AI Project

Design Implementation of Multi-Agent Systems.

Outline:

1. Introduction
2. Implementation
3. Demo

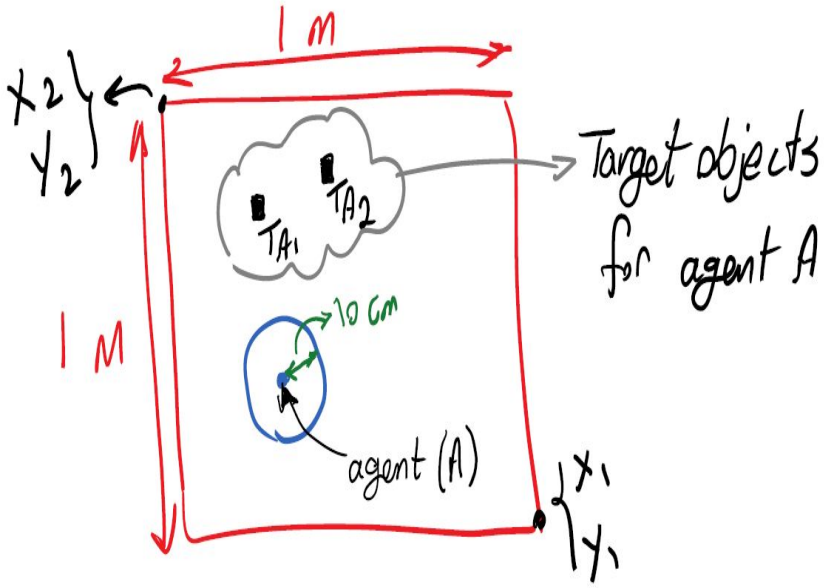
Team Members:

Saleh Nawar | 100536488

Anwar Almukhtar | 100481975

Introduction:

- Environment: a 1 m x 1 m field.
- Five agents (A,B,C,D,E) can move up and down and left and right, one centimeter in every step.
- Assume the agents have unlimited power supply.
- All targets and agents are randomly scattered all over the field. Agents do not have any information about the location of none of the targets and other agents.



Implementation:

Scenario 1: Competition

- The game will be over as soon as one of the agents collects all its targets.
- Only public communication channel is open for all agents. The private channels are closed.

Scenario 2: Collaboration

- The game is not over until all agents collect their own targets.
- Both public and private channels are open. Example of a private communication: Agent A locates TB1. It may or may not notify the agent B the location of TB1.

Scenario 3: Compassionate agents

- The game will be over as soon as one of the agents collects all its targets.
- Both public and private channels are open. Example of a private communication: Agent A locates target TB1. It may or may not notify the agent B the location of TB1.

- Java vs Python
- intelliJ IDEA
- 4 Classes

Target.java

```java
public class Target {

    //positions of target
    private int x;
    private int y;

    //name and color of agent
    private String name;
    private Color color;

    /**
     * constructor
     *
     * @param x
     * @param y
     * @param name
     * @param color
     */
    public Target(int x, int y, String name, Color color) {
        this.x = x;
        this.y = y;
        this.name = name;
        this.color = color;
    }
```

Agent.java

```java
 */
public Agent(MultiAgentSystem system, int x, int y, String name, Color color) {
    this.system = system;
    this.x = x;
    this.y = y;
    this.name = name;
    this.color = color;
}
```

```java
//check if target in the neighbors
private boolean isInNeighbors(List<Object> neighbors, Target target){...}

//move by sensor that check within circle
private boolean moveBySensor(List<Object> neighbors){...}

//move by channel (private or public)
private boolean moveByChannel(List<Target> channel){...}
/**...*/
public void move(){...}
/**...*/
public int getX() { return x; }
/**...*/
public void setX(int x) { this.x = x; }
/**...*/
public int getY() { return y; }
/**...*/
public void setY(int y) { this.y = y; }
/**...*/
public String getName() { return name; }
/**...*/
public Color getColor() { return color; }

/**...*/
public boolean isRunning() { return running; }

/**...*/
public void setRunning(boolean running) { this.running = running; }

/**...*/
public void receiveToPrivateChannel(Target target) { privateChannel.add(target); }

/**...*/
public void receiveToPublicChannel(Target target) { publicChannel.add(target); }

/**...*/
public int getSteps() { return steps; }
```

Board.java

```java
import ...

/serial/
public class Board extends JPanel{

    //reference to MultiAgentSystem
    private MultiAgentSystem system;

    /**
     * constructor
     * @param system
     */
    public Board(MultiAgentSystem system) { this.system = system; }

    @Override
    public void paintComponent(Graphics g) {

        super.paintComponents(g);

        //clear
        g.clearRect( i: 0,  i1: 0, getWidth(), getHeight());

        //draw agents
        for (Agent agent: system.getAgents()){
            g.setColor(agent.getColor());
            g.fillOval( i: agent.getX() * 8 - 4,  i1: agent.getY() * 8 - 4,  i2: 8,  i3: 8);
        }

        //draw targets
        for (int i = 0; i < system.getTargets().length; i++){
            Target target = system.getTargets()[i];
            if (target != null){
                g.setColor(target.getColor());
                g.fillRect( i: target.getX() * 8 - 4,  i1: target.getY() * 8 - 4,  i2: 8,  i3: 8);
            }
        }
    }
}
```

MultiAgentSystem.java

```java
    public MultiAgentSystem(){...}

    /**
     * simulate
     */
    private void move(){...}

    //thread that draw the panel
    class MoveThread extends Thread{...}


    /**...*/
    public void gotTarget(Agent a, Target t){...}

    /**...*/
    public boolean isEmpty(int x, int y){...}

    //check 2 points are adjacency
    public boolean isAdjcency(int x, int y, int x1, int y1){...}

    /**...*/
    public List<Object> getObjects(int x0, int y0, int radius){...}

    /**...*/
    private void createRandomAgentsTargets(){...}

    /**...*/
    public int getScenario() { return scenario; }



    /**...*/
    public Agent[] getAgents() { return agents; }

    /**...*/
    public Target[] getTargets() { return targets; }

    /**...*/
    public static void main(String[] args) {...}
}
```

Demo !!!

Thank You