Intermediate Competition

by MiMaCh System – *Course of Intelligent Systems* a.a. 2021/2022

Contents

Mean-Max Game

- The Game
- The Environment
- The Goal
- The Players

Different Leagues

- Wood3
- Wood2
- Wood1
- Bronze

The Strategy

- The Initial Approach
- Bronze League Code

Conclusion

- The results
- Further Improvement

Mean-Max Game
Bot Programming

The Environment

- Circular map of radius **6000** with **3 players**.
- Players move **simultaneously**.
- **Water Town** is in (0,0) and has a radius of **3000**.
- Each player has 3 **Looters** (**Reaper**, **Destroyer** and **Doof**).
- **Tankers** go to and from Water Town.
- When a Tanker is destroyed by a Destroyer, a **Wreck** is created.

The Goal

- **Gather more water** than your enemies.
- **Victory Conditions**: have more water than your opponents at the end of the game.
- The game is over once any player reaches **50 water** or after **200 turns**.

The Players

Looters

- **Reaper**: takes water from a wreck.
- **Destroyer**: destroys tankers.
- **Doof**: generates rage.
- In Yellow controlled by our code.
- In Blue and Magenta our enemies.

Others

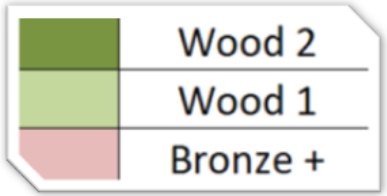
- **Tankers**: autonomous movement.
- **Wrecks**: a wreck appears after a tenker is destroyed.

Summary

		Name	Type	Throttle	Mass	Friction	
Looters = Player controlled		**Reaper**	0	0-300	0,5	0,2	ACC = Throttle / Mass
		**Destroyer**	1	0-300	1,5	0,3	
		**Doof**	2	0-300	1	0,25	
Autonomous		**Tanker**	3	500	2,5+W/2	0,4	
		**Wreck**	4				

Action	Range	Radius	Duration	Effect
Tar	0-2000	1000	3 turns	Mass + 10 Tankers are indestructibles
Grenades	0-2000	1000	Instant	Ejects all units around
Oil	0-2000	1000	3 turns	No friction Can't recolt water

Made with love by Orabîg



For more details, see the referee :
https://github.com/CodinGame/MeanMax/blob/master/Referee.java

Different Leagues

Wood3

- **3 Players (Reapers)** in the map
- **Wrecks** of water
- At each turn the player must either:
  - Go to a position (X, Y)
  - WAIT

Wood2

- **3 Players** and **3 Destroyers** in the map
- **Tankers** and **Wrecks**
- At each turn our player and our destroyer move or wait

Wood1

- **3 Players**, **3 Destroyers** and **3 Doofs**.
- Tankers and Wrecks.
- Now destroyers can use a **SKILL** on (X,Y)
- At each turn looters can move or wait

Bronze

- **All the looters** can use a SKILL
- At each turn looters can move, wait or use skill
- 2 **new skills**
  - Tar pool (Reaper skill)
  - Oil pool (Doof skill)

The Initial Approach

- We created a new class **Entity**
- We focused on the **goal**
- Our player has to reach more **wrecks** as possible

```
//new structures
List<Entity> wrecks=new ArrayList<Entity>();
List<Entity> enemies=new ArrayList<Entity>();
List<Entity> destroyers=new ArrayList<Entity>();
List<Entity> doofs=new ArrayList<Entity>();
Entity playerEntity=new Entity();
List<Entity> tanks=new ArrayList<Entity>();
Entity destroyerEntity=new Entity();
```

```
//adding a new Entity temp in the new structures
Entity temp=new Entity( unitId,
                        unitType,
                        player,
                        mass,
                        radius,
                        x,y,vx,vy,
                        extra,extra2);

//lista delle pozzanghere
if(unitType==4)
    wrecks.add(temp);
//lista dei tanks
else if(unitType==3)
    tanks.add(temp);
//lista delle auto nemiche
else if(unitType==0 && unitId!=0)
    enemies.add(temp);
//lista dei destroyer nemici
else if(unitType==1)
    destroyers.add(temp);
```

The Initial Approach - Wood3

- We calculated the **closer wreck** from our Reaper

```
//raggiungo la pozzanghera con distanza minima
double distance=12000;
Entity wreckMinDistance=new Entity();
for(Entity w:wrecks)
{
    if(distance > playerEntity.dis(w.x,w.y))
    {
        distance = playerEntity.dis(w.x,w.y);
        wreckMinDistance = w;
    }
}
```

The Initial Approach - Wood2

- We defined **Destroyer**'s behaviour
- We calculated the **closer Tanker** from our Destroyer

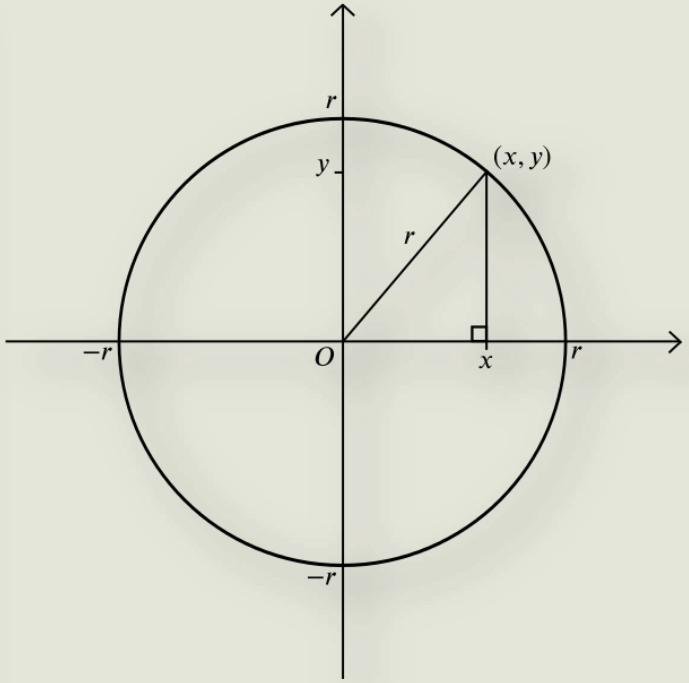
```
//raggiungo il tank con distanza minima
double newDistance=6000;
Entity tankMinDistance=new Entity();
for(Entity t:tanks)
{
    if(newDistance > destroyerEntity.dis(t.x,t.y)-t.radius)
    {
        newDistance = destroyerEntity.dis(t.x,t.y)-t.radius;
        tankMinDistance = t;
    }
}
```

The Initial Approach - Wood1

- We defined **Doof**'s behaviour
- We calculated a **circular motion** using trigonometry formulas

```
int R=(int) Math.hypot(doofEntity.x, doofEntity.y);
if(R<5500){
    int sinA=doofEntity.y/R;
    int R1= 6000-R;
    int angle=(int) Math.toDegrees(Math.asin(sinA));
    int x1=(int) ((Math.cos(Math.toRadians(angle)))*R1);
    int y1=(int) ((Math.sin(Math.toRadians(angle)))*R1);

    System.out.println((x1+doofEntity.x) + " " + (y1+doofEntity.y) + " 300");
}
else{
    int sinA=doofEntity.y/R;
    int angle=(int) Math.toDegrees(Math.asin(sinA))+30;
    int x1=(int) ((Math.cos(Math.toRadians(angle)))*R);
    int y1=(int) ((Math.sin(Math.toRadians(angle)))*R);
    System.out.println(x1+ " " + y1 + " 300");
}
```



Bronze League Code - I

- In **Bronze League** two new skills are added:
  - **Doof skill**: an oil pool that lasts 3 turns in the game and affected in a radius of 1000
  - **Reaper skill**: a tar pool that lasts 3 turns in the game and affected in a radius of 1000

Bronze League Code - II

- Now our player does not just reach the nearest wreck:
  - It first checks if there are more **overlapping wrecks**, giving them priority;
  - otherwise looks for the **closest** and **freest** one, that is the one with as few entities as possible nearby.
- Otherwise it will simply go to the **nearest one**.

```
//COMPORTAMENTO DEL PLAYER
if(!wrecks.isEmpty())
{
    if(!playerEntity.playerOnWreck(wrecks))
    {
        Entity wreckMinDistanceFromPlayer=playerEntity.minDistanceEntity(wrecks, 12000, -1);
        Entity wreckToChoose=playerEntity.rightWreck(wrecks,tanks,destroyers,doofs,unitCount);
        Entity wreck2 = wreckWithMoreWrecks(wrecks);

        if(wreck2!=null)
            System.out.println(wreck2.x+" "+wreck2.y+" 300");

        else if(wreckToChoose!=null)
            System.out.println(wreckToChoose.x+" "+wreckToChoose.y+" 300");

        else
            System.out.println(wreckMinDistanceFromPlayer.x+" "+wreckMinDistanceFromPlayer.y+" 300");
    }
    else
        System.out.println(playerEntity.x+" "+playerEntity.y+" 200");
}
else
    System.out.println(destroyerEntity.x+" "+destroyerEntity.y+" 300");
```

Bronze League Code - III

- Our **Destroyer** tries to reach the **nearest Tanker** to eliminate it.
- Also, when the rage is grater then a specific value, it uses its skill (**Nitro Grenade**) to push vehicles away.
- In case there are no Tankers it simply goes near the closest **enemy Doof**.

```
Entity doofMinDistanceFromDestroyer = destroyerEntity.minDistanceEntity(doofs, 12000, 0);
Entity tankMinDistanceFromDestroyer=destroyerEntity.minDistanceEntity(tanks, 12000, 0);
int newX = (enemies.get(1).x)+enemies.get(1).vx;
int newY = (enemies.get(1).y)+enemies.get(1).vy;

if(myRage>=160 && playerEntity.distance(newX, newY)>1000)
    System.out.println("SKILL "+newX+" "+newY);

else if(tankMinDistanceFromDestroyer!=null && tankMinDistanceFromDestroyer.distance(0,0)<5000)
    System.out.println(tankMinDistanceFromDestroyer.x+" "+tankMinDistanceFromDestroyer.y+" 300");

else
    System.out.println(doofMinDistanceFromDestroyer.x+" "+doofMinDistanceFromDestroyer.y+" 300");
```

Bronze League Code - IV

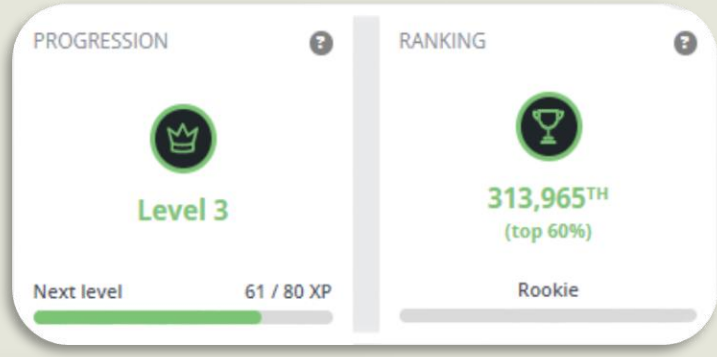
- In the previous league, our doof was just trying to move as much as possible in a circular way to accumulate rage.
- Now it uses its skill (the **Oil Pool**) if the rage is greater than a specific value and our player (**Reaper**) is far enough.
- Otherwise it will chase the opposing Reaper to annoy him so that it has difficulty reaching the wrecks.

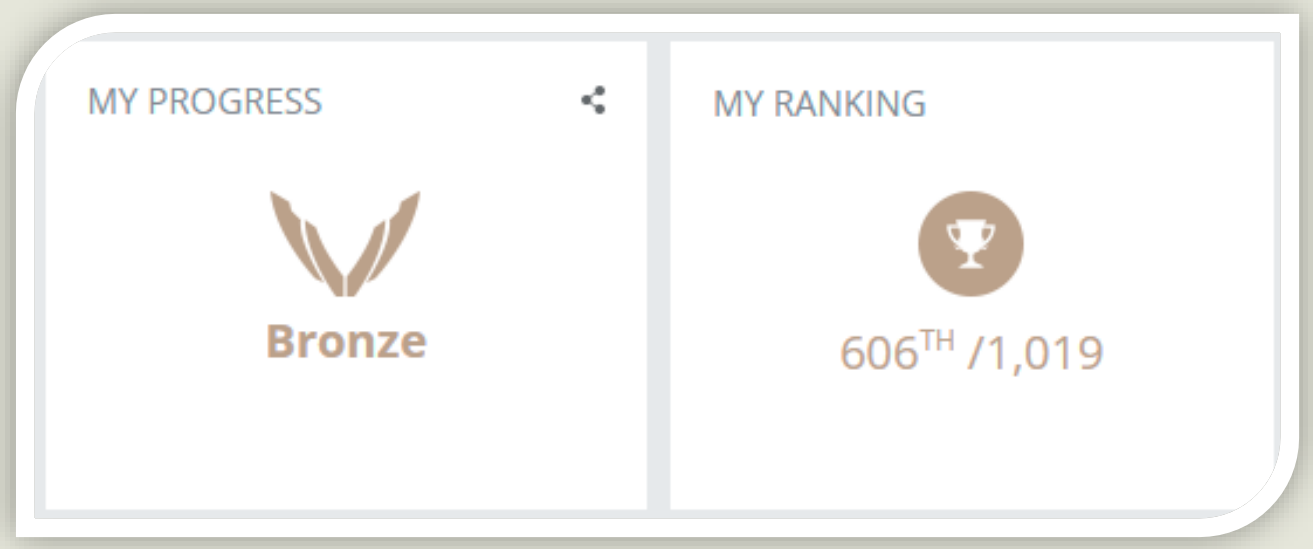
```
int X = (enemies.get(0).x)+enemies.get(0).vx;
int Y = (enemies.get(0).y)+enemies.get(0).vy;

if(myRage>=160 && playerEntity.distance(X,Y)>1000)
    System.out.println("SKILL "+X+" "+Y);

else
    System.out.println(enemies.get(0).x+" "+enemies.get(0).y+" 300");
```

Conclusion - Reached Result





Conclusion - Possible Improvements

Strengths

- Our Destroyer and our Doof do not constitute a barrier to our Reaper
- They target one of the enemies (also with their skills)

Weaknesses

- Managing the collision of our Reaper
- Target both the enemies at the same time

Resources

- Bot programming, Mean-max
- Codingame.com
- MiMaCh System – Profile
- Github - Mean-Max - Referee.java
- Github - MiMaCh System Proj - Intermediate Competition

Thanks for your attention

MiMaCh System
- Canonaco Martina [231874]
- Morello Michele [223953]
- Passarelli Chiara [223971]