Universidad Politecnica Salesiana

Estudiante: Angel Jadan

Materia: Inteligencia artificial 1

Fecha: 6/2/2021

In [15]:

```
from easyAI import TwoPlayersGame, Human_Player, AI_Player, Negamax
#Interfaz grafica
from tkinter import *
from tkinter import messagebox
from neo4j import GraphDatabase
```

In [16]:

```
import turtle


class Pin_pong(TwoPlayersGame):

    def _init_(self, numeroJugadores):

        self.numeroJugadores=1

        #Ventana
        wn = turtle.Screen()
        wn.title("Pong by Mundo Python")
        wn.bgcolor("black")
        wn.setup(width=800, height=600)
        wn.tracer(0)

        #Marcador
        marcadorA = 1
        marcadorB = 1

        #JugadorA
        jugadorA = turtle.Turtle()
        jugadorA.speed(0)
        jugadorA.shape("square")
        jugadorA.color("white")
        jugadorA.penup()#Para eliminar linea que queda marcado.
        jugadorA.goto(-350, 0)#Posicion
        jugadorA.shapesize(stretch_wid=5, stretch_len=1)

        #JugadorA
        maquina = turtle.Turtle()
        maquina.speed(0)
        maquina.shape("square")
        maquina.color("white")
        maquina.penup()
        maquina.goto(350, 0)
        maquina.shapesize(stretch_wid=5, stretch_len=1)


        #Pelota
        pelota = turtle.Turtle()
        pelota.speed(0)
        pelota.shape("square")
        pelota.color("white")
        pelota.penup()
        pelota.goto(0,0)

        #Modificar estas variables para cambiar la velocidad de la pelota
        pelota.dx = 0.3
        pelota.dy = 0.3


        #Pen para dibujar el marcador.
        pen = turtle.Turtle()
        pen.speed(0)
        pen.color("white")
        pen.penup()
        pen.hideturtle()
        pen.goto(0, 260)
```

```
        pen.write("Jugador A: 0     jugadorB: 0", align="center", font=("Courier", 25,


        #Teclado
        wn.listen()
        wn.onkeypress(jugadorA_up, "w")
        wn.onkeypress(jugadorA_down, "s")



        while True:
            wn.update()

            pelota.setx(pelota.xcor() + pelota.dx)
            pelota.sety(pelota.ycor() + pelota.dy)

            #Revisa colisiones con los bordes de la ventana
            if pelota.ycor() > 290:
                pelota.dy *= -1
            if pelota.ycor() < -290:
                pelota.dy *= -1

            # Si la pelota sale por la izq o derecha, esta regresa al centro.
            if pelota.xcor() > 390:
                pelota.goto(0,0)
                pelota.dx *= -1
                marcadorA += 1
                pen.clear()

                #Esta línea de codigo vuelve a pintar el marcador, utilizo "format" de
                #Si tienes python menor a la versión 3.6 esta parte no te funcionará.
                pen.write(f"Jugador A: {marcadorA} Maquina: {marcadorB}", align="cente

            if pelota.xcor() < -390:
                pelota.goto(0,0)
                pelota.dx *= -1
                marcadorB += 1
                pen.clear()
                #Esta línea de codigo vuelve a pintar el marcador, utilizo "format" de
                #Si tienes python menor a la versión 3.6 esta parte no te funcionará.
                pen.write(f"Jugador A: {marcadorA}     Maquina: {marcadorB}", align="c


            #Revisa las colisiones
            if ((pelota.xcor() > 340 and pelota.xcor() < 350)
                and (pelota.ycor() < maquina.ycor() + 50
                    and pelota.ycor() > maquina.ycor() - 50)):
                pelota.dx *= -1

            if ((pelota.xcor() < -340 and pelota.xcor() > -350)
                    and (pelota.ycor() < jugadorA.ycor() + 50
                    and pelota.ycor() > jugadorA.ycor() - 50)):
                pelota.dx *= -1



    def possible_moves(self):
        #Movimientos arriba
        y = maquina.ycor()
        y += 20
        #Movimientos abajo
```

```
        y = maquina.ycor()
        y -= 20
        maquina.sety(y)
        return y
        #maquina.sety(y)

    def make_move(self, y):
        maquina.sety(y)

    def unmake_move(self, y):
        maquina.sety(y)

    def lose(self):
        if marcadorA==10 and marcadorB<marcadorA:
            return True
        else:
            return False
    def show(self):
        print(marcadorA+marcadorB)

    def scoring(self):
        return -100 if self.lose() else 0

    def is_over(self):
        return self.lose()



    #Funciones
    def jugadorA_up():
        #Movimientos arriba
        y = jugadorA.ycor()
        y += 20
        jugadorA.sety(y)

    def jugadorA_down():
        y = jugadorA.ycor()
        y -= 20
        jugadorA.sety(y)


    """def maquina_up():
        y = maquina.ycor()
        y += 20
        maquina.sety(y)"""

    """def maquina_down():
        y = maquina.ycor()
        y -= 20
        maquina.sety(y)"""



    #wn.onkeypress(maquina_up, "Up")
    #wn.onkeypress(maquina_down, "Down")
```

In [13]:

```
"""if __name__=="__main__":
    ai_algo = Negamax(6)
    pinpon = Pin_pong([Human_Player(), AI_Player(ai_algo)])
    pinpon.play()"""
```

Out[13]:

```
'if __name__=="__main__":\n    ai_algo = Negamax(6)\n    pinpon = Pin_pong
([Human_Player(), AI_Player(ai_algo)])\n    pinpon.play()'
```

In [46]:

```
class Algoritmo():
def _init_():
    crear_catalogo()

    uri="localhost"
    driver = GraphDatabase.driver(uri, auth=('neo4j', 'Angel2019'))
    session = driver.session(database="system")
    session = driver.session()

'''Algoritmo A*'''
def algaestrella(lugar):
    result = session.run("""CALL gds.alpha.allShortestPaths.stream({
  nodeProjection: '"""+lugar+"""',
  relationshipProjection: {
    ROAD: {
      type: 'DISTANCIA',
      properties: 'distancia'
    }
  },
  relationshipWeightProperty: 'distancia'
})
YIELD sourceNodeId, targetNodeId, distance
WITH sourceNodeId, targetNodeId, distance
WHERE gds.util.isFinite(distance) = true

MATCH (source:"""+lugar+""") WHERE id(source) = sourceNodeId
MATCH (target:"""+lugar+""") WHERE id(target) = targetNodeId
WITH source, target, distance WHERE source <> target

RETURN source.name AS source, target.name AS target, distance
ORDER BY distance DESC, source ASC, target ASC
LIMIT 10""")

    return result
    #print(result)
    #for record in result:
     #   print("Origen => "+record["source"]+" | Destino => "+record["target"]+" | Dist

    #names = [record["source"] for record in result]

    #print(names)
    session.close()
    driver.close()

'''Algoritmo de la ruta mas corta'''
def rutamascorta(origen, destino):
    result = session.run("""MATCH (start:Lugar {name: '"""+origen+"""'}), (end:Lugar {r
    CALL gds.alpha.shortestPath.stream({
      nodeProjection: 'Lugar',
      relationshipProjection: {
        ROAD: {
          type: 'DISTANCIA',
          properties: 'distancia',
          orientation: 'UNDIRECTED'
        }
      },
      startNode: start,
      endNode: end,
      relationshipWeightProperty: 'distancia'
```

```
    })
    YIELD nodeId, cost
    RETURN gds.util.asNode(nodeId).name AS name, cost""")

    return result
    #print(result)
    #for record in result:
     #   print("Origen => "+record["name"]+" | Costo => "+str(record["cost"]))

    #names = [record["source"] for record in result]

    #print(names)
    session.close()
    driver.close()

'''Funcion para crear el catalogo para correr algoritmos'''
def crear_catalogo():
    result = session.run("""CALL gds.graph.create('myGraph', 'Lugar', 'DISTANCIA', { re

    session.close()
    driver.close()

'''Algoritmo de amplitud'''
def algaplitud(nombreNodo,lugar):
    result = session.run("""MATCH ("""+nombreNodo+""":Lugar{name:'"""+lugar+"""'})
WITH id("""+nombreNodo+""") AS startNode
CALL gds.alpha.bfs.stream('myGraph', {startNode: startNode})
YIELD path
UNWIND [ n in nodes(path) | n.name ] AS names
RETURN names
ORDER BY names""")
    return result
    #print(result)
    #for record in result:
     #   print("Ruta => "+record["names"])


    #names = [record["source"] for record in result]

    #print(names)
    session.close()
    driver.close()

'''Algoritmo de profundidad'''
def algprofundidad(nombreNodoOrigen,lugarOrigen,nombreNodoDestino, lugarDestino):

    result = session.run("""MATCH ("""+nombreNodoOrigen+""":Lugar{name:'"""+lugarOriger
    ("""+nombreNodoDestino+""":Lugar{name:'"""+lugarDestino+"""'})
WITH id("""+nombreNodoOrigen+""") AS startNode, [id("""+nombreNodoDestino+""")] AS targ
CALL gds.alpha.dfs.stream('myGraph', {startNode: startNode, targetNodes: targetNodes})
YIELD path
UNWIND [ n in nodes(path) | n.name ] AS names
RETURN names
ORDER BY names""")
    #print(result)
    for record in result:
        print("Ruta => "+record["names"])

    #names = [record["source"] for record in result]

    #print(names)
```

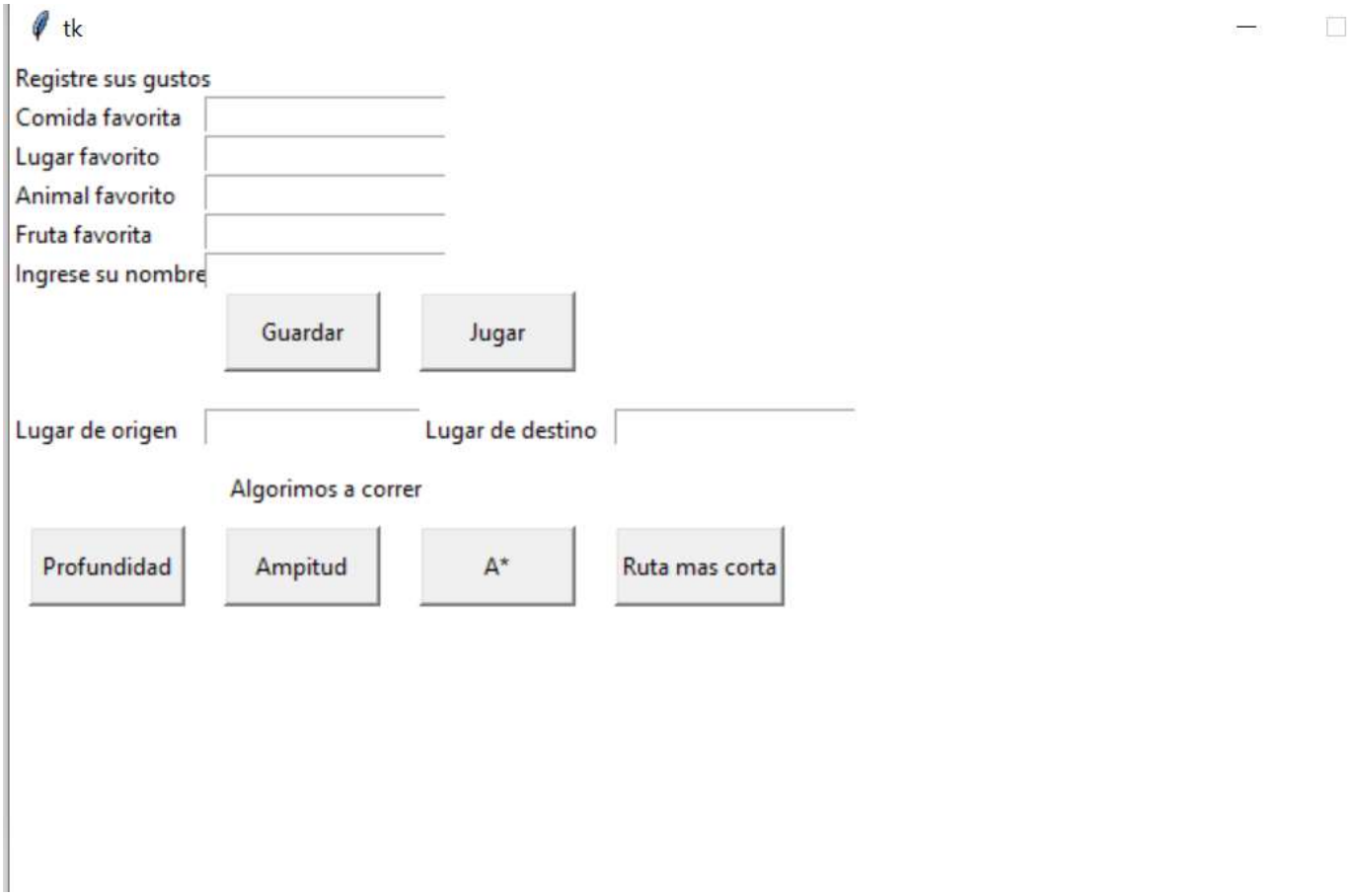
```
121     session.close()
122     driver.close()
123 
124 def crearNodo(nombre, comida, lugar, animal, fruta):
125 
126     result = session.run("""CREATE ("""+nombre.lower().strip()+""":Gustos{name:'"""+nom
127     lugar='"""+lugar+"""', animal='"""+animal+"""', fruta='"""+fruta+"""'})""")
128     #print(result)
129     session.close()
130     driver.close()
131     return True
```

In [50]:

```
#Crear la ventana raiz
ventana = Tk()

#Cambio den el tamaño de la ventana
ventana.geometry("750x450")

ventana.configure(background="white")
#Bloquear el tamaño de la ventana
ventana.resizable(0,0)

#Etiqueta de texto
lbltitle = Label(ventana,text="Registre sus gustos",background="white").place(x=0,y=0)
lblnombre=Label(ventana,text="Comida favorita",background="white").place(x=0,y=20)
lbldireccion=Label(ventana,text="Lugar favorito",background="white").place(x=0,y=40)
lbltelefono=Label(ventana,text="Animal favorito",background="white").place(x=0,y=60)
lblcorreo=Label(ventana,text="Fruta favorita",background="white").place(x=0,y=80)
lblcorreo=Label(ventana,text="Ingrese su nombre",background="white").place(x=0,y=100)

comida=StringVar()
txtnombre=Entry(ventana,textvariable=comida).place(x=100,y=20)
#name=nombre.get()

lugar=StringVar()
txtdireccion=Entry(ventana,textvariable=lugar).place(x=100,y=40)

animal=StringVar()
txttelefono=Entry(ventana,textvariable=animal).place(x=100,y=60)

fruta=StringVar()
txtcorreo=Entry(ventana,textvariable=fruta).place(x=100,y=80)

nombre=StringVar()
txtcorreo=Entry(ventana,textvariable=nombre).place(x=100,y=100)


def guardar():
    com = comida.get()
    lug = lugar.get()
    frut = fruta.get()
    ani = animal.get()
    nomb = nombre.get()


    algoritmo = Algoritmo()
    res = algoritmo.crearNodo(nombre, com, lug, ani, frut)
    if res == True:
        messagebox.showinfo(message="Datos guardados", title="Sms")
        comida.set("")
        fruta.set("")
        lugar.set("")
        animal.set("")
        nombre.set("")
    else:
        messagebox.showinfo(message="No se ha podido guardar revise por favor", title=

def jugar():
    ai_algo = Negamax(6)
    pinpon = Pin_pong([Human_Player(), AI_Player(ai_algo)])
    pinpon.play()
```

```
60
61
62  #Boton de comando
63  cFuncion=Button(ventana, command = guardar , text="Guardar",width=10,height=2).place(x
64  cFuncion=Button(ventana, command = jugar , text="Jugar",width=10,height=2).place(x=210
65
66
67  lblorigen=Label(ventana,text="Lugar de origen",background="white").place(x=0,y=180)
68  origen=StringVar()
69  txtorigen=Entry(ventana,textvariable=origen).place(x=100,y=180)
70
71  lbldestino=Label(ventana,text="Lugar de destino",background="white").place(x=210,y=180
72  origen=StringVar()
73  txtdestino=Entry(ventana,textvariable=origen).place(x=310,y=180)
74
75  algoritmo = Algoritmo()
76
77  def prof():
78      nombreOrigen = txtorigen.get().strip().lower()
79      lugarOrigen = txtorigen.get()
80      nombreDestino = txtdestino.get().strip().lower()
81      lugarDestino = txtdestino.get()
82
83      res = algoritmo.algprofundidad(nombreOrigen,lugarOrigen,nombreDestino, lugarDestin
84      messagebox.showinfo(message=res, title="Sms")
85
86  def aestrella():
87      lugar = txtorigen.get()
88      res = algoritmo.algaestrella(lugar)
89      messagebox.showinfo(message=res, title="Sms")
90
91  def ruta():
92      origen = txtorigen.get()
93      destino = txtdestino.get()
94      res = algoritmo.rutamascorta(origen, destino)
95      messagebox.showinfo(message=res, title="Sms")
96
97  def amp():
98      nombreNodo = origen.get().lower().strip()
99      lugar = origen.get()
100     res = algoritno.algaplitud(nombreNodo,lugar)
101     messagebox.showinfo(message=res, title="Sms")
102
103
104
105 lblcorreo=Label(ventana,text="Algorimos a correr",background="white").place(x=110,y=21
106
107 cFuncion=Button(ventana, command = prof , text="Profundidad",width=10,height=2).place(
108 cFuncion=Button(ventana, command = amp , text="Ampitud",width=10,height=2).place(x=110
109 cFuncion=Button(ventana, command = aestrella , text="A*",width=10,height=2).place(x=21
110 cFuncion=Button(ventana, command = ruta , text="Ruta mas corta",width=11,height=2).pla
111
112
113 ventana.mainloop()
```

Interfaz grafica


tk
Registre sus gustos
Comida favorita
Lugar favorito
Animal favorito
Fruta favorita
Ingrese su nombre
Guardar
Jugar
Lugar de origen
Lugar de destino
Algorimos a correr
Profundidad
Ampitud
A*
Ruta mas corta

In [ ]: