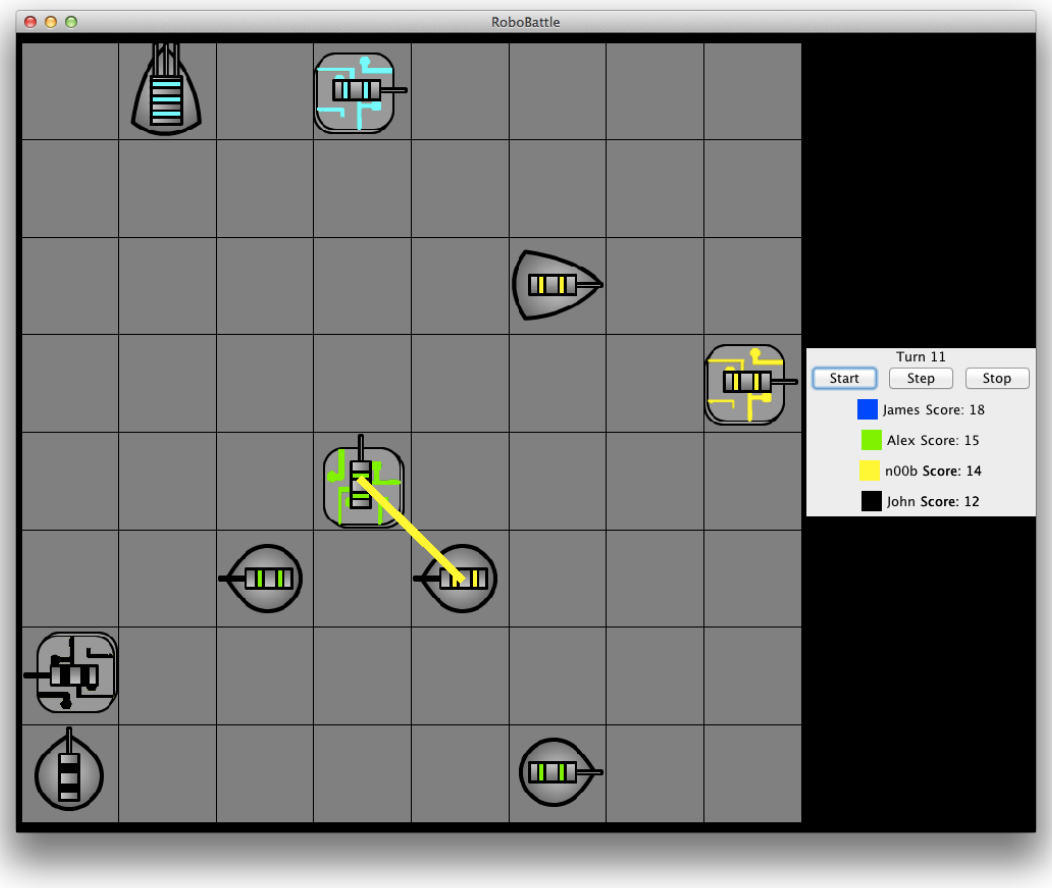
RoboBattle

PROGRAMMABLE STRATEGY ROBOT BATTLES

James Lennon
January 23, 2014
Version: 1.0

THE GAME



Objective

In RoboBattle, users program robot artificial intelligences to battle each other in the RoboBattle Arena. To win, a team must either:

- Be the last team standing (destroy all other robots)
- Gain the most points

The Arena

The RoboBattle Arena is a 2D grid with a specified width and height. Specifically, the upper-right corner has the coordinates (0,0) and the lower-left corner has the coordinates (width, height). No robot may move off the grid, and no two robots may occupy the same location.

TURNS & SCORING

Turns

RoboBattle is a turn-based game, meaning that each robot takes turns acting. One "turn" is defined as either moving in one of the cardinal directions, shooting another robot in range, or skipping a turn. Each turn, a team gains 1 "resource point" that it can spend to make a robot at an adjacent location (only mainframe robots can do this, see "robots" section below).

Scoring

For each turn that a team is still alive (has at least one robot), that team gets 1 point. When a robot on a team shoots another robot, that team gets a number of points equal to the damage inflicted. Each time a mainframe robot makes another robot, its team gets as many points as that robot's cost in resource points.

MAKE A ROBOT STRATEGY

The RobotStrategy Interface

To start creating your own RoboBattle team, you must first create a class that implements the RobotStrategy interface, which requires the following methods:

Method Signature	Description
public Turn getTurn(RobotState state)	Called each turn to determine what the robot will do next
public void receiveMessage(String msg, Point p)	Called when the robot receives a message from one of its teammates
public void onHit(int health, Point source)	Called when the robot is shot by another robot

The Turn Class

The getTurn(RobotState state) method returns a Turn object. There are three types of Turn objects:

Turn Type	Constructor	Description
MoveTurn	public MoveTurn(int direction)	Moves the robot in one of four directions. The integer parameter can be MoveTurn.UP, MoveTurn.RIGHT, MoveTurn.DOWN, or MoveTurn.LEFT.
ShootTurn	public ShootTurn(int x, int y) or public ShootTurn(Point tar)	Shoots at the specified coordinates (if they are within range)
MakeTurn	public MakeTurn(Point loc, int damage, int health, int range, int speed, RobotStrategy strategy)	Creates a new robot at the specified coordinates with the specified damage, health, range, speed, and strategy.
SkipTurn	public SkipTurn()	Skips the turn (duh)

The RobotState Object

In the getTurn method, you are given a RobotState object. This object contains all the information and functions you will need to create your strategy. Its methods are listed below:

Method Signature	Description
public boolean isInBounds(int x, int y)	Return true if the point is on the grid, false otherwise.
public boolean isValid(int x, int y)	Return true if the point is on the grid and is empty, false otherwise.
public boolean isInRange(int x, int y)	Return true if the point is within the range of the robot.
public int getResourcePoints()	Returns the amount of resource points the team has.
public ArrayList<Robot> getRobotsInRange()	Returns an ArrayList containing all other robots within the range of the robot.
public Robot get(int x, int y)	Returns a Robot object with information about the robot at the specified coordinates, if it is within the range of the robot, returns null if it is empty or out of range.
public Point getLocation()	Returns a Point object with the coordinates of the robot.
public int getX()	Returns the x coordinate of the robot.
public int getY()	Returns the y coordinate of the robot.
public Point randomValidAdjacentLocation()	Returns an adjacent point to the robot that is both on the grid and empty, null if there is no such point.
public String getTeam()	Returns the name of the robot's team.
public int getHeight()	Returns the height of the grid.
public int getWidth()	Returns the width of the grid.
public void transmitMessage(String msg, Point p)	Transmits a message (in the form of a String and a Point) to the all other members of the robot's team.
public Robot getRobot()	Returns a Robot object with information about the robot.

Building Robots

If you are programming the strategy of a mainframe robot, you can build more robots on your team with MakeTurn objects. Building a robot costs as many resource points as the sum of that robot's damage, health, range, and speed. For example, a robot with damage 1, health 1, range 1, and speed 1 will cost 4 resource points.

ADD YOUR TEAM TO THE ARENA

The RoboBattleArena Class

The RoboBattleArena class is how you will add your robots to the arena. Here its methods:

Method Signature	Description
public RoboBattleArena(int w, int h)	Constructor; initializes the arena with the specified width and height.
public void addMainframe(String string, Point loc, RobotStrategy strategy)	Adds a mainframe robot to the arena at the specified point with the specified strategy.
public void addRobot(String string, Point loc, int damage, int health, int range, int speed, RobotStrategy strategy)	Adds a normal robot to the arena at the given point with the specified damage, health, range, speed, and strategy.
public void show()	Shows the RoboBattle Arena

Example

The following code snippet shows an example of how you could set up a battle:

```
//Initialize the Arena
RoboBattleArena rb = new RoboBattleArena(8, 8);
//Add a Mainframe with a GeniusStrategy on team “James”
rb.addMainframe("James", new Point(0, 0), new GeniusStrategy());
//Add a Mainframe with a SwarmStrategy on team “Alex”
rb.addMainframe("Alex", new Point(7, 7), new SwarmStrategy());
//Add a Mainframe with a SimpleStrategy on team “John”
rb.addMainframe("John", new Point(0, 7), new SimpleStrategy());
//Add a Mainframe with a DumbStrategy on team “n00b”
rb.addMainframe("n00b", new Point(7, 0), new DumbStrategy());
//Show the Arena
rb.show();
```

EXAMPLE TEAM

SimpleStrategy.java

```
public class SimpleStrategy implements RobotStrategy {
    @Override
    public Turn getTurn(RobotState state) {
        if (state.getResourcePoints() >= 8) {
            return new MakeTurn(state.randomValidAdjacentLocation(), 2, 2, 2, 2,
                                new CenterStrategy());
        }
        return state.randomValidMove();
    }

    @Override
    public void receiveMessage(String s, Point point) {
        //Do nothing
    }

    @Override
    public void onHit(int i, Point point) {
        //Do nothing
    }
}
```

CenterStrategy.java

```
public class CenterStrategy implements RobotStrategy {

    private RobotState s;

    private Turn checkAndShoot() {
        ArrayList<Robot> range = s.getRobotsInRange();
        for (Robot r : range) {
            if (!r.getTeam().equals(s.getTeam())) {
                return new ShootTurn(r.getX(), r.getY());
            }
        }
        return null;
    }
```

```
    @Override
    public Turn getTurn(RobotState state) {
        s = state;
        int x = state.getX(), y = state.getY();
        int cx = state.getWidth() / 2, cy = state.getHeight() / 2;
        Turn t = checkAndShoot();
        if (t != null) {
            return t;
        }
        if (x > cx && state.isValid(x - 1, y)) {
            return new MoveTurn(MoveTurn.LEFT);
        } else if (x < cx && s.isValid(x + 1, y)) {
            return new MoveTurn(MoveTurn.RIGHT);
        } else if (y < cy && s.isValid(x, y + 1)) {
            return new MoveTurn(MoveTurn.DOWN);
        } else if (y > cy && s.isValid(x, y - 1)) {
            return new MoveTurn(MoveTurn.UP);
        }
        return null;
    }

    @Override
    public void receiveMessage(String s, Point point) {
        //Do nothing
    }

    @Override
    public void onHit(int i, Point point) {
        //Do nothing
    }
}
```