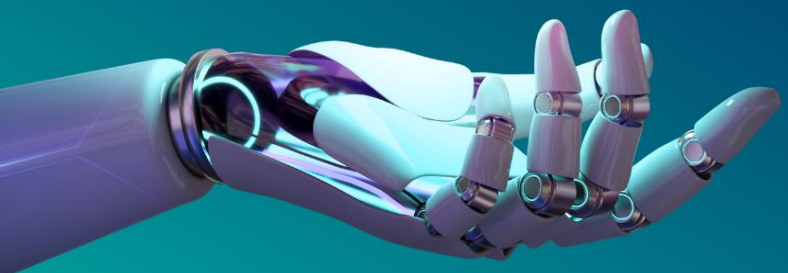
CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

UNIVERSIDAD DE GUADALAJARA

PRÁCTICA I

SIMULADOR ASPIRADORA



Alumnos:

Carbajal Armenta Yessenia Poala

Leon Estrada Rafael

Medina Bolaños Danna Paola

Códigos:

220286482

220286121

220286938

Maestro:

Oliva Navarro Diego Alberto

Materia:

Inteligencia Artificial

Sección:

D05

Guadalajara, Jal. a 10 de marzo del 2023

Practica 1 – Simulador de Aspiradora

Instrucciones

Implementar un simulador que determine la medida de rendimiento para el entorno de una aspiradora (ver archivo anexo). La implementación deber modular de manera que los sensores, actuadores y las características del entorno puedan ser modificadas. (El lenguaje de programación es libre).

Código

Importaciones

La primera sección del código se encarga de importar las librerías y módulos necesarios para la implementación de la simulación. En este caso, se importan las siguientes librerías y módulos.

```
from enum import Enum
from random import random, randint
import os
import time
from time import sleep
```

Definición del enum TipoDeCelda

La segunda sección define una enumeración llamada **TipoDeCelda**. Esta enumeración tiene tres valores: **Limpia**, **Sucia** y **Ocupada**, representados mediante emojis. Esta enumeración se utiliza para representar el estado de las celdas del entorno.

```
class TipoDeCelda(Enum):
    Limpia = "□"
    Sucia = "■"
    Ocupada = "🔬"
```

Definición de la clase Entorno

La tercera sección define la clase **Entorno**, que representa el entorno de la aspiradora. Esta clase tiene los siguientes métodos:

- **__init__**: es el constructor de la clase. Recibe como parámetros el ancho del entorno y la proporción de celdas sucias. El método crea una lista de celdas con la proporción de suciedad especificada y el resto de celdas limpias.

- **__str__**: devuelve una cadena que representa el estado del entorno en un momento determinado.

- **esta_dentro**: recibe como parámetro un número entero que representa la posición de una celda en el entorno. Devuelve **True** si la posición está dentro del rango de la lista de celdas y **False** en caso contrario.

- **esta_sucia**: recibe como parámetro un número entero que representa la posición de una celda en el entorno. Devuelve **True** si la celda está sucia y **False** en caso contrario.

- **limpiar**: recibe como parámetro un número entero que representa la posición de una celda en el entorno. Si la posición está dentro del rango de la lista de celdas, el método cambia el estado de la celda a limpio.

- **esta_todo_limpio**: devuelve **True** si todas las celdas del entorno están limpias y **False** en caso contrario.

```
class Entorno:
    def __init__(self, ancho: int, proporcion_sucia: float = .5) -> None:
        self.espacio = []
        for i in range(ancho):
            if random() > proporcion_sucia:
                self.espacio.append(TipoDeCelda.Limpia)
            else:
                self.espacio.append(TipoDeCelda.Sucia)
        self.espacioAux = self.espacio.copy()#espacioAux tendrá el
espacio sin la aspiradora, es necesario para mostrar la aspiradora y que
no intervenga en las condiciones

    def __str__(self):
        cadena = ""
        for celda in self.espacio:
            cadena += celda.value
        return cadena
```

```
    def esta_dentro(self, x: int) -> bool:
        if x >= 0 and x < len(self.espacio):
            return True
        else:
            return False

    def esta_sucia(self, x: int) -> bool:
        return self.espacioAux[x] == TipoDeCelda.Sucia

    def limpiar(self, x: int) -> None:
        if self.esta_dentro(x):
            self.espacioAux[x] = TipoDeCelda.Limpia

    def esta_todo_limpio(self) -> bool:
        return TipoDeCelda.Sucia not in self.espacioAux
```

Definición de la clase Aspiradora

La cuarta sección define la clase **Aspiradora**, que representa la aspiradora del sistema. Esta clase tiene los siguientes métodos:

- **__init__**: es el constructor de la clase. Recibe como parámetros el entorno y la posición inicial de la aspiradora en el entorno. El método marca la celda ocupada por la aspiradora en el entorno.
- **actuar**: si la celda donde se encuentra la aspiradora está sucia, el método la limpia. Si no, el método llama al método **mover**.
- **mover**: el método marca la celda donde se encuentra la aspiradora como limpia, la mueve a la siguiente celda en la dirección en la que se está moviendo y la marca como ocupada. Si la aspiradora llega al final del entorno, cambia de dirección y vuelve sobre sus pasos.

```
class Aspiradora:
    class Direccion(Enum):
        Izquierda = -1
        Derecha = 1

    def __init__(self, ent: Entorno, x: int):
        self.entorno = ent
```

```
        self.entorno.espacio[x] = TipoDeCelda.Ocupada
        self.x = x
        self.direccion = Aspiradora.Direccion.Izquierda

    def actuar(self):
        if self.entorno.esta_sucia(self.x):
            self.entorno.limpiar(self.x)
        else:
            self.mover()

    def mover(self):
        self.entorno.espacio[self.x] = TipoDeCelda.Limpia
        self.x += self.direccion.value
        if not self.entorno.esta_dentro(self.x):
            if self.direccion == Aspiradora.Direccion.Derecha:
                self.direccion = Aspiradora.Direccion.Izquierda
            else:
                self.direccion = Aspiradora.Direccion.Derecha
            self.x += self.direccion.value * 2

        self.entorno.espacio[self.x] = TipoDeCelda.Ocupada
```

Función auxiliar limpiar_pantalla

La quinta sección define la función **limpiar_pantalla**, que se encarga de limpiar.

```
def limpiar_pantalla():
    os.system('cls')
```

Impresiones del entorno

Esta sección es el punto de entrada del programa. Primero, se establece el tamaño del entorno con la variable "tamanio". Luego, se crea una instancia de la clase Entorno con el tamaño especificado y se almacena en la variable "entorno". A continuación, se crea una instancia de la clase Aspiradora con el entorno creado anteriormente y la posición inicial 0, y se almacena en la variable "Aspiradora".

A continuación, se imprime el estado inicial del entorno en la consola mediante la función "print(entorno)" y se espera 0.7 segundos antes de comenzar el ciclo principal.

El ciclo principal se ejecuta mientras el entorno no esté completamente limpio, lo que se comprueba mediante la función "esta_todo_limpio()" del objeto "entorno" almacenado en el objeto "Aspiradora". En cada iteración del ciclo, se llama a la función "actuar()" del objeto "Aspiradora", que realiza una acción (limpiar o mover) en función del estado actual del entorno y la posición de la aspiradora. Luego, se limpia la pantalla, se imprime el estado actual del entorno y se espera 0.7 segundos antes de continuar con la próxima iteración.

```
tamanio  = 6
limpiar_pantalla()
entorno = Entorno(tamanio)
Aspiradora = Aspiradora(entorno, 0)
print(entorno)
sleep(0.7)

while not Aspiradora.entorno.esta_todo_limpio():
    Aspiradora.actuar()
    limpiar_pantalla()
    print(entorno)
    sleep(0.7)
```

Ejecución

Conclusiones

El programa es una implementación simple de un simulador de aspiradora para determinar la medida de rendimiento en un entorno dado. El programa utiliza dos clases, "Entorno" y "Aspiradora", para modelar el entorno y la aspiradora, respectivamente. La aspiradora se mueve por el entorno y limpia las celdas sucias a medida que se mueve. El rendimiento se mide cuando la aspiradora limpia todo el entorno. Este programa es modular y permite la modificación de los sensores, actuadores y las características del entorno, lo que lo hace flexible y fácil de adaptar para diferentes casos de uso.