Principles of Computer Game Design and Implementation

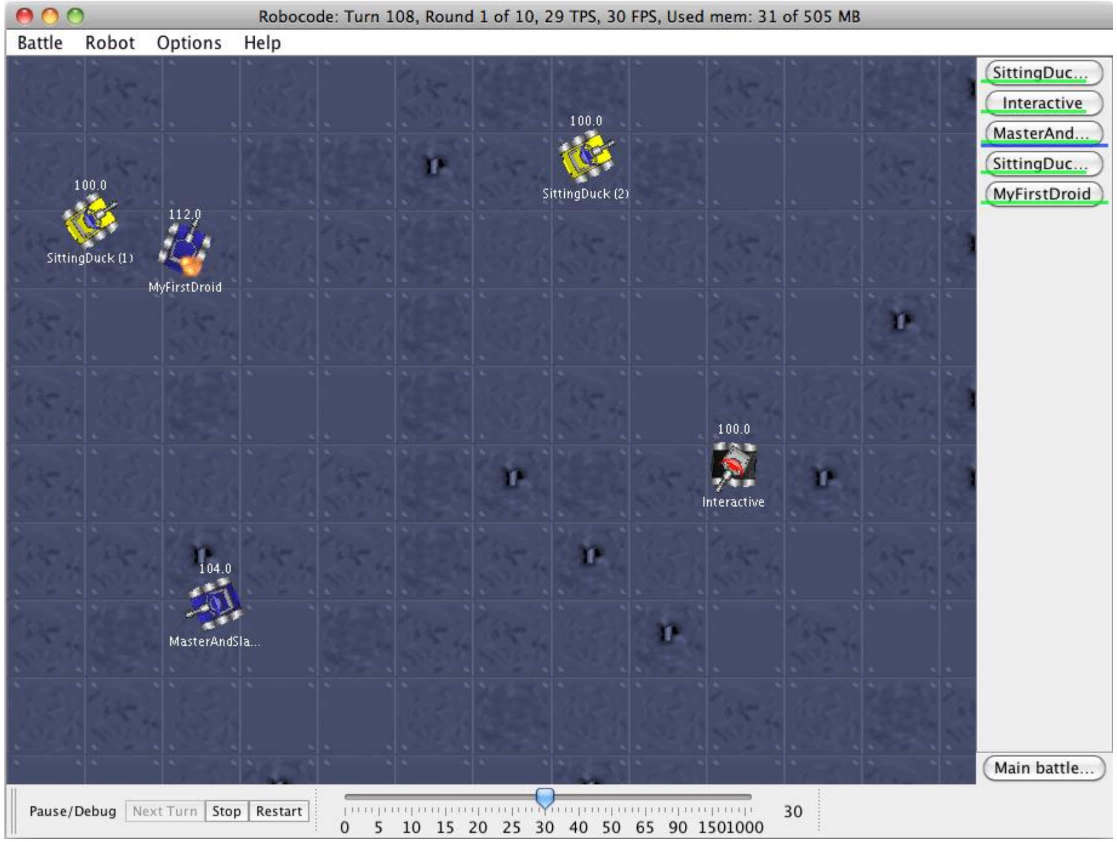
Lecture 21

Outline for today

- Robocode: our second continuous assignment

Robocode

- Educational game with the aim to develop a robot battle tank to battle against other tanks.

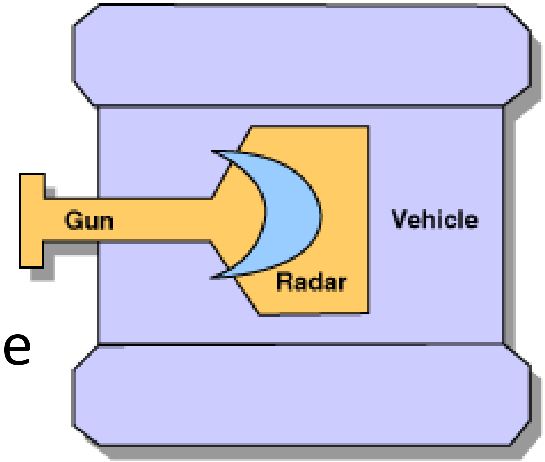

Robocode: Turn 108, Round 1 of 10, 29 TPS, 30 FPS, Used mem: 31 of 505 MB

Every tank is controlled by Java (or C#) code

Anatomy of a Robot

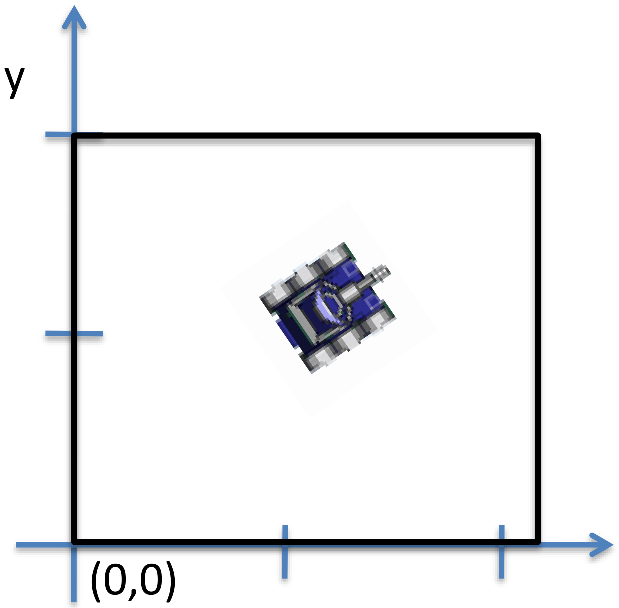
- Every tank is a vehicle equipped with
  - A rotating gun
  - A rotating radar

- The vehicle, gun and radar can rotate independently
  - Initially, all aligned
    - May not be a good idea to decouple the gun and radar (at first at least)



Battle Field

- Rectangular arena
  - getBattleFieldHeight()
  - getBattleFieldWidth()
  - getX()
  - getY()
- Size varies between 400x400 and 5000x5000



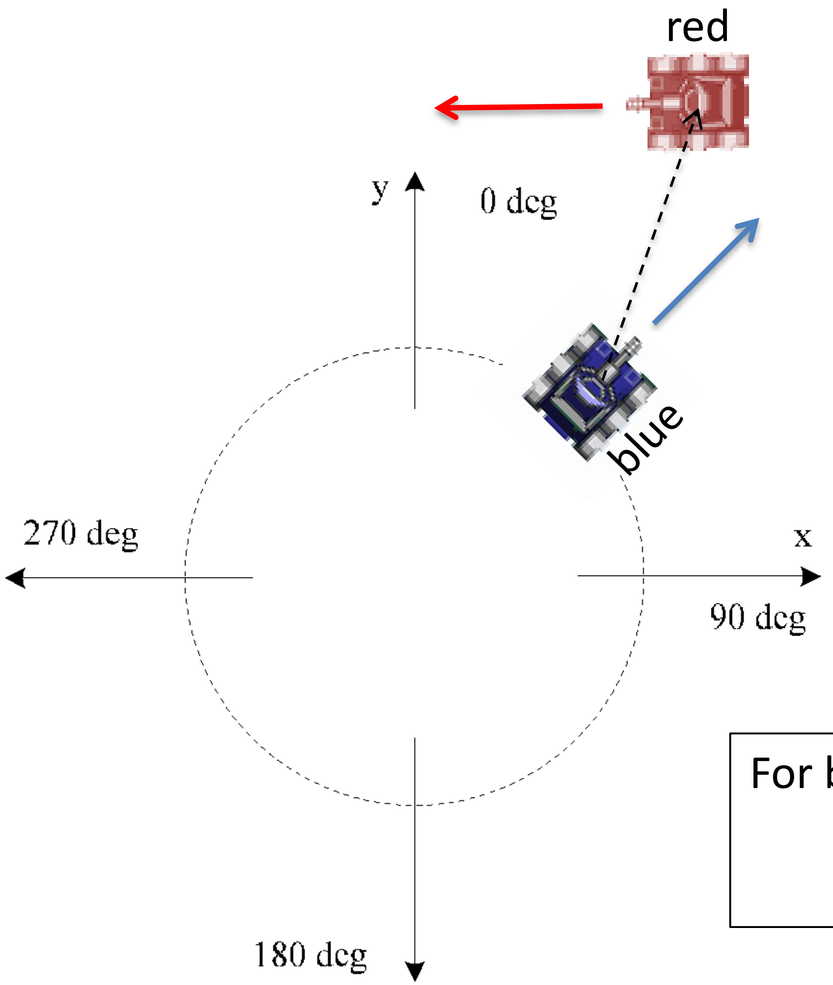
Game Rules

- Every bot has some **energy** (100 at start)
  - When energy is 0 the bot is disabled
  - When a disabled bot is hit it is destroyed
- **Shooting** costs energy
  - New energy = energy – bullet **firepower**
  - Bullet **firepower** is a (double) number between 0.1 and 3
- **Hitting** an enemy bot with a bullet **gives** energy
- **Being hit takes** energy
- **Ramming** into a wall **takes energy**
  - For **AdvancedRobot** only

Time and Space

- Time is measured in ***ticks***
  - 1 tick = 1 turn
    - Every bot executes commands for 1 tick
    - If action is unfinished, it is halted
- Distance is measured in *pixels*
- Angles are measured in degrees

Directions



- **Heading**
  - The direction of bot movement
- **Bearing**
  - Direction *relative* to heading

For blue robot:
Heading = 45°
Bearing to the red robot ≈ 340°

Bot Motion

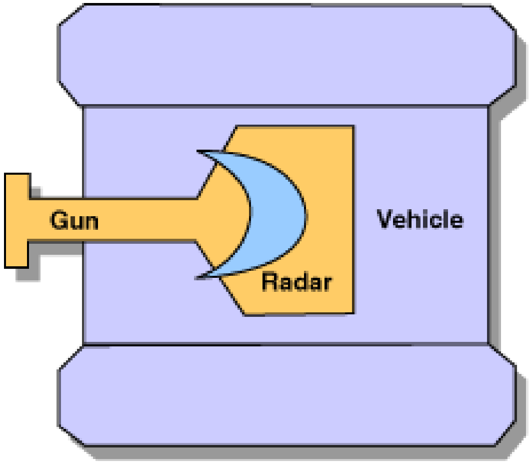
- A robot
  - Accelerates at the rate of 1 pixel/turn/turn
  - Decelerates at the rate of 2 pixels/turn/turn

$$\mathbf{V} = \mathbf{a}\,t$$

  - Velocity cannot exceed 8 pixels / turn
  - Automatically accelerates/decelerates based on the distance to move

Robot, Gun and Radar Rotation Limits

- Max rate of robot rotation
  - (10 - 0.75 * abs(velocity)) deg / turn
  - The faster you're moving, the slower you turn
- Max rate of gun rotation
  - 20 deg / turn
- Max rate of radar rotation
  - 45 deg / turn



Bullets

$0.1 < firepower \le 3$

- **Damage:**
  - 4 * firepower, if firepower ≤ 1
  - 4 * firepower + 2 * (firepower – 1), if firepower > 1
- **Velocity:**
  - 20 - 3 * firepower
- **Power returned on hit:**
  - 3 * firepower
- **GunHeat generated:**
  - 1 + firepower / 5 You cannot fire if gunHeat > 0
  - The gun cools down at the rate of 0.1 per turn

Processing Loop

- Battle view is (re)painted.
- All robots execute their code until they take action (and then paused).
- Time is updated (time = time + 1).
- All bullets move and check for collisions. This includes firing bullets.
- All robots move (gun, radar, heading, acceleration, velocity, distance, in that order).
- All robots perform scans (and collect team messages).
- All robots are resumed to take new action.
- Each robot processes its event queue.

public class myRobot extends …

- A Robocode bot `extends` one of
  - `Robot`
  - `AdvancedRobot`
  - `JuniorRobot`
    - ~~Not in the labs~~
    - For those who are not used to “getters”
      - this.getEnergyLevel()
    - Please do not use

Default Robot (1)

```
package comp222;
import robocode.*;
public class XiaoweiH extends Robot
{
  public void run() {
    while(true) {
      ahead(100);turnGunRight(360);
      back(100);turnGunRight(360);
    }
  }
…
```

Default Robot (2)

```
…
   public void onScannedRobot
                     (ScannedRobotEvent e) {
      fire(1);
   }
   public void onHitByBullet
                     (HitByBulletEvent e) {
      back(10);
   }
   public void onHitWall(HitWallEvent e) {
      back(20);
   }
}
```

Robot vs AdvancedRobot (1)

Blocking method inherited from Robot	Non-blocking methods inherited from AdvancedRobot
turnRight()	setTurnRight()
turnLeft()	setTurnLeft()
turnGunRight()	setTurnGunRight()
turnGunLeft()	setTurnGunLeft()
turnRadarRight()	setTurnRadarRight()
turnRadarLeft()	setTurnRadarLeft()
ahead()	setAhead()
back()	setback()

Robot vs AdvancedRobot (2)

- Non-blocking calls return immediately
  - One can do more than one action per turn
  - Call `execute()` to run pending actions

- If an advanced robot rams into a wall, it looses
  - Velocity / 2 + 1 energy

More Info

- Robocode web page
  - http://robocode.sourceforge.net/
- Robowiki
  - http://robowiki.net/
- Robocode API
  - http://robocode.sourceforge.net/docs/robocode/

Assignment 2

- Code (30%)
- Documentation (40%)
- Tournament (30%)

You need to implement one of behaviour models considered in the module

- FSM
- Behaviour trees
- Decision trees
- ….

Documentation **(40%)**

- Describe the behaviour model of your choice **(10%)**
- Design the bot using this model **(20%)**
  - E.g. for FSMs, draw states and transitions
- Describe your implementation **(10%)**

Implementation **(30%)**

- Providing response to battle events **10%**
  - onScannedRobot(),…
- Following the design **10%**
- Clarity and style of code **10%**

Naming Convention

- Package name: `comp222`
- Robot name: any unique name
  - FirstnameLastname
    - E.g. `XiaoweiHuang`
  - `Astudentnumber`
    - E.g. if the student number is 200812345
    - `A200812345 (can compromise the ID)`
  - `Ayourfullbirthday`
  - …
- `Clearly identify authorship in the comments!`

Tournament **(30%)**

- Randomly split into groups of **around 10 bots** each
- Winners will progress into the next round
- ~~Details to be finalised~~

Use of Sources

Any robot with code borrowed (with or without acknowledgment of sources) from elsewhere will be disqualified from the tournament

Crime Does Not Pay!

- When the module was run for the first time, some students submitted code downloaded from the Internet to improve their chances in the tournament

- This is NOT a good idea, and here’s why…

Case Study (1)

- Student A cheated and got 30% in the tournament (initially)
  - Got caught and had Tournament marks stripped
  - Did not understand the code and got only 5% for the design
  - The implementation did not match the design -> poor description, low mark

Total final mark: 35%

Case Study (2)

- Student B submitted 40 lines of code
  - Code matched the design
  - Decent performance in the tournament
  - Good explanation of the design
  - Good description of the implementation

<u>Total final mark: 90%</u>