# Implementación de un Agente Reflejo Simple seguidor de líneas

Salazar Vega Edwin Martín (edwin@iartificial.io)
Llana Chavez Walter Rodolfo (walter.wr7@gmail.com)
Luna Jaramillo Juan Marcos (jmlunaj@pucp.edu.pe)
Alejos Yarasca Fiorella Andrea (fiorella.alejos@yahoo.com)
Medina Rodríguez Henry (hmedinar@uni.pe)

MIA-103 Fundamentos de Inteligencia Artificial - Maestría en Inteligencia Artificial - 2024-I
Facultad de Ingeniería Industrial y Sistemas - Universidad Nacional de Ingeniería

***Resumen*—Este informe explica la creación y funcionamiento de un agente reflejo simple, implementado en Python, que se desplaza por un entorno dinámico de 10x10 celdas generado aleatoriamente. El agente está equipado con sensores básicos diseñados para detectar y seguir líneas oscuras, buscando maximizar el tiempo que permanece sobre estas. La implementación incluye una visualización de cada movimiento del agente dentro de su entorno, facilitando la observación directa de sus decisiones y acciones.**

***Abstract*--This report explains the creation and operation of a simple reflex agent, implemented in Python, that moves through a dynamically generated 10x10 cell environment. The agent is equipped with basic sensors designed to detect and follow dark lines, aiming to maximize the time it remains on these lines. The implementation includes a visualization of each movement of the agent within its environment, facilitating direct observation of its decisions and actions.**

## I. Introducción

La capacidad de los agentes autónomos para interactuar y navegar en entornos complejos representa un pilar clave en la inteligencia artificial aplicada. En este proyecto, se ha desarrollado un agente reflejo simple utilizando Python, específicamente en la plataforma Google Colab, para explorar las dinámicas de la navegación autónoma mediante un entorno de simulación visual. El objetivo principal es implementar y evaluar la capacidad del agente seguidor de líneas negras basado en reglas de percepción-acción.

## II. Diseño del Agente

### *A. Diseño de la Arquitectura*

*Descripción del ambiente físico*
El ambiente consiste en una malla de cuadrados de 10x10 celdas. Conformada por celdas oscuras, celdas claras y paredes. Las celdas son de color blanco o negro. Mientras que las paredes son de color gris.
El agente se ubica inicialmente en la celda superior izquierda (1,1) y está solo.
El medio ambiente es accesible, estático, episódico y discreto.

*Representación del ambiente físico*

*Diseño de variables*

- Variable matriz de percepción-acción (Matriz de enteros de dimensión 55x6)

Percepciones:
Orientación, choque, celda actual, celda frontal izquierda, celda frontal central, celda frontal derecha

Acciones:
Rotar 90 grados en sentido horario, rotar 90 grados en sentido antihorario, avanzar

- Variable malla de cuadrados (Matriz de enteros de dimensión 10x10)

- Variable clase agente

Atributos: Entorno, posición, orientación, sensor de contacto, cámara inferior, cámara frontal derecha, cámara frontal izquierda, cámara frontal centro Métodos: Percibir, decidir, actuar, mostrar

*Inicialización de variables*

Matriz de percepción-acción

**Tabla de Percepción - Acción**

| Orientacion | Choque | C. Posicion | C. Izquierda | C. Central | C. Derecha | Unica Accion |
|---|---|---|---|---|---|---|
| Cualquiera | Contacto | ---- | ---- | ---- | ---- | Rotar -90° |
| Cualquiera | No contacto | Blanca | Blanca | Blanca | Blanca | Avanzar |
| Cualquiera | No contacto | Blanca | Blanca | Blanca | Oscura | Rotar -90° |
| Cualquiera | No contacto | Blanca | Blanca | Blanca | Borde | Avanzar |
| Cualquiera | No contacto | Blanca | Blanca | Oscura | Blanca | Avanzar |
| Cualquiera | No contacto | Blanca | Blanca | Oscura | Oscura | Avanzar |
| Cualquiera | No contacto | Blanca | Blanca | Oscura | Borde | Avanzar |

*Figura 1. Fragmento de la matriz percepción-acción*

Malla de cuadrados
Las paredes se ubican en los 4 bordes de la malla.
Las celdas claras y oscuras se determinan mediante una función random.

Agente
La posición inicial del agente es (1,1) y orientación fija a la derecha.

*Estructuras de datos para representar el ambiente*

Matriz percepción acción

```
#Fragmento de la Matriz Percepción - Acción
matriz_per_ac = [[1,0,0,0,0,1],
[0,1,1,1,1,0],
[0,1,1,1,0,1],
[0,1,1,1,2,0],
[0,1,1,0,1,0],
[0,1,1,0,0,0],
[0,1,1,0,2,0],
[0,1,1,2,1,1]....
```

Malla de cuadrados

```
# Fragmento de Clase para el entorno
matriz = np.array([[random.choice([PISO_OSCURO, PISO_NO_OSCURO]) for _ in range(tamano)] for _ in range(tamano)])
```

Agente

```
# Fragmento de Clase para el agente
class Agente:
    def __init__(self, entorno, posicion=(1, 1), orientacion=0):
    def percibir(self):
```

*Representación gráfica del ambiente*

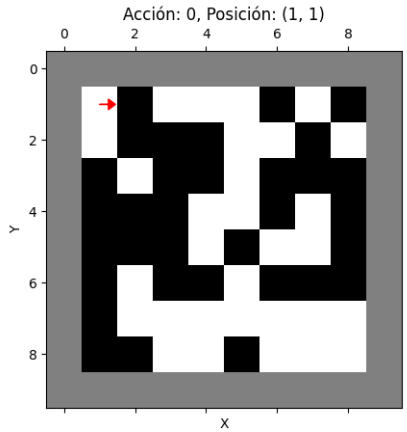



*Figura 2. Ambiente inicial*

*Ejemplos de ambientes:*

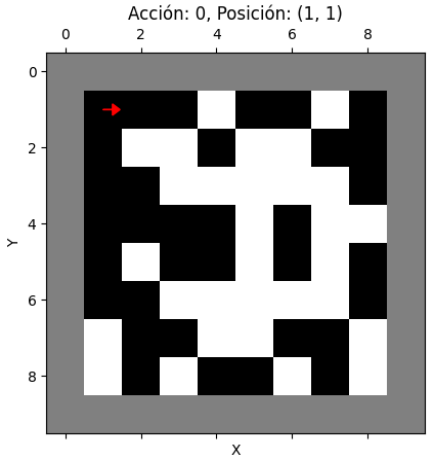



*Figura 3. Ejemplo 1 de ambiente*

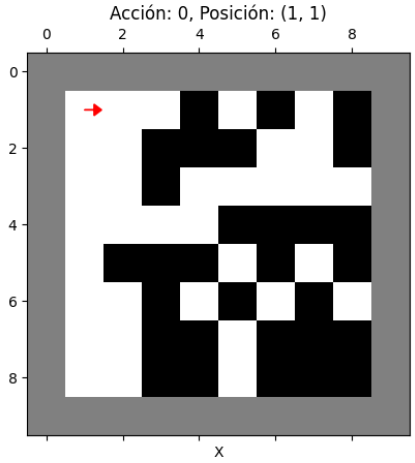



*Figura 4. Ejemplo 2 de ambiente*

*B. Objetivo del Agente*
Seguir las líneas (Celdas oscuras) dentro de la malla de cuadrados.

*Sensores*
Propioceptor
Sensor de contacto
Camara debajo del agente
Cámara delante del agente

*Preparación de Datos.*
Se genera la malla de cuadrados aleatoriamente

*C. Percepciones*
- Celda oscura
- Celda clara
- Pared

## I. CREAR LA TABLA DE PERCEPCIÓN ACCIÓN

La tabla o matriz de Percepción - Acción ha sido diseñada con la capacidad de maximizar la eficiencia de seguimiento de líneas negras del agente reflejo simple desarrollado. En esta se intenta que el agente sepa cómo actuar frente a los datos obtenidos por medio de sus sensores, descritos en los acápites previos. El "creador" considera la mejor decisión a tomar frente a cada información percibida y le brinda esta información al agente para saber que hacer frente a tal o cual situación descrita en la Tabla 1.

Tabla 1. Matriz de Percepción – Acción

| Orientacion | Choque | C. Posicion | C. Izquierda | C. Central | C. Derecha | Accion |
|---|---|---|---|---|---|---|
| Cualquiera | Contacto | ---- | ---- | ---- | ---- | Rotar -90° |
| Cualquiera | No contacto | Blanca | Blanca | Blanca | Blanca | Avanzar |
| Cualquiera | No contacto | Blanca | Blanca | Blanca | Oscura | Rotar -90° |
| Cualquiera | No contacto | Blanca | Blanca | Blanca | Borde | Avanzar |
| Cualquiera | No contacto | Blanca | Blanca | Oscura | Blanca | Avanzar |
| Cualquiera | No contacto | Blanca | Blanca | Oscura | Oscura | Avanzar |
| Cualquiera | No contacto | Blanca | Blanca | Oscura | Borde | Avanzar |
| Cualquiera | No contacto | Blanca | Blanca | Borde | Blanca | Rotar -90° o Rotar +90° |
| Cualquiera | No contacto | Blanca | Blanca | Borde | Oscura | Rotar -90° |
| Cualquiera | No contacto | Blanca | Blanca | Borde | Borde | Rotar +90° |
| Cualquiera | No contacto | Blanca | Oscura | Blanca | Blanca | Rotar +90° |
| Cualquiera | No contacto | Blanca | Oscura | Blanca | Oscura | Avanzar |
| Cualquiera | No contacto | Blanca | Oscura | Blanca | Borde | Rotar +90° |
| Cualquiera | No contacto | Blanca | Oscura | Oscura | Blanca | Avanzar |
| Cualquiera | No contacto | Blanca | Oscura | Oscura | Oscura | Avanzar |
| Cualquiera | No contacto | Blanca | Oscura | Oscura | Borde | Avanzar |
| Cualquiera | No contacto | Blanca | Oscura | Borde | Blanca | Rotar +90° |
| Cualquiera | No contacto | Blanca | Oscura | Borde | Oscura | Rotar -90° o Rotar +90° |
| Cualquiera | No contacto | Blanca | Oscura | Borde | Borde | Rotar +90° |
| Cualquiera | No contacto | Blanca | Borde | Blanca | Blanca | Avanzar |
| Cualquiera | No contacto | Blanca | Borde | Blanca | Oscura | Rotar -90° |
| Cualquiera | No contacto | Blanca | Borde | Blanca | Borde | Avanzar |
| Cualquiera | No contacto | Blanca | Borde | Oscura | Blanca | Avanzar |
| Cualquiera | No contacto | Blanca | Borde | Oscura | Oscura | Avanzar |
| Cualquiera | No contacto | Blanca | Borde | Oscura | Borde | Avanzar |
| Cualquiera | No contacto | Blanca | Borde | Borde | Blanca | Rotar -90° |
| Cualquiera | No contacto | Blanca | Borde | Borde | Oscura | Rotar -90° |
| Cualquiera | No contacto | Blanca | Borde | Borde | Borde | Rotar -90° |
| Cualquiera | No contacto | Oscura | Blanca | Blanca | Blanca | Avanzar |
| Cualquiera | No contacto | Oscura | Blanca | Blanca | Oscura | Rotar -90° |
| Cualquiera | No contacto | Oscura | Blanca | Blanca | Borde | Rotar +90° |
| Cualquiera | No contacto | Oscura | Blanca | Oscura | Blanca | Avanzar |
| Cualquiera | No contacto | Oscura | Blanca | Oscura | Oscura | Avanzar |
| Cualquiera | No contacto | Oscura | Blanca | Oscura | Borde | Avanzar |
| Cualquiera | No contacto | Oscura | Blanca | Borde | Blanca | Rotar -90° o Rotar +90° |
| Cualquiera | No contacto | Oscura | Blanca | Borde | Oscura | Rotar -90° |
| Cualquiera | No contacto | Oscura | Blanca | Borde | Borde | Rotar +90° |
| Cualquiera | No contacto | Oscura | Oscura | Blanca | Blanca | Rotar +90° |
| Cualquiera | No contacto | Oscura | Oscura | Blanca | Oscura | Rotar -90° o Rotar +90° |
| Cualquiera | No contacto | Oscura | Oscura | Blanca | Borde | Rotar +90° |
| Cualquiera | No contacto | Oscura | Oscura | Oscura | Blanca | Avanzar |
| Cualquiera | No contacto | Oscura | Oscura | Oscura | Oscura | Avanzar |
| Cualquiera | No contacto | Oscura | Oscura | Oscura | Borde | Avanzar |
| Cualquiera | No contacto | Oscura | Oscura | Borde | Blanca | Rotar +90° |
| Cualquiera | No contacto | Oscura | Oscura | Borde | Oscura | Rotar -90° o Rotar +90° |
| Cualquiera | No contacto | Oscura | Oscura | Borde | Borde | Rotar +90° |
| Cualquiera | No contacto | Oscura | Borde | Blanca | Blanca | Rotar -90° |
| Cualquiera | No contacto | Oscura | Borde | Blanca | Oscura | Rotar -90° |
| Cualquiera | No contacto | Oscura | Borde | Blanca | Borde | Avanzar |
| Cualquiera | No contacto | Oscura | Borde | Oscura | Blanca | Avanzar |
| Cualquiera | No contacto | Oscura | Borde | Oscura | Oscura | Avanzar |
| Cualquiera | No contacto | Oscura | Borde | Oscura | Borde | Avanzar |
| Cualquiera | No contacto | Oscura | Borde | Borde | Blanca | Rotar -90° |
| Cualquiera | No contacto | Oscura | Borde | Borde | Oscura | Rotar -90° |
| Cualquiera | No contacto | Oscura | Borde | Borde | Borde | Rotar -90° o Rotar +90° |

## I. CALCULAR DESEMPEÑO

El cálculo del desempeño se realizará de la siguiente forma:

Se realizarán 10 ejecuciones.
Cada ejecución contendrá 100 iteraciones.

El cálculo del desempeño será:

$$desempeño = \frac{pasos\ repetidos}{total\ de\ pasos}$$

$$desempeño\ total\ = \frac{\sum_{i=1}^{100} desempeño_i}{10}$$

A continuación se muestra el ejercicio completo de la iteración 1
Tabla de resultados de iteración 1

| i | Pos | i | Pos | i | Pos |
|---|---|---|---|---|---|
| 1 | 1,1 | 35 | 3,3 | 69 | 8,1 |
| 2 | 1,2 | 36 | 3,2 | 70 | 8,1 |
| 3 | 1,3 | 37 | 3,1 | 71 | 8,2 |
| 4 | 1,4 | 38 | 3,1 | 72 | 8,2 |
| 5 | 1,4 | 39 | 2,1 | 73 | 8,3 |
| 6 | 2,4 | 40 | 1,1 | 74 | 8,4 |
| 7 | 3,4 | 41 | 1,1 | 75 | 8,5 |
| 8 | 4,4 | 42 | 1,2 | 76 | 8,6 |
| 9 | 5,4 | 43 | 1,3 | 77 | 8,6 |
| 10 | 6,4 | 44 | 1,4 | 78 | 8,6 |
| 11 | 7,4 | 45 | 1,4 | 79 | 8,6 |
| 12 | 7,4 | 46 | 2,4 | 80 | 8,6 |
| 13 | 7,3 | 47 | 3,4 | 81 | 8,6 |
| 14 | 7,2 | 48 | 4,4 | 82 | 7,6 |
| 15 | 7,1 | 49 | 5,4 | 83 | 6,6 |
| 16 | 7,1 | 50 | 6,4 | 84 | 6,6 |
| 17 | 8,1 | 51 | 7,4 | 85 | 6,7 |
| 18 | 8,1 | 52 | 7,4 | 86 | 6,8 |
| 19 | 8,1 | 53 | 7,3 | 87 | 6,8 |
| 20 | 7,1 | 54 | 7,2 | 88 | 5,8 |
| 21 | 6,1 | 55 | 7,1 | 89 | 4,8 |
| 22 | 5,1 | 56 | 7,1 | 90 | 3,8 |

| | | | | | |
|---|---|---|---|---|---|
| 23 | 4,1 | 57 | 7,1 | 91 | 2,8 |
| 24 | 3,1 | 58 | 7,2 | 92 | 1,8 |
| 25 | 2,1 | 59 | 7,3 | 93 | 1,8 |
| 26 | 1,1 | 60 | 7,4 | 94 | 1,8 |
| 27 | 1,1 | 61 | 7,5 | 95 | 1,7 |
| 28 | 1,2 | 62 | 7,5 | 96 | 1,6 |
| 29 | 1,3 | 63 | 7,5 | 97 | 1,6 |
| 30 | 1,4 | 64 | 7,4 | 98 | 1,6 |
| 31 | 1,4 | 65 | 7,3 | 99 | 1,6 |
| 32 | 2,4 | 66 | 7,2 | 100 | 1,6 |
| 33 | 3,4 | 67 | 7,1 | | |
| 34 | 3,4 | 68 | 7,1 | | |

Desempeño de iteración 1= 90/100 =0.9

Muestreo de las 10 iteraciones

| Iteración | Desempeño |
|---|---|
| 1 | 0.9 |
| 2 | 0.59 |
| 3 | 0.75 |
| 4 | 0.89 |
| 5 | 0.71 |
| 6 | 0.52 |
| 7 | 0.63 |
| 8 | 0.84 |
| 9 | 0.78 |
| 10 | 0.91 |

Desempeño Total Promedio=0.752
Desempeño Total Promedio en Porcentaje= 75.2%.

## III. DISEÑAR EL PROGRAMA

Para el desarrollo del programa, hemos elegido Python y utilizado Google Colab, dada su facilidad de implementación y desarrollo en la nube, hemos implementado el agente con un enfoque de programación orientado a objetos, lo cual facilita la organización y escalabilidad del código. Además, en cada iteración del agente, se genera una visualización gráfica de su entorno, mejorando significativamente la comprensión de las acciones tomadas por el agente. Este enfoque visual no solo clarifica el proceso de decisión, sino que también realza la interactividad y el análisis de la simulación.

**Clases Entorno:**

- Método **init**: encargado de construir la matriz del entorno de forma aleatoria con pisos oscuros, claros y bordes.

**Clase Agente:**

- Método **init**: encargado de inicializar las variables.
- Método **percibir**: encargado de dar lectura acorde a los sensores del agente.
- Método **decidir**: encargado de buscar la acción acorde a la matriz percepción-acción de acuerdo a los parámetros percibidos.
- Método **actuar**: encargado de ejecutar la acción, ya sea moverse o rotar.
- Método **dibujar_agente**: encargado de graficar al agente acorde a su posición y orientación.

Método general **mostrar_entorno_con_agente**: encargado de graficar al entorno y su agente.

**Definición de Constantes**

```
# Definir las constantes
PISO_OSCURO = 0  # Representado por el color negro
PISO_NO_OSCURO = 1  # Representado por el color blanco
BORDE = 2  # Representado por el color gris
PISO_TEXTO = ["Piso Oscuro", "Piso Blanco", "Borde"]


ACCION_AVANZAR = 0
ACCION_ROTAR_N90 = 1
ACCION_ROTAR_P90 = 2
ACCION_TEXTO = ["Avanzar", "Rotar -90", "Rotar +90"]

ORIENTACION_ARRIBA = 0
ORIENTACION_DERECHA = 1
ORIENTACION_ABAJO = 2
ORIENTACION_IZQUIERDA = 3
ORIENTACION_TEXTO = ["Arriba","Derecha","Abajo","Izquierda"]
```

**Definir la matriz de relación percepción acción**

```
#Matriz Percepción - Acción
matriz_per_ac = [[1,0,0,0,0,1],
                 [0,1,1,1,1,0],
                 [0,1,1,1,0,1],
                 [0,1,1,1,2,0],
                 [0,1,1,0,1,0],
                 [0,1,1,0,0,0],
                 [0,1,1,0,2,0],
                 [0,1,1,2,1,1],
                 [0,1,1,2,0,1],
                 [0,1,1,2,2,2],
                 [0,1,0,1,1,2],
                 [0,1,0,1,0,0],
                 [0,1,0,1,2,2],
                 [0,1,0,0,1,0],
                 [0,1,0,0,0,0],
                 [0,1,0,0,2,0],
                 [0,1,0,2,1,2],
                 [0,1,0,2,0,1],
```

Notar que esta tabla es el mapeo de las reglas percepción acción descritas en el punto 3, esta imagen es un fragmento de la matriz, la tabla real consta de 55 reglas. Se puede ver en el código completo.

**Métodos representativos:**

**Percibir:** método encargado de realizar las capturas acorde a sus sensores.

```
def percibir(self):
    x, y = self.posicion
    self.camara_debajo = self.entorno.matriz[x][y]

    #Percepcion de las camaras
    if self.camara_debajo<2:
      if self.orientacion == ORIENTACION_ARRIBA:
        self.camara_centro = self.entorno.matriz[x-1][y]
        self.camara_izquierda = self.entorno.matriz[x-1][y-1]
        self.camara_derecha = self.entorno.matriz[x-1][y+1]
      elif self.orientacion == ORIENTACION_ABAJO:
        self.camara_centro = self.entorno.matriz[x+1][y]
        self.camara_izquierda = self.entorno.matriz[x+1][y+1]
        self.camara_derecha = self.entorno.matriz[x+1][y-1]
      elif self.orientacion == ORIENTACION_IZQUIERDA:
        self.camara_centro = self.entorno.matriz[x][y-1]
        self.camara_izquierda = self.entorno.matriz[x+1][y-1]
        self.camara_derecha = self.entorno.matriz[x-1][y-1]
      elif self.orientacion == ORIENTACION_DERECHA:
        self.camara_centro = self.entorno.matriz[x][y+1]
        self.camara_izquierda = self.entorno.matriz[x-1][y+1]
        self.camara_derecha = self.entorno.matriz[x+1][y+1]

      #Print Percepciones
      #print("Matrix:", self.entorno.matriz)
      print("Captura de Percepción:")
      print("Posicion: ",x,y)
      print("Orientacion: ", ORIENTACION_TEXTO[self.orientacion])
      print("Camara Debajo: ", PISO_TEXTO[self.camara_debajo])
      print("Camara Centro: ", PISO_TEXTO[self.camara_centro])
      print("Camara Izquierda: ", PISO_TEXTO[self.camara_izquierda])
      print("Camara Derecha: ", PISO_TEXTO[self.camara_derecha])
      print("Sensor de contacto: No Contacto")
    else:
      self.sensor_contacto = 1

      print("Captura de Percepción:")
      print("Posicion: ",x,y)
      print("Orientacion: ", ORIENTACION_TEXTO[self.orientacion])
      print("Sensor de contacto: Contacto")
```

**Decidir:** método encargado de buscar la regla que coincide con los parámetros capturados (valores de los sensores) para identificar la regla definida, notar que este proceso funciona ya que se ha definido para todos los casos una regla específica.

```
def decidir(self, percepcion):
    #Choque, Camara Debajo, Camara Izq, Camara Centro, Camara Derecha, Accion
    accion = -1
    if self.sensor_contacto == 0:
      for row in matriz_per_ac:
        if row[0] == 0:
          if row[1] == self.camara_debajo and row[2] == self.camara_izquierda
            and row[3] == self.camara_centro and row[4] == self.camara_derecha:
            if row[2] == row[4] and (row[3]==1 or row[3]==2):
              accion = random.choice([1,2])
            elif row[3] == 1 and (row[2]==0 or row[4]==0):
              accion = random.choice([0,1])
            else:
              accion = row[5]
            break
        else:
          accion = row[5]
      print("Accion: ", ACCION_TEXTO[accion])
    else:
          accion = 1

    return accion
```

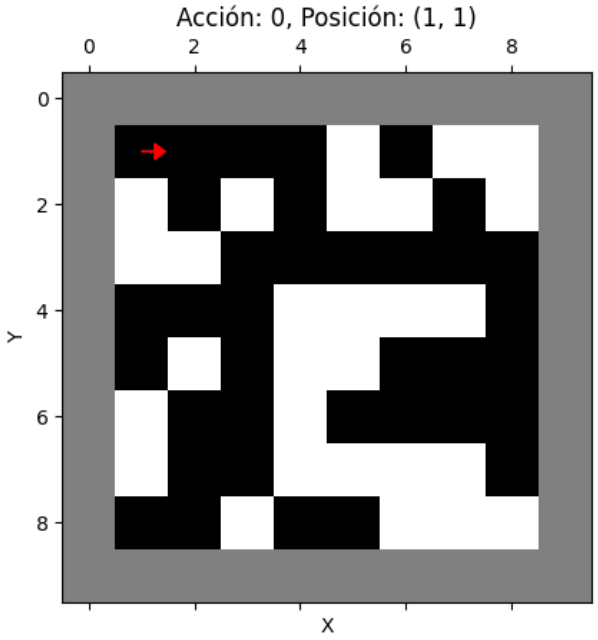
**main:** Ejecución del agente en su entorno, hemos definido que el agente iniciará desde la posición 1,1 y tomara una orientación hacia la derecha, luego invocamos a un iterador por 100 veces para que el agente interactúa con su entorno.

```
# Simulación del agente en el entorno
entorno = Entorno()
agente = Agente(entorno, posicion=(1, 1), orientacion=1)
mostrar_entorno_con_agente(entorno, agente, 0)

# Ciclo de simulación
for _ in range(100):  # Número de pasos de la simulación
    percepcion = agente.percibir()
    accion = agente.decidir(percepcion)
    agente.actuar(accion)
    mostrar_entorno_con_agente(entorno, agente, accion)
```

Resultados:

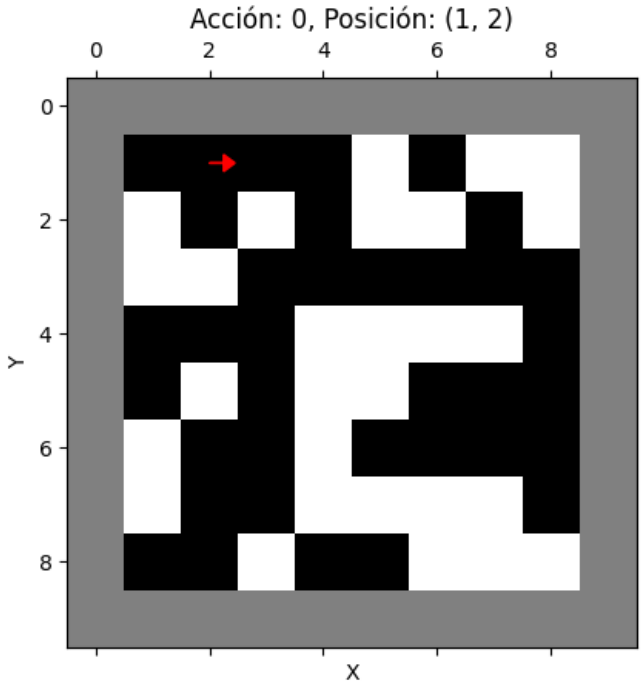
**Posición inicial:**




```
Captura de Percepción:
Posicion:  1 1
Orientacion:  Derecha
Camara Debajo:  Piso Oscuro
Camara Centro:  Piso Oscuro
Camara Izquierda:  Borde
Camara Derecha:  Piso Oscuro
Sensor de contacto: No Contacto
Accion:  Avanzar
```

En la parte inferior incluimos los valores de los sensores con su acción tomada acorde a la matriz de percepción - acción, notar que el agente ha optado por avanzar.

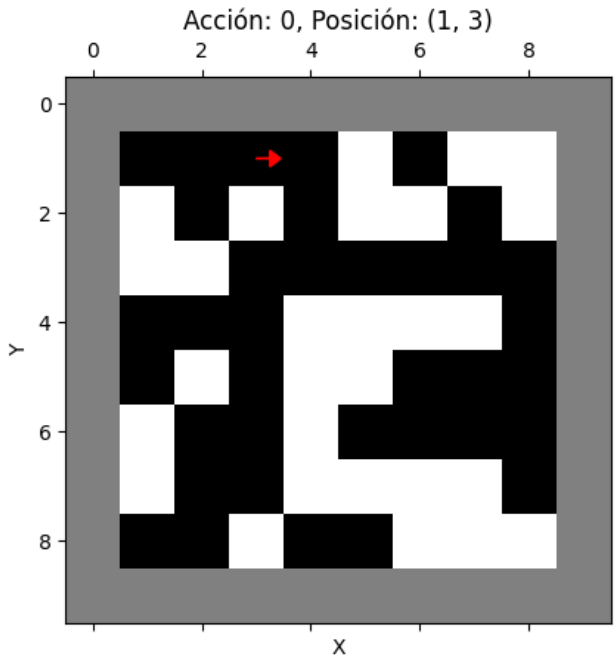
**Iteración 1:**




```
Captura de Percepción:
Posicion:  1 2
Orientacion:  Derecha
Camara Debajo:  Piso Oscuro
Camara Centro:  Piso Oscuro
Camara Izquierda:  Borde
Camara Derecha:  Piso Blanco
Sensor de contacto: No Contacto
Accion:  Avanzar
```

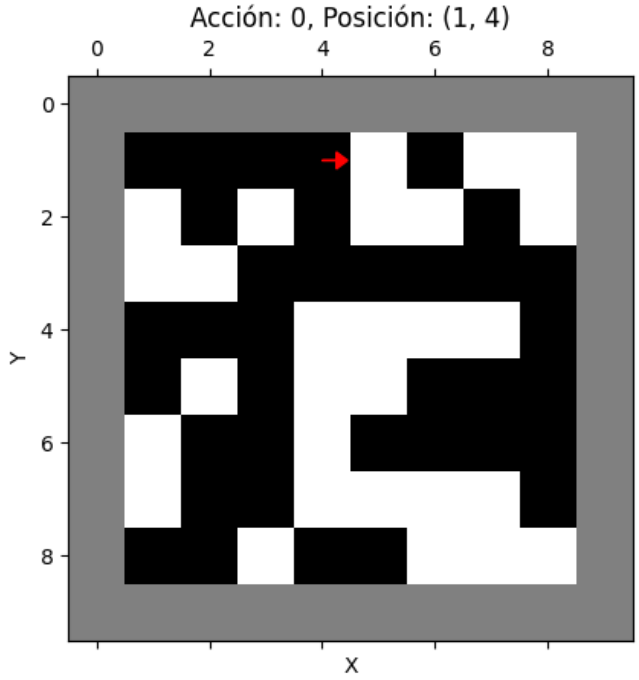
Notar que el agente ha optado por avanzar.

**Iteración 2:**




```
Captura de Percepción:
Posicion:  1 3
Orientacion:  Derecha
Camara Debajo:  Piso Oscuro
Camara Centro:  Piso Oscuro
Camara Izquierda:  Borde
Camara Derecha:  Piso Oscuro
Sensor de contacto: No Contacto
Accion:  Avanzar
```

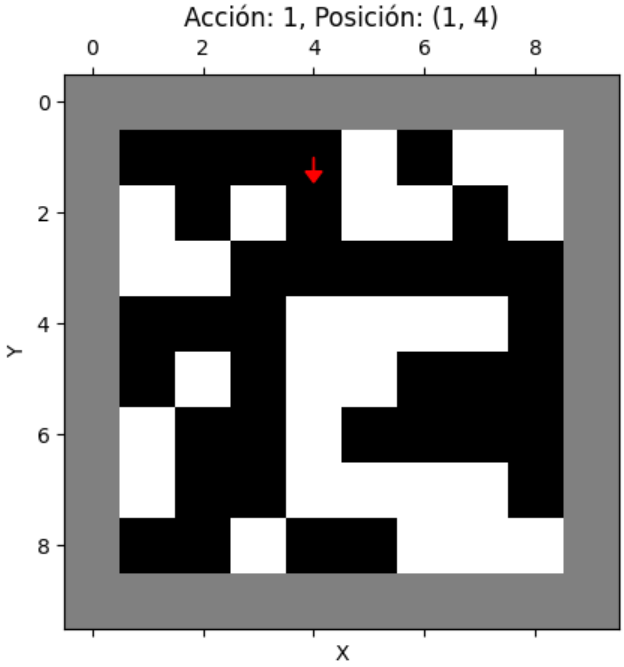
**Iteración 3:**




```
Captura de Percepción:
Posicion:  1 4
Orientacion:  Derecha
Camara Debajo:  Piso Oscuro
Camara Centro:  Piso Blanco
Camara Izquierda:  Borde
Camara Derecha:  Piso Blanco
Sensor de contacto: No Contacto
Accion:  Rotar -90
```

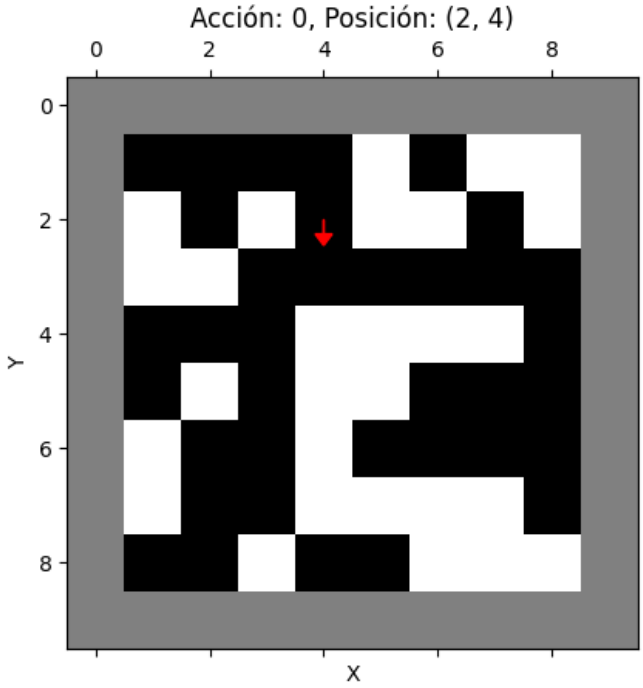
Notar que el agente ha optado por rotar en sentido horario.

**Iteración 4:**



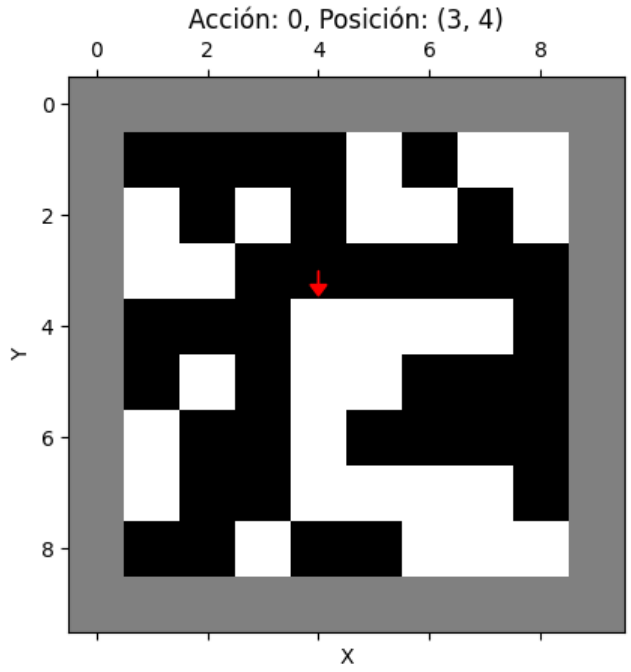

```
Captura de Percepción:
Posicion:  1 4
Orientacion:  Abajo
Camara Debajo:  Piso Oscuro
Camara Centro:  Piso Oscuro
Camara Izquierda:  Piso Blanco
Camara Derecha:  Piso Blanco
Sensor de contacto: No Contacto
Accion:  Avanzar
```

**Iteración 5:**



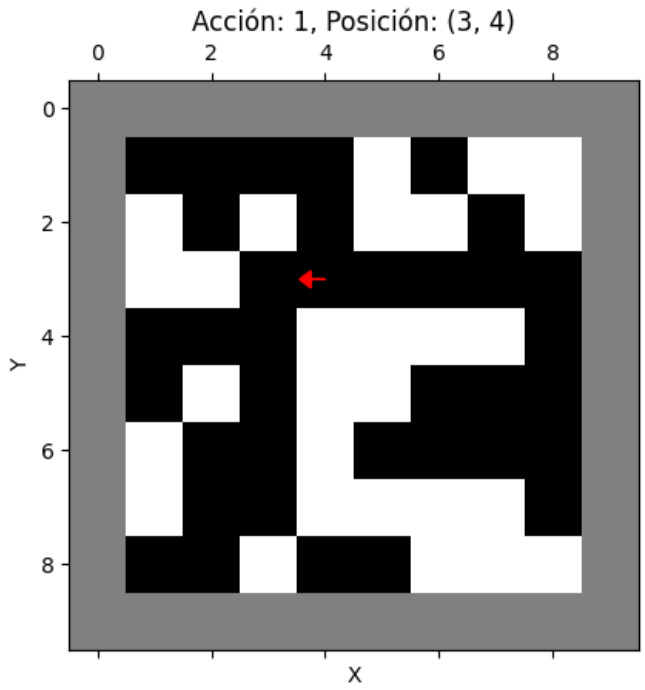

```
Captura de Percepción:
Posicion:  2 4
Orientacion:  Abajo
Camara Debajo:  Piso Oscuro
Camara Centro:  Piso Oscuro
Camara Izquierda:  Piso Oscuro
Camara Derecha:  Piso Oscuro
Sensor de contacto: No Contacto
Accion:  Avanzar
```

**Iteración 6:**



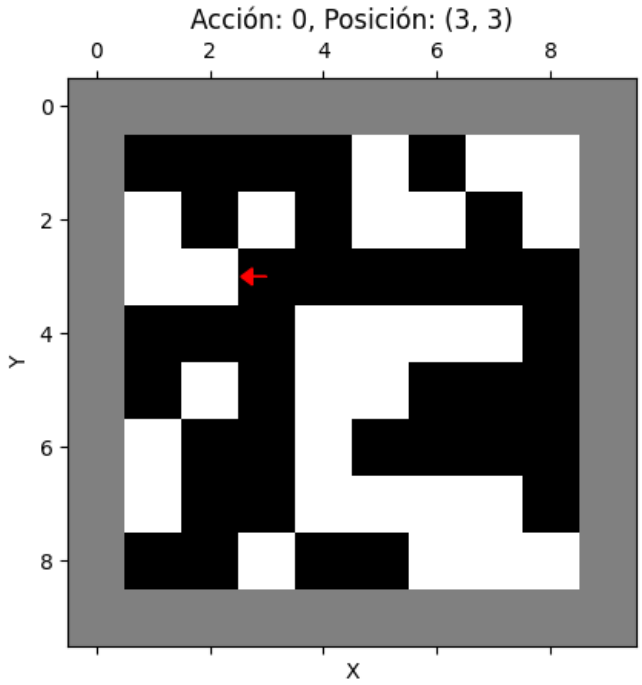

```
Captura de Percepción:
Posicion:  3 4
Orientacion:  Abajo
Camara Debajo:  Piso Oscuro
Camara Centro:  Piso Blanco
Camara Izquierda:  Piso Blanco
Camara Derecha:  Piso Oscuro
Sensor de contacto: No Contacto
Accion:  Rotar -90
```

**Iteración 7:**




```
Captura de Percepción:
Posicion:  3 4
Orientacion:  Izquierda
Camara Debajo:  Piso Oscuro
Camara Centro:  Piso Oscuro
Camara Izquierda:  Piso Oscuro
Camara Derecha:  Piso Blanco
Sensor de contacto: No Contacto
Accion:  Avanzar
```

**Iteración 8:**




```
Captura de Percepción:
Posicion:  3 3
Orientacion:  Izquierda
Camara Debajo:  Piso Oscuro
Camara Centro:  Piso Blanco
Camara Izquierda:  Piso Oscuro
Camara Derecha:  Piso Oscuro
Sensor de contacto: No Contacto
Accion:  Rotar +90
```

Sucesivamente el agente va explorando el entorno con el objetivo de mantenerse en las líneas negras

En el siguiente GitHub Repositorio se encuentra el código e información complementaria.

https://github.com/edwinsalazar/ArtificialIntelligence

## IV. Conclusión

- Considerando el tipo de agente de reflejo simple diseñado en esta ocasión, es necesario diseñar adecuadamente la Tabla de Percepción tomando en cuenta todos los sensores de los que dispondrá el agente y estableciendo una acción lógica para cada ocurrencia de sucesos. Su desempeño está estrechamente ligado a la pericia con la que el "creador" pueda establecer las acciones para cada caso.

- El hecho de no disponer de memoria limita la capacidad de acción del agente ya que no es capaz de "aprender" de las acciones previamente desarrolladas, por ende es que el porcentaje de desempeño calculado en este informe puede ser inferior a lo aceptable, siempre dependiendo de la iteración realizada.

- Se puede notar que con la Tabla de Percepción presentada en este informe [Figura 1] el agente es capaz de seguir correctamente las líneas negras presentadas de forma aleatoria en el tablero generado.

## Referencias

[1] G. Eason, B. Noble, and I.N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," *Phil. Trans. Roy. Soc. London,* vol. A247, pp. 529-551, April 1955.