Dokumentacja Programu : Sonogram

Autor: Juliusz Stańczyk 107408

1.Temat

a. Program pokazuje wykres oraz sonogram sygnału dźwiękowego

b. Pobiera dźwięk z pliku (.wav) – mono i stereo

c. Nagrywa własny dźwięk (warunkiem jest mikrofon)

d. Wyświetlenie sonogramu w wersji Hamminga

e. Odtwarza dźwięk poprzez otworzenie wbudowanego programu Windows do odtwarzania muzyki

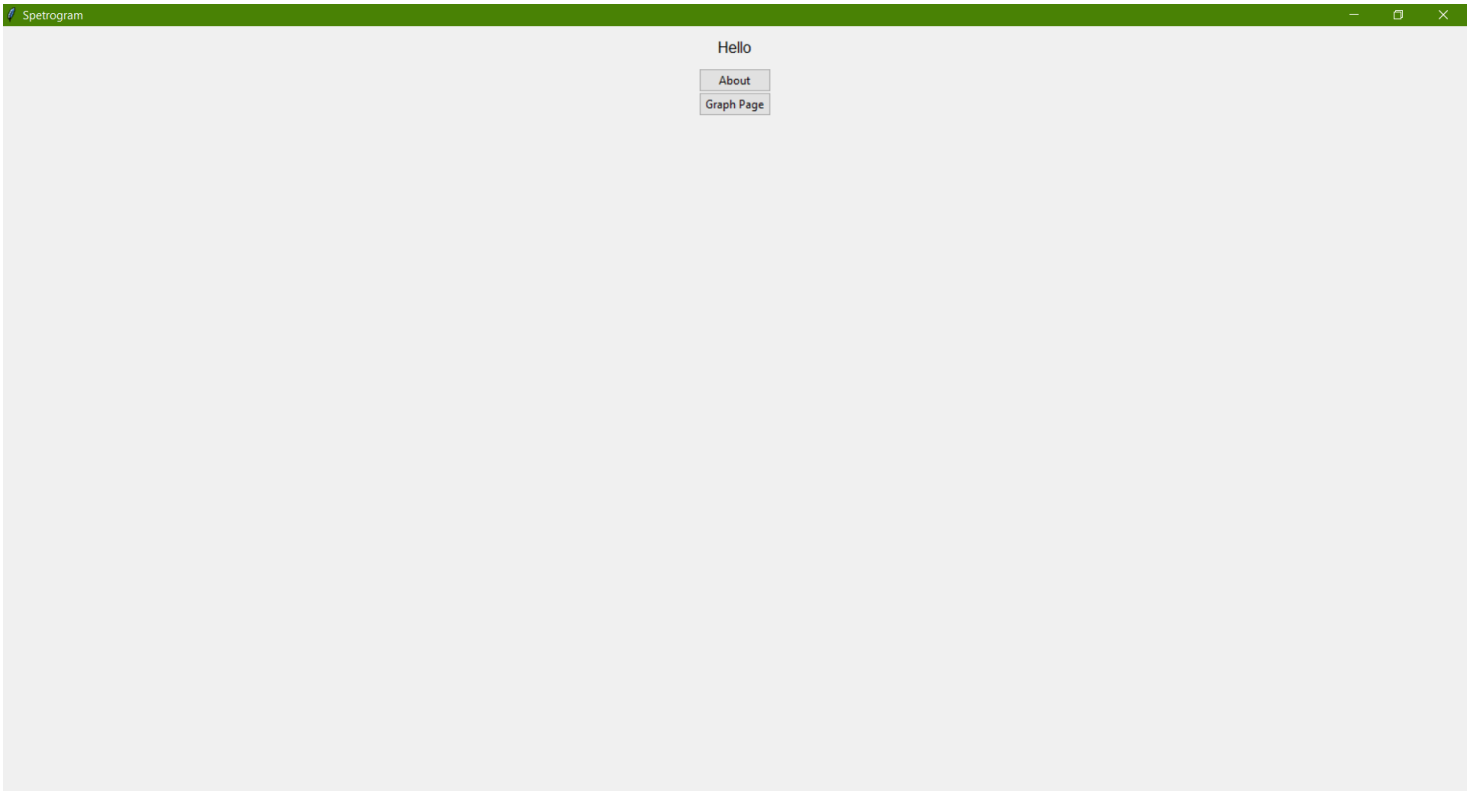
f. Przybliżenie wykresu spektrogramu, przesówanie oraz powrót do oryginalnego rozmiaru

2.Specyfikacja użytkownika

Program tworzony był języku programowania Python 3.8 i tylko na tej wersji był testowany. Środowiskiem programistycznym był PyCharm w wersji 2019.2.3.

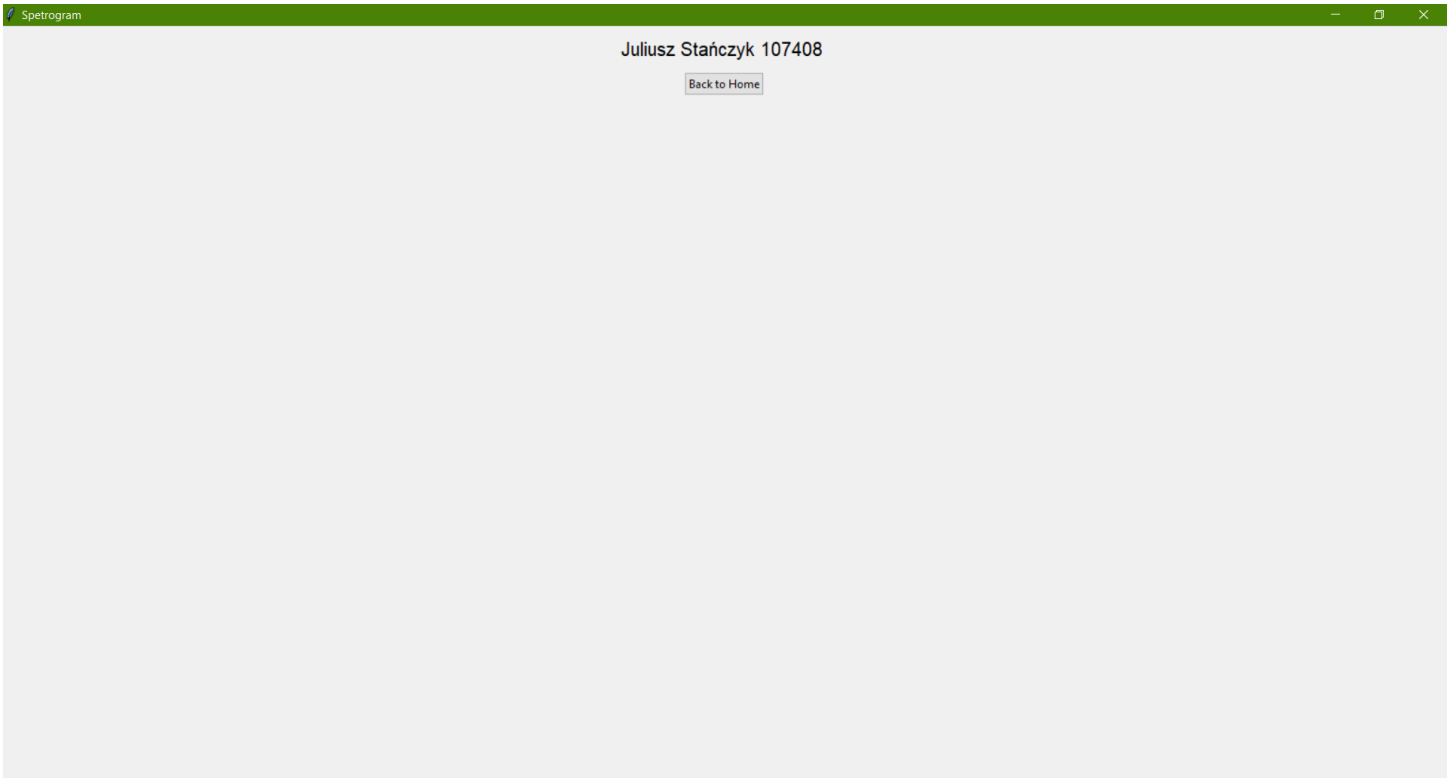
Po wypakowaniu programu należy uruchomić środowisko PyCharm. Następnie File -> Open... -> (wyszukać folder programu o nazwie SonogramProjektPy) -> OK. Program jest gotowy do skompilowania i uruchomienia. Plikiem głównym programu jest „sonomain.py”. Plik zawiera również przykładowe dźwięki.

Po uruchomieniu pojawia się menu:



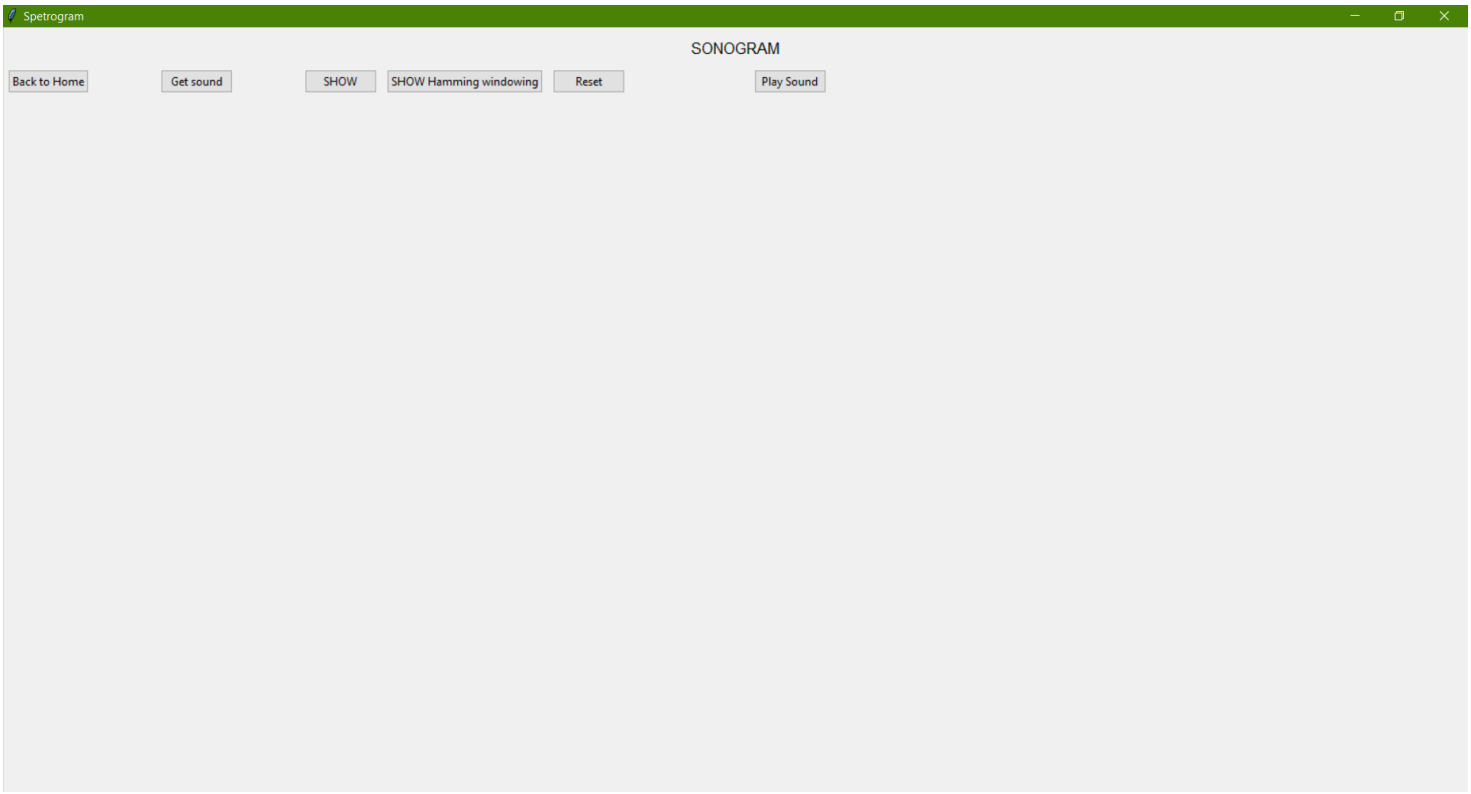
Mamy do wyboru:

1. About – informacje: moje imię, nazwisko i numer indeksu



Przycisk Back to Home wraca do menu.

2. Graph Page – przejście do głównego menu sonogramu

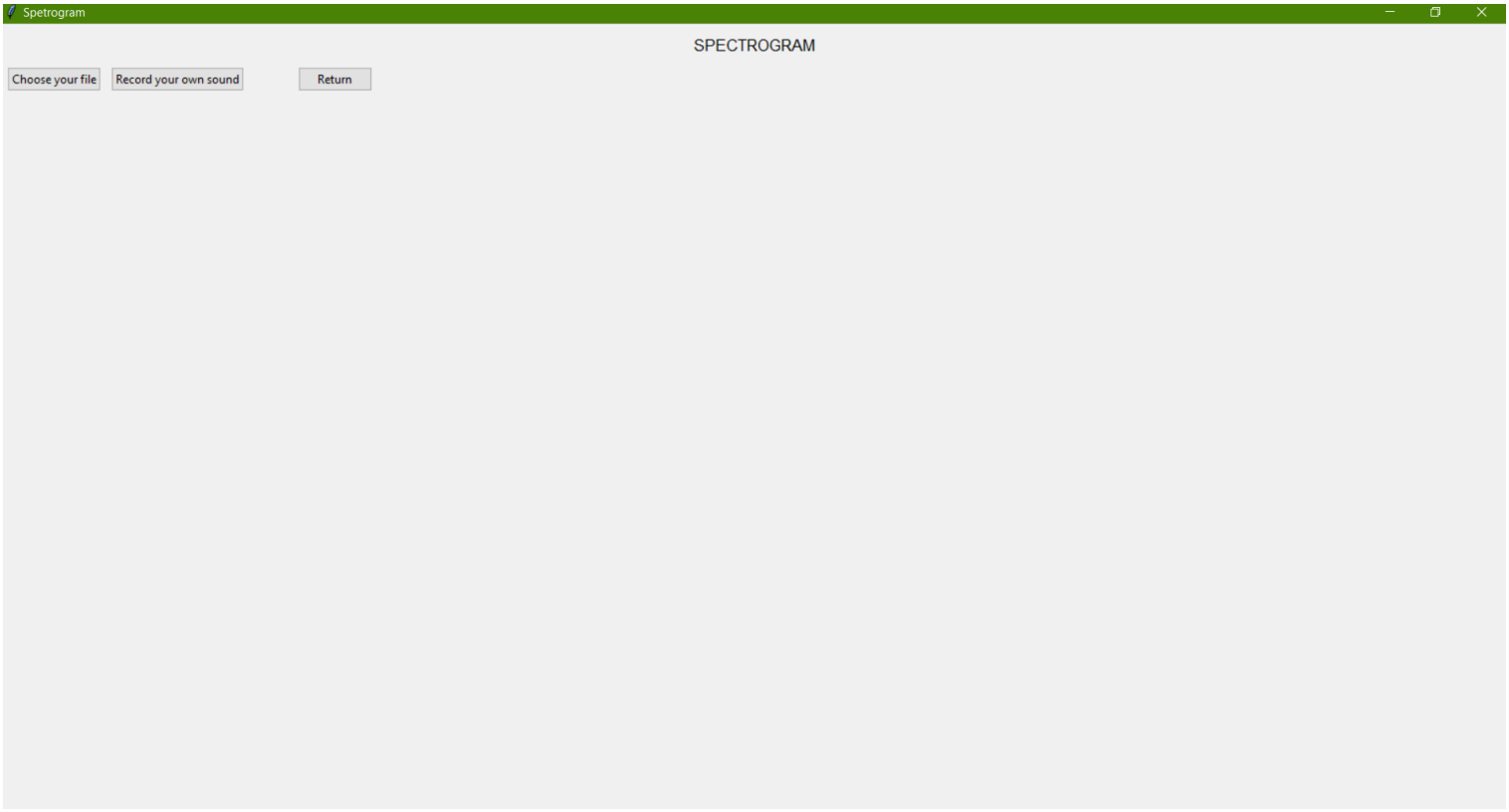


Mamy do wyboru:

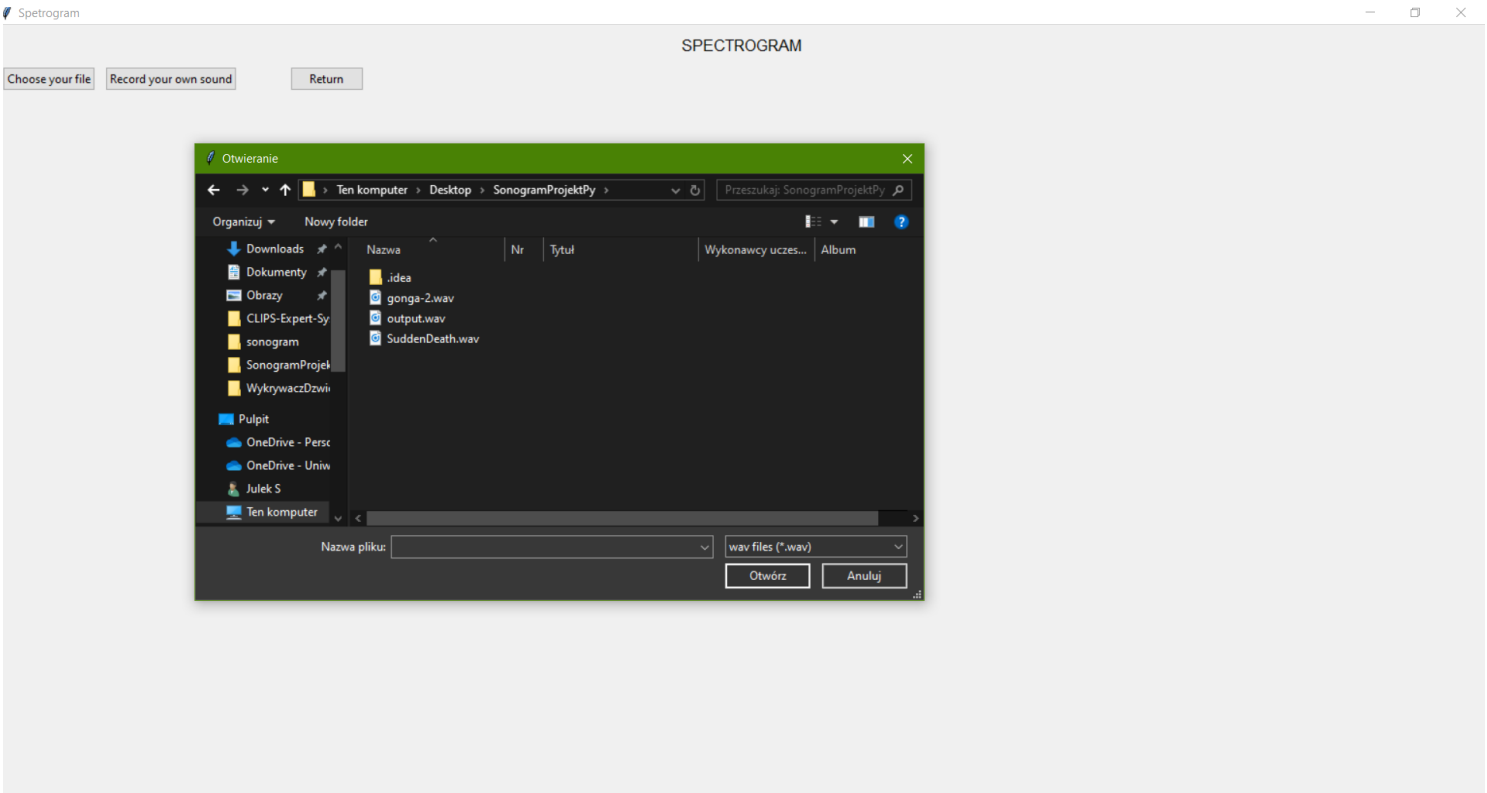
1. Back to Home – wraca do głównego menu
2. Get sound – Przechodzi do panelu z pobraniem dźwięku (będzie omówione potem)
3. SHOW – pokazuje sonogram, dźwięk oraz pasek narzędzi (dopiero po wczytaniu/nagraniu dźwięku)
4. SHOW Hamming Windowing – pokazuje sonogram w wersji Hamminga (dopiero po wczytaniu/nagraniu dźwięku)
5. Reset – resetuje wyświetlony sonogram (wymagane do ponownego wgrania kolejnego dźwięku lub wyświetlenia sonogramu)
6. Play sound – otwiera okno Windows z odtwarzaczem dźwięku i wybranym dzi

a.kiem (wymagane jest wcześniejsze wybranie dźwięku)

Po kliknięciu Get sound mamy:



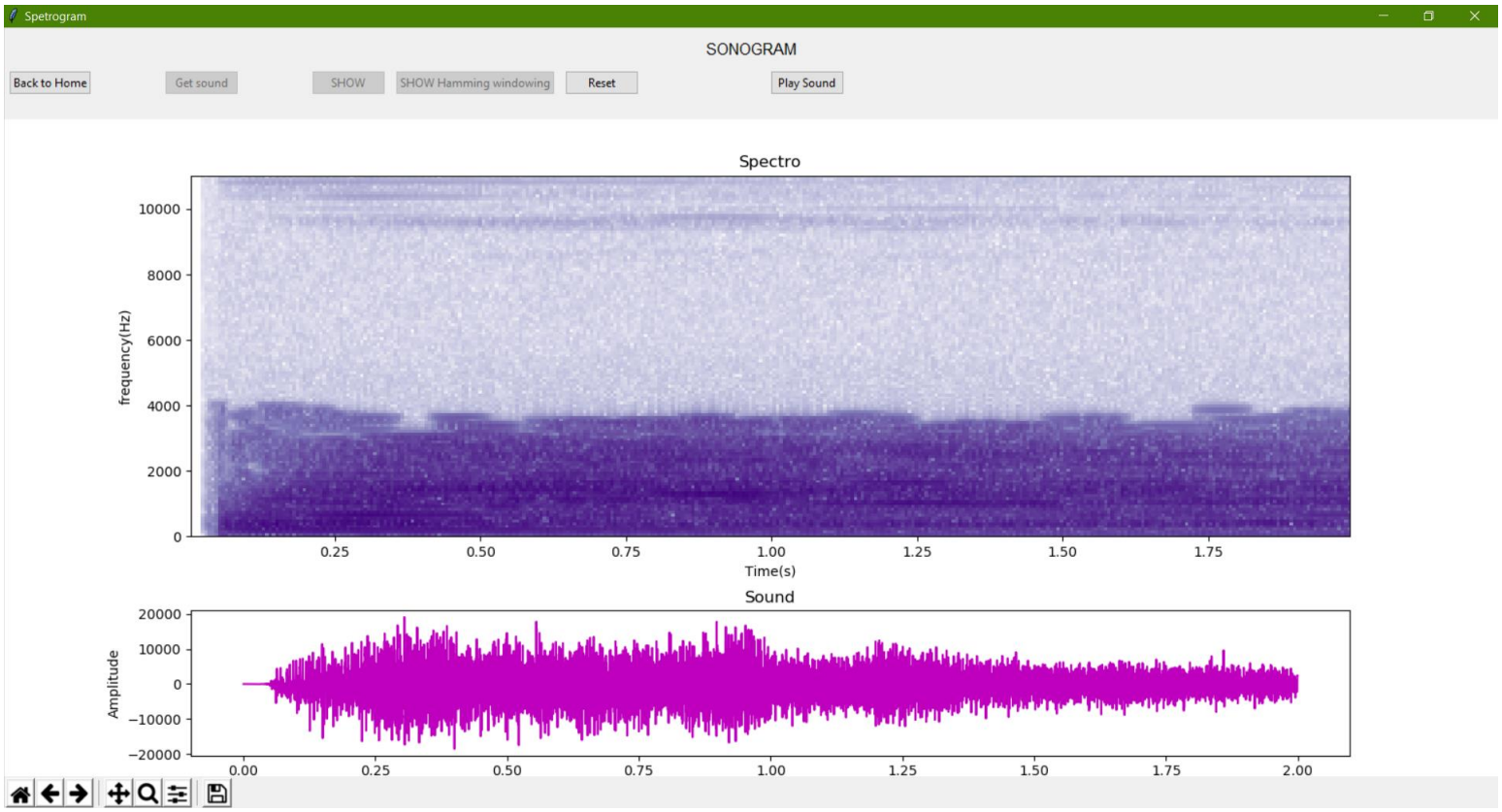
1. Choose your file – Otwiera okno windows z możliwością wybrania dźwięku typu .wav



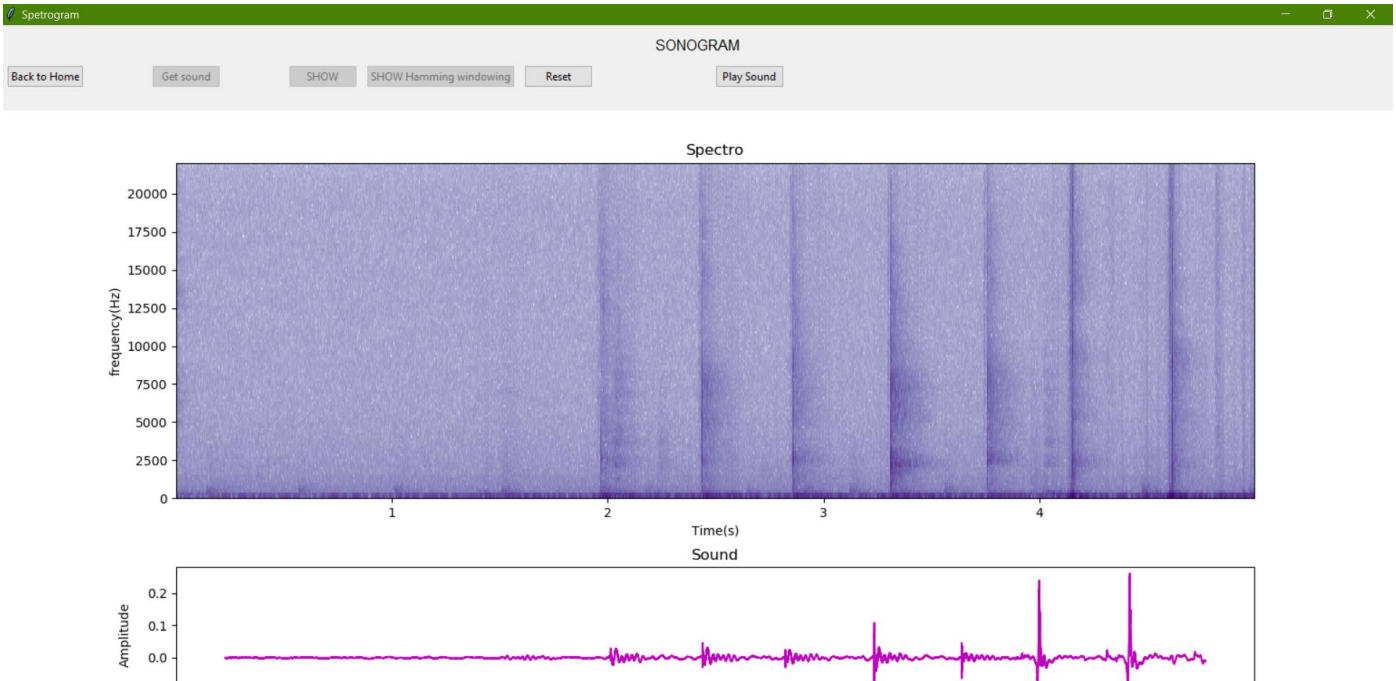
2. Record your own sound – Pokazuje przycisk Start Recording, który aktywuje 5 sekundowe nagrywanie (warunkiem jest mikrofon)

3. Return – Powrót do menu z sonogramem

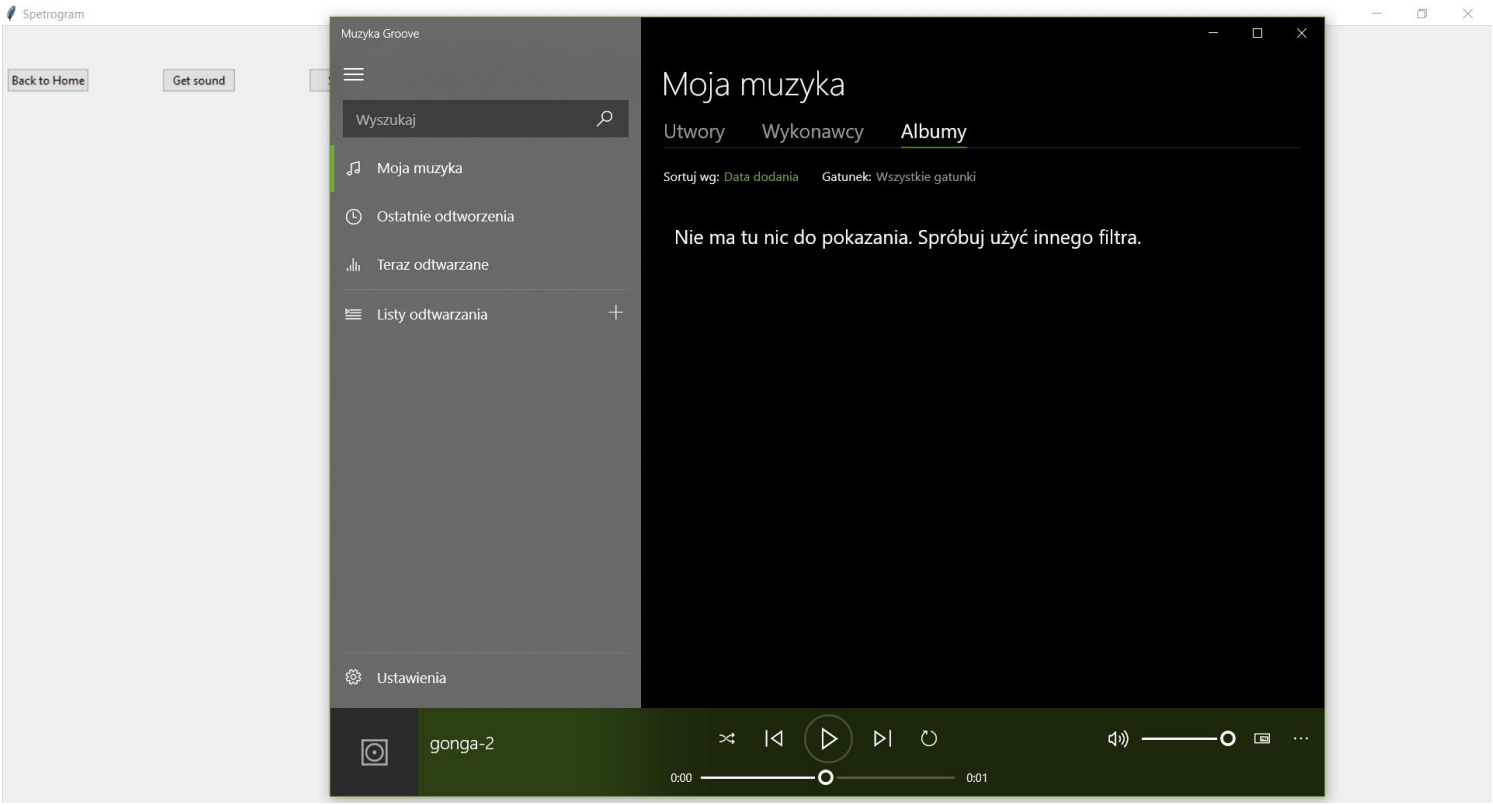
Po załadowaniu/nagraniu dźwięku i kliknięciu SHOW mamy:



Po załadowaniu/nagraniu dźwięku u kliknięciu SHOW Hamming windowing mamy:



Po kliknięciu Play Sound mamy:



Po kliknięciu SHOW lub SHOW Hamming windowking u dołu ekranu pokazuje się również pasek z narzędziami:

1. „Dom” - powraca do głównego widoku (warunkiem jest zmiana widoku np. Użyciu „lupy”)
2. „Strzałki” – cofają ostatni krok w użyciu paska np. „Lupy” (warunkiem jest, że któreś z narzędzi zostało użyte)
3. „Kursor krzyż z czterema strzałkami” – Pozwala przesuwać wykres sonogramu
4. „Lupa” pozwala przybliżać wykres sonogramu. Po wybraniu „Lupy” należy zaznaczyć na wykresie fragment do przybliżenia
5. „Suwaki” – otwiera okno z możliwością zmiany wielkości wykresu

6. „Dyskietka” – otwiera okno Windows z możliwością zapisu wykresu

3. Kod programu

a. Używane biblioteki:

```
import matplotlib
import scipy

matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.backends.backend_tkagg import NavigationToolbar2Tk

import tkinter as tk
from tkinter import ttk
from tkinter import filedialog as fd

from scipy.io import wavfile
import matplotlib.pyplot as plot

import sounddevice as sd
from scipy.io.wavfile import write
from pathlib import Path

import os

import numpy as np
```

b. Fragment odpowiadający za wczytanie i pobranie parametrów sygnału (klasa GetSound)

Funkcja getsnd - wczytanie z pliku:

```
def getsnd(self):
    global filename
    filename = None
    filename = fd.askopenfilename(filetypes=[("wav files", ".wav")])
    self.button6.config(state="disabled")
    if(filename != None and filename!= ""):
            self.label8 = tk.Label(self, text="Your file has been successfully uploaded")
            self.label8.pack(anchor="nw",side='left', padx=5)
    else:
            self.label7 = tk.Label(self, text="Your file has NOT been successfully
uploaded. Return and try to load file again")
            self.label7.pack(anchor="nw", side='left', padx=5)
```

Funkcja RecordPom - nagranie własne:

```
def RecordPom(self):
    fs = 44100
    seconds = 5  # Duration of recording
    global filename
    sound = sd.rec(int(seconds * fs), samplerate=fs, channels=1)
    sd.wait()  # Wait until recording is finished
    self.label5 = tk.Label(self, text="Finished")
    self.label5.pack(anchor="nw", side='left', padx=5)
    write('output.wav', fs, sound)
    dirname = 'C:/Users/julek/Desktop/SonogramProjektPy'
    fname = 'output'
    suffix = '.wav'
    filename = Path(dirname, fname).with_suffix(suffix)
```

c. Wyświetlanie wykresu wave i sonogramu razem z wczytaniem ścieżki pobranej w wyższych funkcjach (klasa PageThree)

Funkcja gtsnd - wyświetlanie wykresu:

```
def gtsnd(self, button5, button8, button4):
    button5.config(state="disabled")
    button8.config(state="disabled")
    button4.config(state="disabled")

    if (self.canvas2 == None):
        self.x = 1

        rate, k = wavfile.read(filename)

        self.f1, self.a1 = plot.subplots(1)
        self.a1.clear()

        self.a1.set_title('Spectro')
        plot.xlabel("Time(s)")
        plot.ylabel("frequency(Hz)")

        try:
            self.a1.specgram(k, Fs=rate, cmap='Purples')
        except ValueError:
            k = k[:,1]
            self.a1.specgram(k, Fs=rate, cmap='Purples')

        self.canvas2 = FigureCanvasTkAgg(self.f1)
        self.canvas2.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

        self.toolbar = NavigationToolbar2Tk(self.canvas2, app)
        self.toolbar.update()
```

```
        self.canvas2._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

        self.f2, self.a2 = plot.subplots(1)
        Time = np.linspace(0, len(k) / rate, num=len(k))
        self.a2.clear()
        self.a2.set_title('Sound')
        plot.xlabel("Time(s)")
        plot.ylabel("Amplitude")
        #formatter = matplotlib.ticker.FuncFormatter(lambda ms, x:
time.strftime('%M:%S', time.gmtime(ms // 1000)))
        #self.a2.xaxis.set_major_formatter(formatter)
        self.a2.plot(Time, k, color="m")

        self.canvas3 = FigureCanvasTkAgg(self.f2)
        self.canvas3.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)


        self.canvas3._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)
```

Funkcja Hamming - wyświetlanie wykresu Hamminga:

```
def Hamming(self, button5,button8, button4):
    button5.config(state="disabled")
    button8.config(state="disabled")
    button4.config(state="disabled")

    if (self.canvas2 == None):
        self.x = 1

        rate, k = wavfile.read(filename)

        self.f1, self.a1 = plot.subplots(1)
        self.a1.clear()

        self.a1.set_title('Spectro')
        plot.xlabel("Time(s)")
        plot.ylabel("frequency(Hz)")

        try:
            self.a1.specgram(k, Fs=rate, window=scipy.hamming(256), NFFT=256,
cmap='Purples')
        except ValueError:
            k = k[:,1]
            self.a1.specgram(k, Fs=rate, window=scipy.hamming(256), NFFT=256,
cmap='Purples')

        self.canvas2 = FigureCanvasTkAgg(self.f1)
        self.canvas2.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

        self.toolbar = NavigationToolbar2Tk(self.canvas2, app)
        self.toolbar.update()
```

```
        self.canvas2._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

        self.f2, self.a2 = plot.subplots(1)
        Time = np.linspace(0, len(k) / rate, num=len(k))
        self.a2.clear()
        self.a2.set_title('Sound')
        self.a2.plot(Time, k, color="m")
        plot.xlabel("Time(s)")
        plot.ylabel("Amplitude")
        #formatter = matplotlib.ticker.FuncFormatter(lambda ms, x:
time.strftime('%M:%S', time.gmtime(ms // 1000)))
        #self.a2.xaxis.set_major_formatter(formatter)

        self.canvas3 = FigureCanvasTkAgg(self.f2)
        self.canvas3.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

        self.canvas3._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)
```

d. Okienka (reszta kodu)

Główna klasa MMain tworząca pozostałe klasy-okna i określająca ich własności

```
class MMain(tk.Tk):

    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)

        # tk.Tk.iconbitmap(self, default="clienticon.ico")
        tk.Tk.wm_title(self, "Spetrogram")
        tk.Tk.wm_state(self,'zoomed')


        cont = tk.Frame(self)
        cont.pack(side="top", fill="both", expand=True)
        cont.grid_columnconfigure(0, weight=1)
        cont.grid_rowconfigure(0, weight=1)

        self.frames = {}

        for FR in (StartPage, PageOne, PageThree, GetSound):
            frame = FR(cont, self)

            self.frames[FR] = frame

            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame(StartPage)

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

Okienko startowe – Klasa StartPage

```
class StartPage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Hello", font=LARGE_FONT)
        label.pack(pady=10, padx=10)

        button = ttk.Button(self, text="About",command=lambda:
controller.show_frame(PageOne))
        button.pack()

        button3 = ttk.Button(self, text="Graph Page",command=lambda:
controller.show_frame(PageThree))
        button3.pack()
```

Okienko autor – klasa PageOne

```
class PageOne(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Author: ", font=LARGE_FONT)
        label = tk.Label(self, text="Juliusz Stańczyk 107408 ", font=14)
        label.pack(pady=10, padx=10)

        button1 = ttk.Button(self, text="Back to Home", command=lambda:
controller.show_frame(StartPage))
        button1.pack()
```

Klasa GetSound – odpowiada za pobranie dźwięku:

Funkcja init – pokazuje przyciski potrzebne do nagrania/wczytania dźwięku:

```
def __init__(self, parent, controller):
    tk.Frame.__init__(self, parent)
    self.label = tk.Label(self, text="SPECTROGRAM", font=LARGE_FONT)
    self.label.pack(pady=10, padx=10)
    self.label5 = tk.Label()
    self.label4 = tk.Label()
    self.label7 = tk.Label()
    self.label8 = tk.Label()
    self.button8 = ttk.Button()
    self.button6 = ttk.Button(self, text="Choose your file", command=self.getsnd)
    self.button6.pack(anchor="nw",side='left', padx=5)

    self.button1 = ttk.Button(self, text="Record your own sound", command=lambda:
self.Record())
    self.button1.pack(anchor="nw", side='left', padx=5)
```

```
    self.button5 = ttk.Button(self, text="Return", command=lambda:
[controller.show_frame(PageThree), self.MsgDestr()])
    self.button5.pack(anchor="nw",side='left', padx=50)
```

Funkcja Record - przyciski potrzebne do nagrania (po kliknięciu Record your own sound):

```
def Record(self):
    self.button6.config(state="disabled")
    self.button1.config(state="disabled")
    self.label4 = tk.Label(self, text="Recording for 5 second. Click Start Recording")
    self.label4.pack(anchor="nw", side='left', padx=5)

    self.button8 = ttk.Button(self, text="Start Recording", command=lambda:
self.RecordPom())
    self.button8.pack(anchor="nw", side='left', padx=5)
```

Funkcja MsgDestr - odpowiedzialna za wyłączenie przycisków i usunięcie przycisków i wiadomości po wyjściu z okna nagrywania:

```
def MsgDestr(self):
    self.button6.config(state="enabled")
    self.button1.config(state="enabled")

    self.label5.destroy()
    self.label7.destroy()
    self.label8.destroy()
    self.button8.destroy()
    self.label4.destroy()
    #tk.Frame.destroy()
```

Oraz funkcje getsnd oraz RecordPom omówione wyżej

Klasa PageThree odpowiedzialna za główny panel oraz wykresy

Funkcja init – Przyciski i wygląd

```
def __init__(self, parent, controller):
    self.x = 0
    tk.Frame.__init__(self, parent)
    self.label = tk.Label(self, text="SONOGRAM", font=LARGE_FONT)
    self.label.pack(pady=10, padx=10)
    self.canvas2 = None

    button1 = ttk.Button(self, text="Back to Home", command=lambda:
controller.show_frame(StartPage))
    button1.pack(anchor="nw",side='left', padx=5)

    button4 = ttk.Button(self, text="Get sound", command=lambda:
```

```
[controller.show_frame(GetSound), self.SetCanv()])
    button4.pack(anchor="nw",side='left', padx=70)

    button5 = ttk.Button(self, text="SHOW", command=lambda: self.gtsnd(button5, button8,
button4))
    button5.pack(anchor="nw",side='left', padx=5)

    button8 = ttk.Button(self, text="SHOW Hamming windowing", command=lambda:
self.Hamming(button5, button8, button4))
    button8.pack(anchor="nw", side='left', padx=5)

    button7 = ttk.Button(self, text="Reset", command=lambda: self.ResetSono(button5,
button8, button4))
    button7.pack(anchor="nw",side='left', padx=5)

    button7 = ttk.Button(self, text="Play Sound", command=lambda: self.Play())
    button7.pack(anchor="nw", side='left', padx=130)
```

Funkcja Play – otwiera okno odtwarzania

```
def Play(self):
    os.startfile(filename)
```

Funkcja SetCanv – tworzy „canvas” czyli pole operacji na którym będą wyświetlane wykresy i przyciski oraz usuwa label3 czyli napis, który jest już niepotrzebny:

```
def SetCanv(self):
    self.canvas2 = None
    self.label3.destroy()
```

Funkcja ResetSono – resetuje przyciski, usuwa niepotrzebne przyciski, napisy i wykresy:

```
def ResetSono(self,button5, button8, button4):
    button5.config(state="enabled")
    button8.config(state="enabled")
    button4.config(state="enabled")
    self.canvas2.get_tk_widget().pack_forget()
    self.canvas3.get_tk_widget().pack_forget()
    self.toolbar.destroy()
    pom = 0
    if (pom == 0):
        pom = 1
        self.label3 = tk.Label(self, text="Please choose your sound file again")
        self.label3.pack(anchor="nw", side='left', padx=5)
```

Oraz funkcje gtsnd oraz Hamming, które były omówione już wcześniej.