Universitatea Natională de Știință și Tehnologie POLITEHNICA BUCURESTI

Facultatea de Electronică, Telecomunicații si Tehnologia Informației

Proiect Interfete Om-Masina

Joc de X si O prin comanda vocala

Studenti: - Dumitrasco Viorel

- Jitianu Stefan

Grupa: 445A

Coordonator:

Ing. Andrei Stefanescu

Cuprins

[1. Lista Figuri 3](#_Toc155709897)

[2. Lista abrevieri 4](#_Toc155709898)

[3. Introducere 5](#_Toc155709899)

[4. Tehnologii folosite 6](#_Toc155709900)

[5. Prototipul aplicatiei 7](#_Toc155709901)

[6. Implementarea interfetei 10](#_Toc155709902)

[6.1 Fereastra „Introducere jucatori” 11](#_Toc155709903)

[6.2 Fereastra „Alege ordinea” 16](#_Toc155709904)

[6.3 Fereastra „Joc X si 0” 18](#_Toc155709905)

[7. Concluzii 26](#_Toc155709906)

[8. Bibliografie 27](#_Toc155709907)

# Lista Figuri

[Figură 5‑1 Prototip Introduce jucatori 7](#_Toc155709322)

[Figură 5‑2 Prototip fereastra joc 7](#_Toc155709323)

[Figură 5‑3 Prototip mesaje consola 9](#_Toc155709324)

[Figură 6‑1 Diagrama conceptuala joc 10](#_Toc155709325)

[Figură 6‑2 Diagrama fereastra"Introducere jucatori" 11](#_Toc155709326)

[Figură 6‑3 Fereastra Introducere jucatori 12](#_Toc155709327)

[Figură 6‑4 Diagrama fereastra "Alege ordinea" 16](#_Toc155709328)

[Figură 6‑5 Fereastra "Alege Ordinea" 17](#_Toc155709329)

[Figură 6‑6 Diagrama fereastra Joc X si 0 18](#_Toc155709330)

[Figură 6‑7 Fereastra inceput joc X si 0 19](#_Toc155709331)

[Figură 6‑8 Diagrama functie proceseaza\_comanda 21](#_Toc155709332)

[Figură 6‑9 Fereastra joc X si 0 dupa prima completare 23](#_Toc155709333)

[Figură 6‑10 Fereastra joc X si 0 cand a castigat un jucator 25](#_Toc155709334)

# Lista abrevieri

1. GUI – Graphical User Interfaces
2. gTTS - Google Text-to-Speech

# Introducere

Acest proiect isi propune sa prezinte o modalitate diferita de a juca binecunoscutul X si O. Este unul dintre primele jocuri pe care le-am invatat in copilarie si pe care l-am jucat de nenumarate ori, fie ca a fost pe foaie cu creionul, pe tabla de la scoala cu creta apoi pe telefoane cu butoane, pe calculatoare folosind mouse-ul si ulterior folosind o interactiune de tip touchscreen pe noile dispozitive. Logica jocului este arhicunoscuta.

Este unul din primele jocuri create pentru calculatoare, unii spun ca este chiar primul, insa el ramane folosit ca element pedagogic si astazi pentru a explica teoria jocurilor si elementele de baza ale inteligentei artificiale ([[1]](#endnote-1)) . Sau datorita frumusetii sale logice arhicunoscute este folosit pentru a pune in evidenta noi tehnologii la fel cum in acest proiect vom dezvolta posiblitatea de a juca X si O prin comanda vocala.

Comanda vocala a fost creata pentru a facilita si imbunatatii experienta utilizatorilor in interactiunea cu tehnologia. Prin utilizarea vocii, oamenii pot controla dispozitivele si accesa informatii fara a fi nevoie sa utilizeze taste sau ecrane tactile([[2]](#endnote-2)). Acesta ofera un mod mai natural si comod de a comunica cu tehnologia, mai ales in situatii in care utilizarea mainilor ar fi dificila sau nerecomandata, cum ar fi atunci cand conducem sau suntem ocupati in alt fel.

Acest proiect va permite utilizatorilor sa isi introduca numele prin comanda vocala, sa aleaga cine va incepe primul tot din comanda vocala, si sa poata plasa X sau O in interfata grafica rostind coordonatele in care doreste sa plaseze simbolul. In cazul in care exista un castigator calculatorul va reda la boxe numele acestuia in caz contrat va rosti „A castigat prietenia !”. La sfarsit utilizatorul este intrebat daca doreste sa rejoace sau sa inchida jocul.

Contributia studentilor in acest proiect consta in implementarea unui algoritm care sa tina cont de situatiile posibile care pot aparea in logica jocului folosind structuri de control pentru a modifica fluxul de executie al programului. Aceasta logica fiind incorporata cu elemente din biblioteca tkinter pentru crearea interfetei grafice si obiecte, metode din biblioteca sppech\_recognition pentru recunoasterea vocala.

# Tehnologii folosite

* Python – limbaj de programare;

Python este un limbaj de programare dinamic multi-paradigma, creat in 1989 de programatorul olandez Guido van Rossum. Van Rossum este si in ziua de astazi un lider al comunitatii de dezvoltatori de software care lucreaza la perfectionarea limbajul Python si implementarea de baza a acestuia, CPython, scrisa in C. Python este un limbaj multifunctional folosit de exemplu de catre companii ca Google sau Yahoo! pentru programarea aplicatiilor web, insa exista si o serie de aplicatii stiintifice sau de divertisment programate partial sau in intregime in Python. Popularitatea in crestere, dar si puterea limbajului de programare Python au dus la adoptarea sa ca limbaj principal de dezvoltare de catre programatori specializati si chiar si la predarea limbajului in unele medii universitare. Din aceleasi motive, multe sisteme bazate pe Unix, inclusiv Linux, BSD si Mac OS X includ din start interpretatorul Cpython([[3]](#endnote-3)).

Python pune accentul pe curatenia si simplitatea codului, iar sintaxa sa le permite dezvoltatorilor sa exprime unele idei programatice intr-o maniera mai clara si mai concisa decat in alte limbaje de programare ca C. In ceea ce priveste paradigma de programare, Python poate servi ca limbaj pentru software de tipul object-oriented, dar permite si programarea imperativa, functionala sau procedurala. Sistemul de tipizare este dinamic iar administrarea memoriei decurge automat prin intermediul unui serviciu „gunoier” (garbage collector). Alt avantaj al limbajului este existenta unei ample biblioteci standard de metode(3).

* Tkinter – biblioteca pentru realizarea de interfata grafica;
* Speech\_recognition – biblioteca pentru recunoastere vocala;
* Numy – biblioteca pentru manipularea matricilor;
* Pyttsx3 si gtts – biblioteci pentru sinteza vocala si redarea unui mesaj audio;
* Os si pygame – biblioteci pentru manipularea fisierelor audio.

# Prototipul aplicatiei

Jocul a fost dezvoltat in doua etape in paralel. O parte ce reprezenta interfata grafica si o parte ce reprezenta functionarea aplicatiei. In urmatoarele randuri voi prezenta aceste doua etape inainte de a face legatura intre ele si de a adauga ultimele functionalitati.

**Partea de GUI():**

Prima data am creata o fereastra care in urma apasarii butoanelor cu textul „Introdu jucator” sa astepte utilizatorul sa rosteasca numele jucatorului ca mai apoi sa il afiseze in interfata grafica, utilizatorul putea reapasa butonul de introducere a numelui in cazul in care recunoasterea vocala nu a functionat corespunzator. In interfata mai exista un butonul pentru start joc care porneste jocul in urma apasarii cand cee doi jucatori sunt gata.

A screenshot of a computer

Description automatically generated

Figură ‑ Prototip Introduce jucatori

A doua fereastra are desenat un tabel cu trei linii si trei coloane fiecare element din tabel corespunde pozitiei conform coordonatelor prezentati in interfata. Am ales reprezentarea asemanatoare unei table de sah adica litere pentru coloane si cifre pentru linii. Aceasta fereastra asculta in permanenta microfonul jucatorilor si atunci cand rostesc o coordonata potrivita jocul pune X sau 0 in pozitia coordonatei.

A screenshot of a computer

Description automatically generated

Figură ‑ Prototip fereastra joc

**Partea de functionalitate a aplicatiei:**

Pentru inceput am creat o functie ce capta comanda de la jucator folosind biblioteca speech\_recognition cu obiectul Recognizer si metodele listen si recognize\_google. Functia salva intr-o variabila ceea ce inregistra de la microfon si returna acea variablia.

def capture\_voice\_input():

while(True):

with sr.Microphone() as source:

print("Ascult...")

audio = recognizer.listen(source,timeout = 5, phrase\_time\_limit=3)

try:

text = recognizer.recognize\_google(audio, language = "ro-RO")

text = text.replace('unu', '1').replace('doi', '2').replace('trei', '3')

return text

except sr.UnknownValueError:

text = ""

print("Scuze, nu te-am înțeles ")

time.sleep(1.5)

continue

except sr.RequestError as e:

text = ""

print("Error; {0}".format(e))

continue

O a doua functie este cea de erori\_citire() care este o functie pentru corectarea inputului potential gresit de exemplu index in afara matricii, utilizatorul a spus ceva in afara de nr liniei si a coloanei, pozitie deja ocupata.

def erori\_citire():

while(True):

text = capture\_voice\_input()

try:

a = int(text[0])

b = int(text[2])

except:

print("Spune doar numarul liniei si numarul coloanei!")

continue

if(a<1) or(b<1) or (a>3) or (b>3):

print("Ai 3 linii si 3 coloane din care sa alegi!")

continue

a = int(a) -1

b = int(b) -1

if(m[a][b]!=4) :

print("Poziție ocupată, mai incearcă")

continue

else : return a,b

O a treia functie este cea de win\_condition() care verifica daca exista castigator. Am implementat functia de win\_condition() folosindu-ne de o matrice initializata la inceputul jocului cu 4 pe fiecare pozitie iar atunci cand utilizatorul rostea o coordonata valida in aceasta matrice era inlocuita valoarea 4 cu 1 pentru jucatorul cu x si cu 0 pentru jucatorul cu 0. Functia in sine face suma pe fiecare linie in parte, pe fiecare coloana in parte si pe cele doua diagonale iar daca a gasit suma pe vreuna din linii, coloane sau diagonale egala cu 0 sau 3 intoarce adevarat in rest fals. Aceasta functie este apelata dupa fiecare completare in tabel.

def win\_condition():

s1=0

s2=0

column\_sums = [sum([row[i] for row in m]) for i in range(0,len(m[0]))]

row\_sums = [sum(row) for row in m]

if(3 in column\_sums) or(0 in column\_sums) :return True

if(3 in row\_sums) or (0 in row\_sums) :return True

for i in range(3) :

s1 += m[i][i];

s2 += m[i][2 - i ];

if(s1 == 3) or (s1 == 0) : return True

if(s2 == 3 ) or (s2 == 0) : return True

return False

Pentru simplitate am apelat acest pachet de functii de 5 ori doar pentru a testa functionalitatea.

Comunicarea cu utilizatorul se face in consola programului si are urmatorul rezultat.

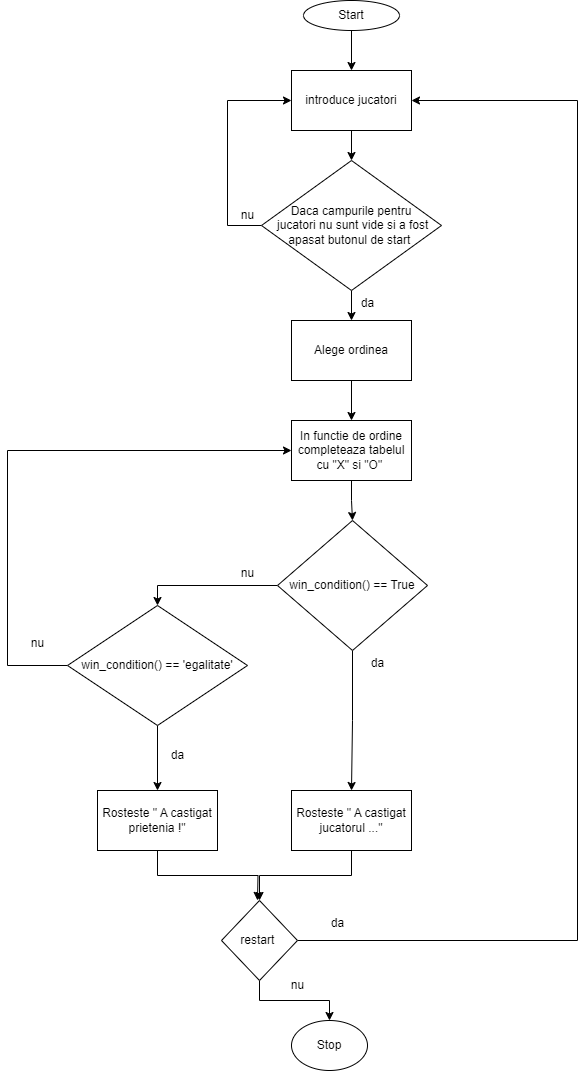
A computer screen shot of a program

Description automatically generated

Figură ‑ Prototip mesaje consola

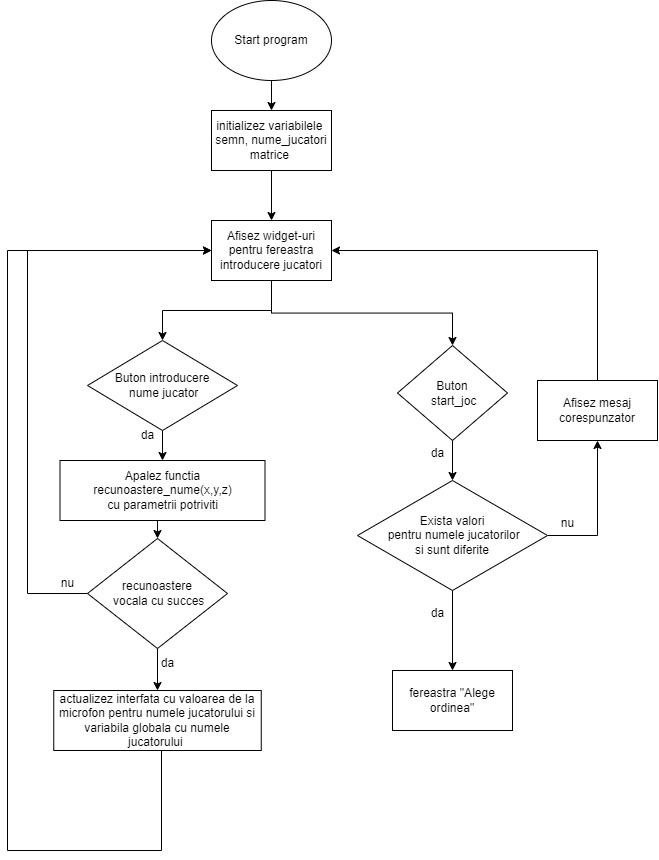
# Implementarea interfetei

Explicarea interfetei este mai usoara folosind diagrama aplicatiei. Voi pleca de la cea de nivel inalt, urmand ca mai apoi sa dezvolt si cu detalii de cod fiecare modul din aplicatie.



Figură ‑ Diagrama conceptuala joc

## Fereastra „Introducere jucatori”



Figură ‑ Diagrama fereastra"Introducere jucatori"

La inceputul aplcatiei inainte de a apela orice functie am importat modulele necesare apoi am initializat variabile pentru numele jucatorilor si am creat o instanta a clasei ‚Tk’ care reprezinta fereastra principala a aplicatiei. Obiectul window va fi folosit pentru a adauga si organiza toate celelalte elemente ale interfetei grafice.

#instanta fereastra principala

window = Tk()

#adaugare titlu interfata

window.title("X și 0")

#variabile pentru numele jucatorilor

jucator\_1 = StringVar()

jucator\_2 = StringVar()

#matricea initializata cu 4 pe baza careia verific castigatorul

m = np.array([ [4,4,4], [4,4,4], [4,4,4] ])

A screenshot of a computer

Description automatically generated

Figură ‑ Fereastra Introducere jucatori

Aceasta prima fereastra a jocului este implementata in functia inceput() si este apelata la pornirea programului. Functia goleste la inceputul sau orice widget exista in fereastra. Vom vedea utilitatea acestui lucru atunci cand vrem sa dam restart la joc fara al inchide.

for widget in window.winfo\_children():

        widget.destroy()

Functia mai are la inceput declarate doua variabile initializate cu liste goale.

etichete\_joc = []

    etichete\_tura = []

semn = "neinitializat"

Setez dimensiunile ferestrei inainte de a adauga continut si stabilesc si culoarea de fundal.

window.geometry("400x470")

    window.config(bg="#e6f7ff")

Ca font de text pentru continut am folost ‚Helvetica’. Si am adaugat padding pentru claritatea ferestrei.

Fereastra contine trei widget-uri de tip Label care au valori diferite pentru text-ul lor iar sub cele doua widget-uri cu valorile Jucator 1 si Jucator 2 se afla doua widget-uri de tip text in care vor aparea numele celor doi jucatori dupa preluarea de la microfon .

titlu = Label(window, text='Joc X și 0',bg="#99ccff",font=("Helvetica", 12, "bold"), relief='ridge', padx=10, pady=10)

    titlu.grid(row=0, column=1, padx=50, pady=20)

    #etichete pentru nume

    label\_eticheta\_1 = Label(window, text='Jucător 1: ',font=("Helvetica", 8, "bold"), bg="#99ccff",relief='ridge', padx=5, pady=5)

    label\_eticheta\_1.grid(row=1,column=1,padx=10,pady=10,)

    eticheta\_nume\_1 = Text(window, height=2, width=13,relief='ridge',state = "disabled")

    eticheta\_nume\_1.grid(row = 2,column=1)

    label\_eticheta\_2 = Label(window, text='Jucător 2: ',font=("Helvetica", 8, "bold"), bg="#99ccff",relief='ridge', padx=5, pady=5)

    label\_eticheta\_2.grid(row=1,column=2,padx=10,pady=10)

    eticheta\_nume\_2 = Text(window, height=2, width=13, relief='ridge', state = "disabled")

    eticheta\_nume\_2.grid(row = 2,column=2)

Fereastra mai contine trei butoane fiecare dintre ele apeland cate o functie. Butoanele cu textul Introduce numele jucatorului apeleaza functia recunoastere\_nume(x,y,z)cu trei parametri pentru ce jucator se face ascultarea, in ce eticheta se insereaza si in ce widget afiseaza mesajele in cazul in care ascultarea nu a fost cu succes. Toate cele trei butoane au fost stilizate folosind instanta clasei ‚Style’ din sub-modulul ‚ttk’.

style = ttk.Style()

    style.configure("TButton", padding=6, relief="flat", background="#ccc")

    buton\_nume1 = ttk.Button(window, text='Introdu numele jucătorului x', command=lambda: recunoastere\_nume(jucator\_1, eticheta\_nume\_1,scena\_joc))

    buton\_nume1.grid(row=3, column=1, padx=10, pady=15)

    buton\_nume2 = ttk.Button(window, text='Introdu numele jucatorului 0', command=lambda: recunoastere\_nume(jucator\_2, eticheta\_nume\_2,scena\_joc))

    buton\_nume2.grid(row=3, column=2, padx=10, pady=15)

Functia recunoastere\_nume(x,y,z) apelata de cele doua butoane creaza o instanta a obiectului Recognizer()

recunoastere = sr.Recognizer()

si foloseste diferite metode pentru a permite recunoasterea vocala a numelui jucatorului.

with sr.Microphone() as source:

Creaza un context pentru utilizarea microdonului. Toate operatiile de recunoastere vocala vor avea loc in interiorul acestui context.

recunoastere.adjust\_for\_ambient\_noise(source)

Ajusteaza recunoasterea pentru zgomotul de fond pentru a imbunatatii calitatea recunoasterii vocale.

        audio = recunoastere.listen(source,timeout = 5, phrase\_time\_limit=3)

asculta sunetul de la microfon pentru o perioada de 5 secunde sau pana cand o fraza este complet recunoscuta, 3 secude in acest caz.

        nume\_jucator = recunoastere.recognize\_google(audio, language="ro-RO")

Utilizeaza Google Speech Recognition pentru a transforma audio-ul in text, folosind limba romana si salveaza acea valoare in variabila nume\_jucator

Daca recunoasterea a fost realizata cu succes actualizeaza variabila in care se stocheaza numele jucatorului.

variabila\_jucator.set(nume\_jucator)

Apoi sterge orice valoare din interfata grafica care se afla in widget-ul cu numele jucatorului si o inlocuieste cu varianta actuala.

eticheta.delete('1.0', 'end')

        # Adaugă noul text

        eticheta.config(state="normal")

        eticheta.delete("1.0", "end-1c")

        eticheta.insert('1.0', str(nume\_jucator))

        eticheta.config(state="disabled")

        # Actualizare interfață grafică

Daca recunoasterea nu a fost realizata cu succes afiseaza un mesaj in GUI in obiectul Text.

Daca apare o eroare in procesul de recunoastere vocala acesta este gestional si afisat in consola.

except sr.RequestError as e:

        print(f"Eroare la cerere pentru recunoaștere vocală; {e}")

In aceasta functie a fost foarte important sa trimitem cei trei parametrii intrucat asupra lor s-au efectuat procesarile din functie.

Aceasta prima fereastra mai contine un buton care apeleaza functia de start a jocului intre cei doi competitori. Aceasta o vom explica imediat intrucat este functia care inglobeaza logica complexa a programului.

buton\_start = ttk.Button(window, text='Start', command=afiseaza\_tabel)

    buton\_start.grid(row=4, column=1, padx=50, pady=20)

Ultimul widget din fereastra este unul de tip Text in care sunt afisate mesajele de eroare in cazul in care recunoasterea vocala nu a fost cu succes sau alte mesaje.

scena\_joc = Text(window, height=5, width=40, wrap=WORD, font=("Helvetica", 10), bg="#e6f7ff", state = "disabled")

    scena\_joc.grid(row=5, column=0, columnspan=3, padx=10, pady=10)

Pentru a incepe jocul am pus **conditia ca cei doi jucatori sa aibe introduse nume si acestea sa fie diferite**. Aceste conditii se verifica la apasarea butonului de start in prima parte a functiei afiseaza\_tabel() prin doua intructiuni if. Primul if verifica daca valorile din variabilele numelor nu sunt nule iar cel de-al doilea if daca valorile sunt diferite intre ele si diferite de null . In fiecare dintre cazuri daca este indeplinita conditia din if afiseaza utilizatorului un mesaj folosind o fereastra de avertisment.

if not jucator\_1.get() or not jucator\_2.get():

        messagebox.showwarning("Avertisment", "Introduceți numele ambilor jucători!")

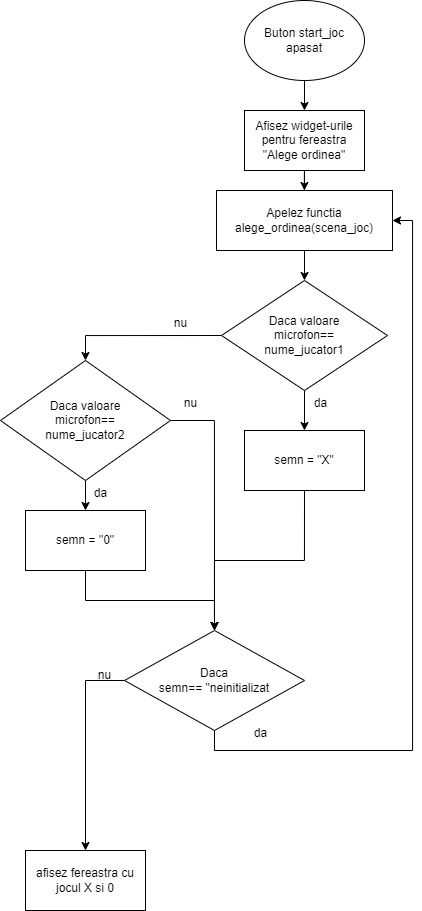
        return

    if jucator\_1.get()== jucator\_2.get() and jucator\_1.get() !="" :

        messagebox.showwarning("Avertisment", "Introduceți nume de jucători diferite!")

        return

## Fereastra „Alege ordinea”



Figură ‑ Diagrama fereastra "Alege ordinea"

Dupa ce am trecut de verificarile pe variabilele de nume sterg toate widget-urile din fereastra si le afisez pe cele cu mesaj corespunzator pentru a alege ordinea.

#afiseaza widget-uri pentru fereastra alege ordinea

    eticheta\_ordine = Label(window, text=f'Cine este primul jucător ? \n {jucator\_1.get()} sau {jucator\_2.get()}', font=("Helvetica", 18, "bold"),bg="#99ccff")

    eticheta\_ordine.grid(row=0, column=0, padx=10, pady=10)

    #chenarul pentru mesaje

    scena\_joc = Text(window, height=10, width=40, wrap=WORD, font=("Helvetica", 10), bg="#e6f7ff")

    scena\_joc.grid(row=7, column=0, columnspan=3, padx=10, pady=10)

Rezultatul in interfata este urmatorul in functie de numele introduse :

A screenshot of a phone

Description automatically generated

Figură ‑ Fereastra "Alege Ordinea"

Dupa afisarea interfetei este apelata functia **alege\_ordinea()**

window.after(100,lambda: alege\_ordinea(scena\_joc))

Functia alege \_ordinea(x) este apelata in cadrul metodei **after** care este utilizata pentru a programa executia functiei dupa o anumita perioada de timp, in acest caz 100 de milisecude. Am folosit aceasta metoda pentru a permite afisarea in interfata a widget-urilor corespunzatoare. Am folosit functia **lambda** pentru ca este modalitatea potrivita de a apela functii in contextul functiilor care primesc alte functii ca argumente.

Aceasta functie este asemanatoare cu functia recunoastere\_nume() intrucat la fel creez un context in care utilizez microfonul pentru recunoasterea vocala doar ca voi afisa mesajele si modificarile corespunzatoare in functie de textul preluat de la microfon.

Am spus ca la inceputul programului am initializat variabila semn=”neinitializat” tocmai pentru aceasta etapa. Daca valoarea de la microfon se potriveste cu vreuna din valorile din variabilele pentru numele jucatorilor atunci modific valoarea din semn potrivit cu ce simbol a ales jucatorul. Daca nici una din valori nu se potriveste atunci nu fac nicio modificare si trec la urmatoarea linie de cod.

#salvez valoarea de la microfon in variabila ordine\_jucator

        ordine\_jucator = recunoastere.recognize\_google(audio, language="ro-RO")

        print(f"Ati spus primul jucator este: {ordine\_jucator}")

        if ordine\_jucator == jucator\_1.get():

            print(f"primul jucator este {jucator\_1.get()}")

            semn = "X"

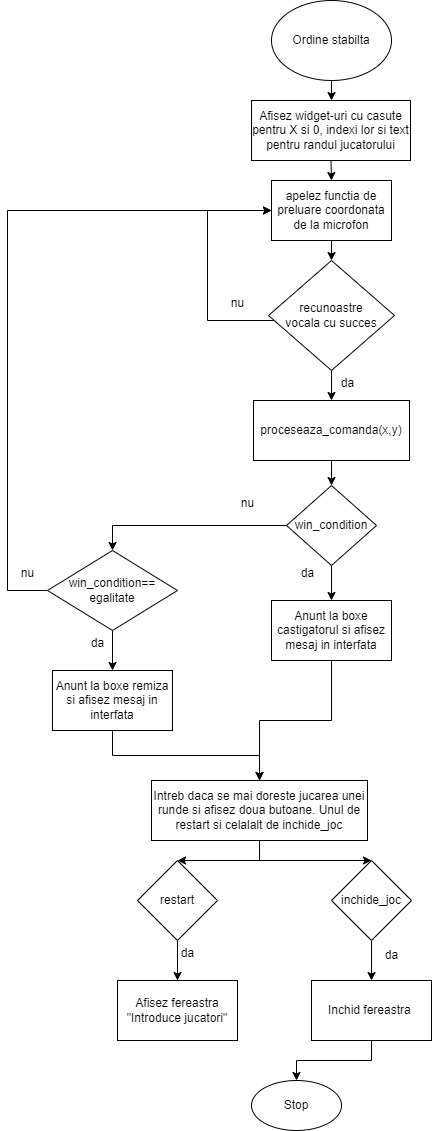
        elif ordine\_jucator == jucator\_2.get():

            print(f"primul jucator este {jucator\_2.get()}")

            semn = "O"

La urma verific daca valoarea din semn a ramas „neinitializat” si in acest caz apelez dinou functia de ascultare de la microfon cu metoda after.

## Fereastra „Joc X si 0”



Figură ‑ Diagrama fereastra Joc X si 0

Daca a fost aleasa ordinea jucatorilor afisez fereastra cu jocul de X si 0 in care am in prima parte eticheta cu randul carui jucator este apoi am tabelul de X si 0 cu indicii pentru alegerea coordonatelor precum o tabla de sah, pe linii sunt cifre iar pe coloane litere. Apoi mai jos am chenarul in care sunt afisate mesaje pentru utilizarea corespunzatoare a joculuim, mesaje de eroare sau de ghidare in utilizarea jocului.

A screenshot of a computer

Description automatically generated

Figură ‑ Fereastra inceput joc X si 0

Afisez un text care sa spuna randul carui jucator este folosindu-ma de valoarea variabilei semn.

#in functie de randul jucatorului afisez un text cu randul carui jucator este

         if semn == 'O':

             eticheta\_tura = Label(window, text=f'{jucator\_2.get()} alege poziția',font=("Helvetica", 12, "bold"), bg="#99ccff")

             eticheta\_tura.grid(row=0, column=0,columnspan=3, padx=10, pady=10)

             etichete\_tura.append(eticheta\_tura)

         else:

             eticheta\_tura = Label(window, text=f'{jucator\_1.get()} alege poziția',font=("Helvetica", 12, "bold"),bg="#99ccff")

             eticheta\_tura.grid(row=0, column=0,columnspan=3, padx=10, pady=10)

             etichete\_tura.append(eticheta\_tura)

Afisez indexii pentru coloane si linii si ma folosesc de doua instructiune for pentru a reduce din redundanta, din liniile de cod.

#afisez indexii pentru coloane folosindu-ma de un for

         for i, litera in enumerate(['A', 'B', 'C']):

             eticheta\_coloana = Label(window, text=litera, font=("Helvetica", 14, "bold"), bg="#e6f7ff")

             eticheta\_coloana.grid(row=1, column=i + 1, padx=10, pady=10)

         #afisez indexii pentru randuri cu un for

         for i in range(1, 4):

             eticheta\_rand = Label(window, text=str(i), font=("Helvetica", 14, "bold"),bg="#e6f7ff")

             eticheta\_rand.grid(row=i + 1, column=0, padx=10)

Afisez casutele in care vom pune X sau 0 si salvez fiecare eticheta intr-un vector pentru ca ma voi folosi de poziitle etichetelor in vector ca sa adaug in casuta X sau 0. Casutele le-am delimitat prin borduri si culoare de fundal diferita.

#afisez casutele in care se marcheaza X sau 0

         for i in range(1, 4):

             for j in range(1, 4):

                 eticheta\_joc = Label(window, text='', width=10, height=2, relief='solid', borderwidth=1,font=("Helvetica", 18, "bold"),bg="#cce5ff")

                 eticheta\_joc.grid(row=i + 1, column=j)

                 etichete\_joc.append(eticheta\_joc)

La urma am adaugat chenarul pentru mesaje si apelez functia care preia coordonata de la microfon.

#afisez chenarul pentru mesaje

         scena\_joc = Text(window, height=10, width=40, wrap=WORD, font=("Helvetica", 10), bg="#e6f7ff")

         scena\_joc.grid(row=7, column=1, columnspan=3, pady=10)

         #apelez functia care preia coordonata de la microfon

         window.after(100, lambda: comanda\_vocala(scena\_joc))

Comanda de la microfon este luata asemanator cu contextul folosit in functiile de mai sus. De aceasta data daca a fost preluata o comanda vocala cu succes este trimisa ca parametru functiei **proceseaza\_comanda(x).** Daca preluarea nu a fost cu succes afisez un mesaj potrivit.

#context pentru utilizarea microfonului.

    with sr.Microphone() as source:

        recunoastere.adjust\_for\_ambient\_noise(source)

        audio = recunoastere.listen(source, phrase\_time\_limit=3)

    try:

        comanda = recunoastere.recognize\_google(audio, language="ro-RO")

        #daca comanda a fost preluata de la microfon cu succes o trimit ca parametru functiei proceseaza\_comanda

        print(f"Ati spus: {comanda}")

        proceseaza\_comanda(comanda,scena\_joc)

    except sr.UnknownValueError:

        #mesaj de eroare

        scena\_joc.insert(END, f"Recunoașterea comenzii vocale a eșuat. Încearcă din nou.\n")

        scena\_joc.see(END)

        print(f"Recunoașterea comenzii vocale a eșuat. Vă rugăm să încercați din nou.")

    except sr.RequestError as e:

        print(f"Eroare la cerere pentru recunoaștere vocală; {e}")

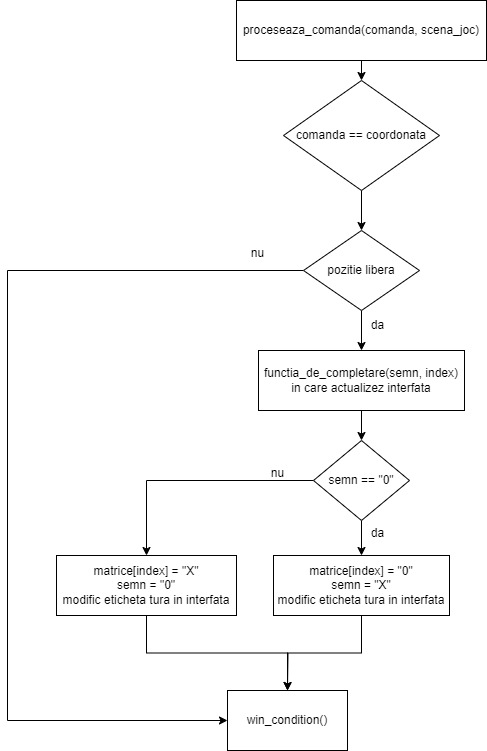
Dupa fiecare ascultare a microfonului verific starea jocului daca a castigat cineva. Daca nu a castigat nimeni si nici egalitate nu este atunci mai apelez o data functia de recunoastere vocala.

if not win\_condition():

        window.after(100, lambda: comanda\_vocala(scena\_joc))

    elif win\_condition() == "egalitate":

**Functia de proceseaza\_comanda(comanda, scena\_joc)**

****

Figură ‑ Diagrama functie proceseaza\_comanda

Aceasta functie este un else if cu noua conditii. Conditiile sunt noua vectori in care am salvate sub forma de string coordonatele casutelor in care se pune X sau 0. Voi da un singur exemplu de astfel de conditie intrucat si celelalte sunt asemanatoare ca logica.

elif comanda in ["unu B", "1B", "1b", "1 b", "B1"," b1", "b 1"]:

daca valoarea din comanda se potriveste cu o coordonata prima data verific daca acel spatiu din casuta este gol. Gol inseamna ca valoarea din matrice este 4 daca are alta valoare este ocupata casuta. (Reamintim ca la inceput am initializat matricea cu 4)

if m[0][1] == 4:

Daca este libera pozitia apelez functia de completare cu parametrul de semn si de pozitie a casutei. In acest caz

functia\_de\_completare(semn, 1)

Aceasta functie preia indexul din vectorul cu etichetele casutelor si updateaza textul sau cu valoarea X sau 0 potrivit jucatorului a carui ordine este atunci.

def functia\_de\_completare(valoare, index):

    #verificare suplimentara ca indexul sa fie in vectorul cu indexi casutelor

    if 0 <= index < len(etichete\_joc):

        # Setează valoarea cheii text a etichetei

        etichete\_joc[index].config(text=valoare)

    else:

        print("Index invalid")

Apoi ma reintorc in functia proceseaza\_comanda() si verific semnul. Cu o instructiune if verific daca semnul este „0” si modific si matricea de stare cu 0, semnul il schimb in „X” si updatez si textul din fereastra care spune acum ca este randul celuilalt jucator.

if semn == 'O':

                    m[0][0]= 0

                    semn = "X"

                    etichete\_tura[0].config(text=f'Este randul jucatorului {jucator\_1.get()}')

            else:

                    m[0][0]= 1

                    semn = "O"

                    etichete\_tura[0].config(text=f'Este randul jucatorului {jucator\_2.get()}')

Daca spatiul este ocupat afisez mesaj ca pozitia este ocupata si ma reintorc sa verific starea de castig ca jocului.

A screenshot of a computer

Description automatically generated

Figură ‑ Fereastra joc X si 0 dupa prima completare

Programul functioneaza in acesta bucla de completare pana cand nu mai sunt spatii libere sau este un castigator. Functia care verifica starea de castig este apelata dupa fiecare completare.

**Functia win\_condition()**

Aceasta functie lucreaza simplu intrucat verifica suma pe linii, pe coloane si pe diagonale iar daca aceasta suma este 3 sau 0 anunta ca este un castigator. Apoi pentru a verifica daca este egalitate verifica daca mai sunt spatii libere in tabel prin verificarea daca mai este vreo valoare de 4 in matrice. Daca nu mai este si nu este niciun castigator inseamna ca este remiza si returnez valoarea „egalitate”.

def win\_condition():

     s1 = 0

     s2=0

     egalitate = "egalitate"

     #fac suma pe coloane

     column\_sums = [sum([row[i] for row in m]) for i in range(0,len(m[0]))]

     #fac suma pe randuri

     row\_sums = [sum(row) for row in m]

     #intorc adevarat daca suma pe randuri sau coloane este 0 sau 3

     if(3 in column\_sums) or(0 in column\_sums) :return True

     if(3 in row\_sums)  or (0 in row\_sums) :return True

     #fac suma pe diagonale

     for i in range(3) :

         s1 += m[i][i];

         s2 += m[i][2 - i ];

    # daca suma pe diagonale este 0 sau 3 intorc adevarat

     if(s1 == 3) or (s1 == 0) : return True

     if(s2 == 3 ) or (s2 == 0) : return True

     #verific daca mai am spatii libere in matrice daca nu este remiza

     if verifica\_egalitate(m):

         print("A castigat prietenia")

         return egalitate

     return False

Starea de castig o verific cu o instructiune else if

Daca functia de win\_condition() intoarce „egalitate” inseamna ca am remiza si atunci updatez eticheta cu text in interfata sa anunte ca este remiza si apelez functia care anunta un mesaj corespunzator la boxe.

etichete\_tura[0].config(text='A câștigat prietenia !',bg="#e6f7ff")

        print("A castigat prietenia")

        mesaj\_castig = f"A câștigat prietenia !"

        window.after(1000, lambda: rostește\_mesaj(mesaj\_castig))

reinitializez variabilele cu numele jucatorilor cu sir gol si matricea de stare cu 4.

jucator\_1.set("")

        jucator\_2.set("")

        for row in range(3):

            for element in range(3):

                m[row][element] = 4

Intreb daca se doreste rejucare unei runde si afisez doua butoane unul de restart si altul de inchidere joc.

#intreb daca se mai doreste jucarea unei runde

        eticheta\_restart = Label(window, text=f'Doriti sa mai jucati o data ?', font=("Helvetica", 12, "bold"),bg="#e6f7ff", foreground='red')

        eticheta\_restart.grid(row=5, column=0,columnspan=3, pady=10)

        #afisez butoanele pentru restart si inchide joc

        buton\_restart = ttk.Button(window, text='Restart ', command=inceput)

        buton\_restart.grid(row = 6, column=0, padx= 50, pady=20)

        buton\_inchidere = ttk.Button(window, text='Închide Joc', command=inchide\_joc)

        buton\_inchidere.grid(row = 6, column=1, pady=20)

Daca functia de win\_condition() intoarce True inseamna ca am un castigator si atunci verific care este valoarea semnului. Daca este 0 inseamna ca a castigat X si invers. Apoi fac aceleasi procese ca in cazul egalitatii doar mesajele se modifica.

A screenshot of a computer

Description automatically generated

Figură ‑ Fereastra joc X si 0 cand a castigat un jucator

Butonul de restart apeleaza functia **inceput()** care afiseaza fereastra „Introducere jucatori” cu valorile variabilelor resetate. Iar butonul de Inchide joc apeleaza functia de **inchide\_joc()** care apeleaza metoda **destroy** pe fereastra principala.

**Functia rosteste\_mesaj(mesaj)**

Aceasta este functia care anunta castigatorul sau remiza la boxe. Am importat o serie de librari.

from gtts import gTTS

import pygame

mai intai am convertit textul in audio cu ajutorul clasei gTTS(2) si l-am salvat intr-un fisier audio.

tts = gTTS(text=mesaj, lang='ro')

tts.save("castigator.mp3")

Apoi am folosit biblioteca pygame pentru a reda fisierul audio salvat mai sus. Initializez modulul pygame.mixer care este responsabil pentru manipularea sunetului.

pygame.mixer.init()

incarc fisierul audio de mai sus si ii dau play dupa care astept 3 secunde inainte de a continua executia programului. Apoi inchid modului pygame.mixer

pygame.mixer.music.load("castigator.mp3")  # Încarcă fișierul audio

    pygame.mixer.music.play()  # Redă fișierul audio

    pygame.time.wait(3000)  # Așteaptă 3 secunde (sau până la sfârșitul melodiei) înainte de a continua

    pygame.mixer.quit()

# Concluzii

In urma efectuarii acestui proiect pe baza problemelor intalnite am invatat mai bine cum functioneaza biblioteca tkinter, cu ce fel de functii trebuie sa lucram cu ea si cum sa impartim programul in parti functionale mai mici care sa faca ca jocul sa se desfasoare fluent.

Am observat ca o buna proiectare de la inceput este importanta intrucat asa cum am lucrat in acest proiect adaugarea de widget-uri noi in fereastra in faze finale ale proiectului este o munca costisitoare intrucat fiecare widget are o pozitie statica si trebuie modificata manual pozitia sa in fereastra. O solutie pentru aceasta problema este folosirea gestionarului de geometrie pack. Dar la inceputul proiectului nu am stiut aceasta problema asa ca am folosit grid.

Python este un limbaj de programare foarte puternic datorita librariilor sale si modului de lucrare cu matricile. Daca ar fi trebuit sa facem acelasi proiect in C sau alte limbaje de programare ar fi fost mult mi dificila implementarea si ar fi necesitat o experienta de programare mai avansata ca sa ajungem la acelasi rezultat.

Dezvoltarea jocului X si 0 (Tic-Tac-Toe) poate fi o unealta didactica excelenta pentru a exersa anumite tehnici de programare. Acest joc simplu ofera numeroase oportunitati de a lucra cu concepte de baza in programare si poate ajuta la dezvoltarea abilitatilor practice. Poti utiliza diferite structuri de date pentru a reprezenta starea jocului, cum ar fi matricea pentru tabla de joc sau vectori pentru a tine evidenta mutarilor jucatorilor. Implementarea regulilor si logicii jocului necesita gandire algoritmica.In interfata grafica, trebuie sa gestionezi starea jocului si sa reactionezi la diferite evenimente, cum ar fi mutarile jucatorilor.

Desigur ca jocul ar putea fi imbunatatit in continuare spre exemplu s-ar putea adauga varianta in care un jucator uman sa joace cu calculatorul care sa poata avea nivele de inteligenta artificiala diferite. Sau o interfata grafica mai moderna poate chiar 3D, si multe altele.

# Bibliografie

1. Tic-tac-toe(joc) <https://koaha.org/wiki/Tris_(gioco)> [↑](#endnote-ref-1)
2. Prezentarea generala si aplicatii ale tehnologiei de recunoastere a vocii <https://ro.shaip.com/blog/voice-recognition-overview-and-applications/> [↑](#endnote-ref-2)
3. Python - <https://ro.wikipedia.org/wiki/Python> [↑](#endnote-ref-3)