

**UNIVERSIDAD SIMÓN BOLÍVAR**

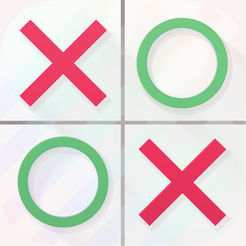
**COORDINACIÓN DE INGENIERIA EN COMPUTACION**

**CI2691: LABORATORIO DE ALGORITMOS Y ESTRUCTURAS I**

**Profesor: Fernando Lovera**

**Kevin Briceño 15-11661**

**LA VIEJA EN 3 DIMENSIONES**

****

**INTRODUCCION**

El juego de 3 en raya conocido como la vieja, es un juego de estrategia sencillo, por ello queremos presentar una versión con otra perspectiva a la formula, consiste en un diseño donde un numero N indicado por el usuario establece el tamaño del tablero, pero al ser de una forma tridimensional NxNxN, filas, columnas y pisos, serán rellenados por los jugadores para conseguir el mayor número de filas hasta agotarse la última posición disponible.

El código de este proyecto está realizado en Python el cual emplea una programación descendiente definida, con el fin de tener un código ordenado para su mantenimiento o desarrollo a nuevas versiones. Pygame es una librería para Python que implementa un motor gráfico para la creación de un programa interactivo.

Teniendo en cuenta las herramientas mencionadas anteriormente, el objetivo del programa es realizar un siclo de varias con matrices de 3 dimensiones.

Esta documentación conlleva todo lo usado para la creación de este proyecto, la estructura descendiente utilizada, estructuración del código y el estado actual del programa.

**DISEÑO**

El diseño aplicando análisis descendiente el programa tiene la forma en GCL:

**do QuieresJugar() == 1 ->**

**otra = 1;**

**do otra == 1 ->**

**Nombres(jugador,jugador1,jugador2,partida);**

**do QuedanFichas(N,fichas)->**

**jugador = Turno(turno, jugador, jugador1, jugador2);**

**x,y,z = ObtenerJugada(x,y,z,N);**

**if ( EsValida(x,y,z,N,T) = True ) ->**

**x,y,z = ObtenerJugada(x,y,z,N)**

**T[z][x][y] = jugador.j**

**jugador.lineas = jugador.lineas + HayLinea(x,y,z,T,N,turno)**

**Resultado(N,T,jugador,jugador1,jugador2)**

**fichas = fichas - 1**

**[] ( EsValida(x,y,z,N,T) =False ->**

**Error**

**fi**

**CambiarJugador(jugador,jugador1,jugador2)**

**turno = turno + 1**

**Total(jugador1,jugador2,jugador)**

**If otra = 1 \/ otra = 0 ->**

**partida = partida + 1**

**T = [ [ [ 0 for x in range(0, N) ] for y in range(0, N) ] for z in range(0, N) ];**

**fichas = N \*\* 3**

**turno = 1**

**jugador1.lineas=0**

**jugador2.lineas=0**

**[] otra > 1 or otra < 0 ->**

**error**

**fi**

**od**

**od**

El programa tiene una cantidad de subprogramas que resuelven diferentes problemas siendo estas la lista de funciones creadas:

* Funcion1\_Tablero
* Funcion2\_QuieresJugar
* Funcion3\_Nombres
* Funcion4\_QuedanFichas
* Funcion5\_Turno
* Funcion6\_ObtenerJugada
* Funcion7\_EsValida
* Funcion8\_HayLinea
  + Funcion\_HayLineaH
  + Funcion\_HayLineaV
  + Funcion\_HayLineaT
  + Funcion\_HayLineaDDI
  + Funcion\_HayLineaDID
* Funcion9\_Resultado
* Funcion10\_CambiarJugador
* Funcion11\_Total

**IMPLEMENTACION**

A continuación, se detallará el funcionamiento de las funciones y procedimientos usados, pero antes se establecerá las variables y constantes usadas en el programa.

#CLASES

class jugador(): # Clase jugador almacena los datos de los jugadores

nombre="" # Definir string como valor de entrada para nombre

j=0 # Definir string como valor de entrada para j que es jugada

lineas=0 # Definir entero como valor de entrada para líneas que es jugada

#INICIALIZACION#

x = 0 #tablero

y = 0 #fila

z = 0 #columna

turno = 1 #turno empieza el jugador 1

fichas = 0 #fichas

N = 0 #tamaño

partida = 1 #Numero de Partidas

jugador1 = jugador() #invoca la clase

jugador2 = jugador() #invoca la clase

jugador1.j = 1 #El valor del jugador1

jugador2.j = 2 #El valor del jugador2

jugador1.lineas = 0 #Cantidad de líneas del jugador1

jugador2.lineas = 0 #Cantidad de líneas del jugador2

**# FUNCIONES**

**def Tablero(fichas,N,T): # Procedimiento el tablero separándolo por pisos**

**assert( N>=1 and fichas == N \*\* 3)**

**print("Fichas disponibles: ",fichas)**

**for i in range(0,N):**

**print("PISO",i)**

**for j in range(0,N):**

**print(T[i][j])**

**def QuieresJugar() -> int: # Función que preguntar al usuario si quiere comenzar el juego**

**try: # try/except**

**print("Quieres comenzar el juego elige:")**

**print("1 = Comenzar")**

**print("0 = Exit")**

**Jugar = int(input("Coloca una opcion:"))**

**assert( Jugar == 1 or Jugar == 0 )**

**except:**

**assert( Jugar > 1 or Jugar < 0 )**

**while Jugar > 1 or Jugar < 0:**

**print("Opcion invalida intente nuevamente.")**

**print("Quieres comenzar el juego elige:")**

**print("1 = Comenzar")**

**print("0 = Exit")**

**Jugar = int(input("Coloca una opcion valida:"))**

**assert( Jugar == 1 or Jugar == 0)**

**assert( Jugar == 1 or Jugar == 0)**

**return Jugar**

**def Nombres(jugador,jugador1,jugador2,partida): #Procedimiento que asigna los nombres de los jugadores**

**print("Es la Partida: ",partida) #Indica el número de partidas que se llevan jugando**

**jugador1.nombre = input("Por favor ingrese nombre del jugador 1: ")**

**jugador2.nombre = input("Por favor ingrese nombre del jugador 2: ")**

**def QuedanFichas( N : int , fichas : int ) -> bool : #Funcion que verifica las fichas disponibles**

**assert( fichas <= N\*\*3 )**

**QuedanFichas=True**

**print("Quedan " + str(fichas) + " fichas")**

**if (fichas > 0):**

**print("Se puede seguir jugando")**

**QuedanFichas = True**

**else:**

**print("Se termina el juego porque no quedan fichas")**

**QuedanFichas = False**

**assert( QuedanFichas == True or QuedanFichas == False )**

**return QuedanFichas**

**def Turno(turno, jugador, jugador1, jugador2): #Funcion que intercambia los jugadores dependiendo el turno.**

**assert( turno>=1 ) #Toma en cuenta que la var turno inicializa en 1.**

**if turno%2!=0: #Turnos impares**

**jugador = jugador1**

**elif turno%2==0: #Turnos pares**

**jugador = jugador2**

**print("Es el Turno: ",turno)**

**print("Turno del jugador ",jugador.nombre)**

**assert( jugador == jugador1 or jugador == jugador2 )**

**return jugador**

**def ObtenerJugada(x,y,z,N): #Funcion pregunta al usuario las coordenadas de la jugada.**

**try:**

**z = int(input("Ingrese Piso: "))**

**x = int(input("Ingrese Fila: "))**

**y = int(input("Ingrese Columna: "))**

**assert(0 <= x < N and 0 <= y < N and 0 <= z < N)**

**except:**

**while(0 > x or x >= N or 0 > y or y >= N or 0 > z or z >= N):**

**print("Jugada Invalida: Fuera de limites")**

**print("Intente Nuevamente")**

**z = int(input("Ingrese Piso: "))**

**x = int(input("Ingrese Fila: "))**

**y = int(input("Ingrese Columna: "))**

**return x,y,z**

**return x,y,z**

**def EsValida(x,y,z,N,T): #Funcion verifica la posicion z,x,y para verificar si esta ocupada**

**try:**

**assert((T[z][x][y] == 0))**

**except:**

**return False**

**return True**

**def HayLinea(x,y,z,T,N,turno,): #Función que llama otras funciones y verifica si hay líneas para sumarlas.**

**jugadas=0**

**if HayLineaT(x,y,z,T,N,turno)==True: #Funcion que verifica si hay linea Transversal.**

**jugadas = jugadas + 1**

**if HayLineaH(x,y,z,T,N,turno)==True: #Funcion que verifica si hay linea Horizontal.**

**jugadas = jugadas + 1**

**if HayLineaV(x,y,z,T,N,turno)==True: #Funcion que verifica si hay linea Vertical.**

**jugadas = jugadas + 1**

**if HayLineaDID(x,y,z,T,N,turno)==True: #Funcion que verifica si hay linea Diagonal Izquierda-Derecha.**

**jugadas = jugadas + 1**

**if HayLineaDDI(x,y,z,T,N,turno)==True: #Funcion que verifica si hay linea Diagonal Derecha-Izquierda.**

**jugadas = jugadas + 1**

**return jugadas**

**# FUNCIONES INTERNAS DE HAY LINEA**

**def HayLineaT(x,y,z,T,N,turno):**

**cont1=0**

**cont2=0**

**for k in range(0,N):**

**if T[k][y][x]==1:**

**cont1 = cont1 + 1**

**if T[k][y][x]==2:**

**cont2 = cont2 + 1**

**if cont1==N or cont2==N:**

**return True**

**def HayLineaH(x,y,z,T,N,turno):**

**cont1=0**

**cont2=0**

**for k in range(0,N):**

**if T[z][x][k]==1:**

**cont1=cont1+1**

**if T[z][x][k]==2:**

**cont2=cont2+1**

**if cont1==N or cont2==N:**

**return True**

**def HayLineaV(x,y,z,T,N,turno):**

**cont1=0**

**cont2=0**

**for k in range(0,N):**

**if T[z][k][y]==1:**

**cont1=cont1+1**

**if T[z][k][y]==2:**

**cont2=cont2+1**

**if cont1==N or cont2==N:**

**return True**

**def HayLineaDDI(x,y,z,T,N,turno):**

**cont1=0**

**cont2=0**

**if x+y==N-1:**

**for k in range(0,N):**

**if T[z][N-k-1][k]==1:**

**cont1=cont1+1**

**if T[z][N-k-1][k]==2:**

**cont2=cont2+1**

**if cont1==N or cont2==N:**

**return True**

**def HayLineaDID(x,y,z,T,N,turno):**

**cont1=0**

**cont2=0**

**if x==y:**

**for k in range(0,N):**

**if T[z][k][k]==1:**

**cont1=cont1+1**

**if T[z][k][k]==2:**

**cont2=cont2+1**

**if cont1==N or cont2==N:**

**return True**

**def Resultado(N,T,jugador,jugador1,jugador2): #Procedimiento indica resultados actuales y lo refleja en el tablero.**

**print("")**

**print("Lineas de hechas por",jugador1.nombre,": ",jugador1.lineas)**

**print("Lineas de hechas por",jugador2.nombre,": ",jugador2.lineas)**

**print("")**

**for i in range(0,N):**

**print("PISO",i)**

**for j in range(0,N):**

**print(T[i][j])**

**print("")**

**def CambiarJugador(jugador,jugador1,jugador2): #Funcion que realiza el cambio de jugador**

**if jugador.j==1:**

**return jugador2**

**else:**

**return jugador1**

**def Total(jugador1,jugador2,jugador): #Funcion que indica el resultado total de la partida.**

**if jugador1.lineas>jugador2.lineas:**

**print("Gana: ",jugador1.nombre)**

**elif jugador2.lineas>jugador1.lineas:**

**print("Gana: ",jugador2.nombre)**

**else:**

**print("Empate")**

**ESTADO ACTUAL**

El estado actual del proyecto es completamente operativo funciona perfectamente, se pueden realizar de todas las partidas que los jugadores deseen, pero no tiene interfaz gráfica, debido a que tuve problemas implementando la parte grafica con lo que realicé con la parte lógica, tanto así que era como hacer el código nuevamente, por lo que no se implementado en esta versión.

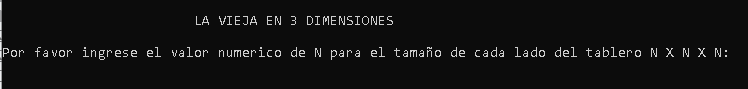
**MEJORAS QUE SE PUEDEN HACER:**

En la próxima versión buscare implementar en su totalidad la interfaz gráfica, con las mismas funcionalidades del programa actual.

**MANUAL DE OPERACIÓN:**

El programa consta de 17 archivos.

Para ejecutarlo se inicia el archivo vieja\_3D.py

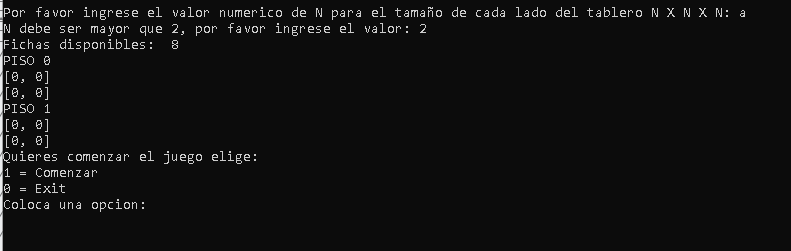


El programa indica cada información requerida.

En caso de error de entrada te indicara una ventana como la siguiente:



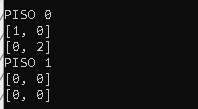
Cuando la entrada sea correcta el programa avanzara.



En algunas ocasiones el programa indica las opciones como se ve en la imagen anterior.

Mientras coloques los valores de entrada correctos el programa avanzará sin interrupciones, en caso de error te pedirá que cortinas la entrada. El programa llenara la matriz indicando

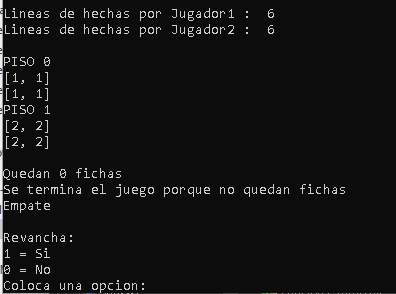
1 = jugada del jugador 1

2 = jugada del jugador 2

También mostrara las líneas hechas en cada jugada realizada:



Al finalizar la partida indicara cual tuvo más puntos he indicara el ganador



El juego termina cuando no hay más fichas, he indica los resultados finales y pregunta si quieres otra partida.

En caso que no quieras introduce 0 y te preguntara si quieres jugar en caso que no quieras introduce 0 y el programa finalizara.

.

**CONCLUSIONES**

Los resultados obtenidos para mi persona son muy gratificantes ya que no había realizado programas completos antes de este curso de laboratorio de algoritmo, he obtenido mucha experiencia y conocimiento realizando este proyecto, la única dificultad fue el uso de interfaz ya que no la usamos dentro del curso por lo que se me dificulto implementarla en programa, pero es algo que mi persona tengo que mejorar para futuras implementaciones.

La experiencia adquirida en las clases fue de mucha utilidad para la realización del proyecto, el uso constante de matrices, veo muy importante que se implemente en los laboratorios la utilización de algoritmos ya existentes para que los estudiantes vallan documentándose, bueno es algo que mi persona debo hacer, investigar y dominar los algoritmos conocidos, este curso genera conocimiento, experiencia y un repertorio funciones que se reutilizaran en el futuro.

**BIBLIOGRAFÍA**

Página Web:

Autor(Es): Python Software Foundation

Nombre: Documentación De Python

URL: <https://docs.python.org/3/contents.html>

Página Web:

Autor(Es): Poketcode.Com

Nombre: Python 2.7.13

URL: <http://www.poketcode.com/es/python/index.html>

Libro:

Autor(Es): Anne Kaldewaij

Nombre: Programming: The Derivation of Algorithms

Editorial: Prentice Hall

Año:1990

Guía:

Autor(Es): Oscar Meza

Nombre: Guía de Programming: The Derivation of Algorithms