

Agentes Inteligentes (AIN)

Presentación de pyGOMAS

- ❖ 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

pyGOMAS

Fichero ASL

- Ejemplo “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.
```

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 corpora 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")  
        def _superhealth(agent, target):  
            self.health=200  
            self.bdi.set_belief(HEALTH, self.health)  
  
            yield
```

¿Cómo añadirlo?

Añadiremos un agente del tipo `BDIIInvencible` en el fichero `JSON`

- Probarlo en el fichero `.asl` del agente

...

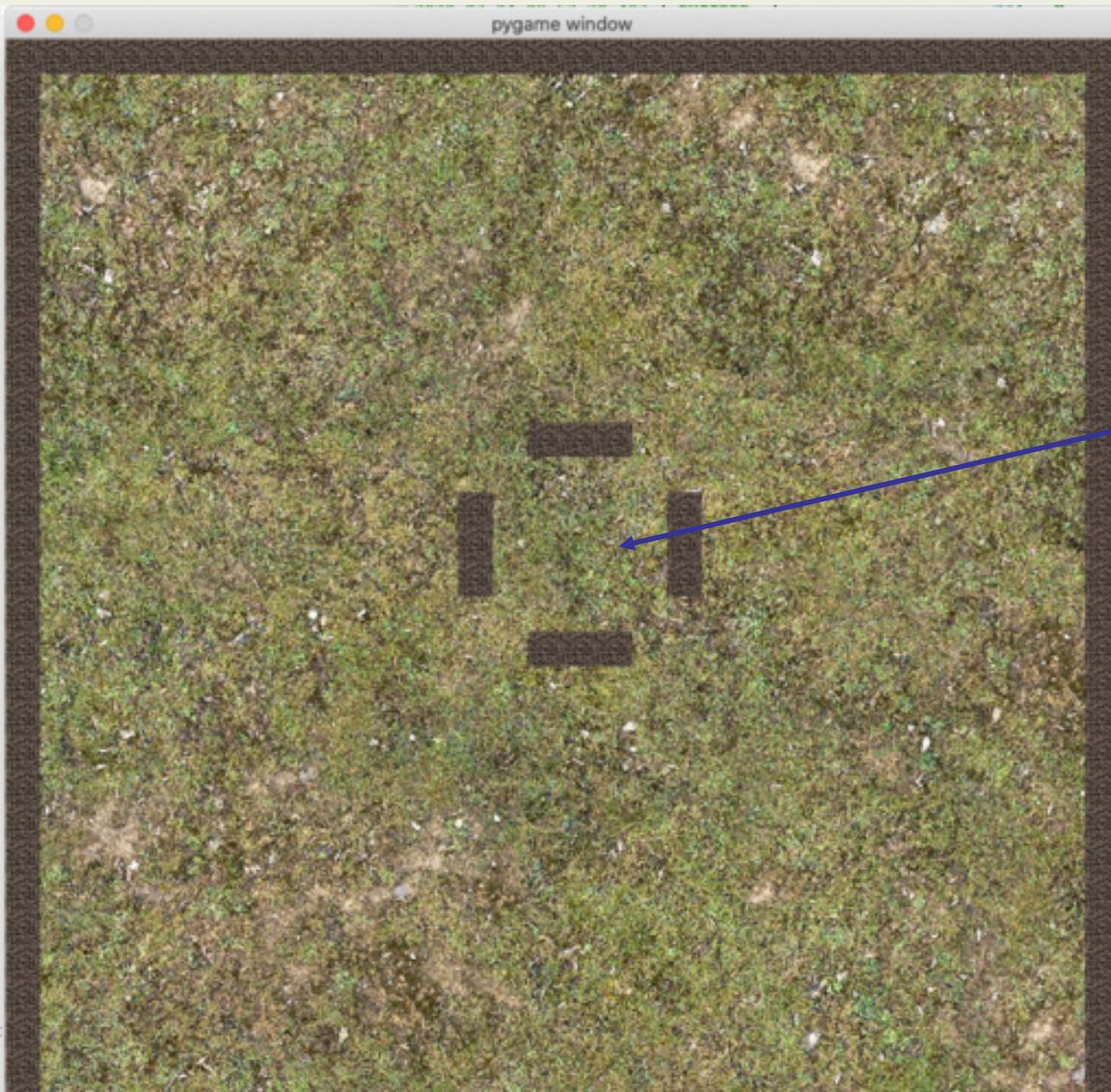
`.superhealth`

- ❖ 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



Escenario ARENA



Los agentes nacen en cualquier punto

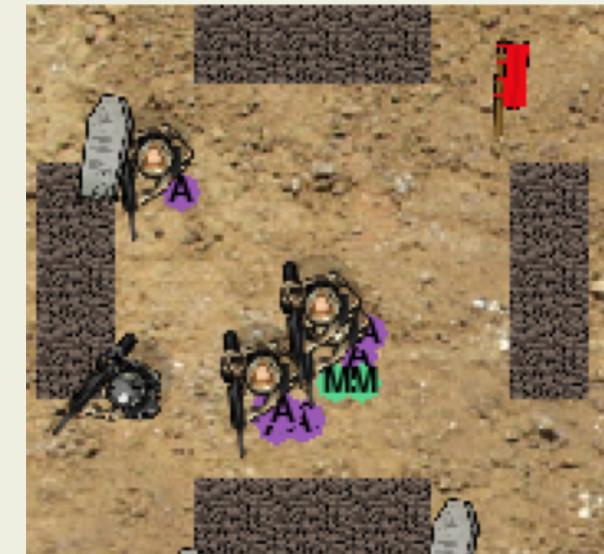
En la zona central se generan paquetes de medicinas y armas

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

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
```

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`
 - ✿ 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 7 de abril (tarea en Poliformat):
 - ✿ Ficheros de código de vuestro agente: .asl, y en su caso, .py
 - ✿ Documento con la descripción de la estrategia
- ✿ Se puede hacer por parejas
- ✿ El día de la entrega haremos una **competición** en directo