LAB1: DOCUMENTATION

Maria Beili Mena Dorado (u199138)
Mireia Pou Oliveras (u198721)

1. Description of the problem

Aquesta primera pràctica consisteix en crear la simulació d'un món que conté diversos agents movent-se. Cada agent tindrà un objectiu que anirà canviant un cop aconsegueix arribar-hi. Després d'implementar els diferents mètodes i classes, s'hauria de poder visualitzar un rectangle (ambient de simulació) amb cercles movent-se (agents) en diferents direccions.

Per a poder arribar a aquest punt, cal crear dues classes:

1. Classe "Agent" → dona forma als agents i defineix el seu comportament.
2. Classe "World" → crea la forma de l'ambient de simulació que contindrà els agents.

Per a cada classe hem definit diferents mètodes:

- Per la classe Agent hem implementat *setTarget()* que determina la posició i la direcció del seu objectiu. *setSpeed()* per establir la velocitat del Agent. També *updatePosition()* que actualitza la seva posició. A més, *targetReached()* verifica si els Agents han arribat al seu objectiu i *isColliding()* que comprova que no xoqui amb cap altre agent. Per últim, *paint()* pinta el cercle del Agent.

```
public void setTarget(Vec2D tar) {
    target = tar;
    dir = new Vec2D(target); // estem creant una instància
    dir.subtract(pos);
    dir.normalize();
}

public void setSpeed(double velocity) {
    speed = velocity;
}
```

```
public void updatePosition() {
    Vec2D multiplication = new Vec2D(dir.getX()*speed, dir.getY()*speed);
    pos = new Vec2D(pos);       //creem instància
    pos.add(multiplication);
}

public boolean targetReached(){
    target = new Vec2D(target);
    target.subtract(pos);
    if (target.getX()<radius | target.getY()<radius ) {
        return true;
    } else {
        return false;
    }
}
```

```
public void paint(Graphics g) {
    int x = (int) (pos.getX() - radius);
    int y = ( int ) (pos.getY() - radius) ;
    int d = ( int ) (2 * radius) ;
    g.setColor(Color.RED);
    g.fillOval(x,y,d,d);
}
```

- La classe World hem implementat *randomPos() genera una posició aleatòria en el World i randomRadius()* similar al mètode anterior, però dona un radi arbitrari. També *simulationStep()* si un Agent arriba al seu objectiu, assignarà un altre o actualitza la seva posició.

```
public Vec2D randomPos() {
    double x = marge + Math.random() * (width - 2* marge);
    double y = marge + Math.random() * (height - 2 * marge);
    return new Vec2D(x,y);
}

public double randomRadius() {
    return 5 + Math.random() * (marge - 5);
}
```

```
public void simulationStep(){
    for (int i = 0; i<agents.length; i++){
        if(agents[i].targetReached() == true){
            agents[i].setTarget(randomPos());
        }else{
            agents[i].updatePosition();
        }
    }
}

public void paint(Graphics g){
    for (int i = 0; i < agents.length; i++){
        agents[i].paint(g);
    }
}
```

2. Alternative solution discussed

- UpdatePosition in agent

Hem tingut molts problemes per a crear el mètode “UpdatePosition” a agent. Aquest consisteix en actualitzar la posició de l’agent fent una sèrie de operacions (multiplicar la direcció per la velocitat i sumar-ho a la posició).

A l’hora de fer-ho ens vam trobar amb el problema que la direcció era un vector 2D i la velocitat un nombre, de manera que java no ens peremtia fer una multiplicació entre ells. Primer vam intentar convertir la velocitat en un vector 2D, però no hi havia la forma de fer-ho. Després vam pensar en canviar l’ordre de l’operació i fer primer la suma i després la multiplicació, però ens trobavem amb el mateix problema i, a més, el resultat no hagués sigut el que voliem.

Finalement, després de rumiar-ho molt, ens vam adonar que podiem dividir el vector 2D en dos nombres enters agafant la seva posició x i la seva posició y. Així doncs, vam decidir crear un nou vector i multiplicant la velocitat pels dos nombres. D’aquesta manera es podia fer la multiplicació i la suma sense problema.

```
public void updatePosition() {
    Vec2D multiplication = new Vec2D(dir.getX()*speed, dir.getY()*speed);
    pos = new Vec2D(pos);       //creem instància
    pos.add(multiplication);
}
```

- World constructor

A l’hora d’inicialitzar els 10 Agents per determinar la posició aleatòria del seu objectiu ens hem trobat amb el dubte de crear una nova funció randomTarget() per a fer-ho. Finalment, vam arribar a la conclusió que randomPos() faria la mateixa funció en el moment de determinar un objectiu de l’ Agent (setTarget()).

```
for(int i = 0; i < agents.length; i++){
    agents[i] = new Agent(randomPos(), randomRadius());
    agents[i].setTarget(randomPos());
    agents[i].setSpeed(velocity: 1);

}
```

- manageCollisions in World

A l’hora de crear la funció addicional ho hem pensat fer de vàries formes. La primera opció que se’ns ha ocorregut fer ha sigut crear una funció change_direction() per a agent i així poder canviar la direcció dels agents quan col·lisionen. Això, però, ho hem descartat perquè tenim funcions que fan coses similars. En segon lloc, també hem pensat fer servir el randomPos() del world, però com és una funció de world no es pot utilitzar a cada agent particularment. Finalment, hem decidit utilitzar la funció de changePosition() i ha quedat així:

```
public void manageCollisions() {
    for (int i=0 ; i<10; i++ ){
        for (int j=i+1; j<10; j++){
            if ((agents[i]!=agents[j]) & (agents[i].isColliding(agents[j]) == true)) {
                agents[i].updatePosition();
                agents[j].updatePosition();
            }
        }
    }
}
```

3. Conclusion

A conclusion that describes how well the solution worked in practice, i.e. did the tests show that the classes were correctly implemented? You can also mention any difficulties during the implementation as well as any doubts you might have had. The source code and documentation should all be uploaded to a directory Lab1 of your Git repository prior to the next lab session.

En resum i mirant-la globalment, creiem que la pràctica ha estat ben implementada. Els mètodes i els atributs han estat fets tal com constava a l'enunciat i el test ha tingut resultats positius.

És cert i en som conscients, però, que no hem tingut en compte cap mena d'error a l'hora de l'input. És a dir, si una funció necessita un "int" i li passem un "vec 2D" la pràctica donarà error i ja està, no hi haurà cap opció per canviar-ho ni al codi s'ha implementat res per evitar-ho.

Finalment, volíem explicar que la funció manageCollisions() està comentada a l'apartat de World. No l'hem posada sense comentar perquè no estàvem segures si s'havia de cridar des del "WorldGUI" o s'havia de fer alguna altra cosa. Tot i això, l'hem intentat fer tal com posava l'enunciat.