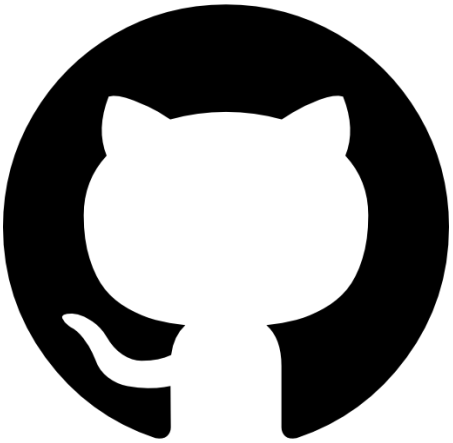
DOCUMENTACIÓ ROBOCODE EX2
FOLLOW THE LEADER TEAM

Denis Vera Iriyari



GitHub Projecte

Plantejament

Havent fet l'exercici del Timidín Robot ja sabía com funcionaba tot i em va ser més fàcil posar-me directament a treballar a l'exercici. És clar la diferència ara és que tenim un equip de robots amb diferents rols, en el meu cas he decidit plantejar-ho amb els 5 robots ordenats jeràrquicament (ara veurem com exactament) tenint els rols {1,2,3,4,5} dels quals cadascú segueix al seu antecessor on el rol 1 és el líder i no te predecessor. Aquests rols s'actualitzen cada 15 segons quan es canvia de sentit invertint l'ordre o quan mor un dels robots aliats per tal de reordenar la llista i que no quedi incongruent.

El programa comença amb tan sols un dels robots escollint aleatòriament un aliat com a líder (si tots haguessin d'escollir aleatòriament a un aliat sería raro), després d'això, segons la distància dels companys respecte del líder aleatòriament escollit s'estableixen els rols d'aquests on el més proper té el rol 2 i el més llunyà té el rol 5. Un cop establerta la jerarquia, el líder va directament a la seva cantonada més propera i comença a moure's en rectangles en sentit horari (canviant aquest sentit cada 15 segons) on el 2 robot li segueix, el 3 segueix al 2, etc. Cada cop que el líder arriba a una de les 4 cantonades, escaneja cercant un enemic, disparant i fent als seus aliats disparar-li també. La primera escannejada agafa el primer robot enemic que troba i tot l'equip el posa com a objectiu comú. Després d'aquest escaneig, cada escaneig donarà les coordenades tan sols d'aquest enemic objectiu, no canviarem d'objectiu fins que aquest mori. Els robots deixaran una distància de 200 per prevenir alguns bugs de moviment. El líder, quan es topa amb un robot, es mourà pel voltant i anirà a la següent cantonada.

Variables

-private enum FaseRobot {HANDSHAKE, CONGA};
-private FaseRobot fase = FaseRobot.HANDSHAKE;

Variables que defineixen la fase actual en què es troba el robot dins la coordinació de l'equip. Faciliten la comprensió de l'estat del robot i el procés de col·laboració entre els membres de l'equip.

-protected String teamLeader;

Variable on es guarda el nom del líder de l'equip. Permet a cada robot saber qui és el líder per seguir les seves ordres o accions.

-protected String predecessor;

Variable que conté el nom del robot que aquest segueix. És `null` si el robot és el líder, ja que no segueix a ningú.

-protected double enemyX;

Variable que guarda la coordenada X actual de l'enemic detectat. Es fa servir per calcular la posició i prendre decisions de combat.

-protected double enemyY;

Variable que guarda la coordenada Y actual de l'enemic detectat, complementant `enemyX` per obtenir la posició exacta.

-protected String enemyTarget;

Variable on es guarda l'objectiu actual (nom del robot enemic) que els seguidors de l'equip dispararan. Centralitza l'objectiu comú per a tots els robots.

-protected int role = 0;

Variable que guarda el rol del robot dins l'equip. Permet definir la jerarquia i els comportaments específics del robot segons la seva funció.

-protected boolean clockwise = true;

Variable booleana que indica la direcció de moviment del robot. Si és `true`, el robot es mou en sentit horari; si és `false`, en sentit antihorari.

-private MyRobotStatus robotStatusGetter;

Variable que guarda l'estat d'un robot aliat. S'utilitza per gestionar la recepció d'estats dels robots i les dades de situació.

-private List<Map.Entry<Double, String>> sortedTeammatesAlive;

Llista que guarda els membres de l'equip vius ordenats per la distància inicial respecte al líder. Facilita la gestió de la jerarquia entre els robots.

-private boolean leaderChosen = false;

Variable booleana que indica si el líder ha estat triat. Evita confusions a l'hora de determinar el cap de l'equip en qualsevol moment.

-private int teammateCounter = 0;

Variable que compta el nombre de membres de l'equip, utilitzada durant l'establiment de la jerarquia.

-private boolean hierarchyEstablished = false;

Variable booleana que indica si la jerarquia entre els membres de l'equip ha estat establerta. Això facilita que el robot sàpiga si ja pot actuar segons el seu rol.

-private List<Double> distances = new ArrayList<>();

Llista que guarda les distàncies inicials entre els seguidors i el líder. S'utilitza per establir l'ordre dels robots dins l'equip.

-private long lastRoleChangeTime = 0;

Variable que guarda el moment en què es va produir el darrer canvi de rols dins l'equip. S'utilitza per gestionar els intervals de canvi de rol.

-private static final int ROLE_CHANGE_INTERVAL = 450;

Variable constant que defineix l'interval entre canvis de rols dins l'equip en ticks, per controlar la freqüència amb què es modifiquen els rols.

-private int deadAllies = 0;

Variable que guarda el nombre de robots aliats que han mort durant la batalla. Facilita l'estratègia i la gestió de recursos dins l'equip.

Funcions

-public void run()

Selecciona els colors del robot, començem a la fase 'HANDHSAKE' on es seleccionarà aleatòriament el líder de l'equip, un cop escollit aquest, si el robot implícit és el líder (role = 1) recopilarà les distàncies de la resta de robots per ordenar-los jeràrquicament en base a la distància a la que estàn. Un cop ordenats, pasem a la fase 'CONGA' no sense fer que el líder es mogui a la cantonada més propera. En aquesta fase simplement es gestiona el moviment tant del líder com dels seguidors.

- private void chooseRandomLeader()

Només un dels robots executa aquesta funció on seleccionarà aleatòriament un dels robots de l'equip com a líder i mandarà aquesta informació a la resta de robots.

-private void goToClosestCorner()

Mou al líder a la cantonada més propera.

-public void runTeamLeader()

Bàsicament truca a moveInRectangle().

-public void runFollower()

Si tenim un enemic objectiu apuntem i disparem. Constantment agafa les coordenades del seu robot predecessor i li segueix. També chequea si han passat 15 segons des de l'últim canvi de sentit i si han passat canvia de sentit altre cop.

-private void changeRoles()

Canvia el sentit del moviment (clockwise = !clockwise), actualitzem els rols de cada robot i invertim la llista dels aliats ordenats jeràrquicament actualitzant també els predecessors de cada robot, el nou líder no tindrà antecessor.

-private void checkAndChangeRoles()

Chequea si han passat 15 segons des de l'últim canvi de sentit.

-private void moveInRectangle()

Gestiona el moviment del líder, on mentre el seu rol no sigui actualitzat a qualsevol que no sigui 1 es mou en rectangles per l'exterior de l'arena (amb un marge de 10%) tenint en compte la variable 'boolean clockwise' per al sentit de la marxa. Cada cop que es mou a una cantonada escaneja les coordenades de l'enemic objectiu (o busca un nou enemic si no en té cap) i dispara també. Chequejant constantment si toca canvi de sentit o no.

-private int getClosestCorner(double[][] corners)

Retorna l'índex de la cantonada més propera.

-private void scanForEnemies()

Escaneja una volta sencera.

-public void onScannedRobot(ScannedRobotEvent event)

Si el líder escaneja un robot i aquest és enemic, si és el nostre objectiu (o si no tenim objectiu i marquem aquest com a objectiu) manda les seves coordenades a la resta de l'equip.

-public void onHitRobot(HitRobotEvent event)

Si el líder choca amb un robot, es mou al voltant i va a la següent cantonada. Si el robot amb el que xoca és un rival, li apunta i li dispara.

-public void moveAroundEnemy(double distance)

Mètode per moure's al voltant de l'enemic amb el que xoca i evitar-ho.

-public void goTo(double x, double y)

Mètode per moure's a les coordenades indicades.

-public void onRobotDeath(RobotDeathEvent event)

Gestiona la mort de qualsevol robot. Si mor un aliat, s'actualitzen els rols, els predecessors i la llista que ordena l'equip. Si mor un enemic, deixem de marcar-ho com a objectiu per tal de buscar el següent.

-public double normalizeBearing(double angle)

Funció per assistir la normalització de l'angle, és a dir per buscar la menor distància a la qual hem de girar per posicionar-nos en la direcció 'angle'.

-public void aimAndShoot(double enemyX, double enemyY)

Apunta a les coordenades indicades i dispara amb una potència calculada segons la distància utilitzant la fórmula de la distància de Euclides.

-public void assignRolesBasedOnDistance(List<Double> distances)

El robot assigna els rols basats en la distància i s'envia la llista ordenada a la resta de robots aliats.

-private MyRobotStatus getRobotStatus(String teammateName)

Manda un missatge al robot 'teammateName' per obtenir el seu 'status', bàsicament les seves coordenades.

-public void onMessageReceived(MessageEvent event)

Gestiona tots els missatges possibles a rebre per qualsevol robot.

-public void onPaint(Graphics2D g)

Mostra radi del líder a la interfície del RoboCode.

-public class MyRobotStatus implements Serializable

Classe per guardar l'estatus dels robots.

Observacions

Tot i que el plantejament del projecte i la majoría de codi ho he pensat, processat i implementat jo, algunes funcions les he fet amb ajuda de chatGPT (no las he simplement copiat, si no que les he redissenyat per implementar–les al meu projecte ja que directament les funcions generades per IA no solen estar 100% correctament lligades al teu programa, però ha sigut de gran ajuda per facilitar funcions secundàries que simplement assisteixen). Aquestes funcions estan indicades amb comentaris al codi.

Com pots veure són funcions de càlculs ràpids o simplement d'assistència a les altres funcions principals que són les que realment tenen pes.

Al principi quan hi han molts robots i la partida és molt caòtica és bastant possible que hi hagin bugs de moviment / col·lisió. Però en partides amb menys robots o més al final de la partida va bastant més net i fluït.

Als comentaris del codi de la classe està tot bastant bé explicat.

Aquesta és una classe potent per a la lluita contra altres equips degut a que tots els robots estàn en constant moviment (el qual fa que siguin objectius més difícils de disparar) i tots els membres de l'equip disparen a un enemic comú i fins que aquest no cau no canvien d'objectiu, això fa que sigui més senzill assolir un avantatge

numèric. I aquest avantatge ens aproparà més a la victoria. Tot i això, al meu cas personal, que estiguin tants robots alhora combatint genera molts bugs de moviment principalment. Comparat a quan hi han 2 o 3 robots per equip que difícilment surten aquests tipos de bugs dels quals no he pogut desfer-me al 100%.