Temas Tratados en el Trabajo Práctico 1

- Diferencia entre Inteligencia e Inteligencia Artificial.
- Concepto de omnisciencia, aprendizaje y autonomía.
- Definición de Agente y sus características. Clasificación de Agentes según su estructura.
- Identificación y categorización del Entorno de Trabajo en tabla REAS.
- Caracterización del Entorno de Trabajo.

Anotaciones

"Acordarse de la definición de agente"

Ejercicios Teóricos

1. Defina con sus propias palabras inteligencia natural, inteligencia artificial y agente.

Inteligencia natural es la inteligencia que posee el ser humano, es creativa, esta basada tanto en experiencias como emociones. La inteligencia artificial por otra parte es la inteligencia asociada a los agentes inteligentes creados por el ser humano. Un agente es todo aquel dispositivo o software que puede percibir el entorno y presenta un cierto nivel de inteligencia.

1. ¿Qué es un agente racional?

Un agente racional es aquel que, para cada secuencia de percepciones que recibe de su entorno, elige la acción que maximiza su medida de rendimiento, basándose en la información provista por esas percepciones, el conocimiento que ya tiene sobre el mundo y las acciones que puede realizar.

1. ¿Un agente es siempre una computadora?

No necesariamente. Decimos que un agente cualquier cosa que tenga cierto nivel de inteligencia, es decir, que reconozca estimulos del entorno y a partir de estos responda en consecuencia

1. Defina Omnisciencia, Aprendizaje y Autonomía.

Omniscinecia: Es la capacidad de conocer el estado real y completo del entorno en todo momento. Es un ideal teórico: ningún agente real puede ser completamente omnisciente, porque el mundo es complejo, dinámico y con información oculta.

Aprendizaje: Es la capacidad de mejorar su rendimiento con la experiencia. Un agente que aprende utiliza sus percepciones no solo para actuar, sino también para ajustar su forma de actuar en el futuro. Detecta patrones, predice consecuencias y optimiza decisiones con el tiempo.

Autonomía: Mide cuánto dependen las acciones del agente de su propia experiencia frente a depender de conocimiento dado de antemano por el diseñador

1. Defina cada tipo de agente en función de su **estructura** y dé un ejemplo de cada categoría.

Agentes reactivos: es un sistema que toma decisiones basadas únicamente en el estado actual del entorno, sin considerar el pasado ni planificar acciones futuras. *Por ejemplo: Termostato, sistemas de deteccion de spam.*

Agentes basados en objetivos: es un tipo de sistema de inteligencia artificial diseñado para alcanzar metas específicas. Estos agentes evalúan su entorno, planifican acciones y toman decisiones basadas en cómo esas acciones afectarán su progreso hacia el objetivo deseado. *Por ejemplo: Sistemas de navegacion GPS, Agentes de atencion al cliente.*

Agentes que aprenden: Los agentes de aprendizaje aprenden continuamente de las experiencias anteriores para mejorar sus resultados. Mediante mecanismos de información sensorial y comentarios, el agente adapta su elemento de aprendizaje a lo largo del tiempo para cumplir con estándares específicos. *Por ejemplo: motores de recomendacion, asistentes personales inteligentes*

1. Para los siguientes entornos de trabajo indique sus **propiedades**:

a. Una partida de ajedrez.

- Totalmente observable
- Determinista
- Secuencial
- Estático
- Discreto
- Multiagente

b. Un partido de baloncesto.

- Parcialmente observable
- Estocástico
- Secuencial
- Dinámico
- Continuo
- Multiagente

c. El juego Pacman.

- Parcialmente observable
- Estocástico
- Secuencial
- Dinámico
- Discreto
- Multiagente

d. El truco.

- Parcialmente observable
- Estocástico
- Secuencial
- Estático
- Discreto
- Multiagente

e. Las damas.

- Totalmente observable
- Determinista
- Secuencial
- Estático
- Discreto
- Multiagente

f. El juego tres en raya.

- Totalmente observable
- Determinista
- Secuencial
- Estático
- Discreto
- Multiagente

g. Un jugador de Pokémon Go.

- Parcialmente observable
- Estocástico
- Secuencial
- Dinámico
- Continuo
- Agente individual

h. Un robot explorador autónomo de Marte.

- Parcialmente observable
- Estocástico
- Secuencial
- Dinámico
- Continuo
- Agente individual

1. Elabore una tabla REAS para los siguientes entornos de trabajo:

    a. Crucigrama.

Caso	Medidas de rendimiento	Entorno	Actuadores	Sensores
Crucigrama	-Palabras Correctas -Tiempo	-Tablero -Pistas	Lapicera	Vista

b. Taxi circulando.

Caso	Medidas de rendimiento	Entorno	Actuadores	Sensores
Taxi Circulando	-Tiempo hasta el destino	Avenidas, Calles, Peatones, Clientes	Pies, manos	Vista

c. Robot clasificador de piezas

Caso	Medidas de rendimiento	Entorno	Actuadores	Sensores
Robot Clasificador de piezas	-Piezas correctamente clasificadas -Tiempo	Lugar de trabajo / Fábrica	Motores, solenoides, cilindros neumáticos	Encoders, Cámara

Ejercicios Prácticos

1. La Hormiga de Langton es un agente capaz de modificar el estado de la casilla en la que se encuentra para colorearla o bien de blanco o de negro. Al comenzar, la ubicación de la hormiga es una casilla aleatoria y mira hacia una de las cuatro casillas adyacentes. Si...

- ... la casilla sobre la que está es blanca, cambia el color del cuadrado, gira noventa grados a la derecha y avanza un cuadrado.
- ... la casilla sobre la que está es negra, cambia el color del cuadrado, gira noventa grados a la izquierda y avanza un cuadrado.

  Caracterice el agente con su tabla REAS y las propiedades del entorno para después programarlo en Python:

  ¿Observa que se repite algún patrón? De ser así, ¿a partir de qué iteración?

Propiedades del Entorno

- Totalmente observable
- Determinista
- Secuencial
- Estático
- Discreto

- Agente individual

Tabla REAS

Caso	Medidas de rendimiento	Entorno	Actuadores	Sensores
Hormiga de Langton	Avanzar y modificar celdas según reglas	Rejilla bidimensional discreta	Giro, avance, cambio de color	Color de la celda actual

In [1]:

```
import pygame
import numpy as np

# --- Config ---
CELL_SIZE = 8
GRID_WIDTH = 100
GRID_HEIGHT = 80
FPS = 60

# Colors
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
ANT_COLOR = (255, 0, 0)

# Directions: 0 = up, 1 = right, 2 = down, 3 = left
DIRS = [(0, -1), (1, 0), (0, 1), (-1, 0)]

pygame.init()
screen = pygame.display.set_mode((GRID_WIDTH * CELL_SIZE, GRID_HEIGHT * CELL_SIZE))
pygame.display.set_caption("Hormiga de Langton")
clock = pygame.time.Clock()

# Grid: 0 = white, 1 = black
grid = np.zeros((GRID_HEIGHT, GRID_WIDTH), dtype=int)

# Ant position and direction
ant_x, ant_y = GRID_WIDTH // 2, GRID_HEIGHT // 2
ant_dir = 0  # facing up

running = True
paused = False

def draw_grid(surface, grid, ant_pos):
    surface.fill(WHITE)
    for y in range(GRID_HEIGHT):
        for x in range(GRID_WIDTH):
            if grid[y, x] == 1:
                pygame.draw.rect(surface, BLACK, (x * CELL_SIZE, y * CELL_SIZE, CELL
    # Draw ant
    ax, ay = ant_pos
    pygame.draw.rect(surface, ANT_COLOR, (ax * CELL_SIZE, ay * CELL_SIZE, CELL_SIZE,
    pygame.display.flip()

def step(grid, ant_x, ant_y, ant_dir):
    if grid[ant_y, ant_x] == 0:  # white → turn right
        ant_dir = (ant_dir + 1) % 4
        grid[ant_y, ant_x] = 1
    else:  # black → turn left
        ant_dir = (ant_dir - 1) % 4
        grid[ant_y, ant_x] = 0
```

```
    dx, dy = DIRS[ant_dir]
    ant_x = (ant_x + dx) % GRID_WIDTH
    ant_y = (ant_y + dy) % GRID_HEIGHT
    return grid, ant_x, ant_y, ant_dir

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE:
                paused = not paused
            elif event.key == pygame.K_c:
                grid[:] = 0
                ant_x, ant_y = GRID_WIDTH // 2, GRID_HEIGHT // 2
                ant_dir = 0

    if not paused:
        grid, ant_x, ant_y, ant_dir = step(grid, ant_x, ant_y, ant_dir)

    draw_grid(screen, grid, (ant_x, ant_y))
    clock.tick(FPS)

pygame.quit()
```

```
pygame 2.6.1 (SDL 2.28.4, Python 3.11.9)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

1. El Juego de la Vida de Conway consiste en un tablero donde cada casilla representa una célula, de manera que a cada célula le rodean 8 vecinas. Las células tienen dos estados: están *vivas* o *muertas*. En cada iteración, el estado de todas las células se tiene en cuenta para calcular el estado siguiente en simultáneo de acuerdo a las siguientes acciones:

- Nacer: Si una célula muerta tiene exactamente 3 células vecinas vivas, dicha célula pasa a estar viva.
- Morir: Una célula viva puede morir sobrepoblación cuando tiene más de tres vecinos alrededor o por aislamiento si tiene solo un vecino o ninguno.
- Vivir: una célula se mantiene viva si tiene 2 o 3 vecinos a su alrededor.

  Caracterice el agente con su tabla REAS y las propiedades del entorno para después programarlo en Python:

Propiedades del Entorno

- Totalmente observable
- Determinista
- Secuencial
- Estático
- Discreto
- Multiagente (según interpretación)

Tabla REAS

Caso	Medidas de rendimiento	Entorno	Actuadores	Sensores
Conways Game of Life	Modificar celdas según reglas	Rejilla bidimensional discreta	Vivo o muerto	Estado de las celdas vecinas

In [2]:

```
import pygame
import numpy as np

# --- Config ---
Celula_SIZE = 10
Grilla_WIDTH = 80
Grilla_HEIGHT = 60
FPS = 30

# Colors
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)

pygame.init()
screen = pygame.display.set_mode((Grilla_WIDTH * Celula_SIZE, Grilla_HEIGHT * Celula
pygame.display.set_caption("Conway's Game of Life")
clock = pygame.time.Clock()

# Grilla: 0 = dead, 1 = alive
Grilla = np.zeros((Grilla_HEIGHT, Grilla_WIDTH), dtype=int)

running = True
paused = True

def draw_Grilla(surface, Grilla):
    surface.fill(BLACK)
    for y in range(Grilla_HEIGHT):
        for x in range(Grilla_WIDTH):
            if Grilla[y, x] == 1:
                rect = pygame.Rect(x * Celula_SIZE, y * Celula_SIZE, Celula_SIZE, Ce
                pygame.draw.rect(surface, WHITE, rect)
    pygame.display.flip()

def update_Grilla(Grilla):
    GrillaNueva = np.copy(Grilla)
    for y in range(Grilla_HEIGHT):
        for x in range(Grilla_WIDTH):
            # Count alive neighbors
            neighbors = np.sum(Grilla[max(0, y-1):min(Grilla_HEIGHT, y+2),
                                      max(0, x-1):min(Grilla_WIDTH, x+2)]) - Grilla[y,
            # Rules
            if Grilla[y, x] == 1 and (neighbors < 2 or neighbors > 3):
                GrillaNueva[y, x] = 0
            elif Grilla[y, x] == 0 and neighbors == 3:
                GrillaNueva[y, x] = 1
    return GrillaNueva

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE:
                paused = not paused
            elif event.key == pygame.K_c:
                Grilla[:] = 0
```

```
        elif event.type == pygame.MOUSEBUTTONDOWN:
            mx, my = pygame.mouse.get_pos()
            gx, gy = mx // Celula_SIZE, my // Celula_SIZE
            Grilla[gy, gx] = 1 - Grilla[gy, gx]  # Toggle Celula

    if not paused:
        Grilla = update_Grilla(Grilla)

    draw_Grilla(screen, Grilla)
    clock.tick(FPS)

pygame.quit()
```

Bibliografía

Russell, S. & Norvig, P. (2004) *Inteligencia Artificial: Un Enfoque Moderno*. Pearson Educación S.A. (2a Ed.) Madrid, España

Poole, D. & Mackworth, A. (2023) *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press (3a Ed.) Vancouver, Canada