Beltran Magaña Ulises – 20170610

# Agentes Deliberativos

Tipo de agente inteligente que toma decisiones basándose en un modelo interno del mundo y en un proceso de razonamiento lógico. Estos agentes evalúan las posibles acciones considerando sus creencias, deseos y objetivos para seleccionar la más adecuada.

## Características principales:

- **Modelo del entorno**: Tiene una representación interna del mundo.
- **Razonamiento basado en objetivos**: Planifica acciones para lograr metas.
- **Capacidad de toma de decisiones**: Evalúa diferentes alternativas antes de actuar.
- **Adaptabilidad**: Ajusta su comportamiento si cambian las condiciones del entorno.

## Ejemplos:

- **Robots Autónomos**: Robots de limpieza (como Roomba) que planifican rutas para cubrir toda una habitación.
- **Asistentes Virtuales**: Siri, Alexa o Google Assistant, que razonan sobre tus preguntas para ofrecerte respuestas útiles.
- **Vehículos Autónomos**: Coches autoconducidos que analizan su entorno para decidir si frenar, girar o acelerar.
- **Sistemas de Diagnóstico Médico**: Software que evalúa síntomas y antecedentes para sugerir diagnósticos.

## Código:

```
import random

class AgenteCarrera:
    def __init__(self, distancia_meta=100):
        self.posicion = 0
        self.distancia_meta = distancia_meta

    def percibir_entorno(self):
        if self.posicion >= self.distancia_meta:
            return "meta"
        return random.choice(["despejado", "obstáculo"])

    def decidir_accion(self, entorno):
        if entorno == "meta":
            return "parar"
        elif entorno == "obstáculo":
            return "saltar"
        else:
            return "avanzar"

    def actuar(self, accion):
        if accion == "avanzar":
            self.posicion += 10
            print("🏃 Avanzando... Posición:", self.posicion)
        elif accion == "saltar":
            print("🤸 Saltando obstáculo...")
        elif accion == "parar":
            print("🎉 ¡Meta alcanzada! Carrera finalizada.")

    def correr(self):
        while True:
            entorno = self.percibir_entorno()
            accion = self.decidir_accion(entorno)
            self.actuar(accion)
            if accion == "parar":
                break

carrera = AgenteCarrera()
carrera.correr()
```

# Agentes Híbridos

Combina múltiples enfoques de razonamiento, como el deliberativo, reactivo, basado en objetivos y aprendizaje, para adaptarse de manera eficiente a entornos complejos y dinámicos. Integra diferentes arquitecturas para aprovechar las ventajas de cada una

## Características principales:

- **Múltiples capas:** Generalmente organizados en capas para gestionar distintas tareas.
- **Razonamiento y reacción:** Responde rápidamente a estímulos mientras planifica a largo plazo.
- **Adaptabilidad:** Aprende y ajusta su comportamiento frente a cambios en el entorno.

## Ejemplos:

- **Vehículos Autónomos:** Combinan agentes reactivos (para evitar obstáculos) y deliberativos (para planificar rutas).
- **Robótica Social:** Robots que interactúan con humanos, usando agentes basados en aprendizaje y comportamiento reactivo.
- **Videojuegos de Estrategia**: Personajes controlados por IA que alternan entre patrones reactivos y planificación estratégica.
- **Recomendadores Inteligentes**: Plataformas como Netflix combinan aprendizaje automático y razonamiento simbólico para sugerencias.

## Código:

```python
import random

class AgenteHibrido:
    def __init__(self, meta=100):
        self.posicion = 0
        self.meta = meta
        self.pasos = 0

    # Capa reactiva: percepción inmediata
    def percibir_entorno(self):
        if self.posicion >= self.meta:
            return "meta"
        return random.choice(["despejado", "obstáculo"])

    # Capa deliberativa: planificación de acciones
    def decidir_accion(self, entorno):
        if entorno == "meta":
            return "parar"
        elif entorno == "obstáculo":
            return "saltar"
        else:
            return "avanzar"

    # Capa de aprendizaje: contar pasos
    def aprender(self):
        self.pasos += 1

    # Capa de acción: ejecuta la decisión
    def actuar(self, accion):
        if accion == "avanzar":
            self.posicion += 10
            print(f"🏃 Avanzando... Posición: {self.posicion}")
        elif accion == "saltar":
            print("🤸 Saltando obstáculo...")
        elif accion == "parar":
            print(f"🎉 ¡Meta alcanzada en {self.pasos} pasos!")
```

```
    def correr(self):
        while True:
            entorno = self.percibir_entorno()
            accion = self.decidir_accion(entorno)
            self.aprender()
            self.actuar(accion)
            if accion == "parar":
                break

carrera = AgenteHibrido()
carrera.correr()
```