



# Agentes Inteligentes (AIN)

Presentación de pyGOMAS

---

---

# Agentes Inteligentes (AIN)

---

---

- ❖ Profesorado:

- ❖ Carlos Carrascosa ([carrasco@dsic.upv.es](mailto:carrasco@dsic.upv.es))
- ❖ Vicente Julián ([vinglada@dsic.upv.es](mailto:vinglada@dsic.upv.es))

## Prácticas

|   |  |  |  |  | P1   | P2  | P3   | P4   | P5  | P6   | P7   | P8  | P9   | P10  |
|---|--|--|--|--|------|-----|------|------|-----|------|------|-----|------|------|
| M |  |  |  |  | 25-2 | 3-3 | 10-3 | 31-3 | 7-4 | 21-4 | 28-4 | 5-5 | 12-5 | 19-5 |

# Introduction

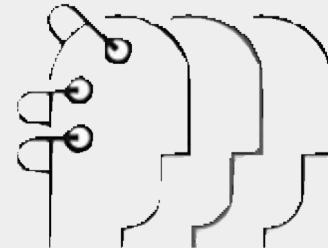
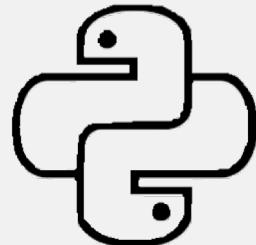
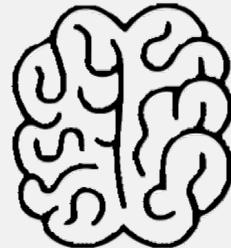
---



SPADE

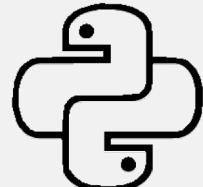
---

## Smart Python Agent Dev Env



based on:

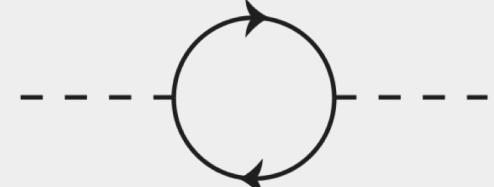
Python >= 3.6



JABBER / XMPP



AsyncIO



---

# Agente Híbrido SPADE

## Instalación

<https://spade-mas.readthedocs.io>

### \* GitHub:

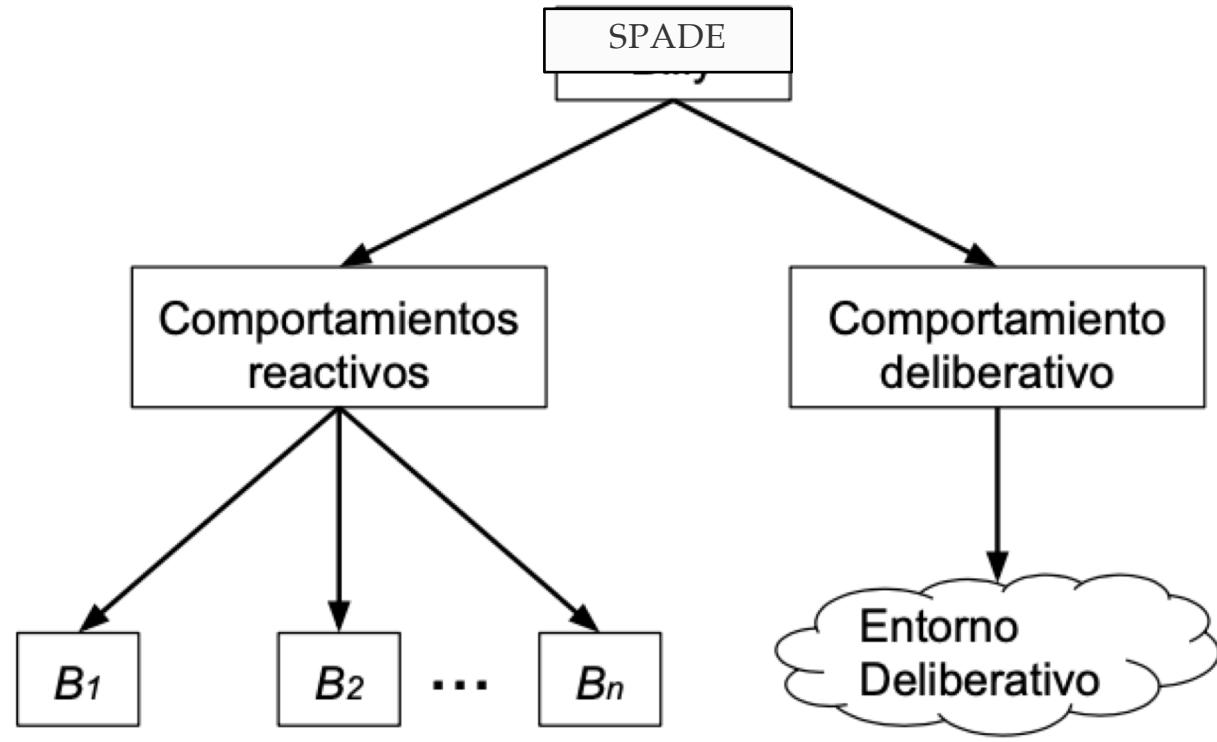
- \* Descargar código de: [https://github.com/javipalanca/spade\\_bdi](https://github.com/javipalanca/spade_bdi)
- \* `python setup.py install`

### \* PyPi:

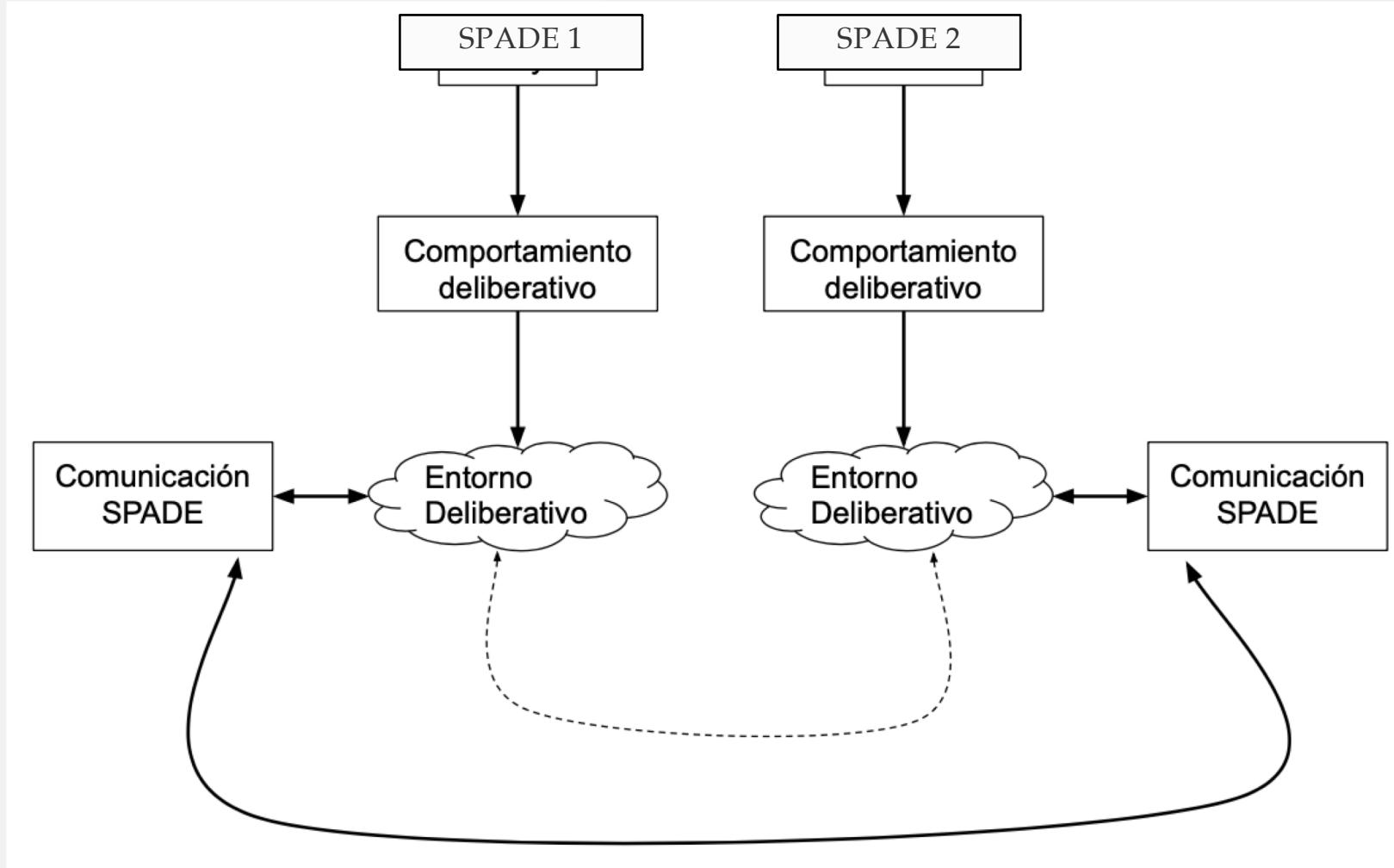
- \* Anaconda: Crear un entorno con Python 3.7
- \* `$ pip install spade_bdi`

```
(Optional) Installing XMPP Server: Prosody IM:  
- Install Homebrew:  
$ /usr/bin/ruby -e "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/install)"  
$ brew tap prosody/prosody  
$ brew install prosody  
- Configure Prosody IM: Activate in-band  
register (in prosody is: allow_registration=true in  
prosody.cfg.lua)  
Launch Prosody IM
```

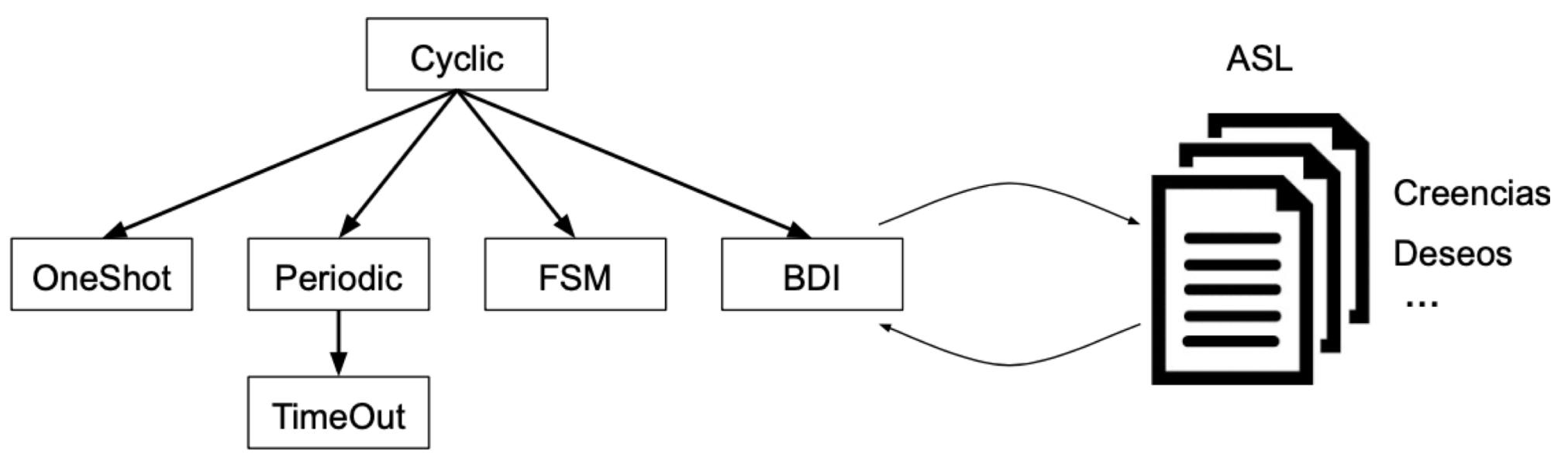
# Agente Híbrido SPADE



# Agente Híbrido SPADE



# Agente Híbrido SPADE



# User Interface

introdu



SPADE



your\_jid

Online

Dashboard

20



your\_jid

## Dashboard

Home > Dashboard

### Behaviours

#### CyclicBehaviour/DummyBehav

Template: <template to="None" from="agent0@fake\_server" thread="...

Kill

#### PeriodicBehaviour/DummyPeriodBehav

Template: <template to="None" from="agent1@fake\_server" thread="...

Kill

#### TimeoutBehaviour/DummyTimeoutBehav

Template: <template to="None" from="agent2@fake\_server" thread="...

Kill

#### FSMBehaviour/DummyFSMBehav

Template: <template to="None" from="agent3@fake\_server" thread="None" ...

### Contacts



agent0@fake\_se...

ONLINE



agent1@fake\_se...

AWAY



agent2@fake\_se...

DND



agent4@fake\_se...

ONLINE



agent3@fake\_se...

OFFLINE



agent5@fake\_se...

OFFLINE

# User Interface

The screenshot shows the SPADE User Interface with a dark theme. On the left, there's a sidebar featuring a large icon of an Ace of Spades card and a smaller icon of a robot head.

The main dashboard area has the following components:

- SPADE** header with a user icon labeled **your\_jid** and status **Online**.
- FSMBehaviour/DummyFSMBehav** section with a **4 Mailbox** icon.
- Template** section with a code snippet:

```
<template to="None" from="agent3@fake_server" thread="None" metadata={}></template>
```
- True** section with an **Is killed?** question and a skull icon.
- 0 Exit Code** section with a right-pointing arrow icon.
- S 1 Current State** section with a state transition diagram icon.
- Chat** window showing messages between **your\_jid** and **agent3**.
  - your\_jid** (11 minutes ago): This is my answer.
  - agent3** (11 minutes ago): Hello from agent3! This is a long message.
  - your\_jid** (11 minutes ago): This is my answer.
- Finite State Machine** diagram with states **S\_1**, **S\_2**, **S\_3**, **S\_4**, and **S\_5** and transitions between them.



agent.start()  
agent.web.start(hostname="127.0.0.1", port="10000")



<http://127.0.0.1:10000/spade>

# Agente Híbrido SPADE

---

---

- ❖ Inserción, eliminación y modificación de creencias desde dentro del comportamiento deliberativo (BDIBehaviour)
  - ❖ get\_belief: consultar una creencia que tenga el agente.
  - ❖ get\_beliefs: consultar todas las creencias.
  - ❖ get\_belief\_value: consultar el valor que toma una creencia.
  - ❖ set\_belief: establecer o modificar una creencia.
  - ❖ remove\_belief: eliminar una creencia que tenga el agente.
- ❖ Comunicación
  - ❖ .send: con el contenido a enviar, destinatario y receptor. En los metadatos se especifica la ilocución y la performativa (“BDI”) con que se enviará el mensaje por el Dispatcher.
  - ❖ Performativas: tell, untell y achieve

# Agente Híbrido SPADE

## Agente Hello

```
import argparse

from spade import quit_spade

from spade_bdi.bdi import BDIAgent

parser = argparse.ArgumentParser(description='spade bdi hello example')
parser.add_argument('--login', type=str, default="basicagent", help='your UPV login.')
parser.add_argument('--server', type=str, default="localhost", help='XMPP server address.')
parser.add_argument('--password', type=str, default="bdipassword", help='XMPP password for the agents.')
args = parser.parse_args()

a = BDIAgent("HelloAgent_{}@{}".format(args.login,args.server), args.password, "hello.asl")

a.start()

import time
time.sleep(1)

a.stop().result()

quit_spade()
```

```
|!start.

+!start <-
    .print("Hello World!").
```

\$ python hello.py --login <user\_login> --server gtirouter.dsic.upv.es

# Agente Híbrido SPADE

## Agente Básico

```
import argparse

from spade import quit_spade

from spade_bdi.bdi import BDIAgent

parser = argparse.ArgumentParser(description='spade bdi basic example')
parser.add_argument('--login', type=str, default="basicagent", help='your UPV login.')
parser.add_argument('--server', type=str, default="localhost", help='XMPP server address.')
parser.add_argument('--password', type=str, default="bdipassword", help='XMPP password for the agents.')
args = parser.parse_args()

a = BDIAgent("BasicAgent_{}@{}".format(args.login,args.server), args.password, "basic.asl")

a.start()

import time
time.sleep(1)

a.bdi.set_belief("car", "azul", "big")
a.bdi.print_beliefs()
print("GETTING FIRST CAR BELIEF")
print(a.bdi.get_belief("car"))
a.bdi.print_beliefs()
a.bdi.remove_belief("car", 'azul', "big")
a.bdi.print_beliefs()
print(a.bdi.get_beliefs())
a.bdi.set_belief("car", 'amarillo')

time.sleep(1)

a.stop().result()

quit_spade()
```

```
!start.

+!start <-
    +car(rojo);
    +truck(azul).

+car(Color)
<- .print("El carro es ",Color).
```

\$ python basic.py --login <user\_login> --server gtirouter.dsic.upv.es

# Agente Híbrido SPADE

## Agente Actions

```
arguments = parser.parse_args()

class MyCustomBDIAgent(BDIAgent):
    def add_custom_actions(self, actions):
        @actions.add_function(".my_function", (int,))
        def _my_function(x):
            return x * x

        @actions.add(".my_action", 1)
        def _my_action(agent, term, intention):
            arg = agentspeak.grounded(term.args[0], intention.scope)
            print(arg)
            yield

a = MyCustomBDIAgent("{}@{}".format(arguments.login, arguments.server), arguments.password, "actions.asl")
a.start()
```

```
!start.

+!start <-
    .my_function(4, R);
    .my_action(R).
```

\$ python actions.py --login <user\_login> --server gtirouter.dsic.upv.es

# Agente Híbrido SPADE

## Agente Launcher

```
import argparse

from spade import quit_spade

from spade_bdi.bdi import BDIAgent

parser = argparse.ArgumentParser(description='spade bdi launcher example')
parser.add_argument('--login', type=str, default="basicagent", help='your UPV login.')
parser.add_argument('--server', type=str, default="localhost", help='XMPP server address.')
parser.add_argument('--password', type=str, default="bdipassword", help='XMPP password for the agents.')
parser.add_argument('--asl', type=str, default="default.asl", help='asl file with JASON code.')
parser.add_argument('--time', type=int, default=1, help='duration time (in seconds) of the execution.')
args = parser.parse_args()

a = BDIAgent("Agent_{}@{}".format(args.login,args.server), args.password, args.asl)

a.start()

import time
time.sleep(args.time)

a.stop().result()

quit_spade()
```

ASL



\$ python asl\_launcher.py --login <user\_login> --server gtirouter.dsic.upv.es  
--asl <nombre\_fichero.asl> --time <máximo\_segs\_ejecución>

# Agente Híbrido SPADE

## Agente Launcher 2

---

---

- Usando el agente *asl\_launcher* comentado, ejecutar el ejemplo del factorial.

```
/* Creencias iniciales */
fact(0,1).

/* Planes */
+fact(X,Y) : X < 5
  <-  +fact(X+1, (X+1) * Y ).

+fact(X,Y) : X = 5
  <-  .print("fact 5 == ", Y ).
```

# Agente Híbrido SPADE

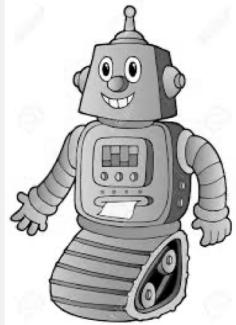
Agente robot limpiador

a

b

c

d



- Usando el agente *asl\_launcher* comentado, ejecutar el ejemplo del robot limpiador con la situación inicial del dibujo.

# Agente Híbrido SPADE

Agente robot limpiador

---

Robot que trata de limpiar basura

Movimientos en una sola dimensión

La basura hay que cogerla y llevarla a la papelera

Las creencias iniciales podrían ser:

adyacente( a, b).

adyacente( b, c).

adyacente( c, d).

localizado( robot, a).

localizado(papel\_usado, b).

localizado(papelera, d).

---

---

# Agente Híbrido SPADE

## Agente robot limpiador

**Objetivo inicial:** !localizado(robot, d).

**Planes:**

+localizado(robot, X) : localizado(papel\_usado, X)

```
<- .print("papel cogido en ", X);
      -localizado(papel_usado, X); // simula el coger papel
      +llevando(papel_usado). //simula el llevar papel
```

+localizado(robot, X) : localizado(papelera, X) & llevando(papel\_usado)

```
<- -llevando(papel_usado);
      .print("todo el papel tirado a la papelera en ", X). //simula tirar a papelera
```

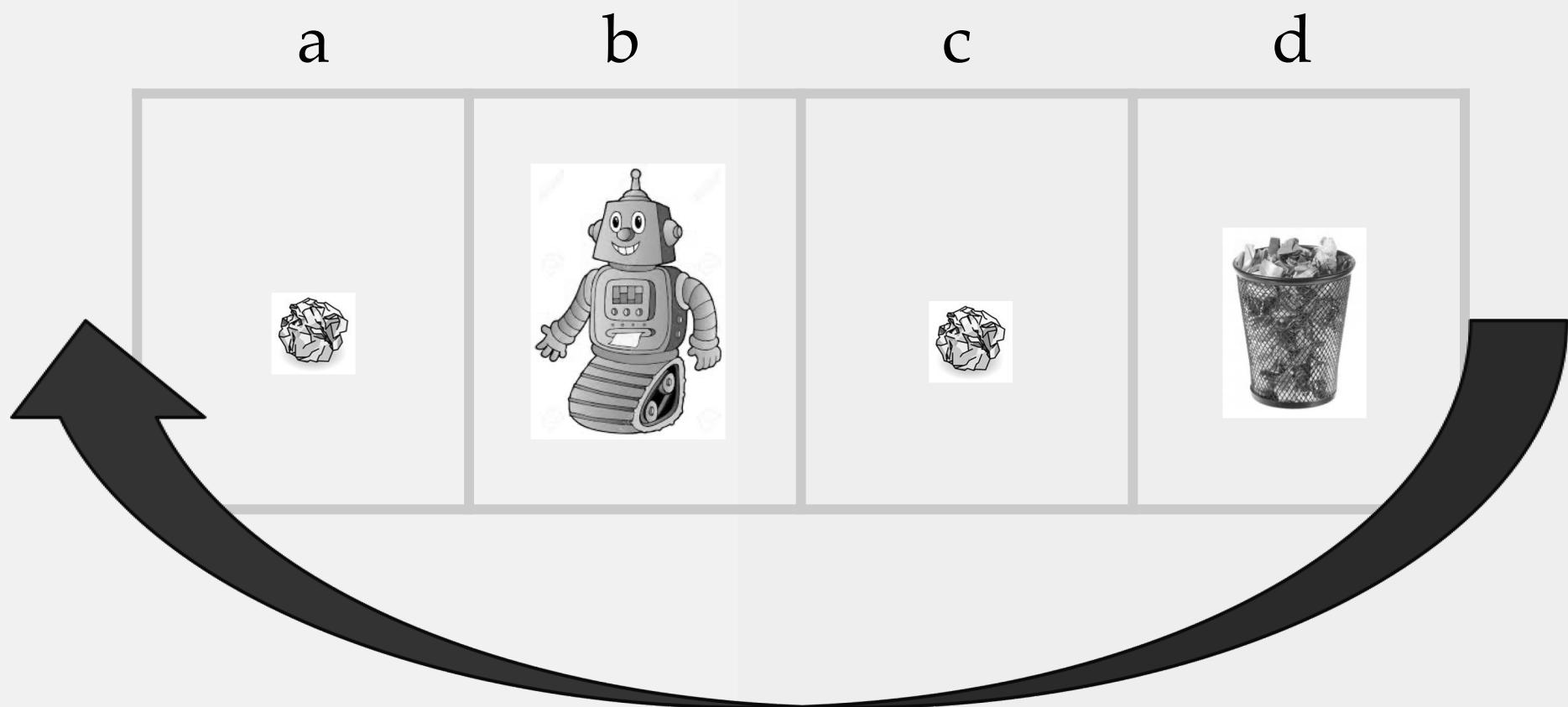
+!localizado(robot, X) : localizado(robot, X) <- .print("Ha llegado a su destino").

+!localizado(robot, X) : localizado(robot, Y) & (not (X=Y)) & adyacente(Y,Z)

```
<- .print("mover de ", Y, " a ", Z);
      -localizado(robot, Y);
      +localizado(robot, Z); // simula el mover de Y a Z
      !localizado(robot, X).
```

# Agente Híbrido SPADE

Agente robot limpiador 2



Entorno circular, con esta nueva situación inicial...

# Agentes Inteligentes (AIN)

Presentación de pyGOMAS

---

- ❖ “**Captura La Bandera**” usando agentes híbridos SPADE.
- ❖ La partida se puede observar a través de visores gráficos que se pueden conectar a un servidor HTTP que se lanza en la aplicación.
- ❖ 2 equipos (**Aliados** y **Eje**) que se enfrentan en un terreno limitado durante un tiempo limitado.
- ❖ Cada equipo tiene su **base**, donde se sitúan inicialmente.
- ❖ Objetivo del juego:
  - ❖ **Aliados**: capturar la bandera y llevarla a su base.
  - ❖ **Eje**: impedir la captura (eliminando todos los aliados o si se agota el tiempo).

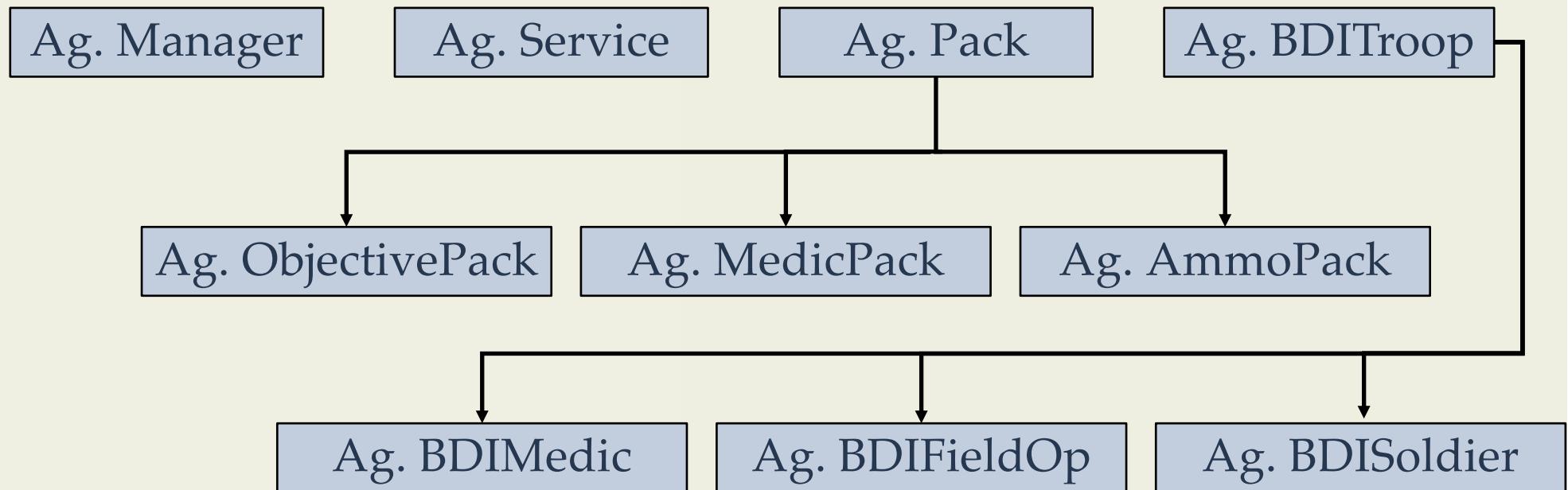
# pyGOMAS

## Descripción - Tropa

- ✿ Llevan un arma de fuego. Probab. de Fallar un disparo por azar: 0.1
- ✿ Inicio de la partida con el máximo de munición (100).
- ✿ Fuego Amigo.
- ✿ Posibles roles (extensible):
  - ✿ **Soldado:** Sus armas hacen el doble de daño. Reciben llamadas de refuerzo.
  - ✿ **Médico:** crean paquetes de medicina que recuperan algo de salud a quien lo coge. Reciben llamadas de asistencia médica.
  - ✿ **Operador de Campo:** crean paquetes de municiones que recuperan algo de munición a quien lo coge. Reciben llamadas de recargo de municiones.
- ✿ Paquetes (**salud** o **munición**) pueden ser cogidos por enemigo.

# pyGOMAS

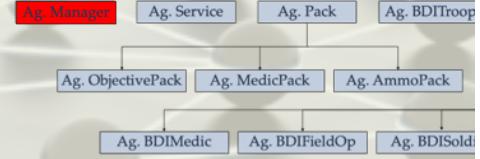
## Descripción - Agentes



# pyGOMAS

Ag. Manager

## Descripción - Agentes

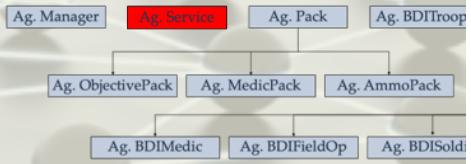


- **Monitoriza y Gestiona** la partida.
- Funciones Principales: **coordinación** y **sincronización** de los otros agentes y la aportación de **información** de lo que está en el **campo de visión** de estos.
- Tareas:
  - **Iniciar la partida.** Mensaje de inicio a los ags. tropa con el mapa del terreno + ubicación inicial de la bandera.
  - **Crear el Agente de Servicios.**
  - **Recibir información de cada agente tropa:** posición, velocidad, orientación, salud y munición.
  - **Servidor de los clientes de visualización.** Enviarles información de la bandera, de los agentes tropa y paquetes.

# pyGOMAS

Ag. Service

## Descripción - Agentes

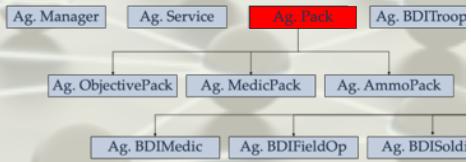


- **Función:** conocer e informar los **servicios** que dan las tropas.
- Al crear un Agente Tropa, se **registra** el **servicio** que ofrece aquí.
- Al morir un Agente Tropa, se informa a este agente para que sepa que ya **no puede ofrecer el servicio** que antes brindaba.
- Sabiendo quiénes son los Agentes Tropa activos, puede responder a **peticiones de solicitud de servicios**.
- Se pueden registrar y posteriormente solicitar servicios nuevos.

# pyGOMAS

Ag. Pack

## Descripción - Agentes

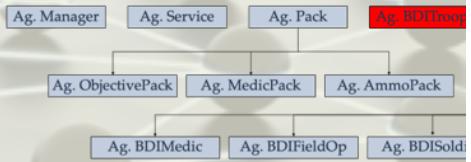


- Los **Agentes Pack** son los paquetes que se crean durante la partida, y pueden ser de tres tipos:
  - **ObjectivePack (id = 1003)**: se crea al inicio del juego por el Agente Manager y representa la **bandera**.
  - **MedicPack (id = 1001)**: creado por los **médicos**.
    - Función: **incrementar la salud** de quien lo reciba en 20 (0..100).
    - Auto-destruye pasado 25 segs. si no es cogido por nadie.
  - **AmmoPack (id = 1002)**: creado por los **operadores de campo**.
    - Función: **incrementar munición** de quien lo reciba en 20 (0..100).
    - Auto-destruye pasado 25 segs. si no es cogido por nadie.

# pyGOMAS

Ag. BDITroop

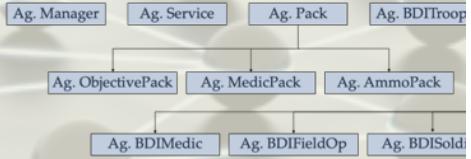
## Descripción - Agentes



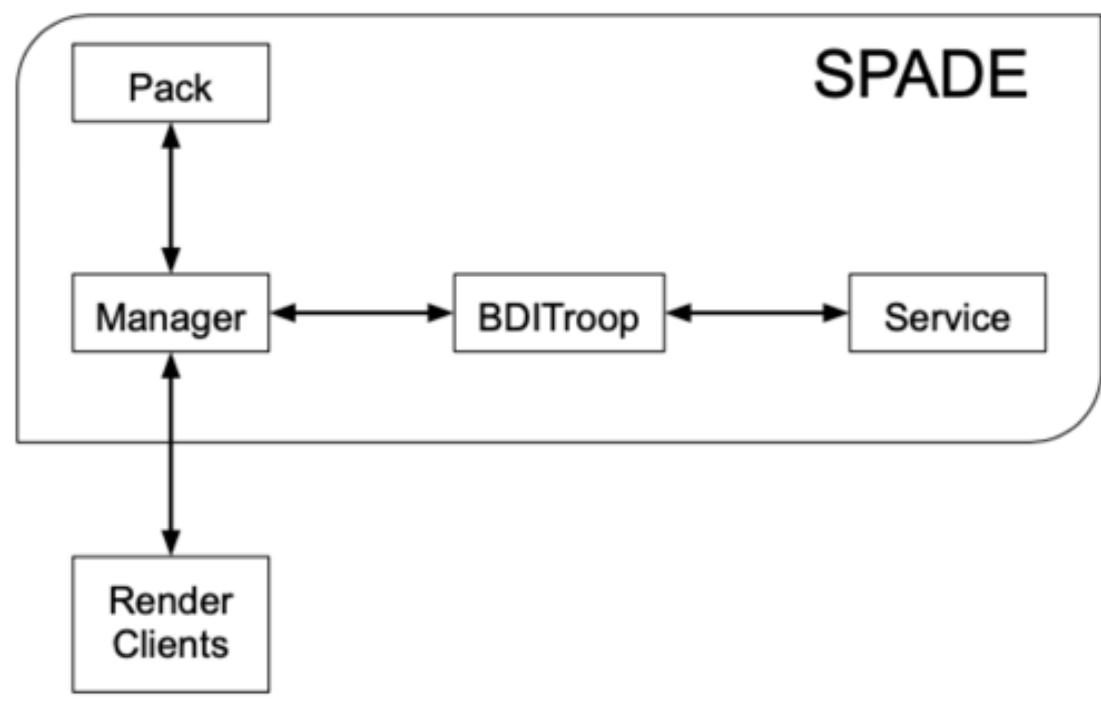
- ❖ Agentes híbridos:
  - ❖ **Capa Reactiva:** acciones de translación, generar puntos de control y disparar entre otras.
  - ❖ **Capa Deliberativa:** ASL + información de las capas reactivas.
  - ❖ Peticiones al **Agente Service:** médicos, operadores de campo y soldados de su equipo disponibles.
- ❖ Tipos de agentes tropa:
  - ❖ **BDIMedic:** servicio médico: crear paquetes de medicina.
  - ❖ **BDIFieldOp:** servicio de recargar municiones: crear paquetes de municiones.
  - ❖ **BDISoldier:** servicio de refuerzo (ir a la posición de un compañero, para reforzar el ataque). Sus disparos hacen el doble de daño.

# pyGOMAS

## Descripción - Agentes



- Agentes heterogéneos:
  - Capacidad de controlar y dirigir.
  - Capacidad de respuesta.
  - Petición de apoyo de soldados.
- Tipos de agentes:
  - **BDIPack:** servicio de medicina.
  - **BDITroop:** servicio de recargar municiones: crear paquetes de municiones.
  - **BDISoldier:** servicio de refuerzo (ir a la posición de un compañero, para reforzar el ataque).



• Puntos de control y respuesta.

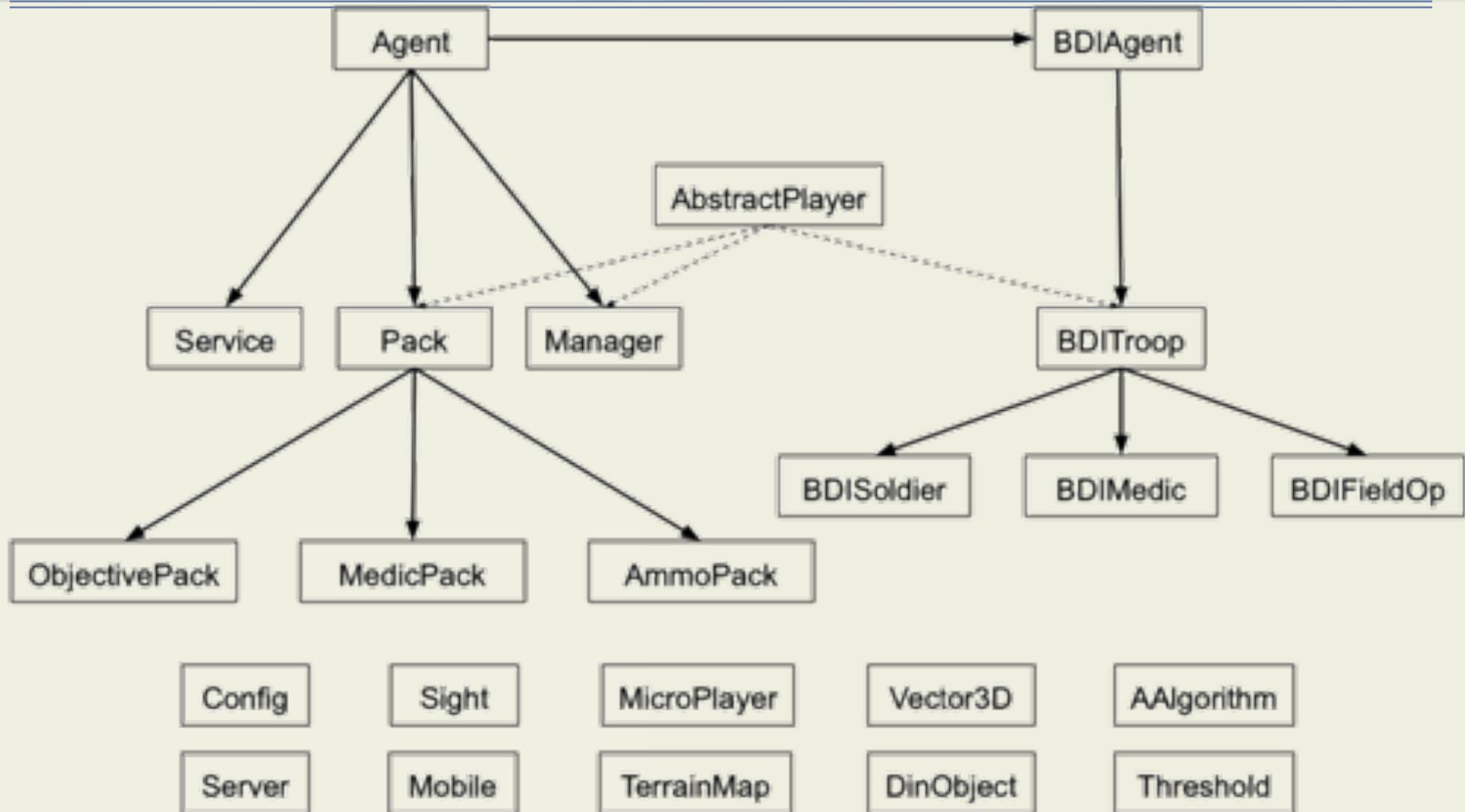
• Puntas reactivas.

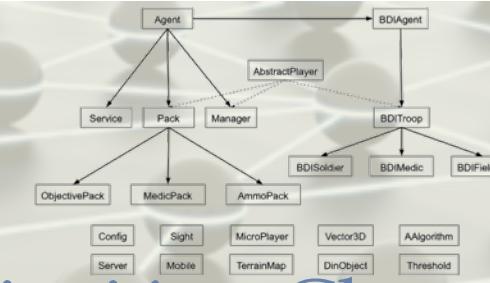
• Unidades de campo y apoyo.

• Medicina.

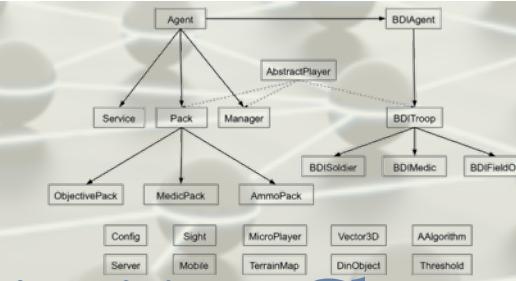
# pyGOMAS

## Descripción - Clases

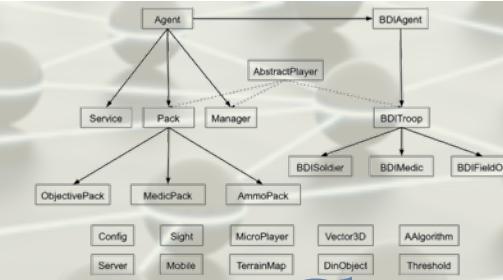




- **Agent**: Es la clase base de agente proporcionada por SPADE.
- **BDIAgent**: agente SPADE híbrido con comportam. deliberativo.
- **AbstractPlayer**: Clase dependiente del juego desarrollado
  - **Atributos**: equipo del agente, nombre (JID) y nombre del Agente de Servicios.
  - **Comportamientos**: registrar y deregistrar servicios.
- **MicroPlayer**: Clase para el Ag. Manager para coordinar la partida.
  - **Atributos de cada Agente Tropa**: nombre, equipo, tipo de agente, posición, velocidad, orientación, salud, cantidad de municiones y si lleva o no la bandera.



- ✿ **DinObject**: Usada por el Manager para representar los paquetes.
  - ✿ **Atributos**: posición, nombre, equipo, tipo, si ha sido tomado o no y quién lo ha tomado en caso afirmativo.
- ✿ **Sight**: Usada por el Manager para saber qué paquete (*DinObject*) o tropa (*MicroPlayer*) hay en el campo de visión de qué agente tropa.
  - ✿ **Información que provee**: tipo de agente, equipo, salud, posición y distancia a la que se encuentra.
- ✿ **Server**: Pone en funcionamiento el servidor que es lanzado por el Manager para que se conecten los visores gráficos.
- ✿ **Threshold**: Atributos **umbrales** de los Agentes Tropa: máximo nº de disparos por ráfaga, de municiones y de salud.
- ✿ **TerrainMap**: Carga el mapa del campo de batalla.



- ✿ **Vector3D**: Clase soporte vectorial.
  - ✿ Creación, suma, resta, normalización, módulo y productos escalares y vectoriales.
- ✿ **Mobile**: Uso de Vector3D para atrib. de mov. de los Ag. Tropa:
  - ✿ **Atributos**: posición, velocidad, orientación y destino.
  - ✿ **Métodos**: cálculo de posiciones, velocidades y orientaciones.
- ✿ **AAlgorithm**: Algoritmo A para mover Ags. Tropa por el mapa.
  - ✿ Heurística modificable (recomendación: distancia Euclidea que permite desplazamientos diagonales).
- ✿ **Config**: Inform. de config. del juego: ruta de los mapas por defecto, ontologías, performativas y valores de precisión.

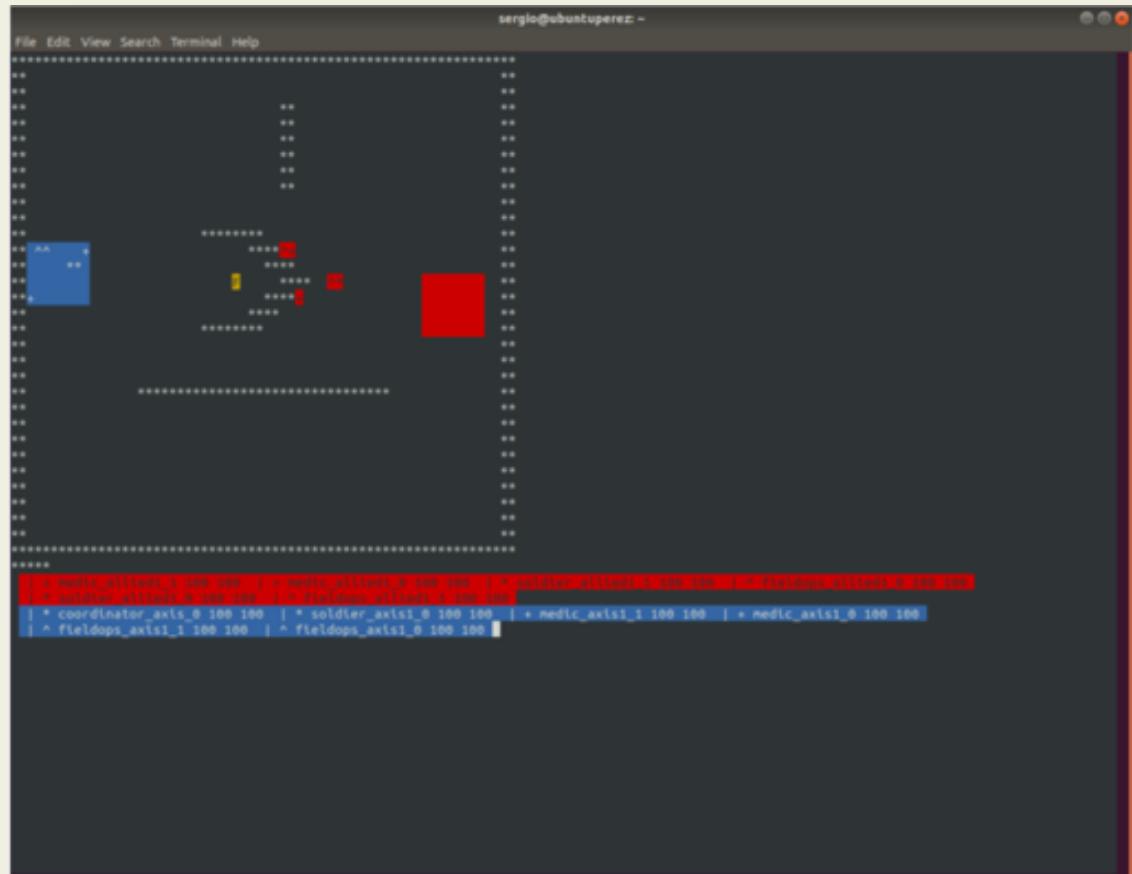
- Ag. Manager crea Serv. para clientes de visores gráficos a los que enviará inform. sobre el estado actual de la partida.
- Para cada Agente Tropa, se enviará la siguiente información:
  - Nombre del Agente Tropa (JID del agente).
  - Tipo (soldado, médico u operador de campo).
  - Equipo al que pertenece.
  - Salud.
  - Número de municiones que lleva.
  - Si lleva la bandera o no.
  - Vectores de posición, velocidad y orientación.
- Para cada paquete, (excepto **bandera** capturada), se enviará:
  - Nombre del paquete (JID del agente).
  - Tipo (bandera, paquete de medicina o munición).
  - Vector de posición.

- ❖ 3 visores gráficos online:
  - ❖ Consola texto.
  - ❖ Consola pygame.
  - ❖ Unity3D.
- ❖ Visor offline:
  - ❖ Consola pygame cargando traza desde fichero.

# pyGOMAS

# Visualización

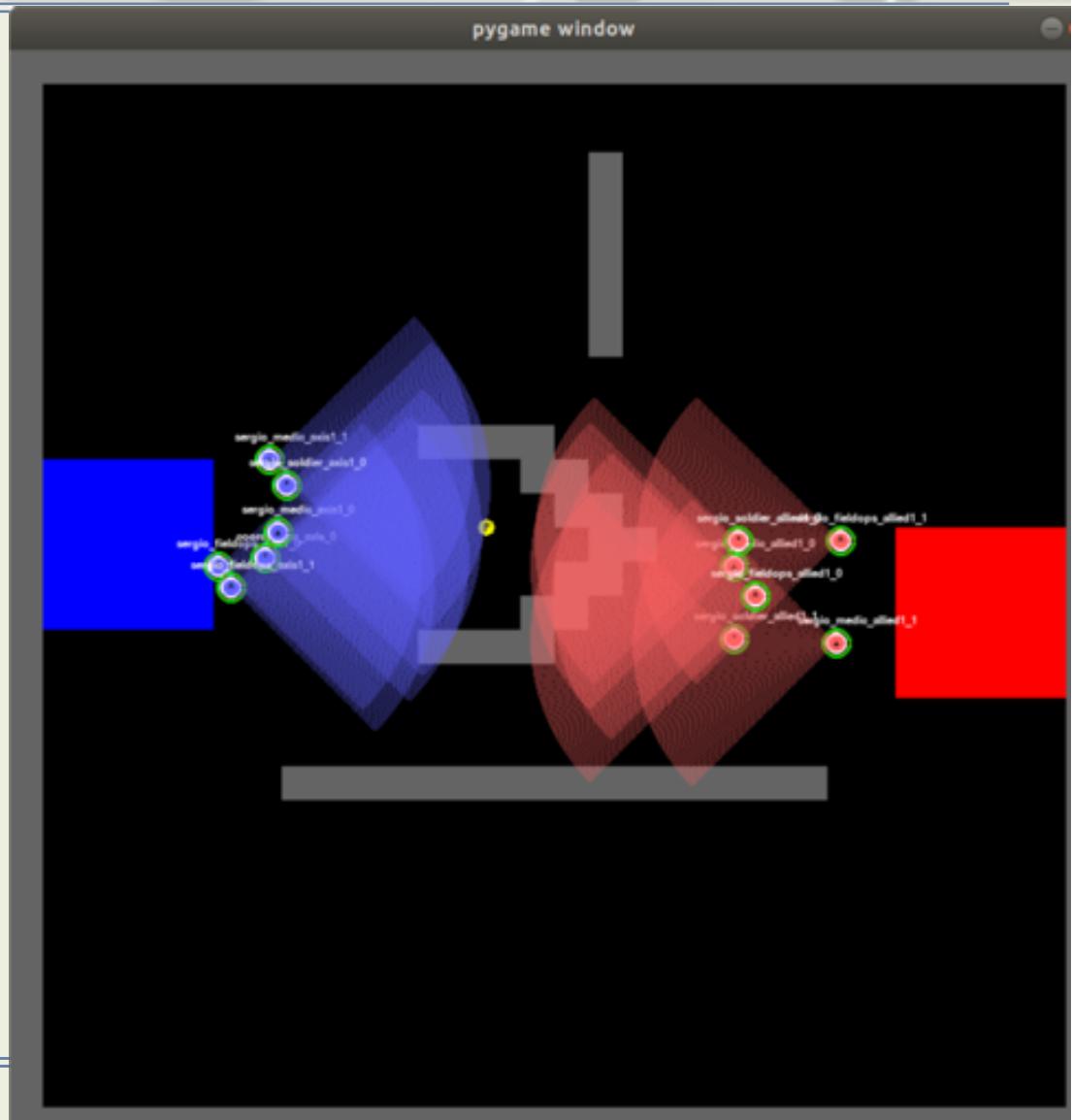
- Consola texto:
    - +: médicos, A: paquetes de municiones, ...
    - Espacios en blanco: pos. sin obst., \* donde sí.
    - Base del Eje: azul, Base Aliada: roja.
    - Cada Ag. Tropa: nombre, salud y n° de munics.



# pyGOMAS

## Visualización

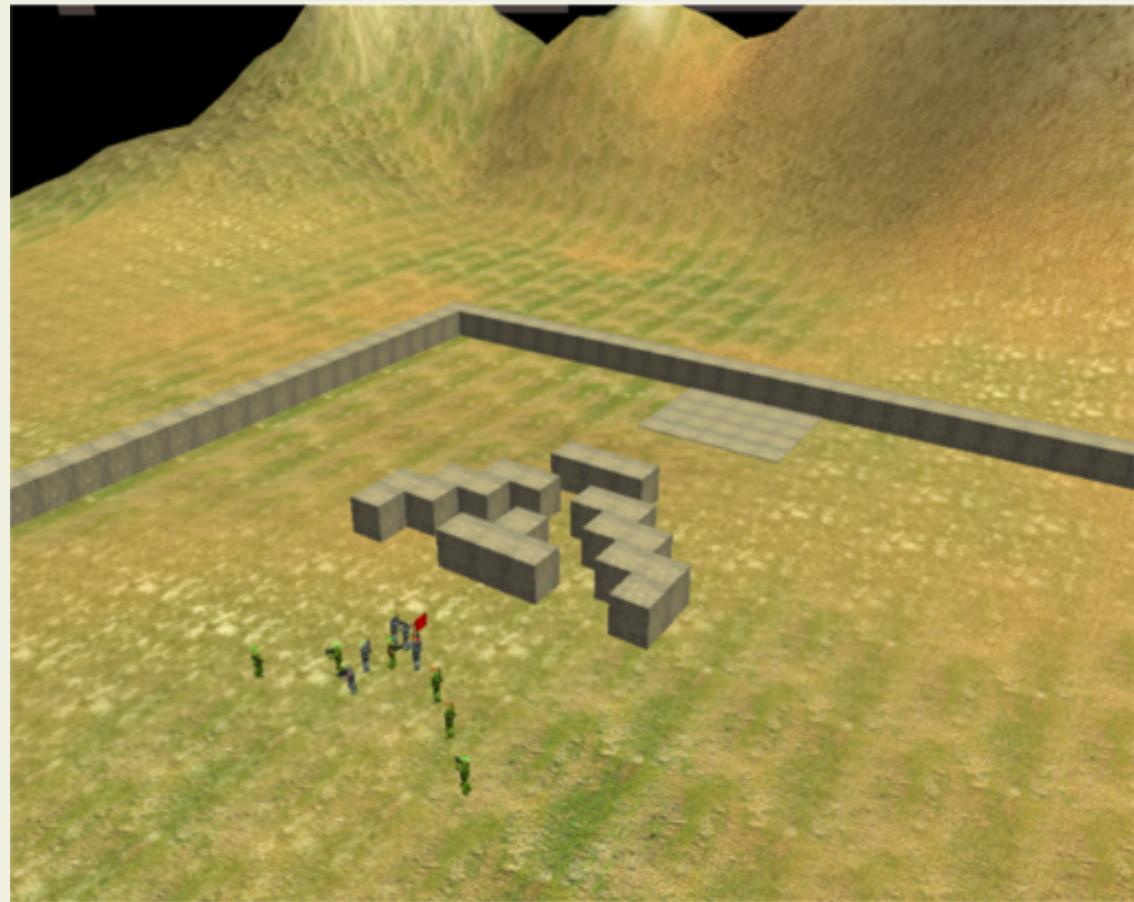
- ❖ Consola pygame:
  - ❖ Campo de visión de cada tropa.
  - ❖ Cursor: mov. de la cámara.
  - ❖ Z, X: Zoom (in / out).
  - ❖ F: Mostrar/ocultar conos visión.



# pyGOMAS

## Visualización

- ❖ Unity3D:
  - ❖ Permite mover la cámara.
  - ❖ H: Ayuda.



# pyGOMAS

# Creación de Mapas

- Mapa: dos ficheros “.txt”:
    - Dimensiones, ubicación de la bandera y bases.
    - Mapa de obstáculos (asteriscos)
  - Movimiento:
    - Comp. Reactivo Periódico.
    - Cola de destinos: puntos del mapa a visitar para llegar a un destino.
    - Alg. JPS (Jump Point Search) para generar camino hacia su destino.
    - Velocidad y Orientación del ag. (cambia su campo de visión).

# pyGOMAS

## Creencias de los Agentes Tropa (I)

- \* **class(X):** X es la clase a la que pertenece el agente:
  - \* NONE = 0, SOLDIER = 1, MEDIC = 2, ENGINEER = 3, FIELOPS = 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).

- \* **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.

- \* **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]).

- ❖ 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.

- ❖ 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.

- **.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

## Implantación

- Instalación:

- GitHub:  
<https://github.com/javipalanca/pygomas>  
python setup.py install
- PyPi:
  - Anaconda: Crear entorno con Python 3.7
  - pip install windows\_curses
  - pip install pygomas

- Lanzar una partida:

- Lanzar el Ag. Manager:

```
pygomas manager -j  
<login_manager>@gtirouter.dsic.upv.es  
-m map_01 -sj  
<login_service>@gtirouter.dsic.upv.es  
-np 6
```

- Lanzar Ags. Tropa:

```
pygomas run -g troops.json
```

- Lanzar Visor(es):

```
pygomas render --text
```

```
pygomas render
```



Usar login como parte  
del nombre de TODOS  
los agentes

# pyGOMAS

## Implantación

### • Instalación:

- GitHub:  
<https://github.com/javipalanca/pygomas>  
python setup.py install
- PyPi:
  - Anaconda: Crear entorno con Python 3.7
  - pip install windows\_curses
  - pip install pygomas

### • Lanzar una partida:

#### • Lanzar el Ag. Manager:

```
pygomas manager -j  
<login_manager>@gtirouter.dsic.upv.es  
-m map_01 -sj  
<login_service>@gtirouter.dsic.upv.es  
-np 6
```

#### • Lanzar Ags. Tropa:

```
pygomas run -g troops.json
```

#### • Lanzar Visor(es):

```
pygomas render --text  
pygomas render
```



# pyGOMAS

## Implantación

- ✿ Fichero JSON con:
  - ✿ **Ags. Tropa**: equipo, nombre, passwd. y rango (soldado, médico, fieldops, ...). Nº ags. de ese tipo a crear + fichero ASL.
  - ✿ **Ags. Manager** y **Service**.
- ✿ La ayuda del juego brinda un fichero JSON para que sirva de ejemplo.
  - ✿ `pygomas help run`

# toyGOMAS



```
{  
    "host": "127.0.0.1",  
    "manager": "cmanager",  
    "service": "cservice",  
    "axis": [  
        {  
            "rank": "BDISoldier",  
            "name": "soldier_axis1",  
            "password": "secret",  
            "asl": "myASL/mybditroop.asl"  
        },  
        {  
            "rank": "BDIMedic",  
            "name": "medic_axis1",  
            "password": "secret",  
            "asl": "myASL/mymedic.asl"  
        },  
        {  
            "rank": "BDIFieldOp",  
            "name": "fieldops_axis1",  
            "password": "secret",  
            "asl": "myASL/myfieldops.asl"  
        }  
}
```

on:

ipo, nombre, passwd. y  
médico, fieldops, ...).  
po a crear + fichero ASL.

Service.

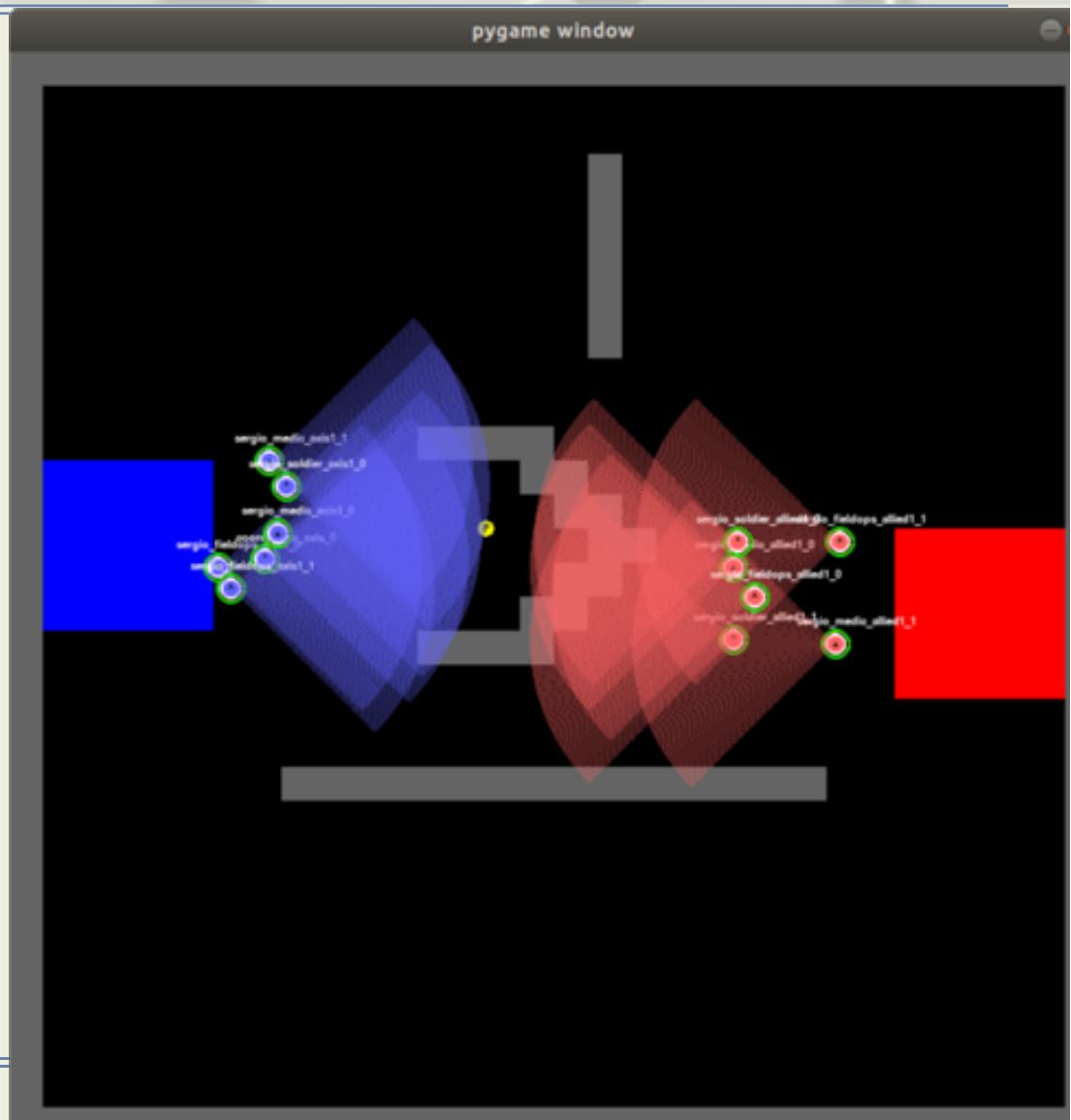
Ejemplo: El  
ego brinda un fichero  
sirva de ejemplo.

run

# pyGOMAS

## Implantación

- ❖ Visualización Offline:
  - ❖ Volcar en un fichero la inform. de la partida:  
`pygomas dump --log  
partida.log`
  - ❖ Visualizar la partida:  
`pygomas replay -game  
partida.log`



# pyGOMAS

## Implantación

- ❖ pygomas\_stats.txt:

- ❖ Generado al acabar la partida en la carpeta del fichero JSON.

```
Winner Team: ALLIED
Duration: [439012h:14m:32s]
Statistics for ALLIED TEAM
-GENERAL:
    * Alive:      3
    * Avg. Health: 74.666666666666667
-OBJECTIVE:
    * Times Taken: 1
    * Times Lost:  0
-SHOTS:
    * EnemyHit:   327
    * TeamHit:    0
    * FailedHit:  23
    * TOTAL:      350
-MEDIC PACKS:
    * Delivered:  0
    * Team Taken: 0
    * Enemy Taken: 0
    * Not Taken:  0
-AMMO PACKS:
    * Delivered:  0
    * Team Taken: 0
    * Enemy Taken: 0
    * Not Taken:  0
-EFICIENCY:
    * Medic:      0
    * FieldOps:   0
    * Army:       1.0
-ANTI-EFICIENCY:
    * Medic:      0
    * FieldOps:   0
    * Army:       0
```

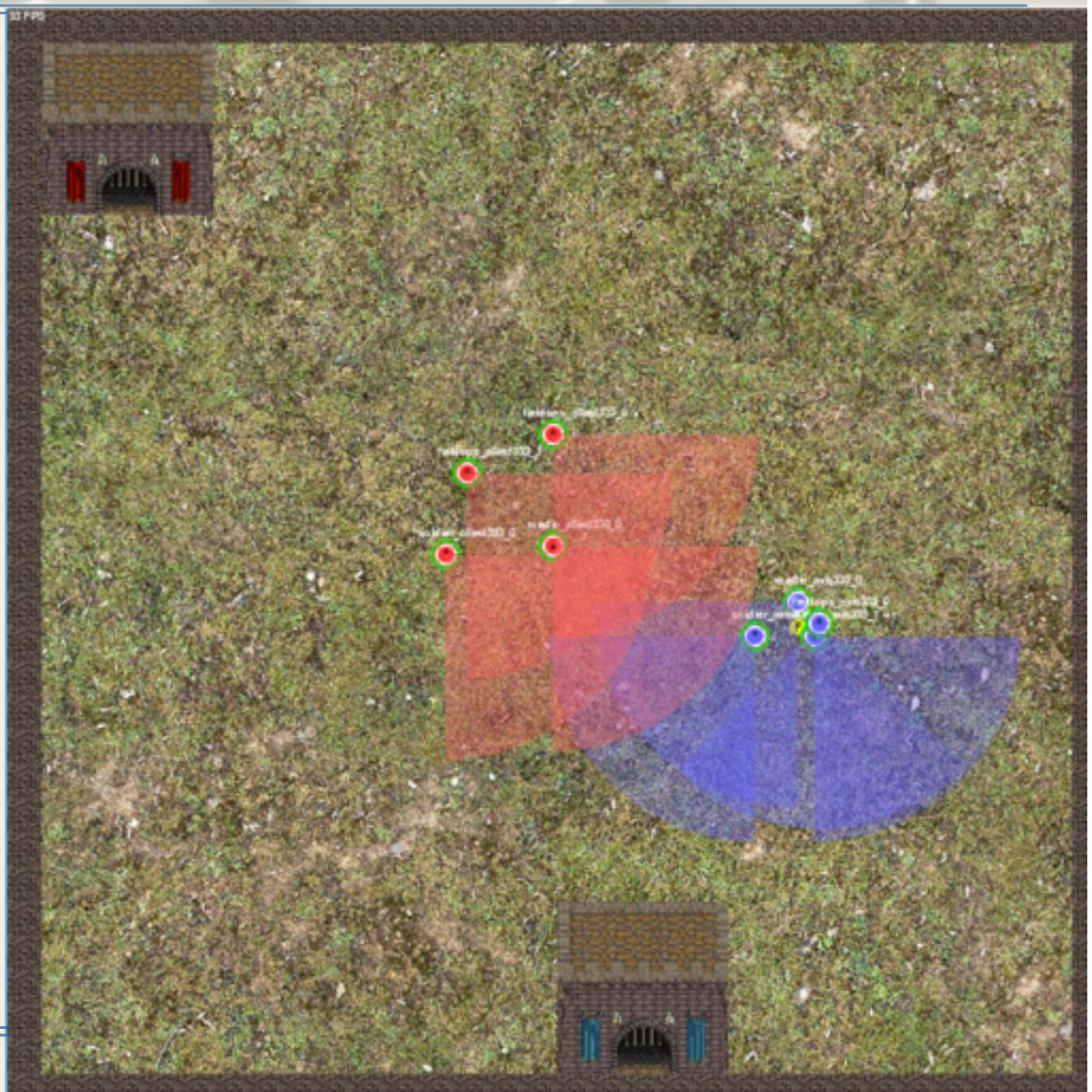
```
Statistics for AXIS TEAM
-GENERAL:
    * Alive:      0
    * Avg. Health: 0
-OBJECTIVE:
    * Times Taken: 318
    * Times Lost:  1
-SHOTS:
    * EnemyHit:   159
    * TeamHit:    0
    * FailedHit:  17
    * TOTAL:      176
-MEDIC PACKS:
    * Delivered:  0
    * Team Taken: 0
    * Enemy Taken: 0
    * Not Taken:  0
-AMMO PACKS:
    * Delivered:  0
    * Team Taken: 0
    * Enemy Taken: 0
    * Not Taken:  0
-EFICIENCY:
    * Medic:      0
    * FieldOps:   0
    * Army:       0.9829545454545454
-ANTI-EFICIENCY:
    * Medic:      0
    * FieldOps:   0
    * Army:       0
```

# pyGOMAS

## Ejemplos de Ejecución

- ❖ Ejecución 1:

- ❖ map\_01
- ❖ 8 soldados
  - ❖ Aliados: 1 soldado, 2 médicos y 1 operador de campo.
  - ❖ Eje: 1 soldado, 2 médicos y 1 operador de campo.



# pyGOMAS

## Ejemplos de Ejecución

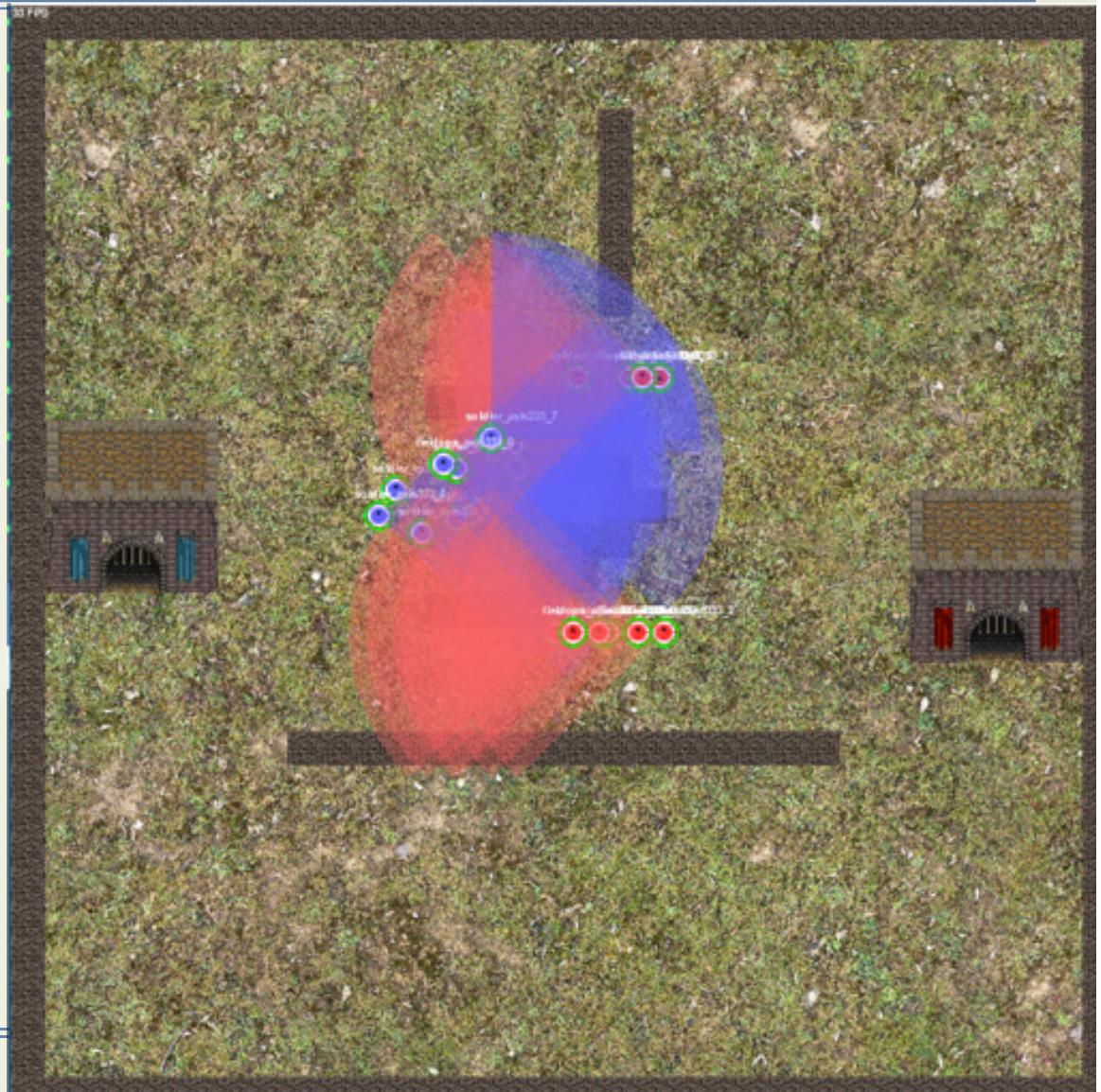
- ❖ Ejecución 2:
  - ❖ map\_04
  - ❖ 12 soldados
    - ❖ Aliados: 4 soldados, 2 médicos y 2 operadores de campo.
    - ❖ Eje: 2 soldados, 1 médico y 1 operador de campo.



# pyGOMAS

## Ejemplos de Ejecución

- ❖ Ejecución 3:
  - ❖ map\_08
  - ❖ 20 soldados
    - ❖ Aliados: 6 soldados, 2 médicos y 2 operadores de campo.
    - ❖ Eje: 8 soldados, 1 médico y 1 operador de campo.



# 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

...

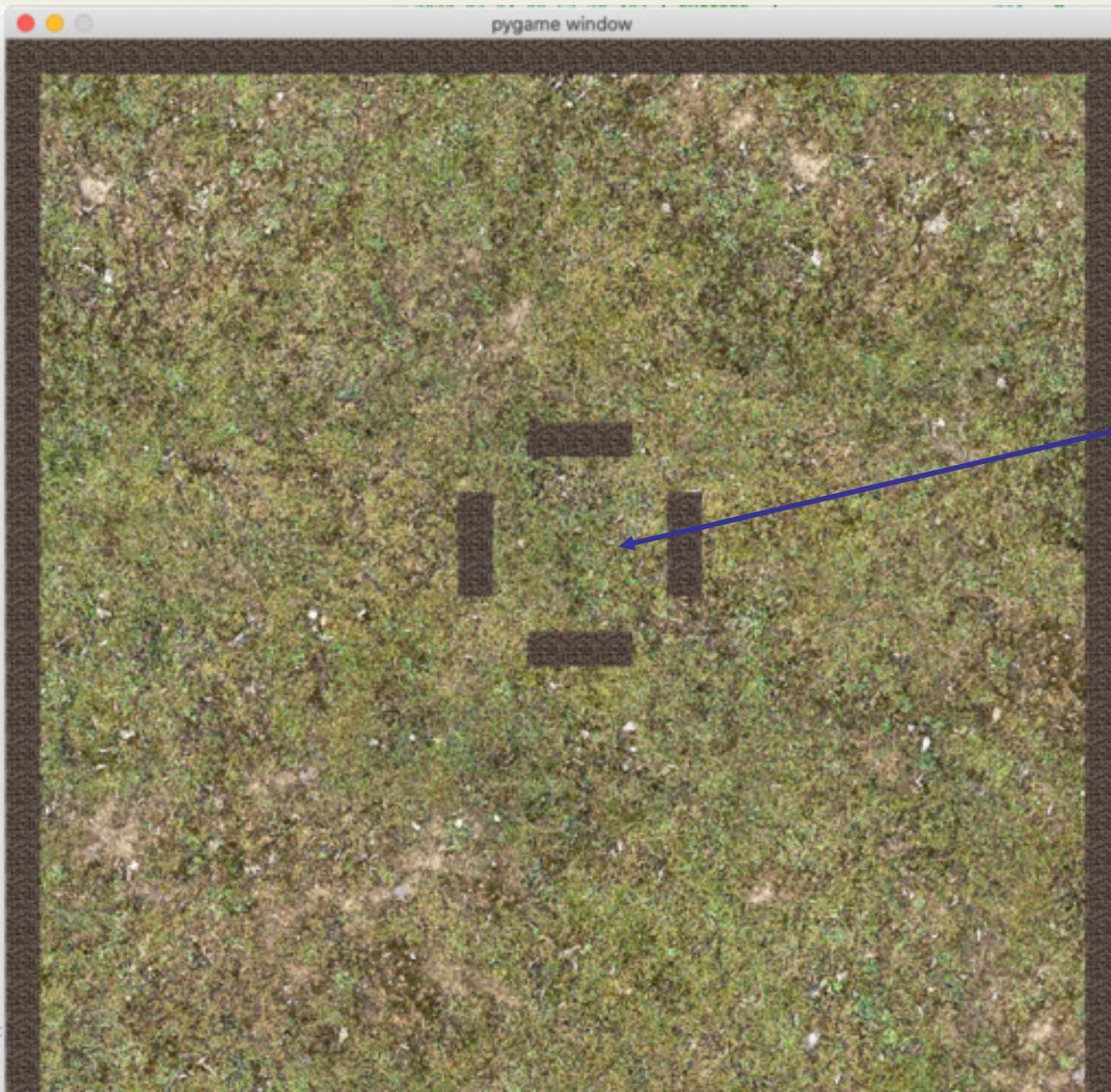
`.superhealth`

- ❖ 1<sup>a</sup> 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

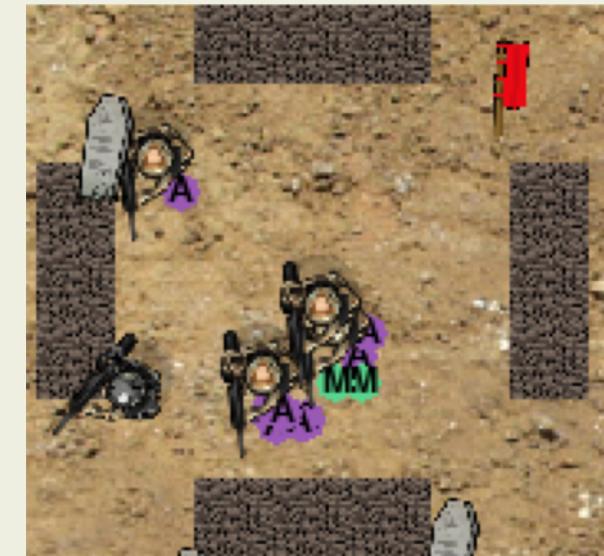
En la zona central se generan paquetes de medicinas y armas

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



# Agentes Inteligentes (AIN)

Práctica 1: Programando mi luchador

---

- ✿ Recordatorio:

**Desarrollar un agente que trate de 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

## Creencias de los Agentes Tropa (I)

### Recordatorio

- ✿ `class(X)`: X es la clase a la que pertenece el agente:
  - ✿ `NONE = 0, SOLDIER = 1, MEDIC = 2, ENGINEER = 3, FIELOPS = 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).

## Creencias de los Agentes Tropa (II)

### Recordatorio

- ✿ **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.

## Creencias de los Agentes Tropa (III)

### Recordatorio

- **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]).

### Recordatorio

- ✿ 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.

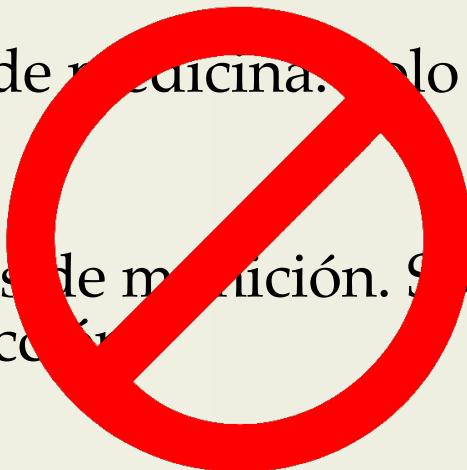
## Acciones de los Agentes Tropa

### Recordatorio

- Envío de mensajes al Service Ag.
  - `.register_service("servicio_a")`: Enviar un mensaje al Service Ag. para registrar un servicio especificado.
  - `.get_medicinas()`: Enviar mens. al Service Ag. para solicitar los medicos de su equipo.
  - `.get_field_soldados()`: Enviar un mensaje al Service Ag. para solicitar los operadores de campo de su equipo.
  - `.get_bacilos()`: Enviar un mensaje al Service Ag. para solicitar los soldados de su equipo.
  - `.get_servicio("servicio_b")`: Enviar un mensaje al Service Ag. solicitando el servicio (distinto de los tres anteriores) a los agentes tropa que lo ofrezcan.

### Recordatorio

- **.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.



- ✿ 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

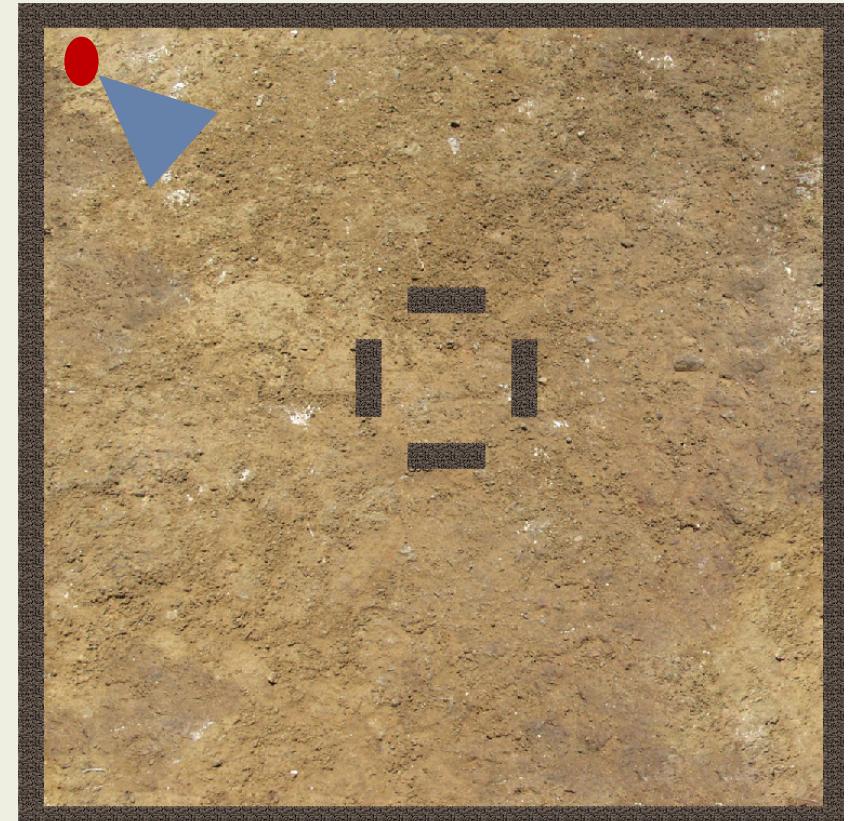
# pyGOMAS



Ejemplo: ir a una esquina

```
+flag(F): team(200)
<-
  +amiesquina;
  .goto([20, 0, 20]);
  +miposicion([20, 0, 20]).
```

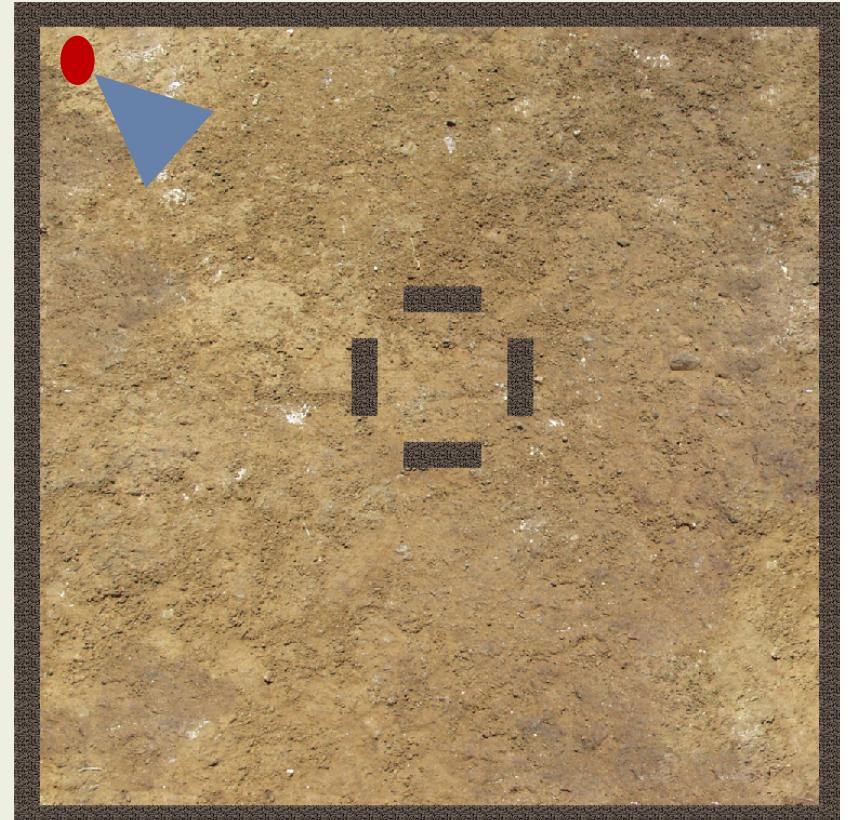
```
+target_reached(T): amiesquina
<-
  -amiesquina;
  .print("Me quedo quieto en pos: ", T);
  ?flag(F);
  .look_at(F);
  -target_reached(T).
```



## Ejemplo: ir a una esquina

A destacar:

- La pared exterior tiene un grosor de 10 puntos en el mapa.  
Ej: `.goto[3, 0, 3]` no funciona
- El ejemplo acude siempre a la esquina superior izquierda, pero un agente nace en cualquier sitio.
  - Mejora: *Ir a la esquina más cercana*



# pyGOMAS



Ejemplo: ir a por paquetes

```
+packs_in_fov(ID,Type,Angle,Distance,Health,Position): Type < 1003
<-
    .goto(Position);
    +aporpaquete.
```

# pyGOMAS



## Ejemplo: girar todo el rato

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

```
<-
```

```
+mirando([[0,0,0],[250,0,0],[250,0,250],[0,0,250]]);
```

```
+estado(0);
```

```
!agirar.
```

```
+friends_in_fov(ID,Type,Angle,Distance,Health,Position)
```

```
<-
```

```
.print("Disparo");
```

```
.shoot(3,Position).
```

```
+!agirar: estado(E) & E<4
```

```
<-
```

```
?mirando(L);
```

```
.nth(E, L, P);
```

```
.look_at(P);
```

```
.wait(1000);
```

```
-estado(_);
```

```
+estado(E+1);
```

```
!agirar.
```

```
+!agirar: estado(E) & E=4
```

```
<-
```

```
-estado(_);
```

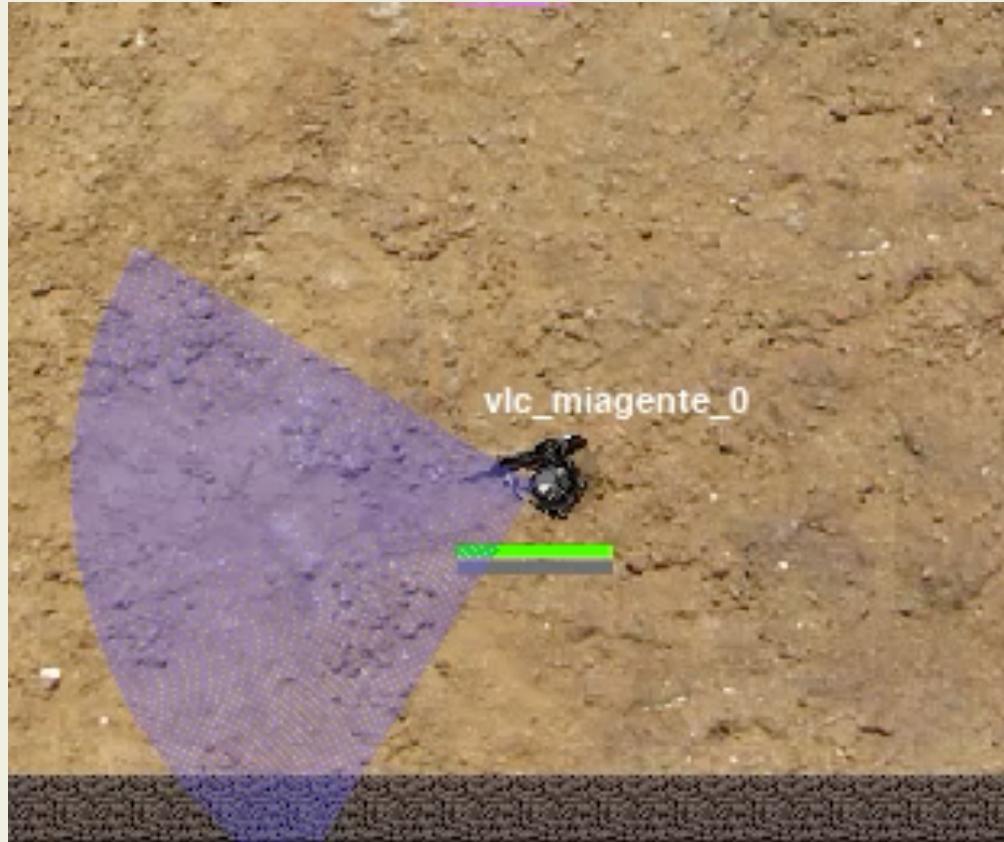
```
+estado(0);
```

```
!agirar.
```

# pyGOMAS



Ejemplo: girar todo el rato



### RECORDATORIO

- ✿ 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~~

Enviad vuestras dudas por:

- Correo: [vjulian@upv.es](mailto:vjulian@upv.es)  
[carrasco@dsic.upv.es](mailto:carrasco@dsic.upv.es)
- o por Poliformat