Agente tele reactivo

Tele reactive agent.

Autor 1: Cediel Autor 2: Santiago Londoño López, Autor 3: Juan Pablo Narváez Arteaga

*Ingeniería de sistemas y Computación, Universidad Tecnológica de Pereira, Pereira, Colombia*

Correo-e: santigo.londono@utp.edu.co

***Resumen*— Un agente tele reactivo es un agente inteligente que toma sus decisiones teniendo en cuenta su estado anterior en este artículo se va a presentar una solución para el juego Snake haciendo uso de estos agentes.**

***Palabras clave—* Agente inteligente, Snake, Python.**

***Abstract*— A tele reactive agent is an intelligent agent that makes its decisions considering its previous state. In this article we are going to present a solution for the game Snake making use of these agents.**

***Key Word* —** **Smart agent, Snake, Python.**

1. INTRODUCCIÓN

La inteligencia artificial cuenta con diferentes ramas que se usan para la resolución de problemas específicos en este articulo científico se hablara del agente tele reactivo que es, sus usos, etc. Y una implementación de este realizada en Python para la resolución del juego Snake.

1. CONTENIDO

**¿Agente inteligente, que es?**

Un agente en inteligencia artificial puede ser considerado como un ente en software que, basado en un conocimiento previo adquirido, toma decisiones o realiza una serie de operaciones para resolver un problema o llevar a cabo una tarea.

Por lo general este conocimiento lo genera evaluando su ambiente a través de sensores y las decisiones que toma se pueden considerar efectores.

Estos pueden ser considerados como entes individuales ya que tiene control sobre si mismos.

Su comportamiento como se puede ver tiene 3 partes fundamentales.

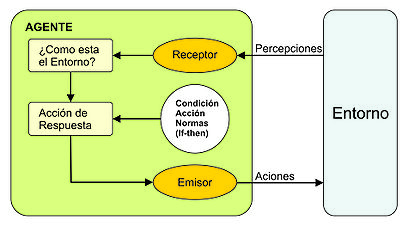
-La inteligencia: ya que puede razonar a partir del conocimiento que va adquiriendo.

-La interactividad: se presenta cuando el agente interactúa con su ambiente.

-La autonomía: Ya que el agente puede tomar decisiones “optimas” basándose en su criterio, sin necesidad de otra opinión.

-Sociabilidad: Por el hecho de que un agente puede interactuar con otros agentes.

Agente inteligente simple:



Jhanlos - Template:Instituto Informatico de Ciencias Cognoscentes (Arequipa - Peru)

**Snake, videojuego.**

Es un videojuego que salió en la década de los 70’s, que ha sido muy reconocido desde entonces, volviéndose un clásico entre los videojuegos. Esta popularidad la obtuvo en 1998, cuando el juego adquirió una audiencia masiva al convertirse en un juego instalado de fabrica en los teléfonos celulares producidos por la empresa Nokia.

El juego consiste en que el jugador controla un ser que está compuesto por una serie de bloques que replica el movimiento de una serpiente o gusano. El ser se mueve en una zona delimitada. La tar4ea del jugador es llevar a este ser por la zona evitando chocar con las paredes y contra si mismo para recoger comida que lo hará crecer provocando que sea cada vez más difícil la movilidad del mismo por el mapa.

**Funcionamiento del código**

Se usan librerías como pygame para el de interfaz gráfica, simpleai para manejo de la inteligencia artificial, random para generación de comida, entre otras:

import pygame

import random

import time

import math

from simpleai.search import SearchProblem, astar, depth\_first, breadth\_first

lo primero es definir las diferentes clases del programa como clases tenemos.

La clase Snake que contara con los atributos de cabeza y cola que serán tratados como Sprites facilitados por la librería pygame

class Snake:

    def \_\_init\_\_(self, lim\_filas, lim\_columnas, tam\_x, tam\_y, game):

        self.head = Head([0, 0], tam\_x, tam\_y)

        self.body = []

        self.game = game

        self.game.allSprites.add(self.head)

        self.lim\_filas = lim\_filas - 1

        self.lim\_col = lim\_columnas - 1

        self.tam\_x = tam\_x

        self.tam\_y = tam\_y

        self.dir = 4

La clase Snake cuenta con métodos para el crecimiento de la serpiente, el movimiento de su cabeza, para la finalización del juego por chocar con los límites de la zona y el movimiento del cuerpo de la serpiente respectivamente.

 def grow(self):

        """

        Funcion que le agregara un cuadro a la serpiente cuando coma

        """

        newBody = None

        if self.body: #Si tiene elementos el cuerpo, le agregamos uno nuevo que sea igual al ultimo

            newBody = Body(self.body[-1].pos, self.tam\_x, self.tam\_y)

        else:

            newBody = Body(self.head.pos\_head, self.tam\_x, self.tam\_y)

        self.game.allSprites.add(newBody)

        self.game.body.add(newBody)

        self.body.append(newBody)

        #print (len(self.body))

    def movement\_head2(self, dir):

        """

        1. Arriba, 2.Abajo, 3.Izquierda, 4.Derecha, se debe tener en cuenta que

        si se le da arriba pero arriba hay cuerpo no puede hacer este desplazamiento

        Se debe recordar que el mapa es una matriz, fila columna, no posicion x, y

        """

        if self.body:

            self.movement\_body()

        self.dir = dir

        if dir == 1:

            self.head.pos\_head = [self.head.pos\_head[0] - 1, self.head.pos\_head[1]]

        elif dir == 2:

            self.head.pos\_head = [self.head.pos\_head[0] + 1, self.head.pos\_head[1]]

        elif dir == 3:

            self.head.pos\_head = [self.head.pos\_head[0], self.head.pos\_head[1] - 1]

        else:

            self.head.pos\_head = [self.head.pos\_head[0], self.head.pos\_head[1]  + 1]

        return self.keepPlaying()

    def keepPlaying(self):

        #Condicional de fin del juego por salirse de los limites

        if (self.head.pos\_head[0] < 0 or self.head.pos\_head[1] < 0 or

            self.head.pos\_head[0] > self.lim\_filas or self.head.pos\_head[1] > self.lim\_col):

            return False

        elif self.head.pos\_head in [i.pos for i in self.body]: #Fin del juego por colision

            return False

        else:

            return True

    def movement\_body(self):

        newBody = Body(self.head.pos\_head, self.tam\_x, self.tam\_y)

        self.game.allSprites.add(newBody)

        self.body.insert(0, newBody)

        self.game.allSprites.remove(self.body[-1])

        self.body.pop(-1)

La clase siguiente es el juego como tal

class Game:

    def \_\_init\_\_(self, lim\_filas, lim\_columnas, tam\_x, tam\_y):

        #Inicializacion

        pygame.init()

        self.lim\_filas = lim\_filas

        self.lim\_columnas = lim\_columnas

        self.tam\_x = tam\_x

        self.tam\_y = tam\_y

        self.gameDisplay = pygame.display.set\_mode([WIDTH, HEIGHT])

        self.clock = pygame.time.Clock()

        self.running = True

En ella se definen la matriz y el tamaño de la zona para jugar

Tiene métodos para crear un nuevo juego, crear el mapa, y darle entendimiento a la serpiente para que sepa como se debe de mover para alcanzar la comida, también se actualiza para mostrar cada nuevo movimiento y es donde se da la gestión de los eventos del programa.

 def new(self):

        #Empieza un nuevo juego

        self.dir = 4

        self.allSprites = pygame.sprite.Group()

        self.body = pygame.sprite.Group()

        self.foods = pygame.sprite.Group()

        self.snake = Snake(self.lim\_filas, self.lim\_columnas, self.tam\_x, self.tam\_y, self) #Enviamos la informacion de la clase game

        self.controlSearchProblem()

    def createMap(self):

        map = []

        #print(self.food.pos)

        for i in range(self.lim\_filas):

            row = []

            for j in range(self.lim\_columnas):

                if [i, j] == self.snake.head.pos\_head:

                    row.append('h')

                elif [i, j] in [i.pos for i in self.snake.body]:

                    row.append('b')

                elif [i, j] == self.food.pos:

                    #print(i, j)

                    row.append('f')

                else:

                    row.append('\*')

            map.append(tuple(row))

        map = tuple(map)

        return map

    def controlSearchProblem(self):

        self.createFood()

        result = self.searchOne()

        try:

            self.actions = [i[0] for i in result.path()[1:]]

            print(self.actions)

        except:

            print("No puede llegar a la comida")

            #map = self.createMap()

            #for i in map:

                #print(i)

        #print("hola")

        #print(self.actions)

        else:

            self.playWithSearchProblem()

    def playWithSearchProblem(self):

        self.playing = True

        """

        print("Goal: ", self.food.pos)

        #print("Head:", self.snake.head.pos\_head)

        print("Actions: ", self.actions)

        map = self.createMap()

        for i in map:

            print(i)

        """

        while self.playing and self.actions: #and i < 10:

            self.clock.tick(20)

            self.events()

            self.update()

            self.draw()

        if self.playing:

            self.controlSearchProblem()

    def searchOne(self):

        food = (self.food.pos[0], self.food.pos[1])

        head = (self.snake.head.pos\_head[0], self.snake.head.pos\_head[1])

        body = [i.pos for i in self.snake.body]

        if body:

            body = [tuple(i) for i in body]

        else:

            body = ()

        state = (head, tuple(body))

        problem = SnakeProblem(state, food, [self.lim\_filas, self.lim\_columnas])

        #result = depth\_first(problem, graph\_search = True)

        result = astar(problem, graph\_search = False)

        return result

    def update(self):

        #Movimiento de la serpiente

        self.transformAction()

        self.actions.pop(0)

        self.playing  = self.snake.movement\_head2(self.dir) and self.playing

        ls\_col = pygame.sprite.spritecollide(self.snake.head, self.foods, True)

        if ls\_col:

            self.snake.grow()

        self.allSprites.update()

    def transformAction(self):

        if self.actions[0] == "up":

            self.dir = 1

        elif self.actions[0] == "down":

            self.dir = 2

        elif self.actions[0] == "left":

            self.dir = 3

        elif self.actions[0] == "right":

            self.dir = 4

    def events(self):

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                #Se termina de jugar y de correr

                if self.playing:

                    self.playing = False

                self.running = False

    def draw(self):

        #ciclo de dibujo

        self.gameDisplay.fill([0, 0, 0])

        self.allSprites.draw(self.gameDisplay)

        pygame.display.flip()

    def createFood(self):

        posNewFood = [random.randrange(0, self.lim\_filas),

                      random.randrange(0, self.lim\_columnas)]

        while (posNewFood == self.snake.head.pos\_head or

               posNewFood in [i.pos for i in self.snake.body]):

               #Para que la comida no aparexca donde este la serpiente

               posNewFood = [random.randrange(0, self.lim\_filas),

                             random.randrange(0, self.lim\_columnas)]

        food = Food(posNewFood, self.tam\_x, self.tam\_y)

        self.food = food

        #print("new food", self.food.pos)

        self.foods.add(food)

        self.allSprites.add(food)

    def showGoScreen(self):

        #Para terminar o seguir el juego

        map = self.createMap()

        print(self.dir)

        for i in map:

            print(i)

        time.sleep(2)

Y por último el main.

if \_\_name\_\_ == '\_\_main\_\_':

    lim\_filas = 10

    lim\_columnas = 10

    tam\_x = int(WIDTH/lim\_columnas)

    tam\_y = int(HEIGHT/lim\_filas)

    g = Game(lim\_filas, lim\_columnas, tam\_x, tam\_y)

    while g.running:

        """

        Este ciclo maneja cada vez que se acaba el juego, ya que dentro hay otro

        ciclo que maneja el juego como tal

        """

        g.new()

        g.showGoScreen()

1. CONCLUSION

La inteligencia artificial puede ser usada para abarcar un gran abanico de resolución de problemas de razonamiento con cierto nivel de aceptación, en el programa realizado solo se tomaba en cuenta el estado anterior por tanto no da la mejor manera de resolver el problema (juego de Snake).

RECOMENDACIONES

REFERENCIAS

1. (2019, octubre 1). Recuperado de <https://es.scribd.com/document/423038318/Agente-Tele-Reactivo>
2. (2019, octubre 1). Recuperado de <http://www.elprofesionaldelainformacion.com/contenidos/1999/abril/agentes_inteligentes_definicion_y_tipologia_los_agentes_de_informacion.html>
3. (2019, octubre 1). Recuperado de <http://www.itnuevolaredo.edu.mx/takeyas/Apuntes/Inteligencia%20Artificial/Apuntes/tareas_alumnos/Agentes_Inteligentes/Agentes_Inteligentes(2005-II).pdf>
4. (2019, octubre 1). Recuperado de <https://es.wikipedia.org/wiki/Agente_inteligente_(inteligencia_artificial)>
5. (2019, octubre 1). Recuperado de <https://es.wikipedia.org/wiki/La_serpiente_(videojuego)>
6. (2019, octubre 1). Recuperado de <https://es.wikipedia.org/wiki/La_serpiente_(videojuego)#Argumento>
7. <https://pypi.org/project/simpleai/>