# ASSIGNMENT REPORT


**Assignment ID:** 2

**Student Name:** Liu Leqi（刘乐奇）

**Student ID:** 12011327

# DESIGN

## Message Processing

The `Message` is changed to send location grid point so that the explorers can arrive at the target more easily.

```java
public class Message {
    private Explorer sender;
    private GridPoint pt;
    private MessageType type;

    public Message(Explorer sender, GridPoint pt, MessageType type) {
        this.sender = sender;
        this.pt = pt;
        this.type = type;
    }

    public Explorer getSender() {
        return this.sender;
    }

    public GridPoint getContent() {
        return this.pt;
    }

    public MessageType getType() {
        return this.type;
    }

    public enum MessageType {
        MATERIAL_FOUND,
        DEST_FOUND,
        NEED_HELP,
        NORMAL_EXPIRED,

        FINDING_GROUP,
    }
}
```

The messages are processed by `MessageManager`

```java
public class MessageManager {

    public void sendMessage(Explorer recipient, Message message) {
        recipient.receiveMessage(message);
    }

    public void receiveMessage(Message message) {
        if (message.getType() == MessageType.NORMAL_EXPIRED) {
            this.messageQueue.removeIf(msg -> msg.getContent().equals(message.getContent()));
        } else {
            this.messageQueue.add(message);
        }
    }

    public void broadcastMessage(Message message, boolean onlyGroup) {
        List<GridCell<Explorer>> nghCells = explorer.getThisGridNeighbour(clazz:Explorer.class, needShuffle:true, communicateRatio,
                communicateRatio);
        for (GridCell<Explorer> cell : nghCells) {
            if (cell.size() > 0) {
                List<Explorer> explorersAround = StreamSupport.stream(cell.items().spliterator(), parallel:false)
                        .filter(otherExplorer -> otherExplorer != explorer).toList();
                for (Explorer otherExplorer : explorersAround) {
                    // send only to explorers in same group
                    if (onlyGroup && explorer.getGroupId() == otherExplorer.getGroupId()) {
                        sendMessage(explorer, message);
                    }
                    if (!onlyGroup) {
                        sendMessage(explorer, message);
                    }
                }
            }
        }
    }

    public void processMessages() {
        Context<Object> context = ContextUtils.getContext(explorer);    Type safety: The expression of type Context needs unchecked co
        Network<Object> net = (Network<Object>) context.getProjection(name:"agents_network");    Type safety: Unchecked cast from Proje
        while (!messageQueue.isEmpty()) {
            Message message = messageQueue.poll();
            switch (message.getType()) {
                case MATERIAL_FOUND -> explorer.addMaterialSpot(message.getContent());
                case DEST_FOUND -> explorer.addDestination(message.getContent());
                case NEED_HELP -> {
                    if (!explorer.nearThanDest(message.getContent())) {
                        explorer.setNextTargetPoint(message.getContent());
                    }
                }
                case FINDING_GROUP -> {
                    if (!explorer.isInGroup() && !message.getSender().isInGroup()) {
                        explorer.setInGroup(inGroup:true);
                        message.getSender().setInGroup(inGroup:true);
                        explorer.setGroupId(message.getSender().getGroupId());
                        net.addEdge(explorer, message.getSender(), weight:0.0);
                    }
                }
                default -> {
                }
            }
            System.out.println("Processed message: " + message.getContent() +
                    "; type: " + message.getType() +
                    "; from: " + message.getSender().getId());
        }
    }
}
```

Notice that the group up process is finished here. `MessageManager` will set the group id and set the `inGroup` flag to true if both are sigle (not in group).

```
case FINDING_GROUP -> {
    if (!explorer.isInGroup() && !message.getSender().isInGroup()) {
        explorer.setInGroup(true);
        message.getSender().setInGroup(true);
        explorer.setGroupId(message.getSender().getGroupId());
        net.addEdge(explorer, message.getSender(), 0.0);
    }
}
```

## Basic Process

The three basic processes are shown as below:

```java
/**
 * mine for materials
 */
public void mineForMaterials(Material material) {
    System.out.println(x:"Mining!");
    double totalWeight = this.materialsCarried.stream()
            .mapToDouble(Material::getWeight)
            .sum();
    if (totalWeight + material.getWeight() < maxCapacity) {
        this.materialsCarried.add(material);
        removeFromContext(material);
    }
}

/**
 * download resources
 */
public void downloadMaterials(Destination destination) {
    System.out.println(x:"Downloading!");
    if (this.materialsCarried.size() > 0) {
        this.surviveLevel = 100.0;
        this.materialsCarried.sort((m1, m2) -> (int) (m1.getWeight() - m2.getWeight()));
        Material material = this.materialsCarried.remove(index:0);
        this.utility += destination.exchangeMaterial(material);
    }
}

/**
 * trade reources
 */
public void tradeMaterials(Trader trader) {
    System.out.println(x:"Trading!");
    List<Material> materialsToDelete = new ArrayList<>();
    if (!materialsCarried.isEmpty()) {
        for (Material material : materialsCarried) {
            double priceLowerBound = (surviveLevel < dyingThreshold) ? 0 : Math.log(1 + material.getWeight());
            double bid = trader.trade(material, priceLowerBound);
            if (bid > 0.0) {
                double finalBid = trader.acceptTrade(material);
                this.utility += bid;
                this.surviveLevel += bid;
                this.extraResources += (finalBid - bid);
                materialsToDelete.add(material);
            } else {
                break;
            }
        }
    }
    for (Material material : materialsToDelete) {
        materialsCarried.remove(material);
    }
    if (extraResources > 0.0) {
        List<Material> materials = trader.acceptTrade(extraResources);
        materialsCarried.addAll(materials);
        this.extraResources = 0.0;
    }
}
```

The first function is used to mine for materials, which is trivial that it just removes the material.

The second function is used to download materials to the destination. The survival level will come back to its maximum and the utility will be increased depending on the weight of the material.

The third function is used to trade with traders. Both survival level and utility will be increased depending on the weight of the material. During trading, the explorer will get some bonus as extra resources. If the explorer already has some extra resources, it can use them to trade for materials from the traders.

# Simulation

The parameters and simulation results can be seen in the source code files. Following are the icons indicating the characters.


refers to the base station.


refers to the destination.


refers to the explorer.


refers to the material.


refers to the trader.

## Setting parameters

The parameters can be set in the simulation runtime GUI. Here is the meaning of the parameters.

| parameter | meaning |
| --- | --- |
| explorerNum | the number of explorers |

| parameter | meaning |
| --- | --- |
| traderNum | the number of traders |

Following are the fixed parameters for every simulation.

| parameter | value |
| --- | --- |
| environment size | (height, width) = (50, 100) |
| number of intervals to change traders | 100 |
| stop tick | 1000 |

## Structure of the source code files

The source code files are under the folder `src_code` . And its structure and descriptions are as below.

```
src_code
    ├─records           // simulation records
    ├─src               // source code
    │  └─marsExplorer
    └─MarsExplorer.rs      // styles and parameters
```

# PROBLEMS

Eclipse is hard to use. But there is no support for IDEA or vscode due to the integration of Eclipse according to the developers' response.