Memoria Práctica 1 (AIN)

Marcos Esteve Casademunt, Jose Gómez Gadea

Abril 2018

1. Objetivo 1: Agente que imprima su posición

```
+!perform_look_action
<-
?debug(Mode);
if (Mode<=3) {
?my_position(X, Y, Z);
.println("Posición actual: (",X,",",Z,")");
}.
```

En este caso, sobrescribimos el plan perform_look_action ya que este plan se ejecuta cada vez que el agente mira hacia algún lugar, es decir, cada vez que se mueve. Instanciamos en las variables X, Y, Z la posición mas reciente del agente y la imprimimos por pantalla. El código de este agente está implementado en el fichero **jasonAgent_ALLIEDobj1.asl**

2. Objetivo 2: Agente que imprima los objetos que ve

```
+!perform_look_action
<-
?debug(Mode);
if (Mode<=3) {
?fovObjects(FOVObjects);
.println("La lista de objetos que veo es ",FOVObjects);
}.
```

Por la misma razón que en el objetivo anterior, sobreescribimos el plan perform_look_action. Instanciamos en FOVObjects la lista de objetos que ve nuestro agente y lo imprimimos por consola.El código de este agente está implementado en el fichero **jasonAgent_ALLIEDobj2.asl**

3. Objetivo 3: Agente que recorra las esquinas

```
+!update_targets
<- ?debug(Mode); if (Mode<=1) { .println("Going to a new position.") }
?estado(E);
if(E == 0) {
!add_task(task(3000,"TASK_GOTO_POSITION",M,pos(50,0,50),""));
-+estado(1);
}
if(E == 1) {
!add_task(task(3000,"TASK_GOTO_POSITION",M,pos(205,0,50),""));
-+estado(2);
}
if(E == 2) {
!add_task(task(3000,"TASK_GOTO_POSITION",M,pos(205,0,205),""));
-+estado(3);
}
if (E == 3) {
!add_task(task(3000,"TASK_GOTO_POSITION",M,pos(50,0,205),""));
-+estado(0);
}.
```

En este objetivo se ha implementado un autómata de 4 estados estado(0..3) para recorrer las 4 esquinas. El cambio de estado añade un plan para ir a la siguiente esquina. Nos gustaría destacar que la mejora en la que el agente comprueba si una posición es válida se podría realizar con el siguiente código:

```
!safe_pos(50,0,50);
?safe_pos(X,Y,Z);
```

De esta forma se instanciarán en X, Y, Z las coordenadas de un punto seguro cercano al (50,0,50). En nuestras pruebas el agente se ha quedado "colgado" al utilizarlo.El código de este agente está implementado en el fichero **jasonAgent_ALLIEDobj3.asl**

4. Objetivo4: Agente que siga a su compañero

```
+!perform_look_action
<-
?debug(Mode); if (Mode<=2) { .println("Looking for agents to aim."); }
?fovObjects(FOVObjects); // Cojo la lista de objetos vistos
.length(FOVObjects, Length);
if (Length > 0) { //si veo objetos
+bucle(0); -+aimed("false");
 while (aimed("false") & bucle(X) & (X < Length)) {
```

```
 //recorro la lista de objetos vistos
  .nth(X, FOVObjects, Object);
  // Object structure
  // [#, TEAM, TYPE, ANGLE, DISTANCE, HEALTH, POSITION ]
 .nth(2, Object, Type);   //si el tipo de ese objeto es >1000--> entonces bandera
 if (Type > 1000) {  ?debug(Mode); if (Mode<=2) { .println("I found some object.");
} else {
.nth(1, Object, Team);
?my_formattedTeam(MyTeam);

  if (Team == 100) {  // Only if I'm ALLIED
.nth(6, Object, Position);
!add_task(task(9999, "TASK_GOTO_POSITION", M, Position, ""));
-+state(standing);
-+aimed("true");
}
}
 -+bucle(X+1);
}.
```

Para realizar este objetivo tenemos que realizar los siguientes pasos

- Recuperamos la lista de los objetos visibles por el agente X(?fovObjects(FOVObjects))
- Recorremos esta lista buscando aliados (Team == 100)
- Si encuentro aliado entonces defino un plan para ir a su posición !add_task(task(9999,"TASK_GOTO_POSITION",M,Position,));
- Cambio el estado a standing (Así el agente cambiará su plan actual por el nuevo)

Al lanzar 15 agentes que siguen y un líder (recorre esquinas) observamos un comportamiento en el que los agentes siguen al líder. Aun así, esta propuesta da ciertos problemas, ya que en ocasiones los agentes que siguen no ven al líder y por tanto, van hacia la bandera. Una posible solución sería implementar comunicación en los agentes para que se den cuenta de quien es el líder. El código de este objetivo está implementado en el fichero **jasonAgent_ALLIEDobj4.asl**