



Agentes Inteligentes (AIN)

Práctica 1 Escenario ARENA

- ❖ Los agentes soldados de pyGomas se implementan con:
 - ❖ Un fichero asl con los planes de alto nivel
 - ❖ Fichero .py con posibles nuevas acciones internas

- ❖ Por defecto, si no se indica en el fichero JSON, los agentes cargan un fichero ASL asociado a su rango:
 - ❖ bdisoldier.asl para Soldados
 - ❖ bdifieldop.asl para operadores de campo
 - ❖ bdimedic.asl para médicos

Disponibles en Poliformat → “ASL agentes básicos”

pyGOMAS

Creencias de los Agentes Tropa (I)

- **flag([X,Y,Z]):** [X, Y, Z] es la posición de la bandera.
- **heading([X, Y, Z]):** el Ag. Tropa está orientado hacia [X, Y, Z].
- **health(X):** X es la salud actual del agente.
- **ammo(X):** X es la munición actual del agente.
- **base([X,Y,Z]):** La base del equipo del agente está en [X, Y, Z].
- **name(X):** X es el nombre del agente.
- **myMedics([id ...]):** Lista de médicos del equipo activos.
- **myFieldops([id ...]):** Lista de FieldOps del equipo activos.
- **myBackups([id ...]):** Lista de Soldados del equipo activos.
- **position([X,Y,Z]):** [X, Y, Z} es la posición actual del agente.
- **team(X):** el Ag. Tropa pertenece al equipo X.

pyGOMAS

Creencias de los Agentes Tropa (II)

- ✿ **class(X):** X es la clase a la que pertenece el agente:
 - ✿ NONE = 0, SOLDIER = 1, MEDIC = 2, ENGINEER = 3, FIELDOPS = 4
- ✿ **enemies_in_fov(ID, TYPE, ANGLE, DIST, HEALTH, [X,Y,Z]):** El Ag. Tropa ha visto un enemigo con identificador ID, del tipo TYPE, a un ángulo ANGLE, a una distancia DIST, con una salud HEALTH, y en la posición [X, Y, Z] .
- ✿ **friends_in_fov(ID, TYPE, ANGLE, DIST, HEALTH, [X,Y,Z]):** El Ag. Tropa ha visto un compañero de equipo...
- ✿ **packs_in_fov(ID, TYPE, ANGLE, DIST, HEALTH, [X,Y,Z]):** El Ag. Tropa ha visto un pack ...
 - ✿ Tipos de Pack: 1000 (None), 1001 (MEDICPACK), 1002 (AMMOPACK), 1003 (FLAG).

pyGOMAS

Creencias de los Agentes Tropa (III)

- **threshold_health(X)**: X es la salud mínima antes de lanzar una acción especial como respuesta.
- **threshold_ammo(X)**: X es la munición mínima antes de lanzar una acción especial como respuesta.
- **threshold_shots(X)**: Límite máximo de disparos simultáneos.
- **velocity([X,Y,Z])**: [X, Y, Z] es la velocidad actual del Ag. Tropa.
- **destination([X,Y,Z])**: Objetivo del Ag. Tropa: [X,Y,Z].
- **pack_taken(TYPE, N)**: Si el agente ha cogido un pack de tipo TYPE (**medic** o **fieldops**) y la cantidad a aumentar de vida / munición.
- **flag_taken**: Si el agente ha cogido la bandera.
- **target_reached([X, Y, Z])**: Se añade cuando el agente llega a su destino ([X, Y, Z]).

pyGOMAS

Acciones de los Agentes Tropa

- Movimiento:

- **.goto([X,Y,Z])**: Establecer [X,Y,Z] como destino del ag. Pone al ag. tropa en marcha hacia dicho lugar, usando un algoritmo JPS para desplazarse por el terreno.
- **.stop**: Detener el mov. del ag. tropa.
- **.turn(R)**: Modificar la orientación del ag. tropa una cantidad (pos. o neg.) R de radianes. **Útil para alterar el campo de visión.**
- **.look_at([X,Y,Z])**: Orientar el ag. tropa hacia [X,Y,Z].
- **.create_control_points([X,Y,Z],D,N,C)**: Crear un grupo de N puntos aleatorios de control a una distancia D dada de una ubicación [X,Y,Z] en el mapa. La lista de puntos se almacena en C. Ej.: patrullar alrededor de la bandera.

pyGOMAS

Acciones de los Agentes Tropa

- ❖ Envío de mensajes al Service Agent:
 - ❖ `.register_service("servicio_a")`: Enviar mens. al Service Ag. para registrar un servicio especificado.
 - ❖ `.get_medics`: Enviar mens. al Service Ag. solicitando los médicos de su equipo.
 - ❖ `.get_fieldops`: Enviar un mensaje al Service Ag. solicitando los operadores de campo de su equipo.
 - ❖ `.get_backups`: Enviar un mensaje al Service Ag. solicitando los soldados de su equipo.
 - ❖ `.get_service("servicio_a")`: Enviar un mensaje al Service Ag. solicitando otro servicio (distinto de los tres anteriores) a los agentes tropa de su equipo que lo ofrezcan.

pyGOMAS

Acciones de los Agentes Tropa

- **.shoot(N,[X,Y,Z]):** Disparar N disparos a [X,Y,Z].
- **.cure:** Crear paquetes de medicina. Solo los médicos pueden realizar esta acción.
- **.reload:** Crear paquetes de munición. Solo los operadores de campo pueden realizar esta acción.

pyGOMAS

Fichero ASL

- ❖ Ejemplo “bdisoldier.asl”

```
//TEAM_ALLIED
```

```
+flag (F): team(100)
```

```
<-
```

```
.goto(F);
```

```
+exploring.
```

```
+flag_taken: team(100)
```

```
<-
```

```
.print("In ASL, TEAM_ALLIED flag_taken");
```

```
?base(B);
```

```
+returning;
```

```
.goto(B);
```

```
-exploring.
```

pyGOMAS

Fichero ASL

- Ejemplo “bdisoldier.asl”

```
// TEAM_AXIS
```

```
+flag (F): team(200)
```

```
<-
```

```
.create_control_points(F,25,3,C);
```

```
+control_points(C);
```

```
.wait(5000);
```

```
.length(C,L);
```

```
+total_control_points(L);
```

```
+patrolling;
```

```
+patroll_point(0);
```

```
.print("Got control points").
```

```
+target_reached(T): patrolling & team(200)
```

```
<- ?patroll_point(P);
```

```
-+patroll_point(P+1);
```

```
-target_reached(T).
```

```
+patroll_point(P): total_control_points(T) & P<T
```

```
<- ?control_points(C);
```

```
.nth(P,C,A);
```

```
.goto(A).
```

```
+patroll_point(P): total_control_points(T) & P==T
```

```
<- -patroll_point(P);
```

```
+patroll_point(0).
```

pyGOMAS

Fichero ASL

- Ejemplo “bdisoldier.asl”

```
+enemies_in_fov(ID,Type,Angle,Distance,Health,Position)
<-
.shoot(3,Position).
```

pyGOMAS

Fichero .py

- Añadimos un nuevo tipo de agente que incorpora más acciones

```
import json  
from pygomas.bditroop import BDITroop  
from ...  
  
class BDIIInvencible(BDITroop):  
  
    def add_custom_actions(self, actions):  
        super().add_custom_actions(actions)  
  
        @actions.add(".superhealth", 0)  
        def _superhealth(agent, term, intention):  
            self.health=200  
            self.bdi.set_belief(HEALTH, self.health)  
            yield
```

¿Cómo añadirlo?

- Añadir agentes del tipo *BDIIInvencible* en el fichero JSON
- Probarlo en el fichero .asl del agente:
...
.superhealth

pyGOMAS

Fichero .py

- Añadimos un nuevo tipo de agente que es invencible y lleva más acciones

```
import json  
from pygomas.bditroop import BDITrooper  
from ...  
  
class BDIIInvincible(BDITrooper):  
  
    def add_custom_actions(self, actions):  
        super().add_custom_actions(actions)  
  
        @actions.add(".superhealth")  
        def _superhealth(agent, target):  
            self.health=200  
            self.bdi.set_belief(HEALTH, self.health)  
  
            yield
```

¿Cómo lo añadir?

Añadir elementos del tipo
`BDIIInvincible` en el fichero
JSON

- Probarlo en el fichero .asl
del agente

...

superhealth

- ❖ 1^a Práctica: **Sobrevivir con sólo información del entorno:**
 - ❖ Programar un agente que trate de sobrevivir en un escenario hostil
 - ❖ La información que dispone es únicamente a través de sus creencias
 - ❖ Ganan los que logran sobrevivir después de un tiempo máximo

pyGOMAS

FORTNAIN

Escenario ARENA



Los agentes nacen
en cualquier punto

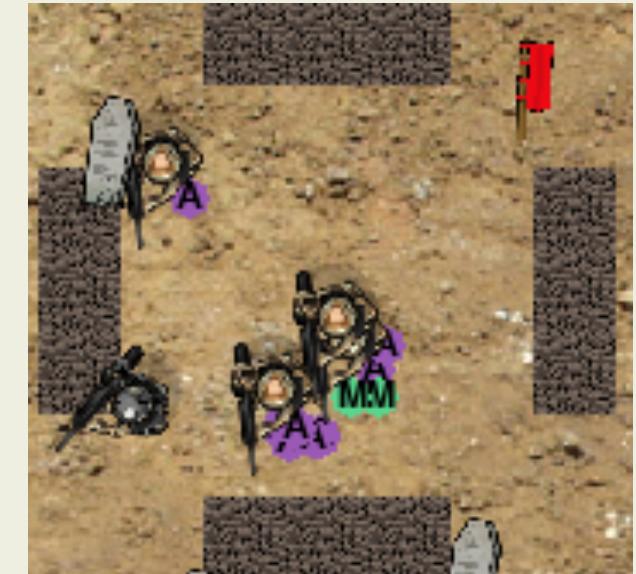
En la zona central
se generan
paquetes de
medicinas y armas

Todos los
participantes son
soldados “Eje” y se
deben disparar
entre ellos

pyGOMAS

FORTNAIN

Escenario ARENA



Soldados especiales
generan paquetes en
el centro

Cada soldado puede
tener su propia
estrategia

pyGOMAS



Escenario ARENA

- ❖ *¿Qué os damos?*
 - ❖ Un escenario arena con soldados “Aliados” en el centro que generan paquetes y son invencibles. No disparan, ni conviene dispararles.
 - ❖ El mapa a utilizar se llama: map_arena
 - ❖ Un conjunto de agentes “Eje” muy sencillos que simplemente se desplazan por puntos de control y disparan a sus amigos (os pueden servir de entrenamiento)
 - ❖ Ej. para ejecutar el manager:

```
shell:> pygomas manager -np 25 -j m_yourlogin@gtirouter.dsic.upv.es  
-sj s_yourlogin@gtirouter.dsic.upv.es -m map_arena
```



Posibles estrategias

- ❖ Conservadoras:
 - ❖ Quedarse quieto
 - ❖ Ir a un punto más seguro: por ejemplo una esquina
 - ❖ Escapar si te disparan o se ve a un amigo
- ❖ Agresivas:
 - ❖ Ir al centro a por munición y medicina
 - ❖ Perseguir a un amigo disparando

- ❖ Podéis ir luchando entre vosotros. ¿Cómo?:
 - ❖ Cada uno desarrolla su estrategia de agente en un fichero .asl
 - ❖ Uno de vosotros lanza un manager en su máquina (con un nº de agentes adecuado y con el mapa “map_arena”)
 - ❖ El resto de luchadores ejecuta en su máquina:
 - ❖ pygomas run -g miluchador.json
 - ❖ En la máquina donde se ha lanzado el manager se puede lanzar el render para ver la partida (para que vaya fluido)

IMPORTANTE:

En el fichero json se debe poner el mismo agente manager y servicio que el que ha lanzado la partida

- Entrega el **27 de abril** (tarea en Poliformat):
 - *Ficheros de código de vuestro agente: .asl, y en su caso, .py*
 - *Fichero de configuración .json para lanzar el agente*
 - *Documento con la descripción de la estrategia*
- Se puede hacer por parejas
- De forma virtual se ejecutará una **competición** entre los agentes