Port Said University

Facility Of Engineering

Audio Manipulation Project

Name:	Abd Elrahman Ashraf Elmorsy Shalaby Abd Elrahman Ahmed Seliman Elsharabasy
Level:	2^{nd} Year
Department:	Electrical Power Engineering Dept
Specializing:	Computer And Control

Introduction:

Our Team With Help Of Some Science Articles Manage To Manipulate The Sound Wave Mono, Or Stereo In A Detail. Which Was Possible Only Because The Deep Level Of Understanding The Structure Of The Wave, And It Been Stored In Memory.

We Built A GUI App Using Python Aimed To Manipulate Sound Waves Using Several Operations Like Increasing The Amplitude, Shifting, Reversing And Change The Speed Of The Sound Wave Which Had Been Studied On The Current Term.

Overview

In This Section We Will Show How We Manage To Make The App Step By Step.

- Frist, We Use The Wave Library. From It We Use The Function Wave. Open After Importing From The Pc.
  Then We Get The Number Of Channels , Max Amplitude, And Sample Rate Then, Show It In The GUI.

- The App Read The Frames In A List Called Raw Then, Convert The Type Of The List To Be A Two From Two-Byte Shape To An Int16 Shape.
  Then It Makes A Row Of Numbers That Equal The Number Of Seconds Of The Main Sound, But With Duration Equal To The Sample Rate Between Each Number.

```
def importfunc():
    updatefr(wvFr)
    updatefr(wvFrMod)
    # open window to select the wav file and get the path to the audio dile then save in variable
    filename = filedialog.askopenfilename(initialdir="\\", title="Select Audio File", filetypes=((
    global direc
    direc = str(filename)
    if direc == '':
        messagebox.showinfo("info", "No File Was Selected")
    else:
        global wav
        wav = wave.open(direc, "rb")
        result = readfile(wav)
        global hasImported
        hasImported = True
        # Update displayed File info
        wavD = AudioSegment.from_file(file=direc, format="wav")
        maxAmp = wavD.max
        fileTypeVal.config(text='wav File')
        channelsVal.config(text=nChannels)
```

The Operations

We First Open A New Wave File To Write The Changes The Program Will Make

- Then We Will Set The Information Of The Wave Like The Number Of Channels
  And The Sample Width By 2.

- The First Operation Will Be The Stretching And Compression We Use Our Knowledge Of The Frame Rate Read Speed And Multiply It By The Number The User Input On The GUI, Which Makes Us Able To Change The Speed Of The Wave And Use This Factor To Execute The Stretching And Compression Operations.

```
def operations(AmpAmount, ShiftAmount, SpeedAmount, ReverseState, echoState):
    updatefr(wvFrMod)
    obj = wave.open('.\\audio\\Modified.wav', 'wb')
    obj.setnchannels(nChannels)
    obj.setsampwidth(2)
    # speed OP
    speed_factor = SpeedAmount
    speed = sampleRate * speed_factor
    obj.setframerate(speed)
```

- Next, The Shift Operation Which Is One Of The Things That Make This Program Unique Here We Take The Number Of Seconds The User Want To Make The Shift With, The Multiply It By The Reading Frame Rate Which We Called Here Sample Rate The With The Help Of A Loop We Write The Amount Of Shift The User Want In New File We Open To Write

```
# Shift OP
pov_sheft_in_sec = ShiftAmount
for i in range(int(sampleRate * pov_sheft_in_sec)):
    zeroinbyte = struct.pack('<h', 0)
    obj.writeframesraw(zeroinbyte)
```

- Next, Changing Amplitude Operation We Will Loop Over The List Of The Sample We Read From The Input Wave And Multiply Each Sample By The Amplitude Factor The Used Input.

You Maybe Wander What If The Amplitude Of The Sample Became Out Of 2byte Range, We Took Care Of This Of Course We Will Walk Through The Solution To This Particular Problem Later On.

- Next, The Reversing Operation Here We Will Reverse The List Which Contain The Samples From The Input Sound File.

  Then Make Sure That Every Sample Did not Exceed The Limit Of 2 Byte Sample
  And If It Exceed The App Will Make It Either 32760 Or -32760 Depending On The Original Value.

  Then Write Each Sample In The Output Sound File By Getting Help From Pack Function From The Struct Library.

```
# Amplification OP
amp = AmpAmount
n = len(data)
# Reverse OP
reverse = ReverseState
if reverse:
    for i in range(data.__len__()):
        twobytesample = data[n - 1 - i] * amp
        if twobytesample > 32760:
            twobytesample = 32760
        if twobytesample < -32760:
            twobytesample = -32760
        sample = struct.pack('<h', int(twobytesample))
        obj.writeframesraw(sample)
else:
    for i in range(data.__len__()):
        twobytesampler = data[i] * amp
        if twobytesampler > 32760:
            twobytesampler = 32760
        if twobytesampler < -32760:
            twobytesampler = -32760
        sample = struct.pack('<h', int(twobytesampler))
        obj.writeframesraw(sample)
```

- Finally, Echo Operation Using Audiolib Library We Make The Value Of The Echo Fixed By 0.4 Sec And Use The Function Set Echo.

```
# set the echo based on state
if echoState:
    sound1 = AudioProcessing('.\\audio\\Modified.wav'
    sound1.set_echo(0.4)
    sound1.save_to_file('.\\audio\\Modified.wav')
```

- Now, It Is Time To Plot The Result And The Original Sound Waves We Use The Matplot Library And The Plot Method To Do So.
  We Open The File out After The Operation Are Finished Then Read All The Frames And The Frame Rate Make A Timeout List To Carry All The Samples Information Then Plot The Amplitude In The Y Axes And The Time In The X Axes.

```
obj = wave.open('.\\audio\\Modified.wav', 'rb')  # open
dataout = obj.readframes(-1)  # get all the frames in data out
dataout = np.frombuffer(dataout, "int16")  # set the data to a sumber of two byte form data out
sampleRateOut = obj.getframerate()  # frame rate HZ (number of frames to be reads in seconds)
Timeout = np.linspace(0, len(dataout) / sampleRateOut, len(dataout))  # time of the output
plott(Timeout, dataout, wvFrMod)
```

- Finally, We Close Both Input And Output Files.

Convolution And Transfer Function

The Program Depends On Convolution Method In Signal Library As We Can See In The Next Figure. It First Generated A Sin Or Rectangular Signals As The User Chose Then Convolute It With.

Finally, It Plot All Of That Signals Under Each Other To Make It Easy For The User To

```
if convVal == 'Sine Wave':
    win = signal.windows.hann(50)
    plottCon(win, ModSginalFr, 'Impulse Response')
    filtered = signal.convolve(sig, win, mode='same') / sum(win)
    plottCon(filtered, ConvSginalFr, 'Filtered Signal')
else:
    win = np.repeat([0., 1., 0.], 50)
    plottCon(win, ModSginalFr, 'Impulse Response')
    filtered = signal.convolve(sig, win, mode='same') / sum(win)
    plottCon(filtered, ConvSginalFr, 'Filtered Signal')
```

Notice The Operations That Happen Clearly.

Text To Speech Section

Here We Used The Pyttsx3 Library To Generate A Sound Wave To The Text The User Input, We Can Change The Rate Or The Gender Of The Speaker The Amplitude Or Nearly Any Thing With Simple Function Included In The Library.

```
def tts(speach):
    engine = pyttsx3.init()  # object creation
    """ RATE"""
    rate = engine.getProperty('rate')  # getting details of current speaking rate
    engine.setProperty('rate', 220)  # setting up new voice rate
    """VOLUME"""
    volume = engine.getProperty('volume')  # getting to know current volume level (min=0 and max=1)
    engine.setProperty('volume', 1.0)  # setting up volume level  between 0 and 1
    """VOICE"""
    voices = engine.getProperty('voices')  # getting details of current voice
    # engine.setProperty('voice', voices[0].id)  #changing index, changes voices. o for male
    engine.setProperty('voice', voices[1].id)  # changing index, changes voices. 1 for female
```