



Prácticas AIN

pyGOMAS

Sesión 4. Añadiendo acciones internas



Objetivos

- ❖ Conocer como se crean acciones internas en pyGOMAS
 - ❖ Conocer y probar algunos ejemplos
-
-

Acciones Internas

- ❖ Una acción interna estará disponible desde el código JASON y comenzará con un '.'
- ❖ Esta acción corresponderá a un método escrito en python, que pertenecerá a la clase de la que es nuestro agente.
- ❖ Hay dos tipos de acciones a crear:
 - ❖ Procedimientos (sin retorno de valor)

```
@actions.add(".nuevaccion", 0)
def _nuevaaccion(agent, term, intention):
    // añadir código de la acción

    yield
```

- ❖ Funciones (con retorno de valor)

```
@actions.add_function(".nuevaccion", (int, ) )
def _nuevaaccion( x ):
    // añadir código de la acción
    return lo_que_sea
```

Acciones Internas

- ❖ Se debe crear un fichero en python donde se cree una nueva clase (ej: NuevoSoldado) sobre BDITroop en la que se defina lo siguiente:

```
import json
from pygomas.bditroop import BDITroop
from ... // añadir imports necesarios

class NuevoSoldado(BDITroop):
    def add_custom_actions(self, actions):
        super().add_custom_actions(actions)

    @actions.add(".nuevaccion", 0)
    def _nuevaaccion(agent, term, intention):
        // añadir código de la acción
        yield
```

Acciones Internas

- ❖ Para poder ejecutar la nueva acción es necesario modificar el fichero json con la especificación de agentes soldados a lanzar en la partida de forma que tengamos un soldado con la siguiente especificación

```
{  
  "rank": "Soldado.NuevoSoldado ",  
  "name": "nuevo_soldado",  
  "password": "secret",  
  "asl": "codigo_del_soldado.asl",  
  "amount": 1  
}
```

En el código asl podrás usar la nueva acción interna

donde en "rank" hay que indicar la nueva clase de la forma:
nombre_del_fichero.nombre_de_la_clase

Nota: En el ejemplo se supone que el fichero Python es Soldado.py

Ejemplo de creación de una acción interna

Información disponible

- ✧ Información disponible desde Python:
 - ✧ Atributos de AbstractAgent:
 - ✧ `team` : Número que identifica el equipo al que pertenece el agente
 - ✧ `services` : Lista con los identificadores de servicio que ofrece el agente.
 - ✧ Atributos de BDI Troop:
 - ✧ `manager` : jid del Agente Manager.
 - ✧ `service` : jid del Agente de Servicios
 - ✧ `is_objective_carried` : (true / false) indica si lleva la bandera o no
 - ✧ `fov_objects` : lista de objetos actualmente en el campo de visión del agente
 - ✧ `aimed_agent` : agente al que actualmente está apuntando (o None)
 - ✧ `health` : salud actual del agente
 - ✧ `ammo` : munición actual del agente
 - ✧ `is_fighting` : indica si el agente está luchando en este momento (True / False)
 - ✧ `is_escaping` : indica si el agente está escapando en este momento (True / False)

Ejemplo de creación de una acción interna

Información disponible

- ✧ Información disponible desde Python:
 - ✧ Atributos de BDITroop:
 - ✧ Relativas al movimiento:
 - ✧ map
 - ✧ `map.can_walk(X, Z)` : indica si es pisable la posición (X, 0, Z) (True/False)
 - ✧ `map.allied_base.get_init_x()` , `map.allied_base.get_init_y()` , `map.allied_base.get_init_z()`
 - ✧ `map.allied_base.get_end_x()` , `map.allied_base.get_end_y()` , `map.allied_base.get_end_z()`
 - ✧ `map.axis_base.get_init_x()` , `map.axis_base.get_init_y()` , `map.axis_base.get_init_z()`
 - ✧ `map.axis_base.get_end_x()` , `map.axis_base.get_end_y()` , `map.axis_base.get_end_z()`
 - ✧ `velocity_value` : velocidad actual del agente
 - ✧ `destinations` : lista ordenada de los próximos destinos del agente.
 - ✧ `movement`
 - ✧ `movement.velocity.x`, `movement.velocity.y`, `movement.velocity.z`
 - ✧ `movement.heading.x`, `movement.heading.y`, `movement.heading.z`
 - ✧ `movement.destination.x`, `movement.destination.y`, `movement.destination.z`
 - ✧ `movement.position.x`, `movement.position.y`, `movement.position.z`

Ejemplo de creación de una acción interna

Información disponible

- ✧ Información disponible desde Python:
 - ✧ Atributos de BDITroop:
 - ✧ `self.soldiers_count = 0`
 - ✧ `self.medics_count = 0`
 - ✧ `self.engineers_count = 0`
 - ✧ `self.fieldops_count = 0`
 - ✧ `self.team_count = 0`
 - ✧ `threshold = Threshold()` Limits of some variables (to trigger some events)
 - ✧ `threshold.health`
 - ✧ `threshold.ammo`
 - ✧ `threshold.aim`
 - ✧ `threshold.shot`

Ejemplos de creación de una acción interna

Fichero .py (II)

```
@actions.add_function(".close", (tuple, tuple,))
```

```
def _close(enemigo, bandera):
```

```
    """
```

```
    Accion interna que determina si el enemigo (pasado como coordenada) se encuentra cerca  
    de la bandera o no.
```

```
    """
```

```
    return np.sqrt(abs((enemigo[0] - bandera[0]) + (0 - 0) + (enemigo[2] - bandera[2]))) < 5
```

```
@actions.add(".superhealth", 0)
```

```
def _superhealth(agent, term, intention):
```

```
    self.health=200
```

```
    self.bdi.set_belief(HEALTH, self.health)
```

```
    yield
```



Ejemplos de creación de una acción interna

Fichero .py (II)

* drunkenMonkey.py

```
import json
import random
from loguru import logger
from spade.behaviour import OneShotBehaviour
from spade.template import Template
from spade.message import Message
from pygomas.bditroop import BDITroop
from pygomas.bdifieldop import BDIFieldOp
from agentspeak import Actions
from agentspeak import grounded
from agentspeak.stdlib import actions as asp_action
from pygomas.ontology import DESTINATION

from pygomas.agent import LONG_RECEIVE_WAIT

class BDIDrunkenMonkey(BDIFieldOp):
    def add_custom_actions(self, actions):
        super().add_custom_actions(actions)
```

```
@actions.add(".drunkenMonkey", 0)
def _drunkenMonkey(agent, term, intention):
    randX = random.randrange(self.map.get_size_x() - 10)
    randZ = random.randrange(self.map.get_size_z() - 10)
    while (self.map.can_walk(randX, randZ) == False):
        randX = random.randrange(self.map.get_size_x() - 10)
        randZ = random.randrange(self.map.get_size_z() - 10)

    self.movement.destination.x = randX
    self.movement.destination.z = randZ
    self.bdi.set_belief(DESTINATION, tuple((randX, 0, randZ)))
    yield
```



Ejemplo de creación de una acción interna

Fichero .asl

❖ bdifieldop_DM.asl

...

+enemies_in_fov(ID,Type,Angle,Distance,Health,Position)

<-


.drunkenMonkey;

.print("Drunken Monkey Fighting...");

.shoot(3,Position).

Ejemplo de creación de acción interna

Fichero .json



```
{
  "host": "gtirouter.dsic.upv.es",
  "manager": "ccc_m",
  "manager_password": "secret",
  "service": "ccc_s",
  "service_password": "secret",
  "axis": [
    {
      "rank": "BDISoldier",
      "name": "ccc_axis",
      "password": "secret",
      "amount": 8,
      "asl": "bdisoldier.asl"
    },
    {
      "rank": "BDIMedic",
      "name": "ccc_mediac_axis",
      "password": "secret",
      "asl": "bdimedic.asl"
    },
    {
      "rank": "drunkenMonkey.BDIDrunkenMonkey",
      "name": "ccc_DM_fieldop_axis",
      "password": "secret",
      "asl": "bdifieldop_DM.asl"
    }
  ],
  "allied": [
    {
      "rank": "BDISoldier",
      "name": "ccc_allied",
      "password": "secret",
      "amount": 8,
      "asl": "bdisoldier.asl"
    },
    {
      "rank": "BDIMedic",
      "name": "ccc_mediac_allied",
      "password": "secret",
      "asl": "bdimedic.asl"
    },
    {
      "rank": "BDIFieldOp",
      "name": "ccc_fieldop_allied",
      "password": "secret",
      "asl": "bdifieldop.asl"
    }
  ]
}
```

- ✦ Fichero JSON:
Añadimos un nuevo tipo de agente que incorpora la acción comentada.

Ejemplo de creación de una acción interna

Más ejemplos

- ✧ Github de pygomas

- ✧ <https://github.com/javipalanca/pygomas>
- ✧ En pygomas/bditroop.py

- ✧ Github de Python-agentspeak

- ✧ <https://github.com/niklasf/python-agentspeak/tree/master/agentspeak>
- ✧ En agentspeak/stdlib.py

Recordatorio: Trabajo a Realizar

Diseñar e implementar **un equipo de 10 agentes** con la distribución de tipos que deseéis (médicos, soldados y fieldops) para jugar a **capturar la bandera** en un mapa cualquiera como **atacante o como defensor**.

Es **necesario** realizar trabajo en los siguientes aspectos:

- 1.**Coordinación vía paso de mensajes** entre agentes del mismo equipo. (30%)
- 2.**Servicios nuevos:** se debe incluir algún servicio nuevo por parte de un agente y el uso del mismo por parte de otros agentes. (30%)
- 3.**Comportamientos internos de los agentes:** Se deben realizar mejoras de **comportamientos** existentes (por ej. tratar de evitar el fuego amigo). (30%)
- 4.**Acción Interna (opcional):** se debe incluir al menos una nueva acción interna en Python. (10%)

Nota máxima: 3

Plazo: 6 (grupo Tarde) y 7 de mayo (grupo Mañana)

Recordatorio: Trabajo a Realizar

Entrega (**tarea en Poliformat**):

Ficheros *.asl y *.py desarrollados, así como el fichero json preparado para lanzar a los agentes del equipo.

IMPORTANTE: los nombres de vuestros agentes deben incorporar vuestro login para diferenciarlos del resto

El código, comentado y documentado debe seguir unas mínimas normas de estilo: tabulado y comentado.

Comprimir todo el directorio en un fichero <nombre_equipo>.zip

Pequeña memoria, indicando las principales ideas de mejora aplicadas al equipo, así como unas breves conclusiones sobre los resultados obtenidos.
