Universidad Tecnológica de la Mixteca

Agente Inteligente

Inteligencia Artificial

Ingeniería en Computación

Profesor

Juan Juarez Fuentes

Integrantes

Bedolla Martinez Beatriz

González Valentín Armando Fabián

Grupo

802-A

Índice

Introducción

En inteligencia artificial se le denomina agente inteligente, que es capaz de percibir su entorno, procesar esta percepción y hacerlo en su entorno de forma razonable (es decir, de forma correcta y tiende a maximizar los resultados esperados) La entidad que reacciona o actúa. Percibe el entorno mediante el uso de sensores que funcionan en el mismo entorno (elementos que responden a estímulos realizando acciones). Las entidades físicas o virtuales pueden considerarse agentes. Aunque el término "agente racional" se refiere a agentes artificiales en el campo de la IA, los animales, incluidos los humanos, también pueden considerarse "agentes racionales".

En informática, el término agente inteligente se puede utilizar para referirse a un agente de software con cierta inteligencia, independientemente de si no es un agente racional en la definición de Russell y Norvig. Por ejemplo, los programas autónomos (a veces llamados robots) que se utilizan para la asistencia del operador o la minería de datos también se denominan "agentes inteligentes".

Contexto

Los agentes inteligentes se describen esquemáticamente como sistemas funcionales abstractos. Por lo tanto, los agentes inteligentes a veces se denominan agentes inteligentes abstractos (AIA) para distinguirlos de las implementaciones del mundo real (como sistemas informáticos, sistemas biológicos u organizaciones). Algunas definiciones de agentes inteligentes enfatizan su autonomía, por lo que prefieren el término "agente inteligente". Otros (especialmente Russell y Norvig (2003)) consideran el comportamiento orientado a objetivos como la esencia de la inteligencia y prefieren el término "agente racional" tomado de la economía.

A continuación se muestran ejemplos de agentes físicos y agentes virtuales

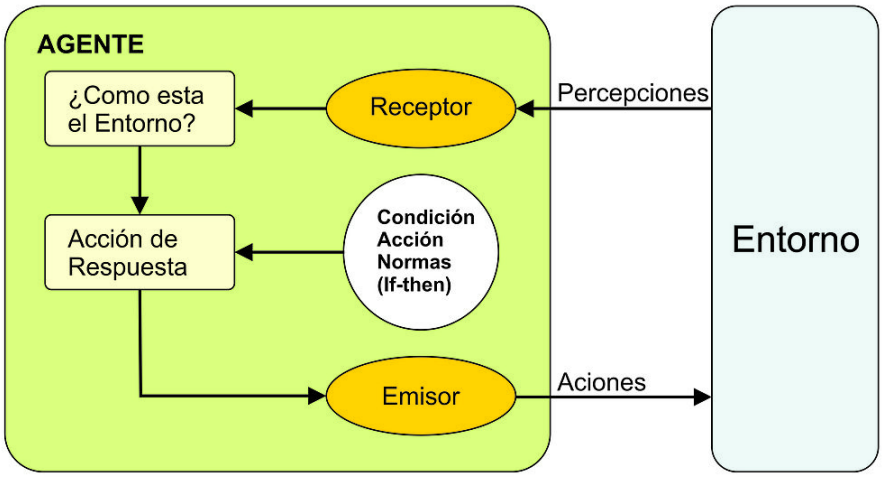
Agentes físicos
Una computadora especial que se usa para controlar un helicóptero para operaciones peligrosas
Robot de comportamiento variable autoajustable (ya sea que su comportamiento esté determinado por software o directamente integrado en equipos electrónicos),

Una computadora que ejecuta software de diagnóstico médico y muestra los resultados en la pantalla para ayudar a los médicos a tomar decisiones.

Agentes virtuales
software de robot que simula personajes en juegos de computadora, como un jugador de ajedrez, un jugador de fútbol del otro lado o un piloto de carreras.
Software de descubrimiento de modo de Internet que solo interactúa con otro software.

Desarrollo

Como práctica se implementa el desarrollo de un agente simple, el cual consta de un mini demo de juego de computadora que simula ser una aspiradora(agente virtual). El objetivo es llenar un tablero de forma aleatoria con objetos que simulan ser basura, el agente debe ser capaz de detectar colisiones con su entorno que en este caso son los límites del tablero y los objetos en el mismo.



Por cada colisión debe desaparecer de la pantalla dicho objeto y mostrarse en un contador.

El demo ha sido implementado en el lenguaje de programación JavaScript con lógica para programación de videojuegos. Se describirán a continuación los pasos del desarrollo como así también la lógica usada.

Implementación

Para cumplir con el objetivo se tuvo que hacer un diseño de lógica puesto que la función del agente es interactuar con el ambiente y producir una respuesta, en este caso la respuesta sería desaparecer los objetos de la pantalla al colisionar con ellos. Este punto hizo uso de teoría de colisiones para detectar la interacción entre objetos.
Existen varios métodos y algoritmos sobre colisiones, pero en este caso se trata de un agente simple por lo que el algoritmo es bastante sencillo.

Creación del entorno

Tablero y marcador

La implementación del tablero y marcador consta de dos funciones que por medio de ciclos realizan el trazado del tablero dibujando directamente sobre el canvas. Se hace un cálculo de coordenadas para poder trazar cada casilla de forma correcta sobre el canvas, así también unas condicionales para alternar casillas pares e impares.

```
function pintaTab(){
    for (let index = 0; index < numCelda; index++) {
        for (let index2 =0; index2 < numCelda; index2++) {
            contexto.beginPath();
            celdaX = index *(tamCelda + celdaPadding)+ celdaOffsetLeft;//cordenada en X
            celdaY = index2 *(tamCelda + celdaPadding)+ celdaOffsetTop;//cordenada en Y
            contexto.rect(celdaX, celdaY, tamCelda, tamCelda);
            //fila impar
            if(((index2 % 2) != 0) ){
                if(((index % 2) != 0)){contexto.fillStyle = cImpar;}//columno impar
                else{contexto.fillStyle = cPar;}                    //columna par
            }
```

```
            //fila par
            if(((index2 % 2) == 0)){
                if(((index % 2) == 0)){contexto.fillStyle = cImpar;} //columno impar
                else{contexto.fillStyle = cPar;}                     //columna par
            }
            contexto.fill();
            contexto.closePath();
        }
    }
}
```

En función del **marcador** solo se imprime texto, contadores y una condicional para restablecer el puntaje..

```
function marcador (){
    contexto.fillStyle = "white";
    contexto.font = "italic bold 25pt Times New Roman";
    contexto.fillText("Puntos: "+ puntos+ "    objetetos: "+ objetos, 35, 50,canvas.width-10);
    if (puntos == fila*col){
        puntos = 0;
    }
}
```

Movimiento del agente

Para realizar el desplazamiento se implementaron 5 funciones; 4 para indicar las direcciones hacia donde se va a desplazar por medio de coordenadas..
Dos funciones más para que el agente permanezca dentro del perímetro.

```
//funciones para la dirección del agente
function movDerecha(){ ejeX = ejeX + 10;}
function movIzquierda(){ ejeX=ejeX - 10;}
function movArriba(){ ejeY = ejeY - 10;}
function movAbajo(){ ejeY = ejeY + 10;}
```

```
//metodos para veificar que el agente no salga del tablero
function perimetroX(){
    if(ejeX <=-32){ejeX =-32;}
    if(ejeX >= canvas.width-128){ejeX = canvas.width-128;}
}
function perimetroY(){
    if(ejeY <=30){ejeY = 30;}
    if(ejeY >= canvas.height-128){ejeY = canvas.height-128;}
}
```

```
//metodo para escuchar los eventos del teclado
function mover(e){
    var evento = e?e:event;
    var tecla = window.Event?evento.which:evento.keyCode;
    if (tecla == 39) {estado = 2; movDerecha();}
    if (tecla == 37) {estado = 1; movIzquierda();}
    if (tecla == 40) {estado = 4; movAbajo();}
    if (tecla == 38) {estado = 3; movArriba();}

}
```

Pintando el Agente

Para dibujar al agente se hace uso de estados los cuales provienen de la función **mover()**, según la dirección se dibuja la posición del agente (izquierda, derecha, abajo, arriba) con sus respectivas coordenadas.

```
//dibuja el agente
function pintaAgente(){
    agente = new Image();                       //se carga la imagen del agente
    agente.src = "/dev/assets/sprite1.png";
    switch(estado){
        case 0: agente.addEventListener("load", function(){   //cuando está quieto
                contexto.drawImage(agente, 0 , 0, 64, 64, ejeX, ejeY, 128, 128);});
                break;
         case 1:
              agente.addEventListener("load", function(){      //hacia la izquierda
              contexto.drawImage(agente, (index*64) , 64, 64, 64, ejeX, ejeY, 128, 128);});
              index = index + 1;
              if (index > 3) {index = 0;} break;
         case 2:
           agente.addEventListener("load", function(){         //hacia la derecha
           contexto.drawImage(agente, (index*64) , 128, 64, 64, ejeX, ejeY, 128, 128);});
           index = index + 1;
           if (index > 3) {index = 0;} break;
```

```
                if (index > 3) {index = 0;} break;
         case 4:
                    agente.addEventListener("load", function(){     //hacia abajo
                    contexto.drawImage(agente, (index*64) , 0, 64, 64, ejeX, ejeY, 128, 128);});
                    index = index + 1;
                    if (index > 3) {index = 0;} break;
        default: estado = 0 ;
                }
}
```

Dibujando objetos

Para dibujar los objetos usamos una matriz la cual en cada posición contiene tres propiedades {x : 0, y: 0, status: 1}, estas propiedades nos sirven para guardar las coordenadas que se generan de forma aleatoria donde se deben colocar los objetos, la variable status sirve para indicar que objeto a colisionado con el agente.

```
function posiciones (){
     fila = Math.floor(Math.random()* (5 -1)+1);
     col = Math.floor(Math.random()*(5 -1)+1);
     objetos = fila*col;
    for (let c = 0; c < numBasura; c++) {
        cordenadas[c] = [];
        for (let r = 0; r <numBasura; r++) {
            cordenadas[c][r] = {x : 0, y : 0, status:1};
        };
    }
    for (let i = 0; i < fila; i++) {
        for (let j = 0; j < col; j++) {
            cordenadas[i][j].x = Math.floor(Math.random()* (600 -200)+1);
            cordenadas[i][j].y = Math.floor(Math.random()* (800 -200)+1);
            console.log(cordenadas[i][j].x + ", "+cordenadas[i][j].y);
        }
    }
}
```

Si la variable status cambia a 0 indica que el objeto ha colisionado con el agente por lo tanto no desaparece de pantalla.

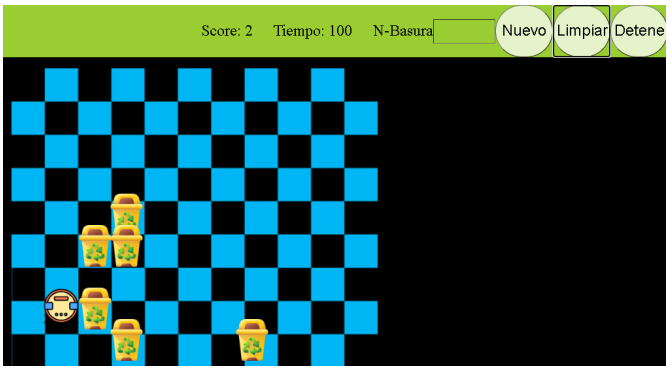
```
//dibuja la basura
function pintaBasura(){
    for (let index = 0; index < cordenadas.length; index++) {
        for (let index2 = 0; index2 < cordenadas.length; index2++) {
            if (cordenadas[index][index2].status == 1) {
               basura = new Image();
               basura.src = "/dev/assets/basura1.png";
               basura.addEventListener("load", function (){
                if( cordenadas[index][index].x!= 0 && cordenadas[index][index2].y != 0){
                 contexto.drawImage(basura,0,0,64,64,(cordenadas[index][index2].x), (cordenadas[index][index2].y),64,64)
                }
               });
            }
        }
    }
}
```

Sin duda la función más interesante en cuanto a funcionalidad y no a complejidad es la función **colisión()** ya que esta es la que le da sentido al agente en sí, detectar las colisiones le permite interactuar con el entorno y por lo tanto producir una respuesta. El algoritmo es simple, basta con comparar las coordenadas del agente y el objeto en colisión para verificar si las coordenadas del agente se encuentran dentro del rango de las coordenadas del objeto, si es así entonces hay una colisión y aumentan los puntos.

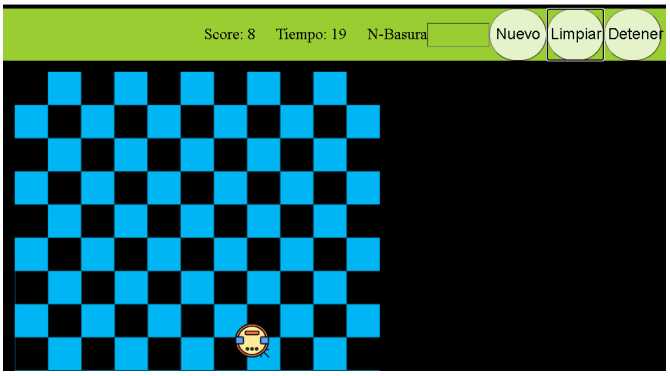
```
function colision() {
    for(let c= 0; c< cordenadas.length; c++) {
        for(let r= 0; r< cordenadas.length; r++) {
            var b = cordenadas[c][r];
         if(b.status == 1) {
                 if((b.x < (ejeX + 32)) && ((b.x + 64) > ejeX)){
                    if(((b.y + 64) > ejeY ) && (b.y < (ejeY + 32))) {
                       b.status = 0;
                       puntos +=1;
                       objetos -=1;
                       if((objetos < 0) || (puntos > (fila*col))){
                           objetos = 0;
                           puntos = fila*col;
                        }
                    }
                }
            }
        }
    }
}
```

Resultados

(estado de inicio)

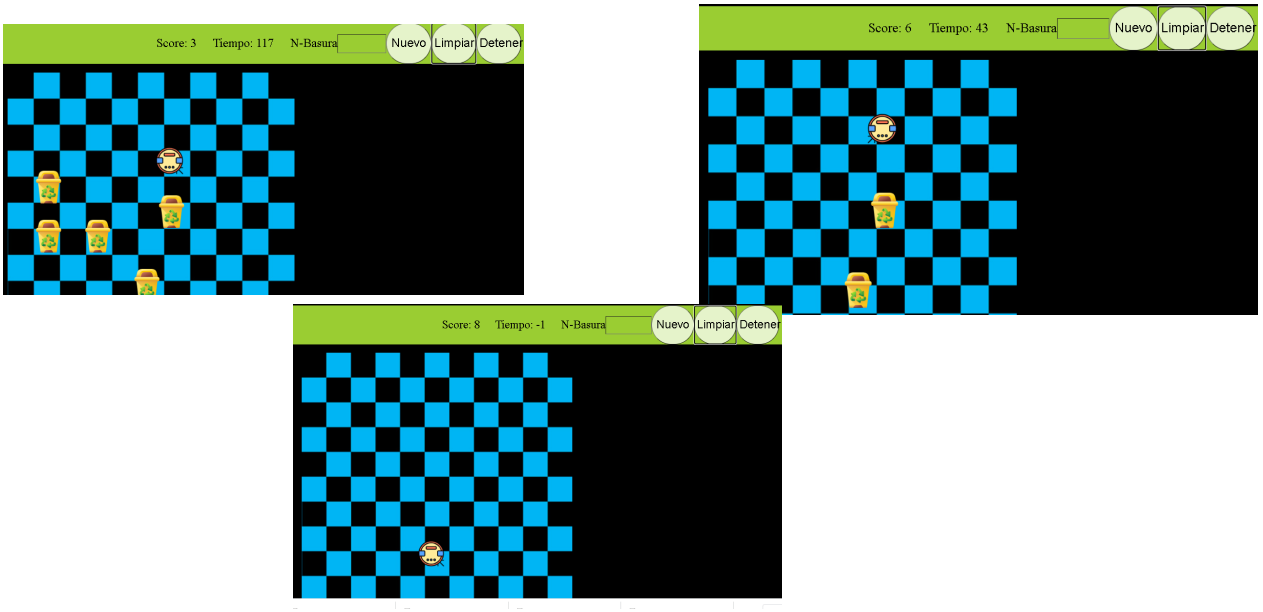


(estado de recolección)



(estado final)

Segunda corrida



Conclusión

En está práctica comprendimos realmente que es un agente inteligente y su funcionamiento, como es que debe de interactuar con su entorno y devolver una acción. Si bien este demo de computadora es un ejemplo básico, es de mucha ayuda para adentrarnos en el mundo de los agentes inteligentes.

Debemos recordar que la aplicación de los agentes toma partido en gran parte de servicios de nuestra vida diaria, desde el Asistente de Google hasta Mapas e indicaciones de la web.

Es evidente entonces la necesidad, que se gesta en las organizaciones tanto públicas como privadas, de profesionales que cuenten con las competencias requeridas para diseñar y aplicar soluciones en IA que permitan el crecimiento de las empresas, mejorando sus procesos y/o productos y habilitando a las mismas para la transformación digital.