Documentació Robocode

Ixent Cornella, Arnau Roca

Exercici 1:

Primerament en el desenvolupament del corner team, vam estar invertint molt temps en una estratègia de enviar els robots al cantó molt eficient. El primer que se'ns va ocórrer (al nostre cap semblava meravellós) era calcular la distància de tots els robots amb el centre i enviar primer el robot que estigués més lluny del centre al cantó més proper. Vam estar treballant molt en desenvolupar aquesta tècnica (ja que ens haviem d'habituar a l'entorn de treball del netbeans i la llibreria del robocode), però finalment vam desestimar-la. El motiu pel qual la vam desestimar és perquè podem trobar casos en el que múltiples robots estiguessin a prop del mateix cantó, i llavors era difícil de gestionar i perdíem l'eficiència que buscavem.

Llavors vam tornar a plantejar el millor mètode per enviar als robots al seu cantó, aprofitant algunes funcions que ja teníem fetes anteriorment (com podria ser el calcular distàncies o calcular angle). Aquest nou mètode consistia en seguint aquest esquema, cantó 0 baix esquerra, cantó 1 baix dreta, cantó 2 dalt esquerra, cantó 3 dalt dreta, aniriem fent un recorregut per cada cantó amb aquest ordre agafant el robot que ens quedés més prop de cada un. Llavors ens quedaria un robot sense assignar, que seria el kamikaze.

A nivell de codi això canvia una mica, per començar si seguim el codi una mica veiem que el primer que es fa és la funció inci. Primer incialitzem els robots indicant que no tenen cap cantó. Llavors mirem la distància a la que està cada robot de cada corner (quan diem corner ens referim a cantó). Això ho fa mitjançant una funció anomenada distanceFromCorner, que es crida per cada corner, i que com ja hem dit fa aquest càlcul. Utilitzem la funció calcDist i llavors fem un broadcast enviant la distància i el corner al que correspon. Al rebre el missatge guardarem totes les distàncies corresponents als corners en 4 HashMaps (un per cada corner).

Seguint amb l'estratègia, el que fem és tenir una variable booleana que ens indica si el robot ja està assignat a un corner. Si no està assignat cridem a la funció findCornerRobot(). Aquí es on utilitzarem tot el calculat d' abans, ja que amb els HashMaps anomenats anteriorment podem saber per a cada corner, quin és el robot que hi està més a prop. Per tant, amb un algoritme bàsic de buscar la mínima distancia, acabem trobant els 4 robots que estan més aprop de cada corner, i al final enviem al robot que ha d'anar al corner corresponent la instrucció per a que hi vagi. Tots aquests missatges tenen un codi per tal que a l'hora de rebre'ls sigui més fàcil identificar la funció (el primer és codi 0, i aquest (goToCorner) es codi 1).

Quan el robot rebi el missatge amb el corner on ha d'anar, entrarà a la funció goToCorner. Aquí teniem també dues opcions per tal de fer-ho. Ens hem volgut cenyir al màxim al que diu l'enunciat i és per això que hem decidit que el robot busqui el angle de gir fins al corner i que vagi directe (en línea recta), a diferència de l'altra opció (més senzilla però més lenta) que és anar al costat i llavors pujar, dibuixant una "L". Per tal que el robot quedés quadrat al lloc, hem fet que just abans d'arribar vaigui fins al costat i llavors faci la L, d'aquesta manera ens queda quadrat al lloc perfecte SENSE XOCAR (ho posem amb majúscules, ja que és molt fàcil que al fer això el robot xoqui).

Llavors, un cop estem al corner marquem una altra variable global que indica que el robot ja està al lloc correctament (això és per quan xoquen amb algun altre robot amb la seva trajectòria, d'aquesta manera sabem que ja està al lloc, si no ho està el tornem a enviar fins que ho estigui).

Per acabar amb l'estratègia dels robots que estan als cantons, el que fem és un moviment sentinella, com ens diu l'enunciat, que fa que vagi endavant i enrere, per tal de no ser un blanc tan fàcil i disparen al primer robot que escanegen (tenen el radar activat i girant tota l'estona). La estratègia del disparar es basa en la que usen molts robots de la col·lecció "sample", concretament MyFirstDroid, que consisteix en calcular la trajectoria del robot que hem detectat i amb un càlcul calculem la posició que té el robot i la que tindrà segons la seva direcció. Després calculem l'angle que hem de posar l'arma, girem la arma, ens parem per poder apuntar millor, i llavors disparem.

Un cop acabada l'estratègia dels robots dels cantons, s'ha d'explicar l'estratègia del kamikaze. Tornem al moment d'assignar el cantons de cada robot, com ja hem dit hi haurà un que no tindrà cantó. Aquest és el líder i kamikaze. Ràpidament el que fa

quan se li assigna aquest rol, es començar a girar com un boig el radar fins trobar un robot enemic. Quan troba a algú, l'estratègia que hem seguit és com abans anar preveïnt la seva trajectòria i anar a xocar amb ell disparant-lo fins que mori. Això també ho fem com abans amb el getBearing i calculant la posició i avançar fins a xocar amb ell, i seguir disparant.
Per implementar la comunicació, mitjançant les crides als mètodes broadcast i sendMessage, hem necessitat implementat la classe missatge. Aquesta té tres variables privades que son les que vam considerar que ens podrien ser més útils, una per identificar el tipus de missatge i dues més per posar dos doubles (pensat per posar la posició de un robot). I la classe té els constructors necessaris per tal de fer els missatges amb el següent format (codi, posicióX, posicióY).

VARIABLES I NOMENCLATURA:

- **4 HashMaps** usats per assignar, per cada membre del team, la seva posició respecte cada corner. Cada robot té la mateixa variable.
- **LIDER**: variable booleana que indica si el robot és el Kamkiaze/Lider, només podrà ser true quan el robot no té corner assignat i per tant, li toca ser kamikaze.
- **CORNER**: variable booleana que indica si el robot és un CornerRobot, és a dir, està assignat a un corner.
- **robotTeCorner**: HashMap que conté, per cada robot del team, si ja té un corner assignat (per evitar que un mateix robot se li assigni dos corners).
- **myCorner**: enter que ens indica el corner del robot, és -1 si no té corner.
- **direccioGir**: variable entera que ens indica la direcció de gir del kamikaze, per poder facilitar calculs i simplificar codi.
- **onCorner**: variable booleana que ens permet saber si el robot ja està al corner que li correspon, només serà true si el mètode goToCorner finalitza amb el robot estant a menys de 30 unitats de distancia del corner.

CODIS:

- Per a cada missatge, hi ha un codi per a poder-lo identificar:

- 0: Ens dona la distància a un cert corner especificat.
- 1: Ens permet saber que el robot està llest per anar a un corner, i l'enviem allà.

- Cada cantonada és representada amb un número: 0 = abaix esquerra, 1 = abaix dreta, 2 = adalt esquerra, 3 = adalt dreta. L'ordre de prioritat a l'hora de repartir els robots per cada corner és 0,1,2,3

.

MÈTODES PRÒPIS i MÈTODES MODIFICATS:

- **onMessageRecieved(Missatge)**: Aquest funció l'hem implementat per tal de poder gestionar els missatges que rebem, té dos possibles codis un per rebre la distància d'un robot a un corner X i un per comunicar que ha d'anar al corner.
- **distanceFromCorner(corner)**: Per al corner "corner", canviem el HashMap corresponent on es guarda la distancia de cada robot al corner.
- **onScannedRobot(Robot)**: Quan escanegem a un robot es cridarà aquesta funció, depenent de si és el lider (kamikaze) o si es un robot de cantó (els altres) farem una cosa o una altra.
- **onHitRobot()**: En cas que un robot es xoqui amb es cridarà automàticament aquesta funció, tenim dues opcions, si es el lider seguirà xocant i si no ho és intentarà anar fins al seu lloc.
- **findCornerRobot(corner):** Per al corner "corner", trobem el robot que hi està més aprop gràcies als hashmaps que guarden les distàncies de cada robot al corner (càlcul previ obligatori).
- **goToCorner (corner)**: Per al corner "corner", el robot segueix els següents passos:
  - 1. Obté l'angle al que s'ha de dirigir per arribar al corner corresponent i va cap allà.
  - 2. 50 unitats de camp abans d'arribar al corner, es para i es dirigeix en direcció vertical / horitzontal, depenent del corner.
  - 3. Avança de forma horitzontal / vertical les unitats que li quedin per arribar al cantó - 20, d'aquesta manera no toca el corner i evitem mal.
  - 4. Gira i avança la distancia que queda per arribar al corner en l'altre sentit (de vertical a horitzontal o de horitzontal a vertical).
  - 5. Si al acabar la funcio està a menys de 30 unitats de camp del corner, considerem que ho està i fiquem la variable "onCorner" a true.

  Aquest moviment vertical / horitzontal quan ja s'ha anat en direcció al corner es fa per evitar que el robot xoqui amb les parets.

- **calcDist(xOrigen, yOrigen, xDesti, yDesti)**: Càlcula la distància entre dos punts a partir de les coordenades d'aquets.
- **getAngleTo(xOrigen, yOrigen, xDesti, yDesti**): Càlcula l'angle del punt Origen al punt Desti.
- **inici():** tria el color pertinent per al robot, i crida altres mètodes com ara distanceFromCorner.