Pràctica d'Agents

Pedro Martí Picó

2021-2022

Índex

1. Comportament del bitxo **2**

1.1. Estratègia ofensiva 2

1.2. Estratègia defensiva 2

2. Explicació del codi **3**

2.1. Mètode 'onGirar' 3

2.2. Mètode 'detectaParet' 3

2.3. Mètode 'esquivar' 4

2.4. Mètode 'filtrarObjecte' 4

2.5. Mètode 'avaluaComportament' 6

1. Comportament del bitxo

El nostre bitxo té de nom Eren Orr, nom que fa referència al protagonista de Shingeki no Kyojin (Ataque a los titanes) i el cognom fa referència a un company que ens cedeix els drets d'imatge que utilitza el nostre bitxo.

Aquest bitxo utilitza una estratègia que es basa en consumir pocs recursos de moviment, utilitzant una velocitat moderada, i amb bastant de tirs. El comportament del nostre bitxo es divideix en dues branques: Comportament ofensiu i defensiu. Així, el que feim, es marcar una estratègia ofensiva i una defensiva:

1.1. Estratègia ofensiva

El que fa es anar recorreguent el mapa alimentant-se de recursos i agafant escuts, mentres, si veu un recurs enemic (mina) el dispara per eliminar-lo i que aquest enemic no pugui alimentar-se de recursos. A part, evita les parets triant el gir més eficient en quant a obstacles, dispara als enemics que veu.

1.2. Estratègia defensiva

La nostra estratègia es basa en tenir un mètode .^esquiva"que si el sector 2 i 3 detecten un tir, l'esquiva girant al sentit contrari del sector d'on ho percep. Si el tir el percep molt a prop, tant que se li fa impossible esquivar-ho, activa l'escut. També el que feim es que si detecta que ha rebut un tir (el més probable es que el rebi per l'esquena, ja que sino el voria), utilitzi l'hiperespai per escapar, ja que es molt perillós tenir un bitxo darrera.

2. Explicació del codi

A aquest apartat explicarem què fa cada mètode creat per nosaltres. En concret, les particularitats de cada un.

2.1. Mètode 'onGirar'

Aquest mètode permet al nostre bitxo donar una orentació en el mapa en quant a la direcció a la que ha d'anar per evitar les parets.

```
public void onGirar() {
    //CAS DE QUE VEU UN OBJECTE A CADA COSTAT
    if (estat.objecteVisor[0] == PARET && estat.objecteVisor[2] == PARET) {
        //CAS EN QUE L'OBJECTE DE L'ESQUERRA ESTIGUI M S APROP QUE EL DE LA DRETA
        if (estat.distanciaVisors[0] < 15 && estat.distanciaVisors[2] < 15) {
            esquerra();
            enrere();
        }else if ((estat.distanciaVisors[0] < estat.distanciaVisors[2])) {
            dreta();
        } else if (estat.distanciaVisors[0] >= estat.distanciaVisors[2]) { //CAS EN QUE
            L'OBJECTE DE LA DRETA ESTIGUI M S APROP QUE EL DE L'ESQUERRA
            esquerra();
        }
        //CAS EN QUE NICAMENT DETECTA L'OBJECTE A LA DRETA
    } else if (estat.objecteVisor[0] == PARET) {
        dreta();
        //CAS EN QUE NICAMENT DETECTA L'OBJECTE A L'ESQUERRA
    } else {
        esquerra();
    }
}
```

2.2. Mètode 'detectaParet'

Detecta si ens esteim a punt de col.lisionar amb una paret a la distància (integer) que es pasa per paràmetre . Activant, així, un booleà que retorna el mètode. Quan aquest està a 'true' vol dir que hi ha pareill de col.lisió, en canvi si es 'false' es que no n'hi ha.

```
public boolean detectaParet(int dist) {
    boolean hihaParet = false;
    if ((estat.distanciaVisors[0] <= dist && estat.objecteVisor[0] == 0) || (estat.
        distanciaVisors[1] <= (dist + 30) && estat.objecteVisor[1] == 0) || (estat.
        distanciaVisors[2] <= dist && estat.objecteVisor[2] == 0)) {
        hihaParet = true;
    }
    return hihaParet;
}
```

2.3. Mètode 'esquivar'

Utilitzat quan detectam que un enemic realitza un llançament cap a nosaltres. Detectam a quina distància ens tira el llançament, de manera que si el llançament es troba a una distància imposible d'esquivar, activam l'escut. Sino, intentam detectar l'agent que ens dispara i esquivam la seva trajectoria, ergo, la del llançament.

```
public void esquivar() {
    int dist = 9999;
    if (estat.distanciaLlan amentEnemic < 10) {
        if (!estat.escutActivat) {
            activaEscut();
        }
    } else if (estat.distanciaLlan amentEnemic > 10) {
        for (int i = 0; i < estat.numObjectes; i++) {
            if (estat.objectes[i].agafaTipus() == Estat.AGENT && estat.objectes[i].
                agafaDistancia() < dist) {
                if (estat.objectes[i].agafaSector() == 2) {
                    esquerra();
                    repetir = 3;
                } else if (estat.objectes[i].agafaSector() == 3) {
                    dreta();
                    repetir = 3;
                }
            }
        }
    }
}
```

2.4. Mètode 'filtrarObjecte'

Mira tots els objectes que l'envolten. Depenent del integer que es pasa per paràmetre, el cual ens diu si hem d'analitzar un recurs (n = 0) o un enemic (n = 1). L'enemic l'analitzam si detectam que els nostres visors veuen un enemic, si el veu, miram cap a l'enemic, disparam, i seguim el nostre camí. De manera continuada (explicada en el mètode 'avaluaComportament') anira mirant i dirigint-se cap als recursos i escuts que es va trobant pel camí. Si troba un recurs enemic (mina) la dispara.

```
public void filtrarObjecte(int n) { //0 si recurs, 1 si enemic
    int dist = 9999;
    for (int i = 0; i < estat.numObjectes; i++) {
        if (n == 0) { //INTERPRETAM RECURS

            /* CONDICIONS: Sigui escut o menjar nostre, estigui mes aprop, estigui dins
                els sectors de visi  2 i 3 */
            if (((estat.objectes[i].agafaTipus() == (MENJAR + estat.id) || estat.
                objectes[i].agafaTipus() == ESCUT)
```

```
                    && estat.objectes[i].agafaDistancia() < dist)
                    && (estat.objectes[i].agafaSector() == 2 || estat.objectes[i].
                        agafaSector() == 3)) {
                mira(estat.objectes[i]);
                dist = estat.objectes[i].agafaDistancia();
            }
            /* CONDICIONS: Sigui un menjar enemic (id = 100), es el que esta m s aprop,
                estigui dins els sectors 2 i 3 */ else if (estat.objectes[i].agafaTipus
            () == MENJAR && estat.objectes[i].agafaDistancia() < dist) {
            if (estat.llan aments != 0 && estat.objectes[i].agafaDistancia() <= 150
                    && (estat.objectes[i].agafaSector() == 2 || estat.objectes[i].
                        agafaSector() == 3)) {
                mira(estat.objectes[i]);
                llan a();
            } else {
                if (estat.objectes[i].agafaSector() == 2) {
                    dreta();
                } else {
                    esquerra();
                }
            }
        }
    } else if (n == 1) { //INTERPRETAM ENEMIC

        /*CONDICIONS: Sigui un bitxo i es el que esta mes aprop, estigui entre els
            sectors 2 i 3, estigui a una dist ncia igual o menor a 150 (m s
            probabilitats d'encertar). */
        if (estat.objectes[i].agafaTipus() == BITXO) {
            if ((estat.objectes[i].agafaSector() == 2 || estat.objectes[i].
                agafaSector() == 3)) {

                dist = estat.objectes[i].agafaDistancia();
                if (dist <= 150) {
                    mira(estat.objectes[i]);
                    llan a();
                }
            }
            if (estat.objectes[i].agafaSector() == 2) {
                esquerra();
                repetir = 3;
            } else if (estat.objectes[i].agafaSector() == 3) {
                dreta();
                repetir = 3;
            }
        }
    }
}
```

2.5. Mètode 'avaluaComportament'

Actualitza l'estat del bitxo. Amb una variable global anomenada 'repetir', entram al condicional. De manera que quan repetir es cero, vol dir que no esteim fent cap moviment que necesiti alguna repetició. Per tant dins el primer condicional únicament repetirem 'repetir' vegades el darrer moviment. Si no entra al condicional 'if', entra al 'else' actualitzant l'estat i filtrant l'entorn en diferents aspectes:

- **Recursos i escuts**: Miram l'entorn continuament cercant algun recurs o escut.

- **Enemics**: Utilitzam el booleà que s'activa si el bitxo veu un enemic, cridant a altres funcions per realitzar l'acció correcte. I en quant al llançament enemic, si detectam que ens han ferit, utilitzam el hiperespai a mode d'escapada d'emergència per no rebrer cap impacte més, al igual que feim amb les col.lisions.

- **Parets**: Miram si ens esteim atracant a una paret o si ens trobam en col.lisió, cridant a mètodes que ens ajuden a evitar la paret.

```
public void avaluaComportament() {

        estat = estatCombat();
        if (repetir != 0) {
            System.out.println("repito");
            repetir--;
        } else {

            atura();
            estat = estatCombat();
            filtrarObjecte(0);
            endavant();

            //DETECTAM QUE UN ENEMIC ENS HA DISPARAT --> ESQUIVAM
            if (estat.llan amentEnemicDetectat) {
                esquivar();
            }
            //SI REBEM UN TIR, ESCAPE D'EMERGENCIA
            if ((impactesRebutsAux < estat.impactesRebuts) && !estat.hiperEspaiActiu) {
                impactesRebutsAux = estat.impactesRebuts;
                hyperespai();
            }

            //ESTEIM EN COL LISIO I VOLEM TORNAR A CAM
            if (estat.enCollisio && !estat.hiperEspaiActiu) {
                hyperespai();
            }

            //EVITAM LA PARET
            if (detectaParet(60)) {
```

```
                System.out.println("paret a 90");
                onGirar();
                repetir = 5;
            }

            if (detectaParet(10)) {
                System.out.println("paret a 10");
                enrere();

                if (estat.distanciaVisors[0] < estat.distanciaVisors[2]) {
                    esquerra();
                } else {
                    dreta();
                }
                repetir = 6;
            }
            //VEIM UN ENEMIC --> ACTUAM DEPENENT DE LA SEVA DISTANCIA I POSICIO
            if (estat.veigAlgunEnemic) {
                filtrarObjecte(1);
            }
        }
    }
```