Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
División División de Tecnologías para la Integración Ciber-Humana

Práctica 1.2 Agente Reactivo Simple

González Camino Edgar Alejandro 214415734
Brandon Hernández Ledezma 215515031

Inteligencia Artificial I - D02
Mtro. Diego Alberto Oliva Navarro

17 de Octubre del 2022

Desarrollo del código.

Se crearon dos clases la clase Aspiradora.py y la clase Enviroment.py.

Clase Enviroment.py

```python
class Enviroment:
    def __init__(self, trash):
        self.trash = trash

    def isClean(self):
        if(self.trash > 0):
            return False
        return True
```

Esta clase simula ser un entorno o cuadro o cuarto dependiendo de la interpretación que se decida darle. Se genera con una cantidad de basura random y una funcion que indica si esta limpia o no.

Clase Aspiradora.py

```python
INACTIVE = 0
CLEANING = 1
MOV_A = 2
MOV_B = 3

class Aspiradora:
    def __init__(self, env_A, env_B, location): #Inicializacion de variables
        self.env_A = env_A
        self.env_B = env_B
        self.status = INACTIVE  #status de la accion
        self.statusPrev = INACTIVE #status de la accion anterior
        self.location = location  #Localizacion de la aspiradora
        self.actions = 0
        self.energy = 100   #Cantidad de energia del robot

    def start(self):    #Inicio del funcionamiento de la aspiradora
        print("Hay: ", self.env_A.trash, "basuras en el Entorno A")
        print("Hay: ", self.env_B.trash, "basuras en el Entorno B")
        while(self.env_A.isClean() == False or self.env_B.isClean() == False):  #Mientras no esten limpios
            self.clean()

        print("Esta todo limpio apagando ...")  #Una vez que termine de limpiar se apaga
        self.statusPrev = self.status
        self.status = INACTIVE
```

Esta clase recibe los paramteros de los dos entornos y la localizacion en la que estará la aspiradora en su constructor de clase. En esta misma imagen también se puede observar que tiene la funcion start() que indica que empezará con su funcionamiento mandando a llamar al metodo clean() mientras no estén limpios ambos entornos.

```
def clean(self):    #Funcion que le indica a la aspiradora como limpiar en cada momento
    self.statusPrev = INACTIVE
    self.status = INACTIVE
    if(self.location == 'A'):   #Si la aspiradora se encuentra en A hay 2 opciones
        if(self.env_A.isClean()):   #La primera: el entorno A esta limpio y se mueve a B
            print("Limpio A")
            self.statusPrev = self.status
            self.status = MOV_B
            print("Moviendo a B")
            self.location = 'B'
            self.actions += 1
            self.energy -= 5    #Disminuye la energia y aumenta la cant de movs
        else:                       #La segunda: El entorno A esta sucio, se limpia y se mueve
            self.statusPrev = self.status
            self.status = CLEANING
            for i in range(self.env_A.trash):
                print('Aspirando A ...')
                self.env_A.trash -= 1

            print("Limpio A")
            self.statusPrev = self.status
            self.status = MOV_B
            print("Moviendo a B")
            self.location = 'B'
            self.actions += 2
            self.energy -= 10    #Disminuye la energia y aumenta la cant de movs

    elif(self.location == "B"): #Si la aspiradora se encuentra en B hay 2 opciones
        if(self.env_B.isClean()):   #La primera: el entorno B esta limpio y se mueve a el entor
            print("Limpio B")
            self.statusPrev = self.status
            self.status = MOV_A
            print("Moviendo a A")
            self.location = 'A'
            self.actions += 1
            self.energy -= 5    #Disminuye la energia y aumenta la cant de movs

        else:                       #La segunda: El entorno B esta sucio, se limpia y se mueve
            self.status = CLEANING
            for i in range(self.env_B.trash):
                print('Aspirando B ...')
                self.env_B.trash-=1

            print("Limpio B")
            self.statusPrev = self.status
            self.status = MOV_A
            print("Moviendo a A")
            self.location = 'A'
            self.actions += 2
            self.energy -= 10    #Disminuye la energia y aumenta la cant de movs
```

Igualmente la clase aspiradora tiene el metodo clean() que le indica a la aspiradora como debería limpiar. En este caso dependiendo de donde esté la aspiradora se tiene distintas opciones de movimiento, pero generalizando se tienen 2 opciones. La primera opción es aspirar el entorno o cuarto o espacio actual si este está sucio. La segunda opción es moverse al siguiente cuarto o entorno, y una vez en el siguiente cuarto se ciclara la misma toma de decision, limpiar o moverse. Y en este caso la modificacion le penaliza por cadsa movimiento que hace restando la cantidad de energia restante

Main.py

```python
from Aspiradora import Aspiradora
from Enviroment import *
import os

trash_A = int(input('Ingrese can. de basura en A: '))
trash_b = int(input('Ingrese can. de basura en B: '))

env_A = Enviroment(trash_A)
env_B = Enviroment(trash_b)
os.system ("cls")

location = input('Ingrese A o B para elegir el area donde iniciara la aspiradora: ').upper()
os.system ("cls")

aspiradora1 = Aspiradora(env_A, env_B, location)

aspiradora1.start()


print("Cantidad total de movimientos: ", aspiradora1.actions)
print("energia restante: ", aspiradora1.energy)
```

En el programa principal se manda a llamar a la clase aspiradora, se crea un objeto y se crean dos entornos (los que terminara por limpiar la aspiradora) con la cantidad de basura que le indiquemos, y por ultimo se pregunta en el entorno inicial en donde estara la aspiradora siendo el entorno A y el Entorno B las dos unicas opciones, una vez terminada la inicializacion de la aspiradora se inicia su fucnionalidad.

Inicializacion:

```
Ingrese can. de basura en A: 4
Ingrese can. de basura en B: 0
```

```
Ingrese A o B para elegir el area donde iniciara la aspiradora: B
```

Resultado:

```
Hay:  4 basuras en el Entorno A
Hay:  0 basuras en el Entorno B
Limpio B
Moviendo a A
Aspirando A ...
Aspirando A ...
Aspirando A ...
Aspirando A ...
Limpio A
Moviendo a B
Esta todo limpio apagando ...
Cantidad total de movimientos:  3
energia restante:  85
```

Responder las siguientes preguntas:

a. ¿Puede un agente reactivo ser racional bajo las condiciones del problema? Justifica la respuesta.

- Sí, debido a las característicasde este problema la aspiradora que diseñamos puede ser lo minimamente racional debido a que puede llegar a decidir por sus propios metodos que si un espacio ya esta limpio entonces debe de moverse a otro espacio que si necesite limpieza.

b. ¿Qué sucedería si se tuviera un agente reactivo capaz de almacenar al menos un estado previo del entorno? Diseña dicho agente.

- Si el agente guardara uno o mas estados previos podria evitar hacer dobles verificaciones en algunas zonas que ya ha limpiado previamente.

Brandon Hernandez Ledezma Conclusión:

Esta practica me sirvio como base para entender el modelo más simple de un agente que resuelve problemas pues con las clases que se crearon a lo largo del desarrollo de la practica pues se logró simular el comportamiento de una spiradora la cual se pone a limpiar dos entornos mientras uno de ellos esté con basura. Se desarrollo de manera que la aspiradora tome la decision de o limpiar el entorno actual o moverse al siguiente entorno. Y si se mueve al siguiente entorno vuelve a tomar la decisión de o moverse o limpiar.

Con el avance de esta actividad se puede observar que el funcionamiento del anteriormente diseñado agente reactivo tuvo un desempeño adecuado como se esperaba, logro limpiar las 2 areas tan solo reaccionando a si el entorno esta limpio o si esta sucio y si tiene que mejor moverse al siguiente entorno.

Y además en adición se penaliza por cada movimiento, en este caso se opto por penalizar con la perdida de energia lo cual, si se tuvieran mas zonas si la aspiradora no se desempeña bien, se quedaria sin energia a mitad de la limpieza de las zonas.