Univerzitet u Beogradu
Elektrotehnički fakultet

Katarina Janković, 2020/0322

MemoryTiles - interaktivna mrežna aplikacija

projekat iz predmeta Principi modernih telekomunikacija

mentor:
prof. dr Milan Bjelica

Beograd, Jul 2023

Sažetak

U okviru ovog rada će se proći kroz uputstvo korišćenja korisničkog interfejsa aplikacije ,,MemoryTiles", kao i kroz način njenog ostvarivanja konekcije između računara i implementacije u programskom jeziku Python.

Ključne reči: Aplikacija, server, klijent, nit, python

Sadržaj

Spisak slika

Glava 1

Uvod

1.1 Predmet rada i motivacija

Cilj pri kreiranju aplikacije ,,MemoryTiles" je da se pokaže da relativno jednostavni mrežni protokoli mogu biti osnova za stvaranje ne toliko jednostavnih interaktivnih aplikacija.
Radi postizanja boljeg korisničkog iskustva, poseban naglasak se stavlja na ostvarivanje bržih veza između korisnika, zaštitu tih veza i jednostavnost njenog interfejsa, radi što lakšeg korišćenja
Dakle, moglo bi se reći da se ,,MemoryTiles" sastoji iz 3 logička dela: GUIa, višenitnog i mrežnog aspekta, od kojih će se svaki obraditi u okviru odvojene celine.

1.2 Opis igre

Pravila igre su krajnje jednostavna:
Postoje 2 igrača koji su međusobno povezani, kada se konekcija između njih ostvari otvara se prozor sa ,,tablom" za igru. Cilj igre je da se pronađu svi parovi koji su nasumično raspoređeni po ,,tabli". Igrači naizmenično otvaraju po dva polja sa ciljem da pronađu par.
Kada se svi parovi ,,otkriju" pobeđuje onaj igrač koji je pronašao više parova.

Glava 2

Uputstvo za korišćenje korisničkog interfejsa

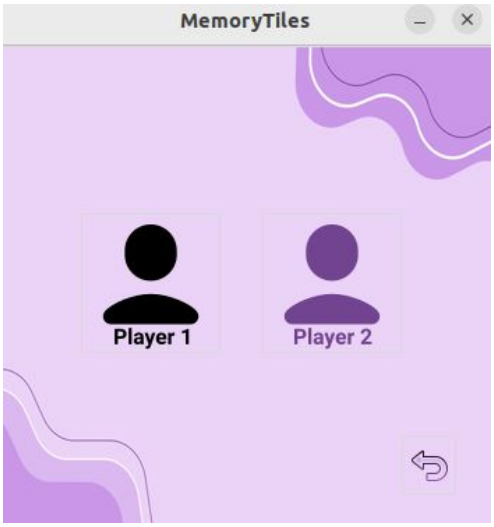
2.1 Pregled prozora i njihovih funkcionalnosti

Pri pokretanju igre prvo vidimo početni prozor sa jednim dugmetom koje služi za započinjanje igre.

Slika 2.1: *Početni prozor*

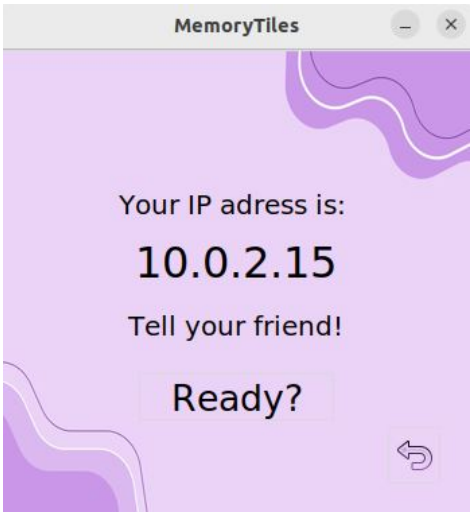
Kada odlučimo da želimo da započnemo igru, pojavljuje se prozor koji nam pruža mogućnost da odaberemo igrača.

U donjem desnom uglu nalazi se strelica za povratak na prethodni prozor. Ona će se nalaziti na svim budućim prozorima sem na poslednjem, prozoru

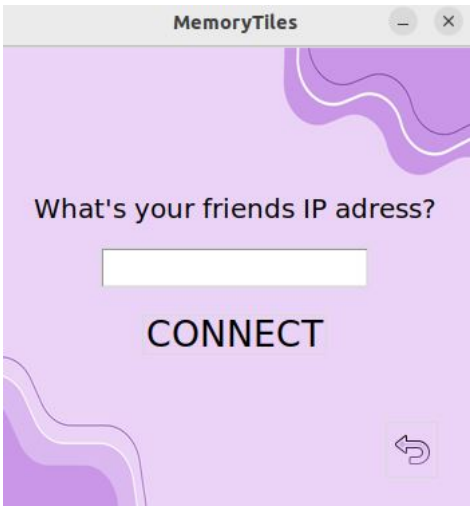


Slika 2.2: *Izbor igrača*

same igre.
U zavisnosti od toga kog igrača izaberemo, pojaviće se sledeći prozori namenjeni za ostvarivanje prvobitne konekcije između igrača:



Slika 2.3: *Konekcija za igrača br.1*

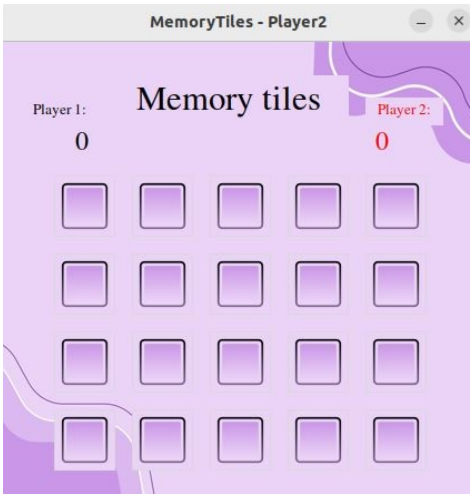


Slika 2.4: *Konekcija za igrača br.2*

U prozoru za igrača br. 1 nalazi se samo ispisana njegova IP adresa i dugme „Ready?" kojim on započinje igru (koje mora biti pritisnuto pre nego što igrač br. 2 pokuša da se konektuje, razlog će biti objašnjen u sledećem poglavlju).

Dok se u prozoru za igrača br. 2 nalazi tekstualno polje u koje on mora da upiše IP adresu igrača br. 1 i pritisne dugme ,,CONNECT" za ostvarivanje konekcije. Pri uspešnom povezivanju otvara se prozor sa ,,tablom" za igru.

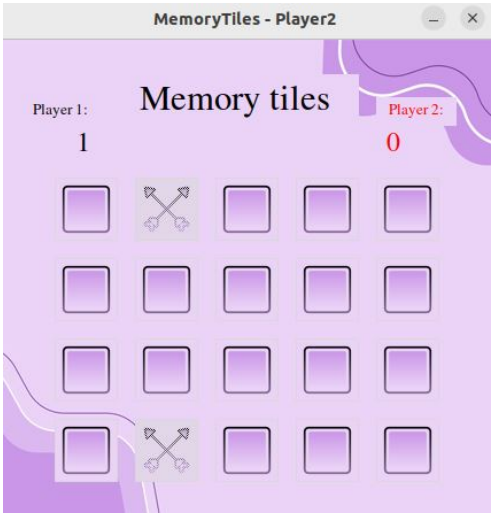
Slika 2.5: *Prozor igrača br.1 / Winodws*



Slika 2.6: *Prozor igrača br.2 / Ubuntu*

Crvenom bojom označen je igrač čiji je trenutno red za potez. Igrač br. 2 uvek igra prvi.
Potez se sastoji od toga da igrač otvori dva ,,polja" koja će se zatvoriti ako igrač nije pronašao par ili ,,pobledeti" ako jeste, pri čemu će se u tom slučaju uvećati broj poena igrača za jedan.
Kada jedan igrač odigra potez, kontrola se prosleđuje drugom igraču automatski.

Igrači naizmenično igraju poteze sve dok se sva polja ne otkriju i pobeđuje onaj igrač koji je ostvario više poena.



Slika 2.7: *Izgled ,,table" u slučaju da je igrač br.1 našao 1 par*

Slika 2.8: *Kraj igre*

2.2 Realizacija funkcionalnosti u kodu

Sada kada smo pregledali izgled korisničkog interfejsa možemo razgovarati o njegovim funkcionalnostima. Programski kod je napisan u jeziku Python, a za kreiranje korisničkog interfejsa je korišćena biblioteka tkinter.
Početni prozori namenjeni za početak igrice i za ostvarivanje konekcije su zapravo samo različiti frejmovi ili tačnije izgledi jednog te istog prozora.

Prozor je definisan u posebnoj klasi ,,windows", koja u svojoj __init__ funkciji sadrži niz svih ostalih frejmova koji će se prikazivati prilikom inicijalizacije

```
self.frames = {}
for F in (MainPage, SidePage, Page2, Page3):
    frame = F(container, self)
```

Slika 2.9: *Lista frejmova početnog prozora*

Unutar same klase ,,windows", definišu se stvari isključivo vezane za prozor kao što su veličina, mesto pojavljivanja i ikonica, sve ostale karakteristike izgleda definišu se u posebnim klasama definisanim za svaki frejm.

Frejmovi su realizovani kao objekti klase tkinter.Frame, sve što se u njima nalazi, kao i u drugim prozorima, su objekti klasa Label, Button i Entry biblioteke tkinter. (više o tome kako se ovi objekti koriste, kao i o samoj biblioteci može se naći u [1])

Kada se ostvari konekcija pojavljuje se novi prozor same igre (klasa ,,Game").

```
class windows(tk.Tk):...

class MainPage(tk.Frame):...

class SidePage(tk.Frame):...

class Page2(tk.Frame):...

class Page3(tk.Frame):...

class Game:...
```

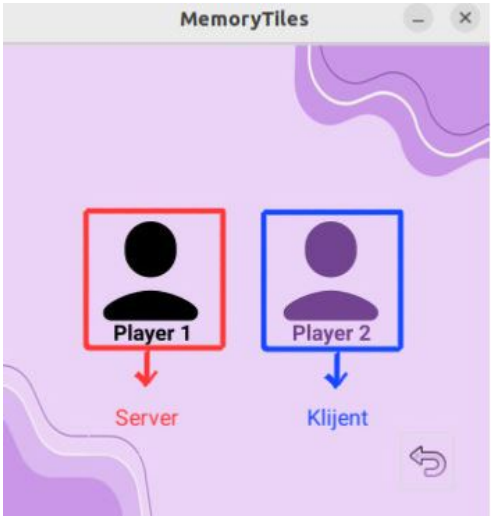
Slika 2.10: *Pregled svih klasa u kodu*

Glava 3

Konekcija

3.1 Protokol

Da bi igra bila što dinamičnija i da korisnici ne bi predugo čekali da se povežu ili na potez suparnika, za ostvarivanje konekcije izabran real-time protokol - UDP.
Za ostvarivanje konekcije potrebna su nam dva učesnika, korisnik i server. Ovo je razlog zašto igrači moraju na početku da biraju da li su jedan ili drugi igrač, kao i zašto prozori za konekciju dva igrača izgledaju drugačije.



Slika 3.1: *Izbor korisnik/server*

Da bi uopšte uspostavili komunikaciju između servera i klijenta, klijent nekako mora da zna na koju adresu šalje podatke, ne može da mu se ,,javi" ko je suparnik, zbog čega je ponuđena opcija da korisnik/klijent ručno unese IP

adresu svog suparnika/servera na stranici za konekciju, dok je na stranici za konekciju servera u labeli ispisana IP adresa računara na kome je aplikacija pokrenuta, da korisnik ne bi morao da traži svoju IP adresu, tj. da bi korisnici što lakše razmenili ovu informaciju neophodnu za pokretanje igre.

3.2 Sigurnost

UDP je poznat po tome što je vrlo jednostavan i brz, što je i razlog zašto je izabran za realizaciju ove aplikacije, ali takođe je i vrlo nesiguran. On ne ostvaruje trajnu konekciju između korisnika i ne garantuje redosled kojim će paketi stizati niti da li će stizati uopšte.

Redosled stizanja paketa i ne predstavlja problem u implementaciji ove aplikacije jer su poruke koje se ovde razmenjuju male veličine (tipa int) i uvek će se slati samo u jednom paketu, ali ostale mane UDP protokola mogu da stvore razne probleme unutar aplikacije, od gubljenja podataka, do upadanja uljeza u igru (ukoliko poznaju IP adresu servera) itd. Za ove probleme morala su se realizovati logička rešenja koja bi obezbedila neometano i pravilno izvršavanje aplikacije.

3.2.1 Međusobno prepoznavanje korisnika

Kao što je već rečeno UDP je nekonektivni protokol što znači da korisnici nemaju nikakvu otvorenu vezu između sebe na koju šalju podatke. Klijent će znati IP adresu servera zato što će je korisnik ručno ukucati, ali server neće znati koje poruke su od klijenta koji zaista želi da započne igru, a koje su greška/poslate od strane nekog drugog.

Ovaj problem može da se reši na bezbroj načina, ali je za potrebe ove aplikacije radi jednostavnosi izabran je najlakši.

Klijent i server će međusobno razmeniti neke početne poruke za koje će oboje znati očekivane vrednosti i kada se one razmene korisnik i server će se međusobno prepoznati kao igrači i na dalje će primati samo poruke jedan od drugog, sve ostale poruke će se odbacivati.
Pošto je jasno da se prvih par poruka razmenjuje automatski čim se pokrene klijent (klikom na dugme ,,Connect"), server mora prvo da se postavi (klikom na dugme ,,Ready") da bi te poruke imao ko da primi.

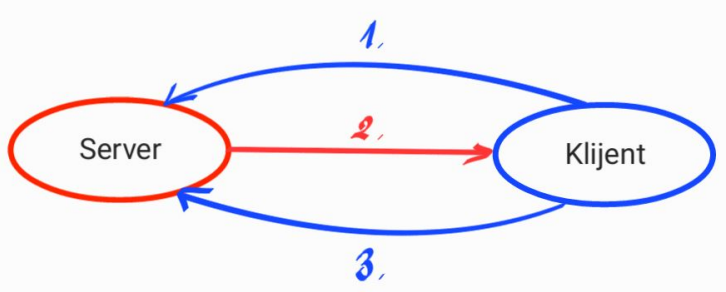
3.2.2 Gubljenje podataka

Iako je rešenje za međusobno prepoznavanje korisnika naznačeno kao najjednostavnije rešenje na prvi pogled ne deluje kao da jeste.

I klijent i server imaju mogućnost da prepoznaju od koga su primili poruke, server bi mogao samo da zapamti prvog klijenta koji mu pošalje poruku koja označava početak igre i samo nastavi da šalje i prima poruke samo od tog klijenta i ne bi bilo potrebe za slanjem više od te jedne početne poruke za početak igre.

Međutim, problem nastaje kod činjenice da UDP nema nikakav način da proveri da li su poruke zapravo stigle, za razliku od nekih drugih protokola koji su napravljeni tako da se uvek pri razmeni pruke pošiljaocu šalje potvrda da je poruka stigla, tako da se može dogoditi da klijent započne igru bez saigrača, zato što serveru nikad nije stigla poruka sa zathevom za početak.

Ovo je razlog zašto se umesto jedne razmenjuju tri poruke. Prva poruka predstavlja zahtev za započinjanje, zatim druga kojom server šalje klijentu potvrdu da je primio zahtev i treća kojom klijent obaveštava da je primio potvrdu. Na ovaj način obe strane znaju da su uspešno razmenile svoje poruke i igra može da počne.



Slika 3.2: *Početne poruke za započinjanje igre*

Ostale poruke će se slati bez ikakve provere, tj bez očekivanja potvrde da su poruke stigle do primaoca. Iako ova projektna odluka možda nije najbezbednija, jer znanje da su neke poruke sigurno razmenjene između korisnika ne garantuje da će sve sledeće poruke sigurno dostići svoju destinaciju, ipak ovaj pristup je odabran zbog očuvanja brzine razmene poruka, što je jedini razlog zašto je izabran ovaj protokol, a pretpostavlja se da osim ukoliko ne dođe do neke promene topologije mreže (usled nekog kvara ili nečeg drugog) velika je verovatnoća da će poruke dostići svoju destinaciju.

3.3 Realizacija same igre

U prethodnom odeljku ovog poglavlja je diskutovano o ostvarivanju konekcije, protokolu koji je izabran, zašto je baš on izabran itd. tako da se sada

može razmatrati šta se to tačno šalje i kako aplikacija logički funkcioniše.

```
def button33_command(self):
    global win
    textBox1 = getattr(self, "textBox1")
    ip = textBox1.get()
    root = tk.Toplevel()
    serverName = ip
    serverPort = 12000
    clientSocket = socket(AF_INET, SOCK_DGRAM)
    order = []

    message = b"1"
    clientSocket.sendto(message, (serverName, serverPort))
    response, serverAddress = clientSocket.recvfrom(2048)
    if (response.decode() == "2"):
        for i in range(20):
            response, serverAddress = clientSocket.recvfrom(2048)
            order.append(int.from_bytes(response, 'little'))
    clientSocket.sendto(b"3", (serverName, serverPort))
    clientSocket.close()

    windows.destroy(win)
    root2 = tk.Tk()
    pg = Game(root2, order, 2, ip)
    root2.mainloop()
```

Slika 3.3: *Kod za konekciju klijenta*

Iz priloženih kodova za konekciju servera i klijenta (koji su modelovani na osnovu primera iz skripte [2]) može se videti da su poruke koje šalje klijent besmislene poruke za potvrđivanje konekcije, dok server šalje 21 poruku, od kojih je prva ,,2", a ostalih 20 su random generisani brojevi.

Tih 20 poruka predstavljaju niz pozicija slika na ,,tabli", tj. raspored slika, tako da oba korisnika imaju isti ,,ključ" za rešavanje igre. Odabrano je da se ovi podaci šalju baš u ovom koraku zato što znamo da je klijent uspeo da nam pošalje jednu poruku i ako dobijemo očekivani odgovor od njega znamo, ne samo da želi da započne igru, već i da je primio raspored u celini.
Sada kada igrači razmenjuju poteze, ono što oni razmenjuju su zapravo po-

```
def connect(self):
    global win, ip_got
    serverPort = 12000  # isto kao za klijent
    serverSocket = socket(AF_INET, SOCK_DGRAM)
    serverSocket.bind(('', serverPort))
    images = []
    numb = []
    order = []
    message, clientAddress = serverSocket.recvfrom(2048)
    if (message.decode() == "1"):
        serverSocket.sendto(b"2", clientAddress)
        for i in range(20):
            numb.append(i)
        for i in range(20):
            a = random.randint(0, 19)
            while (numb[a] == -1):
                a = random.randint(0, 19)
            numb[a] = -1
            order.append(a)
            serverSocket.sendto(a.to_bytes(5, 'little'), clientAddress)
        message, clientAddress = serverSocket.recvfrom(2048)
        if (message.decode() == "3"):
            serverSocket.close()
            windows.destroy(win)
            root2 = tk.Tk()
            pg = Game(root2, order, 1, ip_got)
            root2.mainloop()
```

Slika 3.4: *Kod za konekciju servera*

zicije u rasporedu, na osnovu čega otkrivaju polja koja je izabrao protivnik i preračunavaju da li je on pronašao par ili ne.

Glava 4

Višenitni aspekt

Za sada je pokazano da se aplikacija sastoji od dva ključna aspetka, interfejsa i komunikacije između korisnika, preostalo je samo još prodiskutovati o tome kako je realizovano da se oni izvršavaju simultano.

Teoretsi, ovaj kod bi mogao da se izvršava sekvencionalno, jer ne postoje dva događaja u toku igre koja moraju ili uopšte mogu da se izvršavaju istovremeno. Pri ostvarivanju konekcije, korisnici su primorani da čekaju poruke od suparnika, tokom same igre korisnik mora prvo da odigra potez da bi se on poslao, pa mora da sačeka potez suparnika da bi mogao da igra dalje itd. Međutim, problem nastaje zbog načina na koji je tkinter biblioteka napravljena. Prozor predstavlja objekat klase tkinter.Tk, i pri njegovoj inicijalizaciji, osim ako to eksplicitno nije navedeno, on se neće prikazati korisniku. Tek pri pozivu funkcije mainloop() će se pojaviti prozor. Mainloop() je napravljena kao beskonačna petlja namenjena za osluškivanje događaja, tako da jednom kada je pokrenuta ništa drugo sem nje više ne može da se izvršava.

Kako je osnovni koncept na kom se zasniva realizacija ove aplikacije razmena podataka iz gore navedenih ograničenja dalo bi se zaključiti da sve poruke moraju da se razmene pre pokretanja prozora, što bi onda učinilo samu realizaciju nemogućom, jer bi se izgubio svaki smisao igre.
Na nekim mestima se ovaj problem mogao izbeći tako što se razmena poruka smeštala u određenim događajima klika na dugme, kao što je slučaj na stranicama ,,Page 2" i ,,Page 3", pa čekanje na poruke nije predstavljalo problem, jer je mainloop() svakako realizovala taj prekid.

Na drugim mestima, kao što je realizacija same igre, ovo rešenje nije bilo moguće primeniti, jer bi se previše izmena vršilo u istom događaju pa se one ne bi sve ni pokazale, a pritom bi imali problem čekanja odgovora. Tako da se za zaobilaženje ovog problema pri implementaciji morao uključiti koncept niti (engl. threads). Sve funkcionalnosti su idalje raspoređenje po događajima za dugmad (poljima na ,,tabli"), samo što su sračunavanje poena i promenu

izgleda za klijenta i za server vršile različite niti.

Glava 5

Mogućnosti za unapređenje

5.1 Poboljšavanje trenutne implementacije

Neke ideje:

- Dakle, kao što je pokazano, koncept niti je iskorišćen samo na jednom mestu i to kada je to bilo neophodno. Bilo bi sasvim moguće podeliti sve poslove prikazivanja formi i widgeta, kao i obezbeđivanja njihovih funkcionalnosti na niti, pa da glavna nit u suštini ne radi ništa ili prikuplja podatke o igri (broj izvršenih poteza, poeni igrača na kraju igre, vreme trajanja igre itd.) i čuva ih u nekoj bazi zarad neke dalje analize ili za potrebe usložavanja igre (dodavanje nivoa itd.).

- Takođe je moguće povećati broj korisnika koji mogu da se uključe u igru (klijenata) tako što bi server pamtio koliko je poruka za započinjanje igre pristiglo u nekom vremenskom intervalu.

5.2 Ideje za buduće projekte

Iako ova igrica poprilično dobro ilustruje jednostavnost korišćenja UDP protokola za realizovanje konekcije, u stvarnosti nema baš idealnu primenu zato što je za korišćenje neophodno da korisnici imaju neki drugi vid komunikacije kako bi igrač 1 mogao da kaže svom saigraču svoj IP.
U nekom ozbiljnijem projektu bi se trebalo zakupiti neki univerzalni server koji bi obezbeđivao konekciju između klijenata i čija bi adresa bila integrisana u kodu automatski.

Glava 6

Zaključak

Na posletku, iz ovog rada se da zaključiti da je interaktivna mrežna aplikacija - ,,MemoryTiles" zadovoljila sve ciljeve koji su na početku bili postavljeni.
Pokazano je da i korišćenjem jednostavnih tehnologija, uz malo nadogradnje, mogu da se dobiju odlični rezulati. I pored svih mana UDP protokola, problema sa radom biblioteke tkinter itd. na kraju se dobila sasvim ispravna i funkcionalna aplikacija, implementirana tako da korisnik ni ne oseti sve nadostatke koji su se javljali prilikom realizacije, što je i najvažnije.

Glava 7

Literatura

[1] *Graphical User Interfaces with Tk* elektronski dokument, 2023; dostupno online:
https://docs.python.org/3/library/tk.html

[2] prof. dr Milan Bjelica - *Programski jezik Python - skripta za studente telekomunikacija drugo izdanje*, Udžbenik Elektrotehničkog fakulteta u Beogradu, 2021;

Prilog

U prilogu se nalazi ceo source kod aplikacije.

```
import ctypes
import inspect
import tkinter as tk
import tkinter.font as tkFont
from socket import *
import random, re, time
from threading import *
import threading
from tkinter import messagebox

global win


class windows(tk.Tk):
    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)
        global win
        win = self

        icon = tk.PhotoImage(file="res/stop.png")

        self.wm_title("MemoryTiles")

        width = 355
        height = 343
        screenwidth = self.winfo_screenwidth()
        screenheight = self.winfo_screenheight()
        alignstr = '%dx%d+%d+%d' % (width, height, (screenwidth -
```

```
        width) / 2, (screenheight - height) / 2)
        self.geometry(alignstr)
        self.resizable(width=False, height=False)
        self.iconphoto(False, icon)

        container = tk.Frame(self, height=400, width=600)

        container.pack(side="top", fill="both", expand=True)

        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}
        for F in (MainPage, SidePage, Page2, Page3):
            frame = F(container, self)

            self.frames[F] = frame
            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame(MainPage)

    def show_frame(self, cont):
        frame = self.frames[cont]
        frame.tkraise()


class MainPage(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        poz1 = tk.PhotoImage(file="res/poz1.png")
        lbl = tk.Label(self, image=poz1)
        lbl.img = poz1
        lbl.place(relx=0.5, rely=0.5, anchor='center')

        play = tk.PhotoImage(file="res/playbutton.png")
        label = tk.Label(image=play)
        label.image = play

        button1 = tk.Button(
            self,
            command=lambda: controller.show_frame(SidePage),
```

```
        )
        button1["image"] = play
        button1["justify"] = "center"
        button1["bg"] = "#ebd3f7"
        button1["activebackground"] = "#ffffff"
        button1["relief"] = "flat"
        ft = tkFont.Font(family='Ariel', size=10)
        button1["font"] = ft
        button1.place(x=90, y=140, width=167, height=109)

        label1 = tk.Label(self)
        ft = tkFont.Font(family='Ariel', size=20)
        label1["font"] = ft
        label1["bg"] = "#ebd3f7"
        label1["fg"] = "#000000"
        label1["justify"] = "center"
        label1["text"] = "Welcome to Memory Tiles"
        label1.place(x=7, y=100, width=340, height=30)


class SidePage(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        play = tk.PhotoImage(file="res/playbutton.png")
        player1 = tk.PhotoImage(file="res/user (1).png")
        player2 = tk.PhotoImage(file="res/user (2).png")
        backi = tk.PhotoImage(file="res/return1.png")
        poz1 = tk.PhotoImage(file="res/poz1.png")

        lbl = tk.Label(self, image=poz1)
        lbl.img = poz1
        lbl.place(relx=0.5, rely=0.5, anchor='center')

        button2 = tk.Button(self)
        button2["bg"] = "#ebd3f7"
        button2["bg"] = "#ebd3f7"
        button2["activebackground"] = "#ffffff"
        ft = tkFont.Font(family='Ariel', size=10)
        label = tk.Label(image=player1)
        label.image = player1
```

```
button2["image"] = player1
button2["font"] = ft
button2["fg"] = "#000000"
button2["justify"] = "center"
button2["relief"] = "flat"
button2["text"] = "Button"
button2["command"] = lambda: controller.show_frame(Page2)

button3 = tk.Button(self)
button3["bg"] = "#ebd3f7"
button3["activebackground"] = "#ffffff"
ft = tkFont.Font(family='Ariel', size=10)
label = tk.Label(image=player2)
label.image = player2
button3["image"] = player2
button3["font"] = ft
button3["fg"] = "#000000"
button3["justify"] = "center"
button3["relief"] = "flat"
button3["text"] = "Button"
button3["command"] = lambda: controller.show_frame(Page3)

back = tk.Button(self)
back["bg"] = "#ebd3f7"
back["activebackground"] = "#ffffff"
ft = tkFont.Font(family='Ariel', size=10)
back["font"] = ft
back["fg"] = "#000000"
back["justify"] = "center"
back["text"] = "Button"
lb = tk.Label(image=backi)
lb.image = backi
back["relief"] = "flat"
back["image"] = backi
back["command"] = lambda: controller.show_frame(MainPage)

back.place(x=290, y=280, width=39, height=42)
button2.place(x=60, y=120, width=100, height=100)
button3.place(x=190, y=120, width=100, height=100)
```

```
class Page2(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        global ip_got
        s = socket(AF_INET, SOCK_DGRAM)
        s.connect(("8.8.8.8", 80))
        ip_got = s.getsockname()[0]
        s.close()

        poz1 = tk.PhotoImage(file="res/poz1.png")

        lbl = tk.Label(self, image=poz1)
        lbl.img = poz1
        lbl.place(relx=0.5, rely=0.5, anchor='center')

        label21 = tk.Label(self)
        ft = tkFont.Font(family='Ariel', size=15)
        label21["font"] = ft
        label21["fg"] = "#000000"
        label21["bg"] = "#ebd3f7"
        label21["justify"] = "center"
        label21["text"] = "Your IP adress is: "

        label22 = tk.Label(self)
        ft = tkFont.Font(family='Ariel', size=24)
        label22["font"] = ft
        label22["fg"] = "#000000"
        label22["bg"] = "#ebd3f7"
        label22["justify"] = "center"
        label22["text"] = ip_got

        label23 = tk.Label(self)
        ft = tkFont.Font(family='Ariel', size=15)
        label23["font"] = ft
        label23["fg"] = "#000000"
        label23["bg"] = "#ebd3f7"
        label23["justify"] = "center"
        label23["text"] = "Tell your friend!"

        back = tk.Button(self)
        back["bg"] = "#ebd3f7"
```

```
        back["activebackground"] = "#ffffff"
        ft = tkFont.Font(family='Ariel', size=10)
        back["font"] = ft
        back["fg"] = "#000000"
        back["justify"] = "center"
        back["text"] = "Button"
        backi = tk.PhotoImage(file="res/return1.png")
        lb = tk.Label(image=backi)
        lb.image = backi
        back["image"] = backi
        back["relief"] = "flat"
        back["command"] = lambda: controller.show_frame(SidePage)

        button33 = tk.Button(self)
        setattr(self, "button33", button33)
        button33["bg"] = "#ebd3f7"
        button33["activebackground"] = "#ffffff"
        button33["activeforeground"] = "#836C92"
        ft = tkFont.Font(family='Ariel', size=20)
        button33["font"] = ft
        button33["fg"] = "#000000"
        button33["justify"] = "center"
        button33["text"] = "Ready?"
        button33["relief"] = "flat"
        button33["command"] = self.connect

        label21.place(x=90, y=100, width=175, height=30)
        label22.place(x=75, y=145, width=205, height=25)
        label23.place(x=77, y=190, width=200, height=30)
        back.place(x=290, y=280, width=39, height=42)
        button33.place(x=105, y=240, width=145, height=35)

    def connect(self):
        global win, ip_got
        serverPort = 12000  # isto kao za klijent
        serverSocket = socket(AF_INET, SOCK_DGRAM)
        serverSocket.bind(('', serverPort))
        images = []
        numb = []
        order = []
        message, clientAddress = serverSocket.recvfrom(2048)
```

```
        if (message.decode() == "1"):
            serverSocket.sendto(b"2", clientAddress)
            for i in range(20):
                numb.append(i)
            for i in range(20):
                a = random.randint(0, 19)
                while (numb[a] == -1):
                    a = random.randint(0, 19)
                numb[a] = -1
                order.append(a)
                serverSocket.sendto(a.to_bytes(5, 'little'), clientAddress)
            message, clientAddress = serverSocket.recvfrom(2048)
            if (message.decode() == "3"):
                serverSocket.close()
                windows.destroy(win)
                root2 = tk.Tk()
                pg = Game(root2, order, 1, ip_got)
                root2.mainloop()


class Page3(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        setattr(self, "controller", controller)

        poz1 = tk.PhotoImage(file="res/poz1.png")
        lbl = tk.Label(self, image=poz1)
        lbl.img = poz1
        lbl.place(relx=0.5, rely=0.5, anchor='center')

        label31 = tk.Label(self)
        ft = tkFont.Font(family='Ariel', size=15)
        label31["font"] = ft
        label31["fg"] = "#000000"
        label31["bg"] = "#ebd3f7"
        label31["justify"] = "center"
        label31["text"] = "What's your friends IP adress?"

        textBox1 = tk.Entry(self)
```

```
setattr(self, "textBox1", textBox1)
textBox1["borderwidth"] = "1px"
ft = tkFont.Font(family='Ariel', size=17)
textBox1["font"] = ft
textBox1["fg"] = "#000000"
textBox1["justify"] = "center"
textBox1["text"] = "Entry"

button33 = tk.Button(self)
setattr(self, "button33", button33)
button33["bg"] = "#ebd3f7"
button33["activebackground"] = "#ffffff"
button33["activeforeground"] = "#836C92"
ft = tkFont.Font(family='Ariel', size=20)
button33["font"] = ft
button33["fg"] = "#000000"
button33["justify"] = "center"
button33["text"] = "CONNECT"
button33["relief"] = "flat"
button33["command"] = self.button33_command

back = tk.Button(self)
back["bg"] = "#ebd3f7"
back["activebackground"] = "#ffffff"
ft = tkFont.Font(family='Ariel', size=10)
back["font"] = ft
back["fg"] = "#000000"
back["justify"] = "center"
back["text"] = "Button"
backi = tk.PhotoImage(file="res/return1.png")
lb = tk.Label(image=backi)
lb.image = backi
back["image"] = backi
back["relief"] = "flat"
back["command"] = self.back

label31.place(x=27, y=100, width=300, height=42)
textBox1.place(x=77, y=150, width=200, height=30)
button33.place(x=108, y=200, width=138, height=30)
back.place(x=290, y=280, width=39, height=42)
```

```
    def button33_command(self):
        global win
        textBox1 = getattr(self, "textBox1")
        ip = textBox1.get()
        root = tk.Toplevel()
        serverName = ip
        serverPort = 12000
        clientSocket = socket(AF_INET, SOCK_DGRAM)
        order = []

        message = b"1"
        clientSocket.sendto(message, (serverName, serverPort))
        response, serverAddress = clientSocket.recvfrom(2048)
        if (response.decode() == "2"):
            for i in range(20):
                response, serverAddress = clientSocket.recvfrom(2048)
                order.append(int.from_bytes(response, 'little'))
        clientSocket.sendto(b"3", (serverName, serverPort))
        clientSocket.close()

        windows.destroy(win)
        root2 = tk.Tk()
        pg = Game(root2, order, 2, ip)
        root2.mainloop()

    def back(self):
        textBox1 = getattr(self, "textBox1")
        controller = getattr(self, "controller")
        controller.show_frame(SidePage)
        textBox1.delete(0)


class Game:
    def __init__(self, root, order, player, ip):
        global r, pl
        pl = player
        r = root
        width = 432
        height = 414
        screenwidth = root.winfo_screenwidth()
        screenheight = root.winfo_screenheight()
```

```
alignstr = '%dx%d+%d+%d' % (width, height, (screenwidth -
width) / 2, (screenheight - height) / 2)
root.geometry(alignstr)
root.resizable(width=False, height=False)
root.protocol('WM_DELETE_WINDOW', self.close_root)

poz1 = tk.PhotoImage(file="res/poz4.png")
lbl = tk.Label(root, image=poz1)
lbl.img = poz1
lbl.place(relx=0.5, rely=0.5, anchor='center')

icon = tk.PhotoImage(file="res/stop.png")
root.iconphoto(False, icon)
global btn, images, j, cnt1, cnt2, pair, pl1, pl2
global label1, label2, square
global clientAddress, serverPort
global serverSocket, serverName, clientSocket
pair = 10
cnt1 = 0
cnt2 = 0
j = {
    0: 0,
    1: None,  # button j
    12: None,  # button j+1
    2: None,  # image j
    3: 2  # Player 2 uvek igra prvi
}
clientAddress = None
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
serverName = ip
clientSocket = socket(AF_INET, SOCK_DGRAM)
btn = []
pom = []
images = []
ft = tkFont.Font(family='Times', size=10)
square = tk.PhotoImage(file="res/square.png")
label = tk.Label(image=square)
label.image = square
for i in range(20):
```

```
    btn.append(tk.Button(root))
    btn[i]["bg"] = "#ebd3f7"
    btn[i]["font"] = ft
    btn[i]["fg"] = "#000000"
    btn[i]["justify"] = "center"
    btn[i]["text"] = str(i)
    btn[i]["image"] = square
    btn[i]["relief"] = "flat"
    if (player == 1):
        btn[i]["command"] = lambda name=i: self.server(name)
    else:
        btn[i]["command"] = lambda name=i: self.client(name)
for i in range(10):
    img = tk.PhotoImage(file="res/(" + str(i + 1) + ').png')
    pom.append(img)
    pom.append(img)
for i in range(20):
    images.append(pom[order[i]])

def place():
    global pl1, pl2, label1, label2

    ft = tkFont.Font(family='Times', size=10)
    label1["font"] = ft
    label1["fg"] = "#000000"
    label1["bg"] = "#ebd3f7"
    label1["justify"] = "center"
    label1["text"] = "Player 1:"
    label1.place(x=20, y=50, width=70, height=25)

    ft = tkFont.Font(family='Times', size=18)
    pl1["font"] = ft
    pl1["fg"] = "#000000"
    pl1["bg"] = "#ebd3f7"
    pl1["justify"] = "center"
    pl1["text"] = "0"
    pl1.place(x=40, y=80, width=70, height=25)

    ft = tkFont.Font(family='Times', size=10)
    label2["font"] = ft
    label2["fg"] = "red"
```

```
label2["bg"] = "#ebd3f7"
label2["justify"] = "center"
label2["text"] = "Player 2:"
label2.place(x=330, y=50, width=70, height=25)

ft = tkFont.Font(family='Times', size=18)
pl2["font"] = ft
pl2["fg"] = "red"
pl2["bg"] = "#ebd3f7"
pl2["justify"] = "center"
pl2["text"] = "0"
pl2.place(x=310, y=80, width=70, height=25)

label0 = tk.Label(root)
ft = tkFont.Font(family='Times', size=23)
label0["font"] = ft
label0["bg"] = "#ebd3f7"
label0["fg"] = "#000000"
label0["justify"] = "center"
label0["text"] = "Memory tiles"
label0.place(x=100, y=30, width=215, height=52)

btn[0].place(x=50, y=120, width=55, height=55)
btn[1].place(x=120, y=120, width=55, height=55)
btn[2].place(x=190, y=120, width=55, height=55)
btn[3].place(x=260, y=120, width=55, height=55)
btn[4].place(x=330, y=120, width=55, height=55)
btn[5].place(x=50, y=190, width=55, height=55)
btn[6].place(x=120, y=190, width=55, height=55)
btn[7].place(x=190, y=190, width=55, height=55)
btn[8].place(x=50, y=260, width=55, height=55)
btn[8].place(x=190, y=260, width=55, height=55)
btn[8].place(x=260, y=190, width=55, height=55)
btn[9].place(x=330, y=190, width=55, height=55)
btn[10].place(x=50, y=260, width=55, height=55)
btn[11].place(x=120, y=260, width=55, height=55)
btn[12].place(x=190, y=260, width=55, height=55)
btn[13].place(x=260, y=260, width=55, height=55)
btn[14].place(x=330, y=260, width=55, height=55)
btn[15].place(x=50, y=330, width=55, height=55)
btn[16].place(x=120, y=330, width=55, height=55)
```

```
    btn[17].place(x=190, y=330, width=55, height=55)
    btn[18].place(x=260, y=330, width=55, height=55)
    btn[19].place(x=330, y=330, width=55, height=55)

def call():
    global j, btn, images, cnt2, cnt1, square, pair
    global clientAddress, serverPort, serverSocket

    def timer2(a, b):
        time.sleep(1)
        a.config(image=square)
        b.config(image=square)
        j[3] = 1

    message, clientAddress = serverSocket.recvfrom(2048)
    if (message == b'0'):
        return
    k1 = int.from_bytes(message, 'big')
    btn[k1].config(image=images[k1])
    time.sleep(1)
    message, clientAddress = serverSocket.recvfrom(2048)
    k2 = int.from_bytes(message, 'big')
    btn[k2].config(image=images[k2])
    if (images[k2] == images[k1]):
        cnt2 += 1
        pl2.config(text=str(cnt2))
        pair -= 1
        btn[k1].config(state="disabled")
        btn[k2].config(state="disabled")
        j[3] = 1
    else:
        thread2 = Thread(target=timer2, args=(btn[k1], btn[k2]))
        thread2.start()

    pl2.config(fg="black")
    pl1.config(fg="red")
    label1.config(fg="red")
    label2.config(fg="black")

pl1 = tk.Label(root)
pl2 = tk.Label(root)
```

```
    label1 = tk.Label(root)
    label2 = tk.Label(root)
    place()
    if (player == 1):
        root.title("MemoryTiles - Player1")
        thread = Thread(target=call)
        thread.start()
    else:
        root.title("MemoryTiles - Player2")

def server(self, i):
    global j, btn, images, cnt2, cnt1, square, pair
    global clientAddress, serverPort, serverSocket

    def call():
        global j, btn, images, cnt2, cnt1, square, pair
        global clientAddress, serverPort, serverSocket

        pl1.config(fg="black")  # server
        pl2.config(fg="red")
        label2.config(fg="red")
        label1.config(fg="black")

        def timer2(a, b):
            time.sleep(1)
            a.config(image=square)
            b.config(image=square)
            pl2.config(fg="black")
            pl1.config(fg="red")
            label1.config(fg="red")
            label2.config(fg="black")
            j[3] = 1

        message, serverAddress = serverSocket.recvfrom(2048)
        if (message == b'0'):
            return
        k1 = int.from_bytes(message, "big")
        btn[k1].config(image=images[k1])
        time.sleep(1)
        message, serverAddress = serverSocket.recvfrom(2048)
        k2 = int.from_bytes(message, "big")
```

```
        btn[k2].config(image=images[k2])

        if (images[k2] == images[k1] and btn[k2] != btn[k1]):
            cnt2 += 1
            pl2.config(text=str(cnt2))
            pair -= 1
            btn[k1].config(state="disabled")
            btn[k2].config(state="disabled")
            if (pair == 0):
                self.game_Over()
                return
            pl2.config(fg="black")
            pl1.config(fg="red")
            label1.config(fg="red")
            label2.config(fg="black")
            j[3] = 1
        else:
            t3 = Thread(target=timer2, args=(btn[k1], btn[k2]))
            t3.start()

    t2 = Thread(target=call)

    def timer():
        global j
        time.sleep(1)
        j[1].config(image=square)
        j[12].config(image=square)
        j[0] = 0
        j[1] = None
        j[12] = None
        j[2] = None
        t2.start()

    if (j[3] == 1):  # red je na plejera 1
        t = threading.Thread(target=timer)
        btn[i].config(image=images[i])
        if (j[0] == 0):
            serverSocket.sendto(i.to_bytes(5, 'big'), clientAddress)
            j[2] = images[i]
            j[1] = btn[i]
            j[0] += 1
```

```
            return
        else:
            j[12] = btn[i]
            serverSocket.sendto(i.to_bytes(5, 'big'), clientAddress)
            if (j[2] == images[i] and j[1] != j[12]):
                cnt1 += 1
                pl1.config(text=str(cnt1))
                pair -= 1
                j[1].config(state="disabled")
                j[12].config(state="disabled")
                if (pair == 0):
                    self.game_Over()
                    return
                j[0] = 0
                j[1] = None
                j[12] = None
                j[2] = None
                j[3] = 2
                t2.start()  # call
            else:
                j[3] = 2
                t.start()

def client(self, i):
    global j, btn, images, cnt2, cnt1, square, pair
    global serverName, serverPort, clientSocket

    def timer2(a, b):
        time.sleep(1)
        a.config(image=square)
        b.config(image=square)
        pl1.config(fg="black")
        pl2.config(fg="red")
        label2.config(fg="red")
        label1.config(fg="black")
        j[3] = 2

    def call2():
        global j, btn, cnt2, cnt1, pair

        pl2.config(fg="black")
```

```
    pl1.config(fg="red")
    label1.config(fg="red")
    label2.config(fg="black")
    response, serverAddress = clientSocket.recvfrom(2048)
    if (response == b'0'):
        return
    k1 = int.from_bytes(response, 'big')
    btn[k1].config(image=images[k1])
    time.sleep(1)
    response, serverAddress = clientSocket.recvfrom(2048)
    k2 = int.from_bytes(response, 'big')
    btn[k2].config(image=images[k2])
    if (images[k2] == images[k1] and btn[k2] != btn[k1]):
        cnt1 += 1
        pl1.config(text=str(cnt1))
        pair -= 1
        btn[k1].config(state="disabled")
        btn[k2].config(state="disabled")
        if (pair == 0):
            self.game_Over()
            return
        pl1.config(fg="black")
        pl2.config(fg="red")
        label2.config(fg="red")
        label1.config(fg="black")
        j[3] = 2
    else:
        t3 = Thread(target=timer2, args=(btn[k1], btn[k2]))
        t3.start()

t2 = Thread(target=call2)

def timer():
    global j
    time.sleep(1)
    pl2.config(fg="black")
    pl1.config(fg="red")
    label1.config(fg="red")
    label2.config(fg="black")
    j[1].config(image=square)
    j[12].config(image=square)
```

```
            j[0] = 0
            j[1] = None
            j[12] = None
            j[2] = None
            t2.start()

        if (j[3] == 2):  # red je na plejera 2.
            btn[i].config(image=images[i])
            t = threading.Thread(target=timer)
            if (j[0] == 0):
                j[2] = images[i]
                j[1] = btn[i]
                j[0] += 1
                clientSocket.sendto(i.to_bytes(5, 'big'),
                (serverName, serverPort))
                return
            else:
                j[12] = btn[i]
                clientSocket.sendto(i.to_bytes(5, 'big'),
                (serverName, serverPort))
                if (j[2] == images[i] and j[1] != j[12]):
                    cnt2 += 1
                    pl2.config(text=str(cnt2))
                    pair -= 1
                    j[1].config(state="disabled")
                    j[12].config(state="disabled")
                    if (pair == 0):
                        self.game_Over()
                        return
                    j[0] = 0
                    j[1] = None
                    j[12] = None
                    j[2] = None
                    j[3] = 1
                    t2.start()
                else:

                    j[3] = 1
                    t.start()

    def game_Over(self):
```

```
            global cnt1, cnt2
            if (cnt1 > cnt2):
                messagebox.showinfo("MemoryTiles", "Game Over - Player 1 wins!")
            elif (cnt2 > cnt1):
                messagebox.showinfo("MemoryTiles", "Game Over - Player 2 wins!")
            else:
                messagebox.showinfo("MemoryTiles", "Game Over - Tie")
            serverSocket.close()
            clientSocket.close()
            r.destroy()

        def close_root(self):
            if (j[3] == 1 and pl == 2):
                serverSocket2 = socket(AF_INET, SOCK_DGRAM)
                serverSocket2.bind(('', 1500))
                serverSocket2.sendto(b'0', (ip_got, 1500))
                serverSocket2.close()
            elif ((j[3] == 2 and pl == 1) or (j[3] == 0 and pl == 1)):
                clientSocket2 = socket(AF_INET, SOCK_DGRAM)
                clientSocket2.sendto(b'0', (serverName, serverPort))
                clientSocket2.close()
            serverSocket.close()
            clientSocket.close()
            r.destroy()


    if __name__ == "__main__":
        testObj = windows()
        testObj.mainloop()
    exit(0)
```