Documentació PROP:
Activitat 1
Robocode

Estratègia

La primera estrategia que vam implementar va ser la que utilitzava la primera versió del robot que hem programat. El nostre tank buscava a un enemic amb el radar i al trobar-lo el perseguía i disparava un cop estava a 200 o menys píxels de distància.

Aquesta estratègia ens servia en un inici ja que guanyava al robot Dummy i al Torre el 100% de les batalles, però al ser penjat el robot sorpresa a atenea ens vam adonar que havien de canviar la nostra estratègia per guanyar-lo.

El robot sorpresa es mou de manera paral·lela i rodejant al nostre tank de manera que s'esquiva les bales mentres es va apropant. Cal destacar que va canviant de sentit i que només dispara a certa distància, si esta molt lluny només es mou i segueix apropant-se.
El robot sorpresa també fa una predicció de les posicions futures del nostre robot. Per tant si el nostre robot es mou en una direcció el robot sorpresa és capaç de preveure el nostre moviment i dispara on fa aquesta predicció.

Al entendre el comportament del robot sorpresa vam començar a formar la nostra nova estratègia. Aquesta consisteix en moure el nostre robot d'una manera semblant a la del robot sorpresa, sempre rodejant al enemic (de manera perpendicular a la direcció on apunta el nostre radar) i fent que es pari i vagi fent canvis de sentit.
De manera que aconseguim esquivar les seves bales ja que al parar i canviar de sentit constantment totes les bales que ens dispara utilitzant la predicció del nostre moviment fallen.

El següent problema que vam tenir que solucionar va ser la implementació d'un algorisme per preveure les posicions futures del robot sorpresa ja que al estar movent-se continuament si disparem a la seva posició actual clarament fallarem perquè la bala triga cert temps en arribar.
Per tant vam implementar un sistema de predicció de moviment.

També volíem que el nostre robot tingués més d'un tipus d'estratègia o comportament, de manera que vam fer que si fallava set bales seguides canviés de tàctica.
Aquesta segona tàctica és la que vam implementar inicialment i em explicat a l'inici d'aquest document d'estratègia, seguir i disparar.

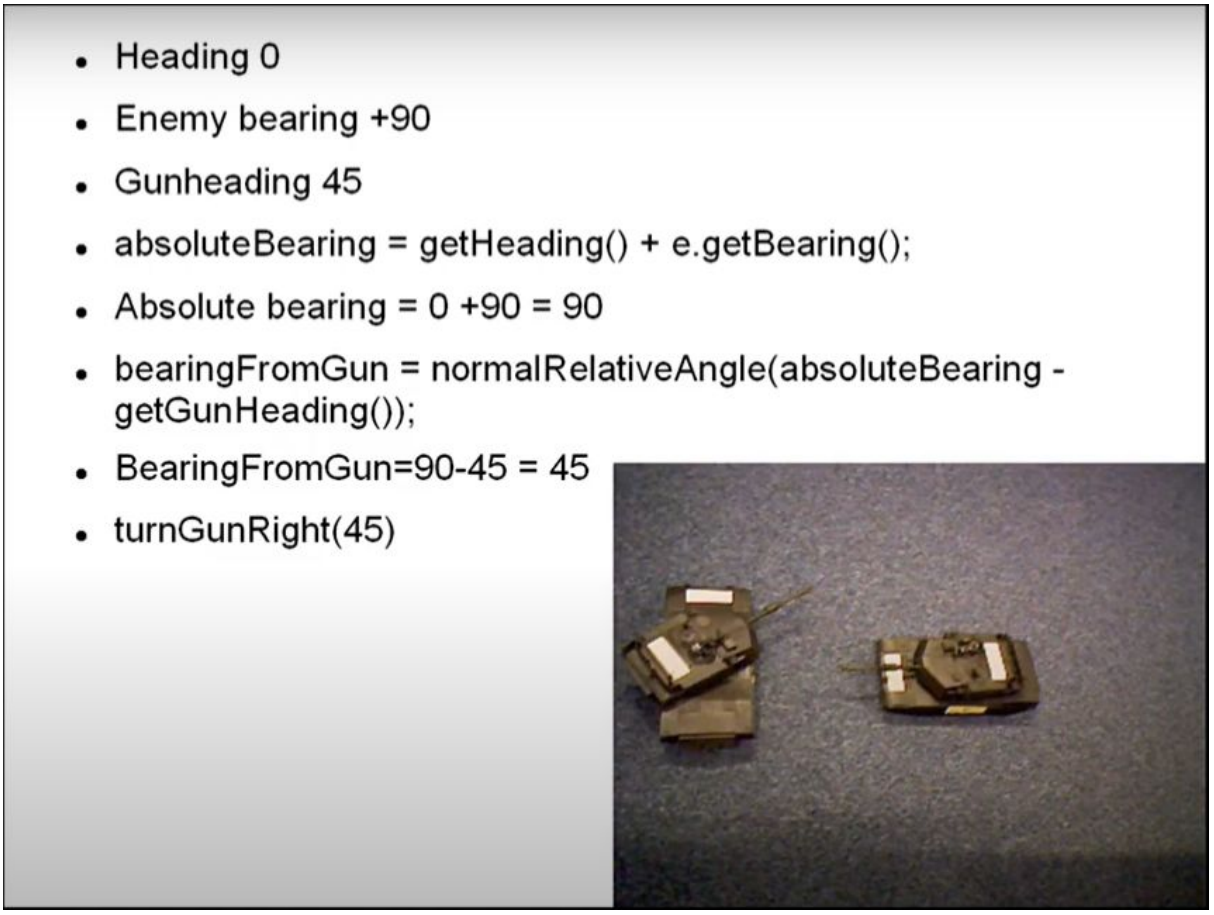
Finalment hem fet que si el robot enemic està suficientment a prop disparem amb més potència.

Detalls de la implementació

Per calcular l'angle que hem de girar l'arma per apuntar al robot enemic fem:

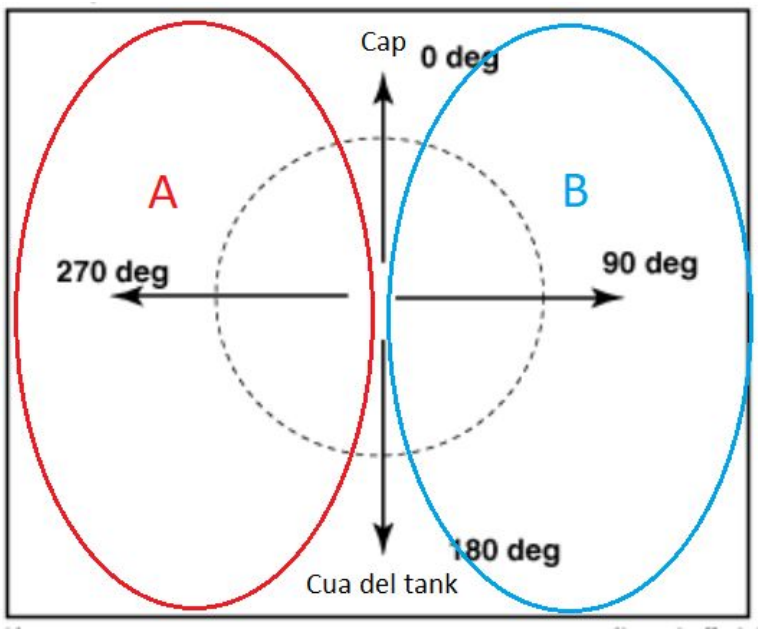
```
angleGirArma = normalRelativeAngleDegrees((getHeading() + e.getBearing())- getGunHeading());
```

La següent imatge mostra la resolució d'aquests càlculs en un cas concret. Les dades proporcionades són del tank de l'esquerra i el tank enemic és el de la dreta.



El càlcul de l'angle de gir del radar és semblant però es fa servir el **getRadarHeading()**.

Per mantenir el nostre robot sempre perpendicular al radar i, per tant, moure'ns de manera perpendicular a les bales que ens dispara el robot enemic, se'ns presenten dos casos que hem de tenir en compte i que queden reflectits en la següent imatge:



Si ens imaginem que el quadrat és el tank en sí i els 0 graus són el seu cap i els 180 la seva cua, en podem trobar en dues situacions diferents.

La primera situació és que el tank enemic es troba a la nostre esquerra o zona A (en vermell) i en aquest cas el que hem de fer per obtenir l'angle de gir necessari per mantenir-nos perpendiculars a ell és:

```
if(e.getBearing()<0){   //El tank enemic esta a l'esquerra
    angleGirTank=e.getBearing()+90;
}
```

Per exemple, si l'enemic està als 315 graus el bearing serà -45 graus.
Si fem -45+90=45 i per tant podem veure que l'angle de gir obtingut és correcte.

La segona situació que ens podem trobar és que el tank enemic es troba a la nostra dreta o zona B (en blau). No podem aplicar el càlcul de la zona A a la B perquè les situacions són diferents i en el cas de trobar-se en la zona B en comptes de sumar els 90 graus els hem de restar. Per tant ens queda el següent:

```
else{
    angleGirTank=e.getBearing()-90;
}
```

Per canviar d'estratègia utilitzem les bales fallades com a referència
Creem una variable que anomenem ***fallDispar*** i la inicialitzem a zero.
Utilitzant la funció onBulletMissed() cada cop que s'activa l'event de fallar una bala incrementem el comptador en 1, com es pot apreciar en la imatge següent:

```
public void onBulletMissed(BulletMissedEvent event){
    fallDispar++;
}
```

Al voler que el robot canviï d'estratègia un cap ha fallat set bales **seguides**, hem de reiniciar el comptador cada cop que encertem una bala i això o aconseguim amb l'event i el codi següent:

```
public void onBulletHit(BulletHitEvent event){
    if(fallDispar<7) fallDispar=0;
}
```

Al estar a prop del robot (hem definit a prop com a menys de 200 pixels de distància) enemic en comptes de disparar amb potencia 1 disparem amb la potència màxima.

```
if (getGunHeat() == 0) {
    if(e.getDistance()<200) fire(Rules.MAX_BULLET_POWER);
    else fire(1);
}
```

Per calcular la predicció dels dispars afegim un valor constant al càlcul de l'angle de gir de l'arma que varia segons la distància a la que ens trobem del robot enemic.
Aquest valor també és positiu o negatiu segons cap a on està girant el radar.
Així ens assegurem de disparar una mica més enllà de la direcció en la que està anant el tank enemic.

```
if(getRadarTurnRemaining()>0){
    if(e.getDistance()<200) angleGirArma = normalRelativeAngleDegrees(((getHeading() + e.getBearing())- getGunHeading())+20);
    else if(e.getDistance()<400) angleGirArma = normalRelativeAngleDegrees(((getHeading() + e.getBearing())- getGunHeading())+25);
    else angleGirArma = normalRelativeAngleDegrees(((getHeading() + e.getBearing())- getGunHeading())+35);
}
```

Si el radar no està realitzant un gir (l'enemic està quiet), llavors es fa el càlcul normal sense sumar ni restar cap valor:

```
else {
    angleGirArma = normalRelativeAngleDegrees((getHeading() + e.getBearing())- getGunHeading());
}
```

Per poder avançar i retrocedir constantment utilitzem un contador que anomenem cont, inicialitzat a 0. Aquest contador s'incrementa a cada iteració de l'event **scan** i l'utilitzem per establir un període de temps.

Durant aquest període de temps (20 iteracions per ser exactes) el robot avançarà o retocedirà segons el valor de la variable distahead, que inicialitzem a 100 .

Cada cop que es superin les 20 iteracions multipliquem distahead per -1 canviant-li així el signe i per tant el sentit en que avança o retrocedeix el tank.

```
if(cont>=20){
    setAhead(distahead);
    distahead *= -1;
    cont=0;
}else{
    cont++;
}
```