TAP:
Minecraft Agent Framework

David Lluis Vidal

12/01/2025

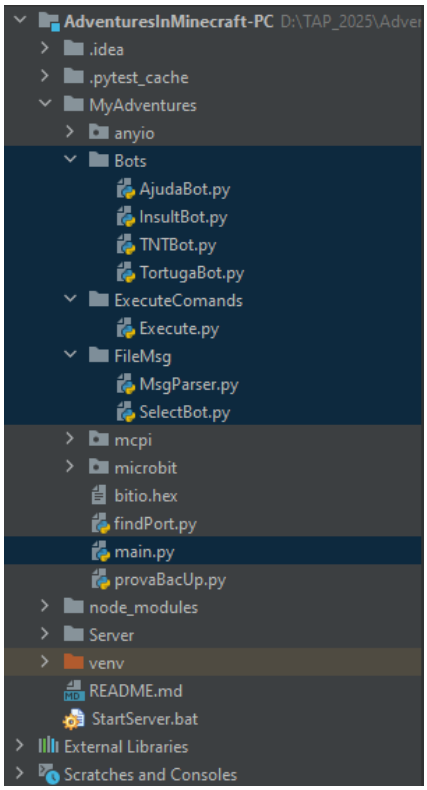
Índex

Introducció

En aquesta practica se'ns demanava la creació amb Python d'un seguit de Agents que fossin capaços d'interactuar amb el mon de Minecraft.
Amb aquesta practica es volia demostrar que senten com implementar un Framework a Python que ens permetes crear i afegir bots de forma fàcil i eficient. També es volia demostrar el correcte us i funcionament dels conceptes de *Funtional programing* i de *Reflective programing*. Finalment es volia incorporar l'us de *Github Actions* amb fi de tenir un bon ***Code coverage*** dintre d'un repositori *git*.

Estructura

Pel que respecta a la estructura hem usat el repositori *AdventuresInMinecraft* com a base i s'ha afegit el codi pertinent:



Com es pot veure a l'imatge s'han creat els directoris **Bots**, **ExecuteComand** i **FileMsg,** també s'ha fet un main per a poder fer us d'aquesta lògica.

Bots: directori on es guarda el codi dels bots.

- AjudaBot.py: Bot que et dona un llistat de bots.
- InsultBot.py: Bot que et retorna un insult.
- TNTBot.py: Bot que col·loca un bloc de TNT i el fa explotar.
- TortugaBot.py: Bot usat per dibuixar un quadrat (Usa la classe tortuga la qual ens permet construir amb un punter).

ExecuteComands:

- Execute.py: Funció que usa *Reflective programing* per a comprovar l'existència dels bots i comprovar si son executables.

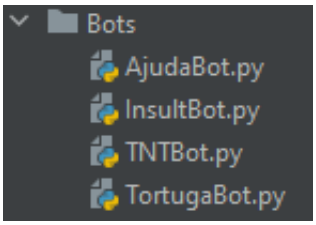
FileMsg:

- MsgParser.py: Funcions que usen *Functional programing* per a poder controlar i guardar la entrada del chat.
- SelectBot.py: Funció que mitjançant un diccionari ens retorna el bot al que s'està cridant.

Desenvolupament

Bots

S'ha decidit crear 4 bots que ens permetin interactuar amb el mon:



AjudaBot.py:

Es el bot mes simple de tots ja que sol mostra informació sobre la resta de bots i com crida los.

```
def main(mc):
    mc.postToChat("<Bot>: Avaliable commads:")
    mc.postToChat("  - bot tnt: Place and activate a tnt near the player.")
    mc.postToChat("  - bot insultame: Gets a random slur from the bot.")
    mc.postToChat("  - bot tortuga: Draws a square using the internal class turtle.")
    mc.postToChat("  - bot ayuda: Show tShis list of commands.")
```

InsultBot.py:

Aquest bot escull un insult aleatori de dintre d'una llista i el mostra pel chat de joc:

```
import random

# List with some mineSpanish slurs
slurs = [
    "¡Eres más lento que un creeper cansado!",
    "¡Tu casa parece hecha por un enderman ciego!",
    "¡Ni un aldeano querría comerciar contigo!",
    "¡El dragón del End se ríe de tus habilidades!",
    "¡Hasta un bloque de tierra tiene más estilo que tú!",
]


# Function that activate slurs
Davidfaena
def main(mc):
    slur = random.choice(slurs)
    mc.postToChat(f"<Bot>: {slur}")
```

TNTBot.py

Bot que ens permet col·locar un bloc de TNT dos blocs d'avant del jugador i que posteriorment l'activa.

Com a modificació es podria fer que el usuari tries les coordenades d'on col·locar el bloc de TNT.

```python
from MyAdventures.mcpi import block
import time

# Function that places some tnt

Davidfaena
def main(mc):

    # Get player location
    pos = mc.player.getTilePos()

    # Coords to place the tnt nest to the player
    x, y, z = pos.x + 2, pos.y, pos.z

    mc.setBlock(x, y, z, block.TNT)
    mc.postToChat("¡TNT placed!")

    time.sleep(2)

    mc.setBlock(x, y+1, z, block.REDSTONE_BLOCK)
    mc.setBlock(x, y+1, z, block.AIR)

    mc.postToChat("¡TNT activated!")
```

TortugaBot.py

Bot que usant la classe *MinecraftTurtle* ens permet generar formes, en aquest cas com a prova genera un quadrat a màxima velocitat.

Com a modificació es podria fer que el usuari tries la velocitat de construcció.

```python
from MyAdventures.mcpi.minecraftstuff import MinecraftTurtle

Davidfaena
def main(mc):

    # Create a turtle instance
    tortuga = MinecraftTurtle(mc, mc.player.getTilePos())

    # Draws a square
    for _ in range(4):
        tortuga.speed(10)
        tortuga.forward(10)
        tortuga.right(90)
```

FileMsg

En aquest directori es guarda tota la lògica que tracta la entrada del text pel chat del joc.

MsgParser.py

En aquest fitxer trobem la lògica que ens permet separar el text en brut del bot que es vol executar, a fi de fer aquesta separació i posterior crida s'han fet dues versions que, de forma latent, es queden escoltant el chat del joc i quan detecten alguna coincidència executen el bot.

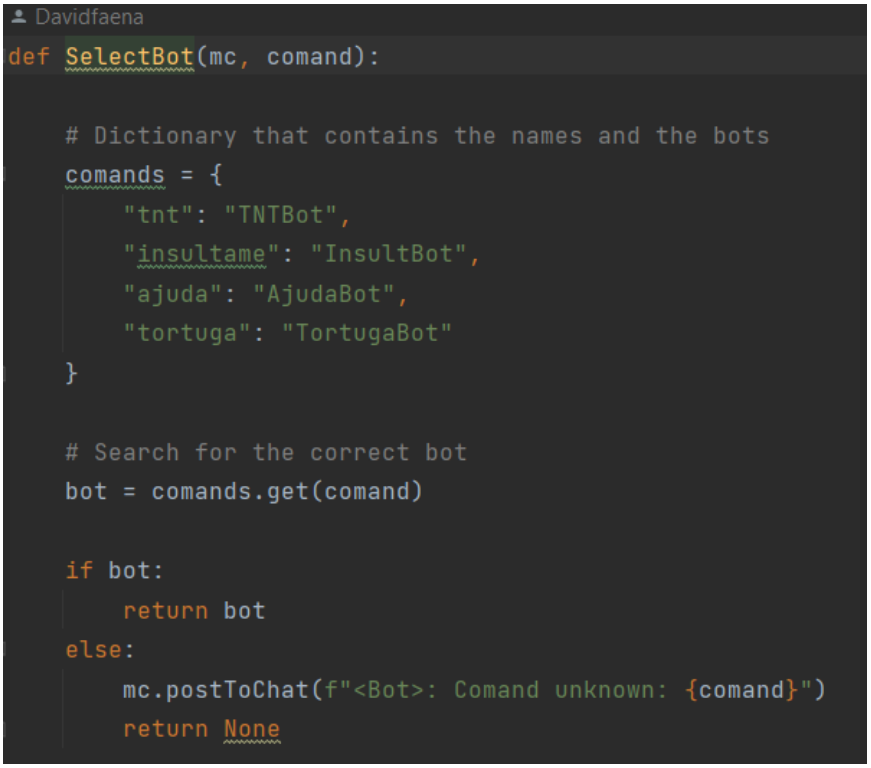
```python
from MyAdventures.ExecuteComands.Execute import apply_command
import time


# Function that listen all the msg
Davidfaena
def listener(mc):...

# Function that listen all the msg
Davidfaena
def filter_message(mc):

    while True:
        mensajes = mc.events.pollChatPosts()
        list(map(lambda mensaje: message_Parser(mc, mensaje), mensajes))
        time.sleep(0.1)

# Functional programing: proces the chat messages
Davidfaena
def message_Parser(mc, msg):

    content = msg.message.lower()

    if content.startswith("bot "):
        comand = content[4:]
        mc.postToChat(comand)
        apply_command(mc, comand)
```

- Listener: Funció que escolta el chat del joc i fa la crida al bot, rep l'element mc al qual estan lligades totes les crides del framework.

- filter_message: Té el mateix funcionament que la funció anterior amb l'única diferencia que usar *Functional programing* en aquest cas usem l'element map per poder fer la crida a la funció message_parser, usem la lambda com a adaptador per al map d'aquesta manera es pot iterar sobre la llista i obtenir l'últim element.
- Message_Parser: Aquesta funció ens permet extreure el nom del bot i cridar a la seva execució.

SelectBot.py

Funció que ens retorna el nom del fitxer que conte el bot, això es fa mitjançant un directori que mapeja el nom del bot amb el nom del seu fitxer.



ExecuteComands

Dintre d'aquest directori trobem l'arxiu el qual s'encarrega de fer totes les crides als bots usant *Reflective programing*.

Execute.py

Aquesta funció ens permet fer una importació dinàmica dels bots que es necessita usar, posteriorment comprova que la existència de una funció comú, en aquest cas li hem dit main, usem la funció getattr per guardar la funció i posteriorment saber si la funció es executable, si ho es executa el bot.

```python
from MyAdventures.FileMsg.SelectBot import SelectBot
import importlib


# Reflective function that let us call the differents bots
Davidfaena
def apply_command(mc, comand):

    bot = SelectBot(mc, comand)  # Name of the bot to be called

    try:
        # Complet name from the module
        modulo_nombre = f"MyAdventures.Bots.{bot}"

        # Dinamic import of the module
        modulo = importlib.import_module(modulo_nombre)

        # Verifies if the module has a comun function name
        if hasattr(modulo, "main"):  # It is assume that all the bots have the same function name
            comando_func = getattr(modulo, "main")

            if callable(comando_func): # Check if we can call that function
                mc.postToChat(f"Executing comand: {bot}")
                return comando_func(mc)  # Executes the main function of the bot
            else:
                mc.postToChat(f"<Bot>: Main function isn't executable {bot}.")
        else:
            mc.postToChat(f"<Bot>: Module {bot} dosent have a main function.")

    except ModuleNotFoundError:
        mc.postToChat(f"<Bot>: Módulo not found: {bot}.")
    except Exception as e:
        mc.postToChat(f"<Bot>: Error while executing {bot}: {str(e)}")

    return None
```

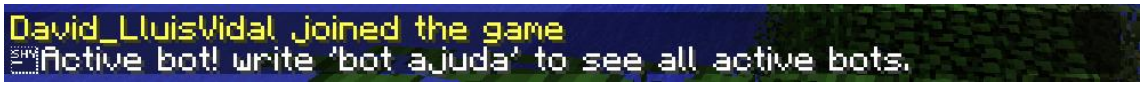
Main.py

S'ha intentat fer el main mes simple possible i d'aquesta manera al main sol es crea/obre la connexió amb el servidor de minecraft, es mostra un missatge a fi que es usuaris tinguin present que els bots estan actius i després es crida a *filter_message,* el qual es queda escoltant i executant els bots.

```python
from mcpi.minecraft import Minecraft
from MyAdventures.FileMsg.MsgParser import listener, filter_message


# Opens a conection to the minecraft server
mc = Minecraft.create()


# Initial message
mc.postToChat("¡Active bot! write 'bot ajuda' to see all active bots.")

# Proceeds to listen the chat log
filter_message(mc)
```

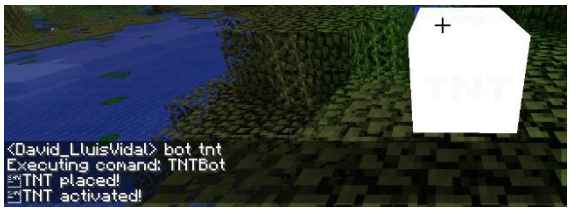
Joc de proves

Activacio del framework que escolta per cridar els bots:



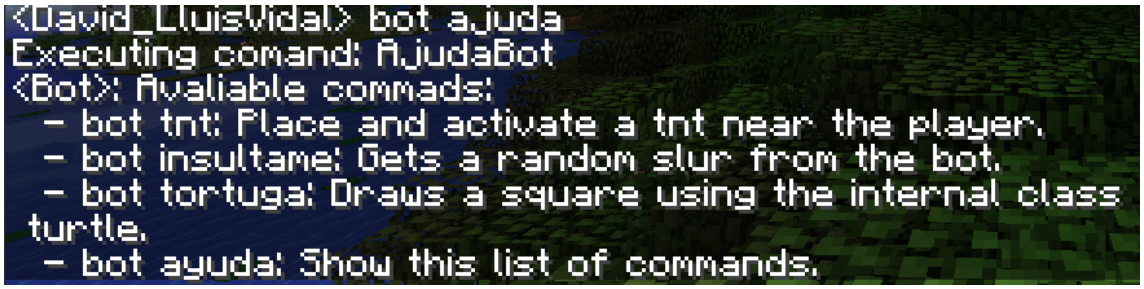
Crida als bots:

- TNTBot:

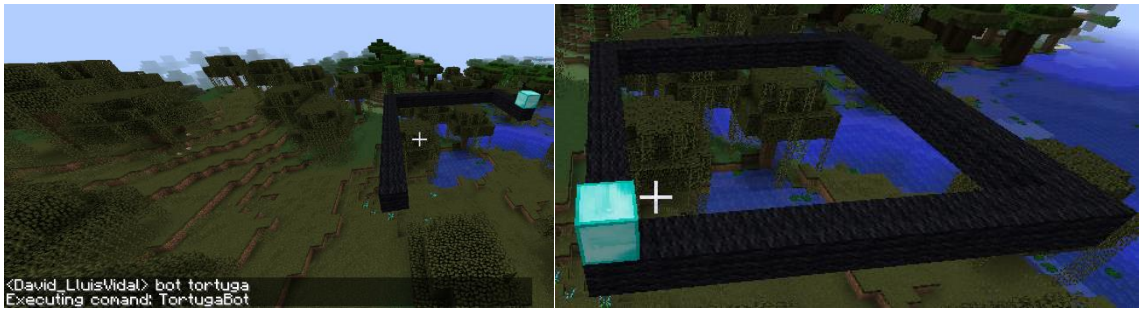


- InsultBot:

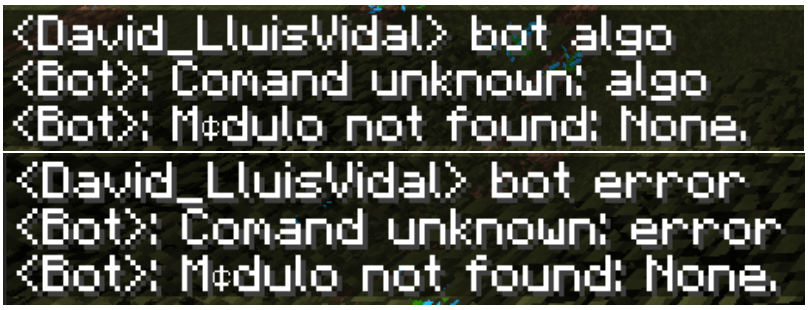
- AjudaBot:



- TortugaBot:



- Crides errònies:



- Bots en arguments: