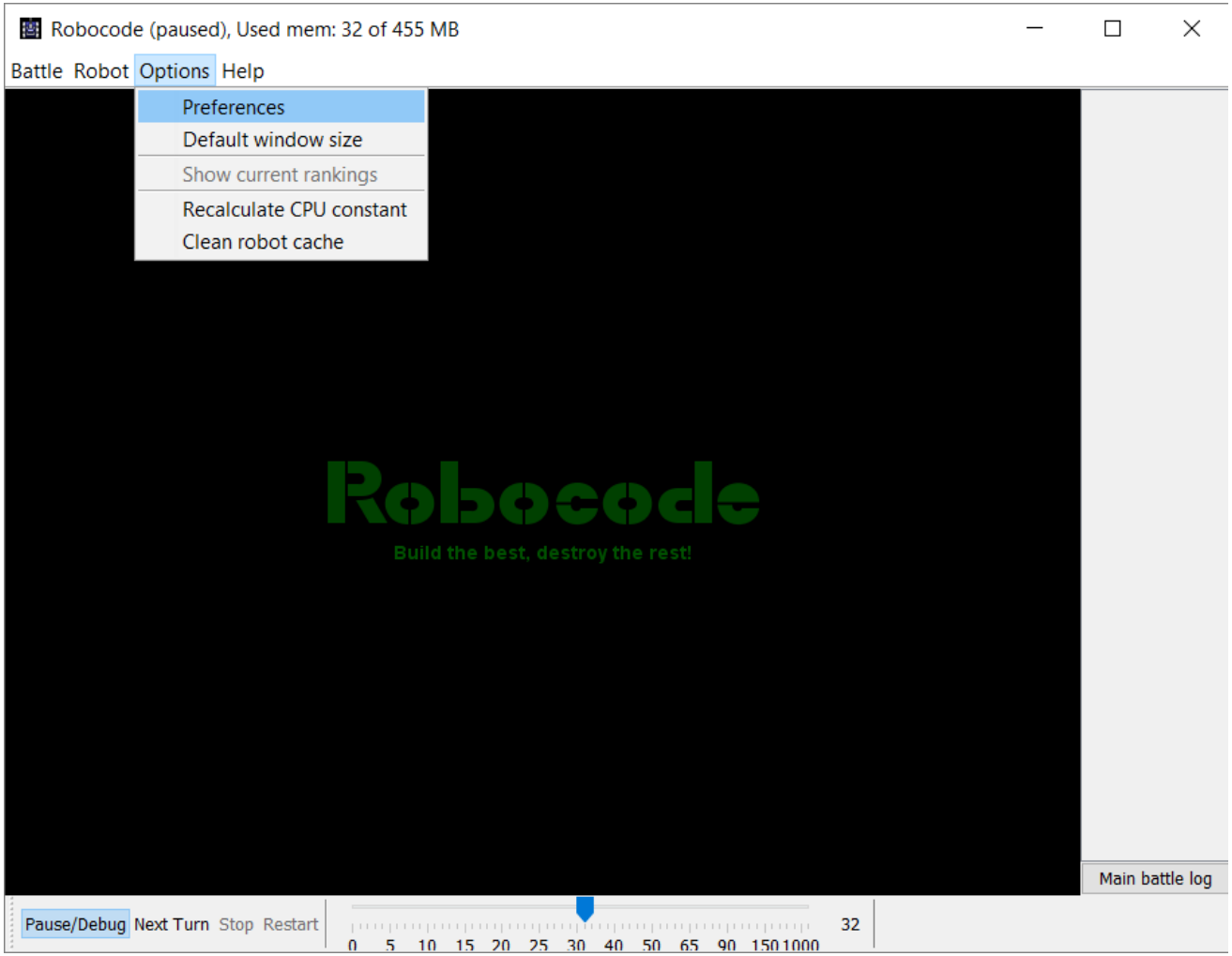
Carregant robots



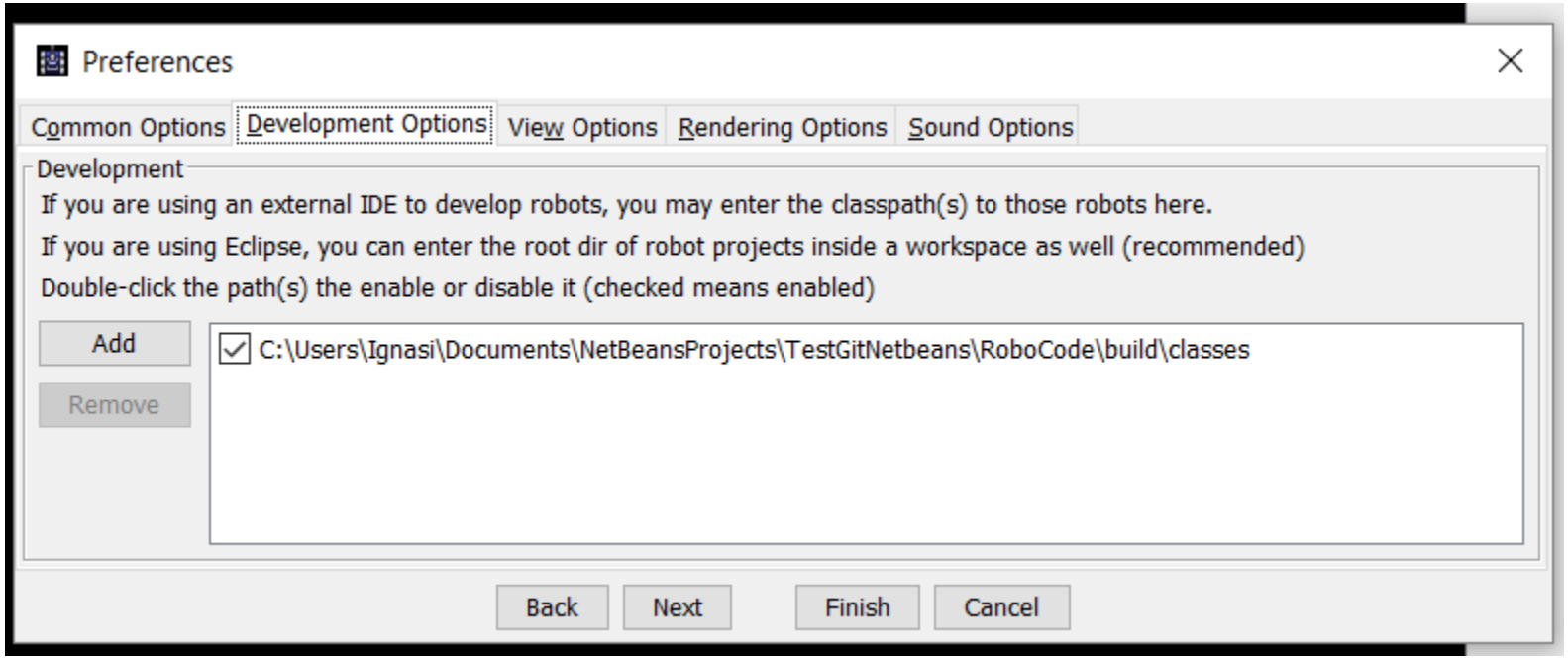
Carregant robots



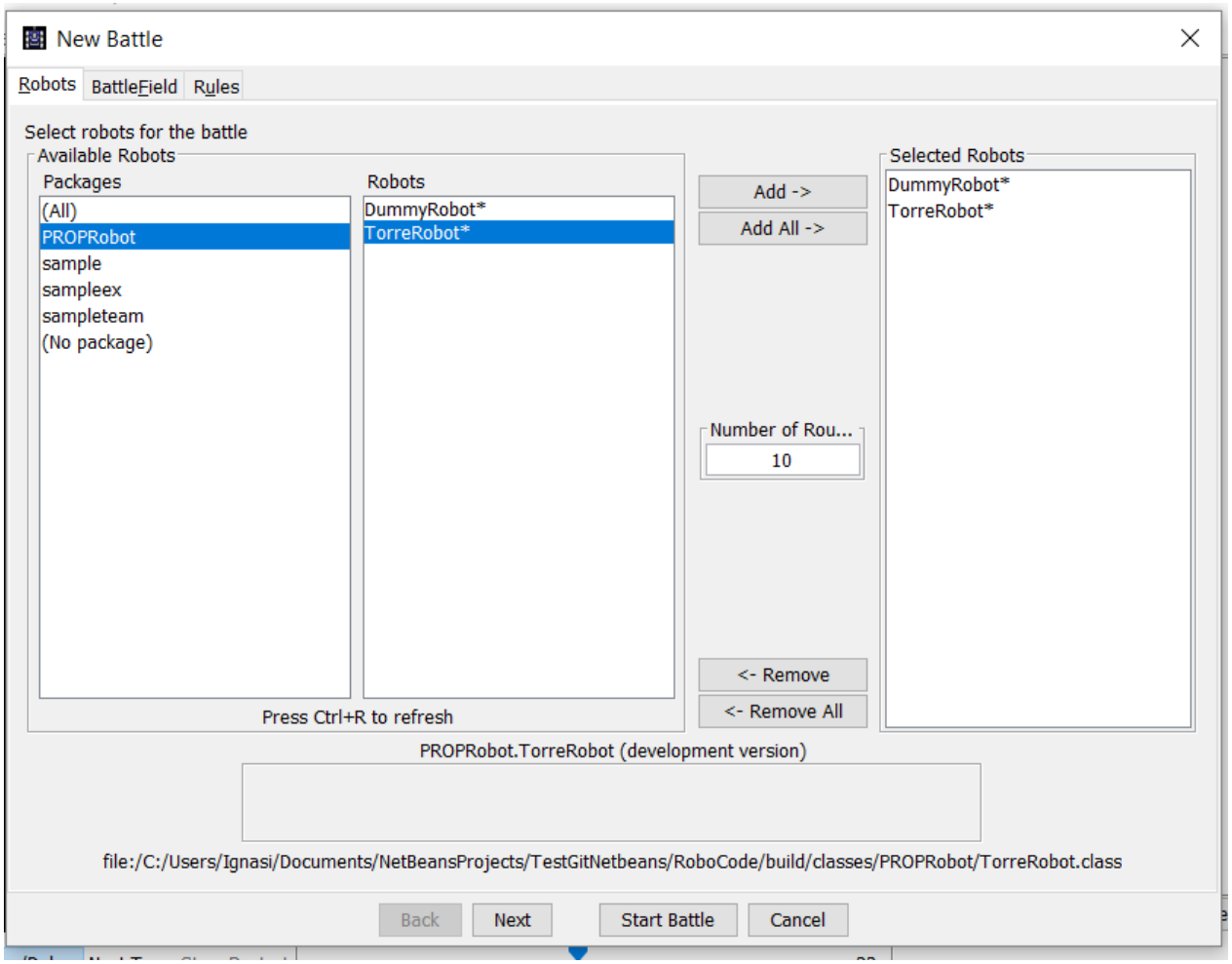
Carregant robots



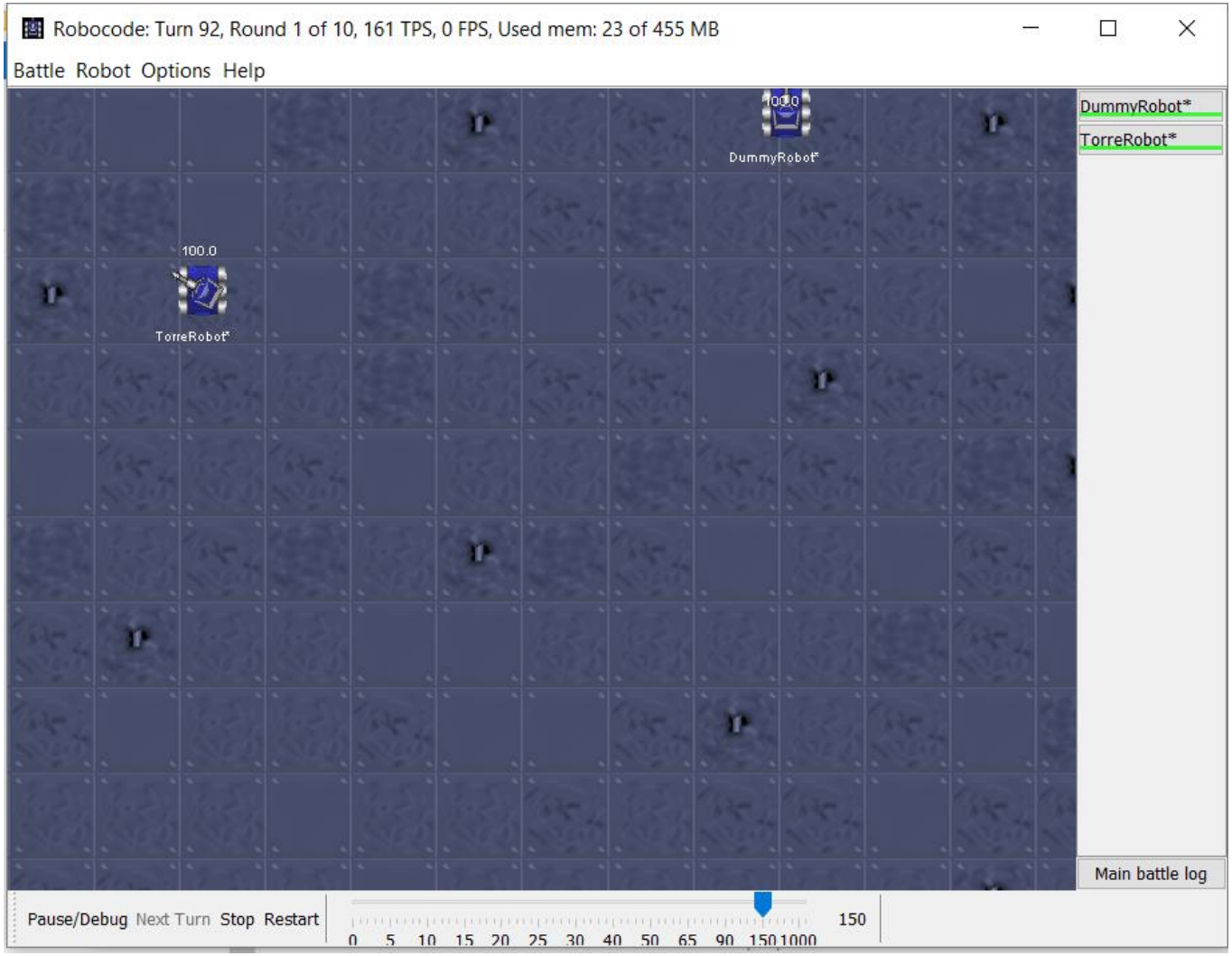
Carregant robots



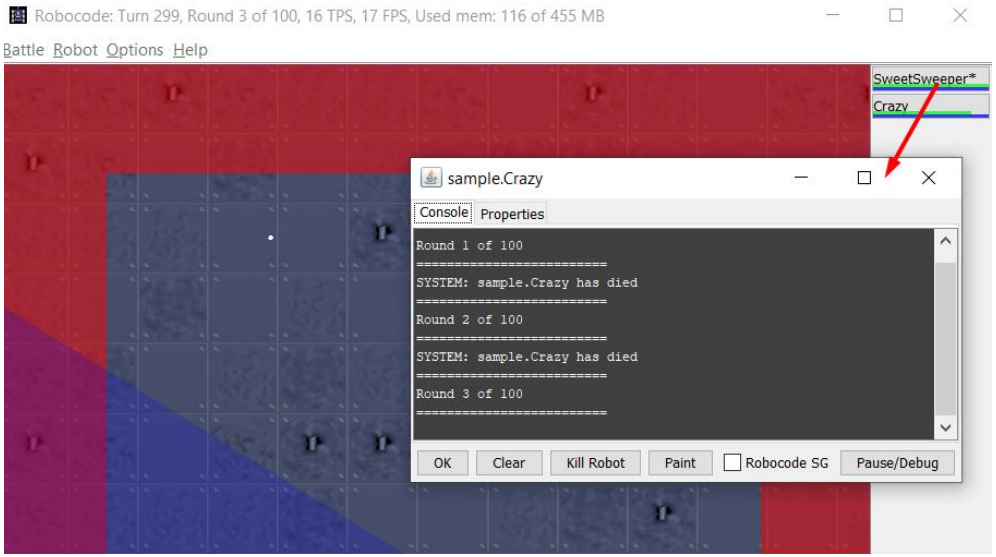
Carregant robots

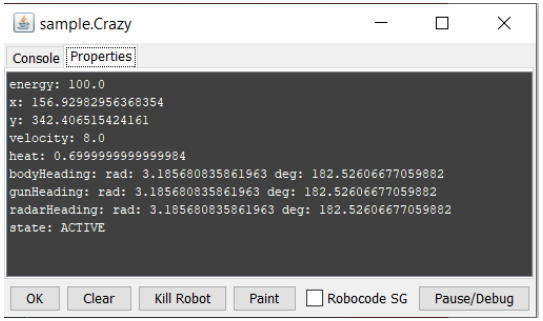


Carregant robots



Depuració (text)

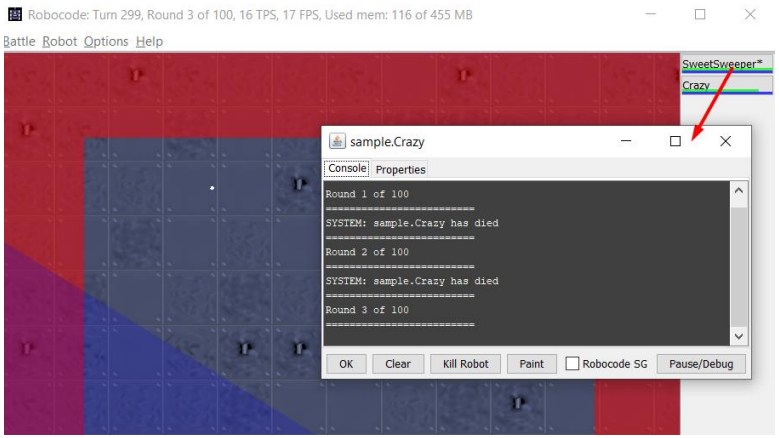




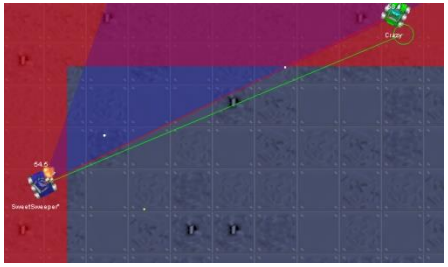

```
setDebugProperty("TIME", "" + getTime());
```

Depuració (gràfica)

- Cal activar-la amb el botó "Paint"

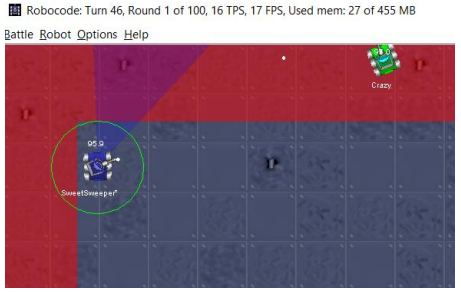


- MOLT
- ÚTIL!



- 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 (veure data límit).
- 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: "TimidínRobot"

En aquest primer exercici desenvolupareu un robots anomenat "**TimidinRobot**" amb l'API d'AvancedRobot per tal que segueixi un comportament prefixat.

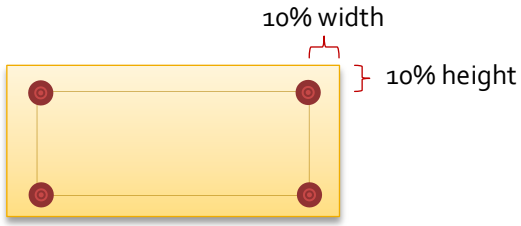
- **FASE 0:**
  - Detecta l'enemic (radar girant) i decideix quina és la cantonada més allunyat d'ell.
- **FASE 1:**
  - Anar a la cantonada calculada anteriorment. La trajectòria es farà en línia recta cap a la posició calculada, amb el radar apuntant cap endavant. En el cas de detectar algun obstacle al camí (un tanc essencialment) disparen i modifiquem l'angle d'aproximació òptim sumant X graus. A cada torn cal recalcular la direcció òptima (en línia recta) cap a l'objectiu, i es torna a sumar l'angle X mentre es continuï detectant l'obstacle.
  - La modificació de l'angle pot ser positiva o negativa, trieu bé per prevenir que no ens porti a xocar contra els marges.
- **FASE 2:**
  - S'escaneja cercant enemics, un cop detectat és centra el radar sobre ell i es dispara. La potència de dispar ha de ser inversament proporcional a la distància.
- Implementació:
  - Useu el patró de màquina d'estats (*state design pattern*) . Fixeu-vos que cada fase és un estat, i en cada estat heu de gestionar els *events* del robot de forma diferent.
  - Podeu trobar més informació sobre el patró a : https://www.digitalocean.com/community/tutorials/state-design-pattern-java
  - Podeu provar el tema d'esquivar obstacles fent un equip de un **TimidínRobot** i enfrontar-lo a un equip de 5 o 6 **SittingDucks**.

Exercici 2: "FollowTheLeaderTeam"

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

- **FASE 0: Handsahe**
  - En aquesta fase inicial, es decideix una jerarquia entre els tancs: Team Leader, segon, tercer, quart, cinquè.
  - L'elecció del TeamLeader (**TL**) es farà de forma aleatòria, i caldrà que els robots ho consensuin. Posteriorment, la jerarquia s'estableix per distàncies. El robot més proper al **TL** és el segon, el robot més proper al segon és el tercer, etc.
  - Useu el mètode *onPaint()* per pintar una circumferència groga al voltant del TL .

- **FASE 1: Conga**
  - El TL segueix una ruta rectangular fixa al voltant del camp de batalla. Inicialment calcularà la cantonada més propera i s'hi dirigirà en línia recta. Un cop allí iniciarà la ruta indicada en la figura següent, en sentit horari:



  - Adapteu la implementació del robot Timidín per tal que el **TL** esquivi als robots rivals per arribar al seu destí. El radar apuntarà endavant per detectar obstacles, però cada cert temps caldria que fes un escombrat per detectar enemics i disparar.
  - Si mor el **TL**, el segon esdevindrà **TL**.
  - Pel que fa a la resta de bots que no són **TL**, cada bot segueix al seu antecessor (p.ex. el tercer segueix al segon). Si mor l'antecessor hem de seguir al immediatament anterior (seguint l'exemple, si mor el segon, seguirem al primer).
  - Mentre els robots van seguir al **TL**, van disparant tots al mateix enemic. Si l'enemic mor, es consensua qui serà el proper enemic a batre.
  - Cada 15 segons, s'inverteixen els rols de l'equip i el sentit de rotació (l'últim passa a ser el TL i gira en sentit contrari).

- **Consells i entorn**
  - Proveu inicialment el vostre equip amb un equip de **SittingDucks.**
  - El camp de batalla serà de 1000x800.

Activitat

- Representa un 10% de la nota total de PROP
- Nota
  - 80% Codi
    - Exercici 1: 30% Desenvolupament del *TimidínRobot*
      Exercici 2: 70% Desenvolupament del *FollowTheLeaderTeam*.
  - 20% Javadoc, documentació

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).
    - **Plantejament i organització del codi** pel desenvolupament del robot.
    - Detalls de la implementació, incloent l'explicació dels càlculs (p.ex. per situar a l'enemic).