Memoria Práctica 2 (AIN)

Marcos Esteve Casademunt, Jose Gómez Gadea

Junio 2018

1. Objetivo 1: Evitar fuego amigo

```
+!get_agent_to_aim
<-  ?debug(Mode); if (Mode<=2) { .println("Looking for agents to aim."); }
?fovObjects(FOVObjects);
.length(FOVObjects, Length);

?debug(Mode); if (Mode<=1) { .println("El numero de objetos es:", Length); }

if (Length > 0) {
    +bucle(0);
+blucle2(0);

    -+aimed("false");

    while (aimed("false") & bucle(X) & (X < Length)) {

// Coge de la lista FOVObjects el elemento n X y lo pone en Object
        .nth(X, FOVObjects, Object);
        // Object structure
        // [#, TEAM, TYPE, ANGLE, DISTANCE, HEALTH, POSITION ]
        .nth(2, Object, Type);

        //.println("Analizando objeto: ", Object);

        if (Type > 1000) {
            //.println("I found some object.");
        } else {
            // Object may be an enemy
            .nth(1, Object, Team);
            ?my_formattedTeam(MyTeam);

            if (Team == 200) {  // If he is an enemy...
```

```
//.println("Aiming an enemy...");
+aimed_agent(Object);
-+aimed("true");
.println("Atacando a: ",Team);

// Evitamos el fuego amigo

while (aimed(A) & A == "true" & bucle2(Y) & (Y < Length)) {
// Coge de la lista FOVObjects el elemento nY y lo pone en Object2
.nth(Y, FOVObjects, Object2);
if(not(X==Y)) {
// Object structure
// [#, TEAM, TYPE, ANGLE, DISTANCE, HEALTH, POSITION ]
.nth(3, Object2, Angle2);
.nth(3, Object, Angle1);
if((Angle1-Angle2) < 0.35 & (Angle1-Angle2) > -0.35) {
.nth(4, Object2, Distance2);
.nth(4, Object, Distance1);
if(Distance1>=Distance2) {
.nth(1, Object2, Team2);
if(100==Team2) { // Si es un aliado...
.println("Dejando de atacar por: ",Team2);
-+aimed("false");
}
}
}
}
-+bucle2(Y+1);
}
-+bucle2(0);
            }
        }
        -+bucle(X+1);
    }
}
-bucle(_).
-bucle2(_).
```

En este objetivo se pretende evitar el fuego entre miembros del mismo equipo. Para ello, en caso de que un aliado este en un angulo similar y distancia menor al enemigo al que se esta apuntando entonces este aliado no disparará ya que puede dañar a su compañero. Este código lo implementan todos los agentes en el plan *get_agent_to_aim*

2. Objetivo 2: El general

En nuestra implementación hemos decidido registrar un general que de ordenes. En primer lugar, el general registra el servicio mediante:

```
.register("JGOMAS","GENERAL_ALIADO");
```

Una vez registrado el servicio envía un mensaje a cada agente para que vayan a una posición determinada y de esta forma ataquen desde múltiples flancos. Además el General esperará 20 segundos antes de ponerse en camino hacia la bandera.

```
+!init
   <- ?debug(Mode); if (Mode<=1) { .println("YOUR CODE FOR init GOES HERE.")}
   .register("JGOMAS","GENERAL_ALIADO"); // Nombre para la comunicacion (servicio)
   .my_team("ALLIED", E1);
   -+cuenta(0);

   for(.member(I,E1)){
 ?cuenta(C);
     .concat("goto(",150+C,",0,",200,")", Content2);
     .send_msg_with_conversation_id(I, tell, Content2, "TEST2");
 -+cuenta(C+10);
   }
   .my_name(YO);
   .wait(20000);
   !add_task(task(9000,"TASK_GOTO_POSITION", YO, pos(40, 0, 40), ""));
   -+state(standing);
   .
```

El resto de agentes implementan un plan para gestionar las ordenes del General. Este plan permite recoger las ordenes impuestas por el general y ir a la posición especifica.

```
+goto(X,Y,Z)[source(A)]
<-
.println("Recibido mensaje goto de ", A);
!add_task(task(9000,"TASK_GOTO_POSITION", A, pos(X, Y, Z), ""));
-+state(standing);
-goto(_,_,_) .
```

3. Objetivo 3: El médico

Todos los agentes implementan el plan *perform_injury_action*. Este plan se dispara cuando un agente recibe un disparo, en ese momento comprueba su vida y en caso de estar por debajo del 30 % envía un mensaje de ayuda al general y huye a la posición 127,210.

```
+!perform_injury_action
    <-
    ?my_health(Hr);
    if(Hr < 30){
.my_team("GENERAL_ALIADO", E);
.concat("help", Content2);
.send_msg_with_conversation_id(E, tell, Content2, "TEST2");
!add_task(task(9000,"TASK_GOTO_POSITION", A, pos(127, 0, 210), ""));
.println("Huyendo a la posicion: ", pos(127, 0, 210));
-+state(standing);
    }    .
```

El general implementa un plan help. Este plan recibe el mensaje de auxilio del agente herido. El general se encargará de enviar un mensaje goto a todos los médicos para que vayan a la posición 127,210.

```
+help[source(A)]
<-
.println("Recibido mensaje de ayuda goto de ", A);
.my_team("medic_ALLIED", E);
.concat("goto(127, 1, 210)", Content2);
.send_msg_with_conversation_id(E, tell, Content2, "TEST2");
-help .
```

Por último, los médicos tienen un plan goto diferente al resto de agentes, esto es porque reciben dos tipos de mensajes: Por una parte los mensajes de ir a una posición determinada al inicio de la partida. Por otra parte los mensajes de auxilio ordenados por el general y que por tanto, es necesario que dejen paquetes de vida al llegar a la posición especificada.

```
+goto(X,Y,Z)[source(A)]
<-
.println("Recibido mensaje goto de ", A);
if(Y==1) {
!add_task(task(9000,"TASK_GIVE_MEDICPAKS", A, pos(X, 0, Z), ""));
} else {
!add_task(task(9000,"TASK_GOTO_POSITION", A, pos(X, 0, Z), ""));
}
    .println("Voy a ", pos(X,Y,Z));
-+state(standing);
-goto(_,_,_)
.
```

4. Objetivo 4: Despistar al enemigo

Por último, cuando un agente coge la bandera y se dirige hacia la base el resto de aliados se ponen a patrullar por la zona de los enemigos con el fin de despistar a los enemigos y evitar que maten al agente que lleva la bandera.

```
+!perform_look_action
 <-
?my_position(X, Y, Z);
!add_task(task(10,"TASK_PATROLLING", A, pos(X, 0, Z), "")).
```

5. Conclusión

Como conclusión comentar que encontramos la práctica muy importante para comprender correctamente el funcionamiento de los agentes y sobretodo la comunicación y actuación frente a otros agentes u objetos.

También nos gustaría comentar que, aunque nos parece una forma muy interesante y atractiva de realizar las prácticas, debido al mal funcionamiento de los agentes ya implementados (con sus respectivos métodos) y a la mala depuración que ofrece Jason (pese a la ayuda del entorno 3D ofrecido por el programa desarrollado con Unity) se dificulta mucho el desarrollo de la práctica, causando la pérdida de tardes enteras por ejemplo solucionando un pequeño error complicado de detectar mediante prints.

Mencionar también que la práctica nos muestra la importancia de desarrollar, desde un primer momento, una arquitectura eficiente y fácilmente estructurable, para de esta forma poder ampliar de forma más sencilla las funciones de los agentes y así poder realizar agentes más complejos, sin incrementar exponencialmente la curva de dificultad para comprender su funcionamiento.