

DOCUMENTACIÓN FRAMEWORK

Índice

DOCUMENTACIÓN FRAMEWORK.....	1
Índice.....	2
Descripción.....	3
InsultBot.....	3
OracleBot.....	3
TNTBot.....	3
Directorios/ficheros relevantes.....	4
1. main.py.....	4
2. manager.py.....	4
3. base_agent.py.....	4
4. event_observer.py.....	5
5. action.py.....	5
6. Carpeta core/Minecraft/.....	5
7. Carpeta agents/.....	5
Instalación.....	6
1. Requisitos.....	6
2. Entorno Virtual.....	6
Uso del framework.....	7
Cómo crear un nuevo agente.....	7
Pruebas y cobertura de código.....	8
Autor.....	8

Descripción

Este framework permite la creación y ejecución de agentes en un servidor de Minecraft. Los agentes pueden moverse, interactuar con el entorno, construir y destruir bloques, y comunicarse a través del chat del juego. Incluye varios agentes predefinidos, como:

[InsultBot](#)

Insulta jugadores en el chat.

Comandos disponibles

insultbot insult [player]	# Insulta a un jugador.
insultbot help	# Muestra los comandos disponibles.

[OracleBot](#)

Responde preguntas con respuestas predefinidas.

Comandos disponibles

oraclebot [pregunta]	# Responde a preguntas predefinidas.
oraclebot add info: pregunta -> respuesta	# Agrega nueva información.
oraclebot remove info: pregunta	# Elimina una respuesta.
oraclebot help	# Muestra los comandos disponibles.

[TNTBot](#)

Coloca y detona TNT en el juego.

Comandos disponibles

tntbot add tnt	# Coloca TNT cerca del jugador.
tntbot fire tnt	# Detona TNT en un radio de 3 bloques.
tntbot line tnt [n]	# Crea una línea de TNT y la detona.
tntbot help	# Muestra los comandos disponibles.

El framework permite a los usuarios crear sus propios agentes personalizados.

Directorios/ficheros relevantes

1. main.py

Propósito

Es el punto de entrada del framework.

Crea una instancia de Manager y llama a su método run() para gestionar los agentes.

Funcionamiento

Llama a Manager.run(), lo que pone en marcha el sistema de detección de eventos en el chat.

2. manager.py

Propósito

Gestiona la carga y ejecución de los agentes.

Escucha eventos del chat y los distribuye a los agentes adecuados.

Funcionamiento

1. Carga dinámica de agentes

Busca archivos en la carpeta agents/ y los importa dinámicamente.

Identifica clases que hereden de BaseAgent y las instancia.

2. Recepción y distribución de eventos

Escucha mensajes en el chat del servidor.

Cuando detecta un mensaje, notifica a los agentes correspondientes.

Si un mensaje menciona a un agente, se activa su método update(event).

3. base_agent.py

Propósito

Define la clase base de todos los agentes.

Proporciona acceso al entorno de Minecraft y al chat.

Funcionamiento

Al instanciar un agente, este recibe:

Environment → Permite modificar el mundo de Minecraft.

Movement → Facilita el movimiento del agente.

Chat → Le permite enviar y recibir mensajes.

Todos los bots (InsultBot, TNTBot, etc.) heredan de esta clase.

4. event_observer.py

Propósito

Define una interfaz para los agentes que reaccionan a eventos.

Permite implementar el patrón Observer (observador de eventos).

Funcionamiento

Define el método abstracto `update()`, que cada agente debe implementar.

Cuando un evento ocurre, Manager llama a `update(event)` en los bots que corresponda.

5. action.py

Propósito

Define una interfaz (Action) para los agentes que ejecutan acciones en el mundo.

Similar a EventObserver, pero enfocada en acciones explícitas en lugar de eventos pasivos.

Funcionamiento

Los bots que realizan acciones periódicas implementan `run()`, por ejemplo, TNTBot colocando TNT.

6. Carpeta core/Minecraft/

Propósito

Contiene módulos que permiten la interacción con Minecraft. Facilita a los agentes la modificación del mundo de Minecraft.

Contenido

`environment.py` → Maneja el mundo del juego (bloques, posiciones).

`chat.py` → Permite enviar y leer mensajes del chat.

`movement.py` → Controla el desplazamiento de los agentes.

7. Carpeta agents/

Propósito

Contiene los diferentes agentes implementados en el framework.

Instalación

1. Requisitos

- Minecraft 1.12
- Servidor Adventures in Minecraft:
[GitHub - Adventures in Minecraft] (<https://github.com/AdventuresInMinecraft>)
- Python 3.8 o superior
- Librerías necesarias (se instalan con pip)
 - mcpi
 - coverage

2. Entorno Virtual

Crear el entorno.

Windows (cmd/powershell)

```
python -m venv minecraft-env
```

Linux/Mac (bash)

```
python3 -m venv minecraft-env
```

Activar el entorno.

Windows (cmd/powershell)

```
minecraft-env\Scripts\activate
```

Linux/Mac (bash)

```
source minecraft-env/bin/activate
```

Sabrás que está activado porque verás (minecraft-env) al inicio de la línea de comandos.

Una vez activado el entorno virtual, tienes que instalar las librerías mencionadas anteriormente:

```
pip install mcpi
```

```
pip install coverage
```

Uso del framework

Para iniciar el framework y ejecutar los agentes:

```
python minecraft-agent-framework/main.py
```

Los agentes responderán automáticamente a eventos en el chat del juego.

Cuando quieras que finalice, solo tienes que hacer "Ctrl + C" desde el terminal.

[Cómo crear un nuevo agente](#)

Puedes añadir un nuevo bot creando un archivo en la carpeta agents/.

Ejemplo: Crear un agente "HelloBot"

```
"""Imports necesarios"""

from core.base_agent import BaseAgent
from core.event_observer import EventObserver


class HelloBot(BaseAgent, EventObserver):

    """Asignar un nombre al agente"""

    def __init__(self, name="HelloBot"):
        super().__init__(name)

    """Iniciar el agente"""

    def run(self):
        self.chat.send_message("¡HelloBot listo para saludar!")

    """Se activa el agente para ejecutar"""

    def update(self, event):
        if hasattr(event, 'message') and isinstance(event.message, str):
            message = event.message.lower()
            if "hello" in message:
                self.chat.send_message("¡Hola, aventurero de Minecraft!")
```

¡Guarda el archivo y ejecuta main.py!

Pruebas y cobertura de código

Este proyecto incluye pruebas unitarias y usa GitHub Actions + Codecov para medir la cobertura de código.

Al realizar un pull/push se ejecutan los test automáticamente.

También se pueden ejecutar los tests manualmente:

```
coverage run -m unittest discover tests
```

```
coverage report -m
```

Para ver un reporte en HTML:

```
coverage html
```

Esto generara la carpeta htmlcov/. Para ver los resultados simplemente tienes que abrir index.html

Autor

Judith García Santiago