Agentes Inteligentes (AIN)

Práctica 1 pyGOMAS

Objetivos

- Aprender a desarrollar un agente soldado básico
- Conocer el entorno de la práctica 1
- Conocer el planteamiento de la práctica 1

pyGOMAS

Implementando Agentes

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

pyGOMAS

Fichero ASL

- 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 GitHub:
https://github.com/javipalanca/pygomas

Comportamiento básico

- Allied: van a por la bandera, si la capturan vuelven a la base
- Axis: van dando vueltas alrededor de la bandera según una lista de puntos de control aleatorios
- Ambos tipos de soldados disparan si ven a un enemigo

Fichero ASL

- Ejemplo comportamiento básico "bdisoldier.asl"

```
//TEAM_ALLIED
+flag (F): team(100)
  <-
  .goto(F).

+flag_taken: team(100)
  <-
  .print("In ASL, TEAM_ALLIED flag_taken");
  ?base(B);
  +returning;
  .goto(B);
  -exploring.
```

Ir a por la bandera
Este plan se dispara al inicio del agente aliado
flag(F) indica la posición inicial de la bandera
El agente aliado por defecto va directo a por la bandera

Si se captura se vuelve a la base
Este plan se dispara cuando el agente coge la bandera
base(B) indica la posición donde nació el agente en la base
El agente aliado por defecto va directo a la base cuando coge la bandera

Fichero ASL

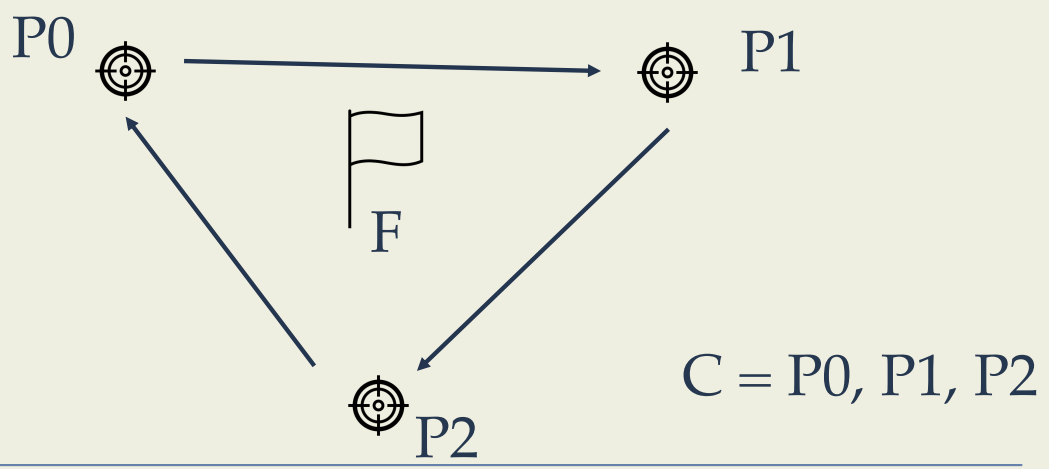
- Ejemplo comportamiento básico “bdisoldier.asl”

```
//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").
```

Al inicio crear puntos de control
Este plan es el inicio del agente eje
flag(F) indica la posición inicial de la bandera
El agente crea puntos de control alrededor de la bandera

Recorrerá dichos puntos de forma cíclica



pyGOMAS

Fichero ASL

- Ejemplo comportamiento básico "bdisoldier.asl"

```
+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).
```

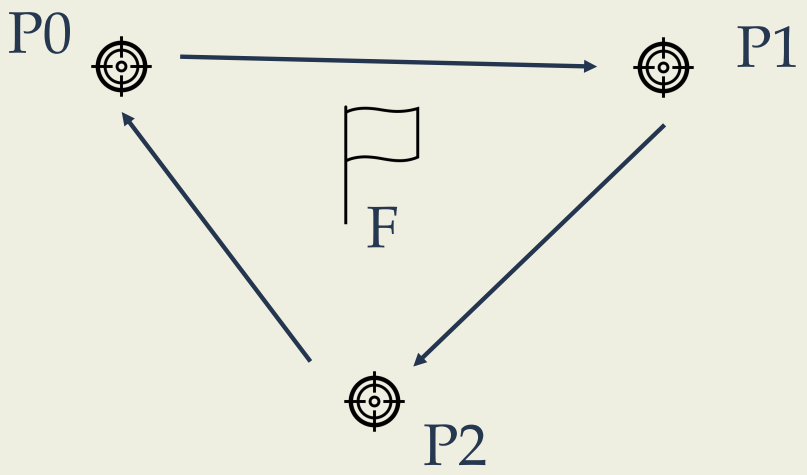
Reglas para recorrer los puntos de control

target_reached se ejecuta cada vez que el agente llega a un punto con .goto

patroll_point guarda el siguiente punto de control

P0→P1→P2

Se recorre como una lista circular



C = P0, P1, P2

Fichero ASL

- Ejemplo comportamiento básico "bdisoldier.asl"

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

Este plan se dispara cuando el agente ve a un enemigo
(se disparan tantas instancias del plan como creencias de enemigos hay)

En este caso dispara tres disparos a la posición donde ha visto al enemigo

Y si quisiese disparar a los amigos (luego sabréis porqué):

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

pyGOMAS

Fichero .py

- Se pueden añadir nuevos tipos de agente que incorporan más acciones

```
import json
from pygomas.agents.bditroop import BDITroop
from …

class BDIInvencible(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(Belief.HEALTH, self.health)
     yield
```

¿Cómo añadirlo?

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

  …
  .superhealth
  …

pyGOMAS

Fichero .py

- Se pueden añadir nuevo… …e incorporan más acciones

```
import json
from pygomas.agents.bdit…  …port BDITroop
from …


class BDIInvencible(…  …oop):


 def add_custom_act…  …self, actions):
   super().add_cust…  …tions(actions)


   @actions.add(".su…  …lth", …
   def _superhealth(ag…  …ention):
     self.health=200
     self.bdi.set_belief(B…  …LTH, self.health)
     yield
```

¿…añadirlo?

- …ha…gentes del tipo *BDIIn…ible* en el fichero JSON

- Probarl… el fichero .asl del age…
  - …
  - .s…health

Escenario ARENA

- 1ª 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

Escenario ARENA

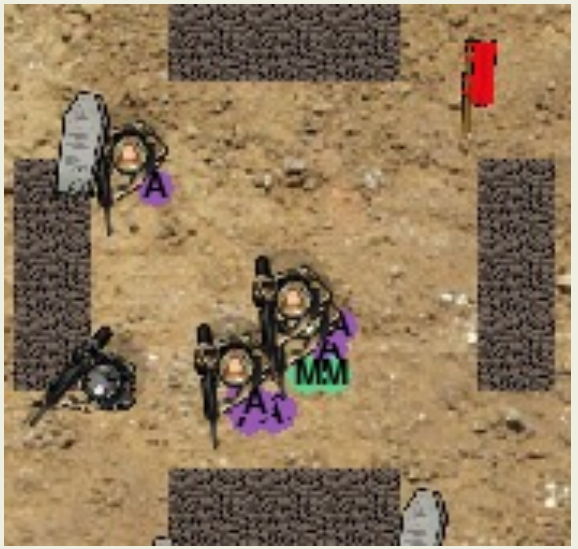
Los agentes nacen en cualquier punto

En la zona central se generan paquetes de medicinas y armas

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

pyGOMAS

Escenario ARENA



Soldados especiales generan paquetes en el centro

Cada soldado puede tener su propia estrategia

Escenario ARENA

- *¿Qué os damos? (Poliformat)*

  - 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
    - Ver código de *luchador.asl*
  - Ej. para ejecutar el manager:

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

  - Ej. para lanzar los soldados:

```
shell:>  pygomas run –g game_arena.json
```

Escenario ARENA

- *¿Qué os damos? (Poliformat)*
  - Ej. para lanzar el render (en pygame)

    shell:> pygomas render

En Polilabs en Linux:

Si al ejecutar el render da un error con *swrast*

Ejecutar esta instrucción desde vuestro directorio home

```
rm ./.conda/envs/pygomas/lib/libstdc++*
```

Y volver a ejecutar el render

pyGOMAS

Escenario ARENA

- ¿Cómo añadir nuestro soldado a la partida?

```
{  "host": "gtirouter.dsic.upv.es",
   "manager": "manager_yourlogin",
   "manager_password": "secret",
   "service": "service_yourlogin",
   "service_password": "secret",
   "axis": [
      {
         "rank": "BDISoldier",
         "name": "luchador_yourlogin",
         "password": "secret",
         "amount": 21,
         "asl": "luchador.asl"
      }
   ],
    "allied": [
            {
         "rank": "invencibleM.BDIMInvencible",
         "name": "medic_yourlogin",
         "password": "secret",
         "amount": 1,
         "asl": "medic_arena.asl"
      },
      {
         "rank": "invencibleF.BDIFInvencible",
         "name": "fieldop_yourlogin",
         "password": "secret",
         "amount": 3,
         "asl": "fieldop_arena.asl"
      }    ]    }
```

Añadimos un nuevo soldado

```
{  "host": "gtirouter.dsic.upv.es",
   "manager": "manager_yourlogin",
   "manager_password": "secret",
   "service": "service_yourlogin",
   "service_password": "secret",
   "axis": [
      {
         "rank": "BDISoldier",
         "name": "luchador_yourlogin",
         "password": "secret",
         "amount": 20,
         "asl": "luchador.asl"
      },
     {
         "rank": "BDISoldier",
         "name": ”miluchador_yourlogin",
         "password": "secret",
         "amount": 21,
         "asl": ”miluchador.asl"
      }
   ],
    "allied": [
            {
         "rank": "invencibleM.BDIMInvencible",
         "name": "medic_yourlogin",
         "password": "secret",
         "amount": 1,
         "asl": "medic_arena.asl"
      },
      {
         "rank": "invencibleF.BDIFInvencible",
```

Cuidado con los nombres de los agentes
Sin nombres repetidos
Usad vuestro login

pyGOMAS

Escenario ARENA

- 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

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

  - En la máquina donde se ha lanzado el manager se puede lanzar el render para ver la partida (para que vaya fluido)

pyGOMAS

Escenario ARENA

- Entrega hasta el **7 de mayo** (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

  Se ejecutará una **competición** entre los agentes el 7 de mayo

  NOTA: la entrega será antes de empezar la práctica de ese día

  - IMPORTANTE: consultar anexos del manual de **pygomas** para conocer todas las creencias y acciones existentes que podéis usar