GUI for JOY HUB

END-TERM REPORT

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By:

S.no.	***Name***	***Roll No.***	***Registration no.***
1.	*Akshat Mishra*	*04*	*11912412*
2.	*Arshi*	*08*	*11911141*
3.	*Mridul jain*	*09*	*11911255*

Courses Code: INT213

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

Objective

The primary objective of this project is to implement what we've learnt throughout this wonderful journey of python learning. Through this project we wanted to showcase the various aspects of python programing language . We tried to create a beautiful interface using python3 which isn't only focussed on the coding but also displays different fields in which this programing language is preffered like game designing , media field(audio player in this project) etc.

Introduction

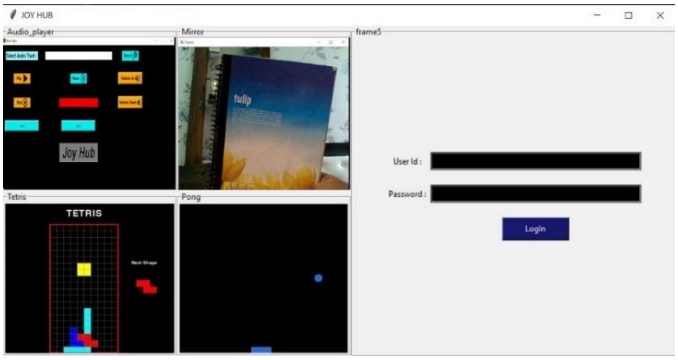
The advancement in the todays lifestyle has made our lives extremely busy and hence increased the mental tension which has made today's generation prone to different kind of mind related issues. Therefore, it becomes important to create a wonderous, productive and playful environment and for that we need to take small breaks and entertain ourselves, that's where JOY HUB come into play. A JOY HUB is a place where anyone can go and refresh there minds with various kinds of activity like playing games, listen to their favorite audio, and if required they could take some pictures etc. This project is no exception, it has been coded in python and comes with a graphical user interface to facilitate the users. This project has 2 main GUI's and 4 modules which includes:-

1) AUDIO PLAYER
2) MIRROR
3) TETRIS
4) PONG GAME

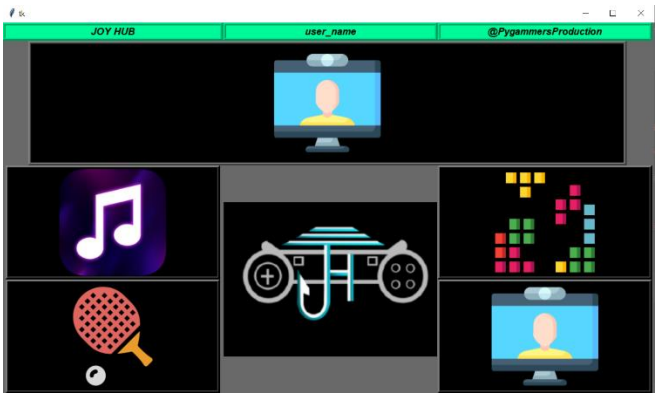
This project also includes SQL database connectivity that helps the use to store their report and then fetch it later. The detailed functional report of each module is given in the later sections of the report.

GUI Screenshots:

1. **GUI (i/ login page)**

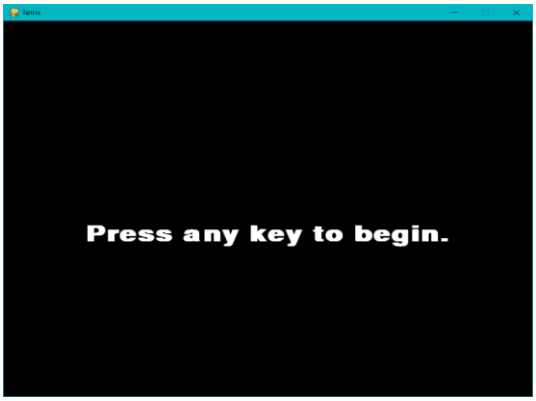


2. **GUI(ii/ Main page)**

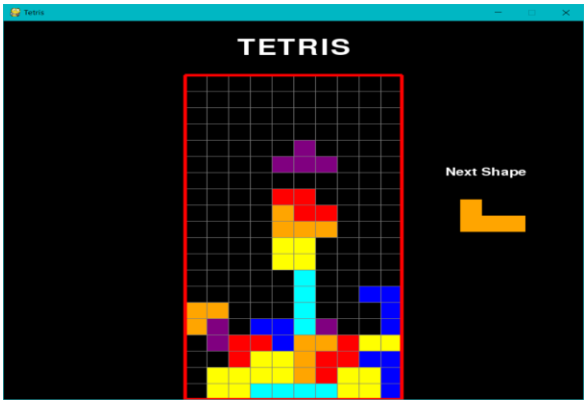


3. **TETRIS**

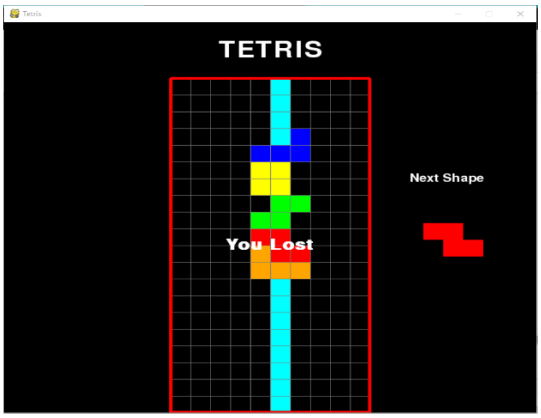
// opening page of the game



//main game window

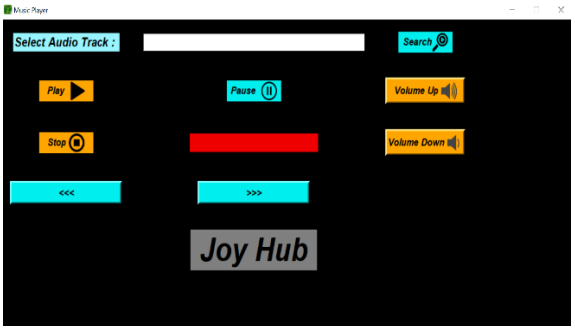


//lost the game

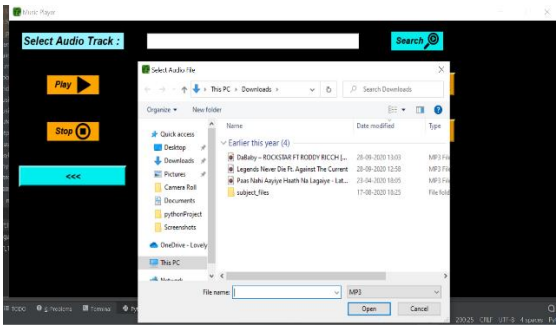


4. AUDIO PLAYER

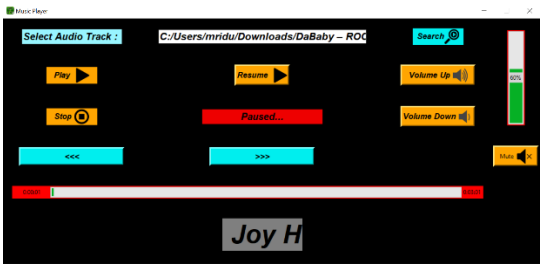
//main screen



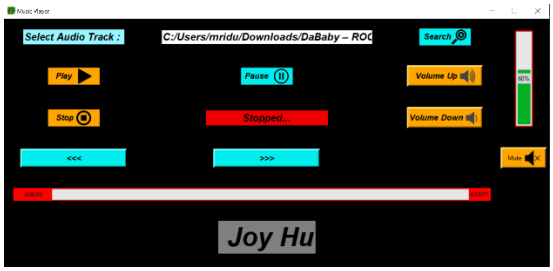
// Audio selection



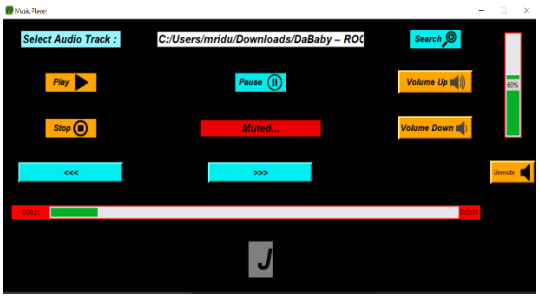
// audio paused



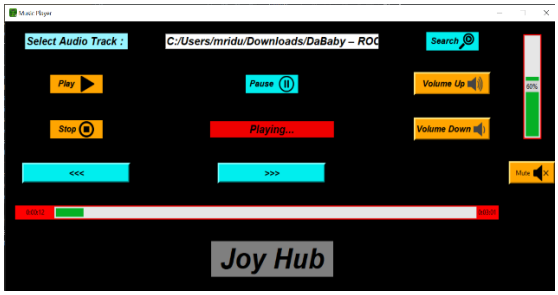
//Audio stopped



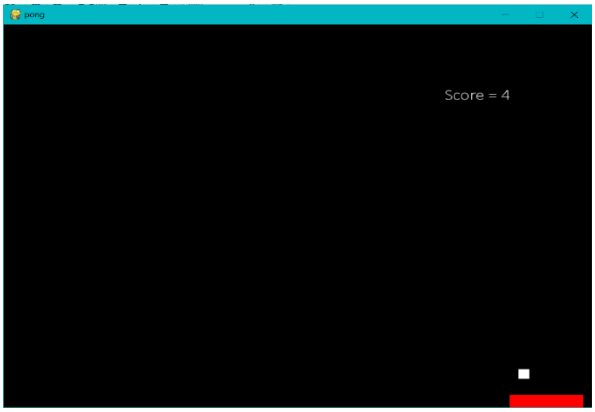
//audio muted



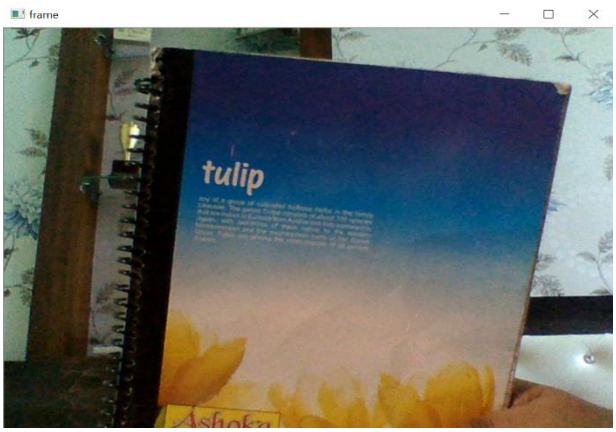
//Audio playing



5. PONG GAME



6. MIRROR



Function wise description

1. TETRIS

a. **draw_window() :- this function creates the window for the game and the grids showing in it.**
b. **draw_next_shape() :- this creates the next random object that falls from the center of the grid .**
c. **clear_rows() :- this will check whether there is a need for clear the row(i.e. row is fully filled) and then empty that row**
d. **draw_grid():- creates the grid the game game window**
e. **Piece() :- creates individual pieces**
f. **check_lost() :- checks whether you have lost (i.e. any column is fully filled)**
g. **valid_space() :- checks spaces valid for the placement of the next shape**
h. **convert_shape_format() :- it converts the shape formats into playable game shapes.**

2. PONG

a. **drawrect() :- this function creates the hitting rectangular platform**
b. **drawball() :- function to create the moving ball**

3. AUDIO PLAYER

a. **IntroLabel() :- This function is used for making slider word animation of "Joy Hub" text. In which string is divided into char and project one after another using loop**
b. **Createlabel() :- This function is used for making and accessing button and label . Required for audio player.**
c. **Mixer.init() :- This function is used for initializing the instructor of pygame mixer file. To control audio.**
d. **Musicurl() :- This function is used for accessing the MP.4,WAV,wav,mp.4 files. For the Rom of devices.**
e. **Progressbarmusictick() :- this function is used to give a track of music length. And this is only visible after Playmusic() function is call**
f. **Playmusic() : - this function is used to play the audio. And make progressbar , mute/unmute , volumebar, visible.**
g. **Paused() :- this function is used for pausing audio.**
h. **Resumed () :- this function is used to resume audio from paused. And paused is also likened to it.**
i. **volumeDown ():- used to decrease volume. IF(volume>25) decrease by 10 unit otherwise by 2 unit**
j. **volumeup ():- used to increase volume. IF(volume<75) increase by 10 unit otherwise by 2 unit**
k. **mute() :- This function is used to mute the audio.**
l. **unmute() :- This function is used to unmute the audio.And mute is also linked with it as same button to access.**
m. **Stop():- this function is used to stop the audio**
n. **Lengthup():- this function is used to forward the music tie by 10 unit.**
o. **Lengthdown():- this function is used to backward the music tie by 10 unit.**
p. **Library important:- mutagen.mp3 to access mp3 files, Pygame mixer file to control audio.tkinter.ttk progressbar to make progress bar**
q. **IntroLabel() :- This function is used for making slider word animation of "Joy Hub" text. In which string is divided into char and project one after another using loop**

r. **Createlabel() :- This function is used for making and accessing button and label . Required for audio player.**
s. **Mixer.init() :- This function is used for initializing the instructor of pygame mixer file. To control audio.**
t. **Musicurl() :- This function is used for accessing the MP.4,WAV,wav,mp.4 files. For the Rom of devices.**
u. **Progressbarmusictick() :- this function is used to give a track of music length. And this is only visible after Playmusic() function is call**
v. **Playmusic() : - this function is used to play the audio. And make progressbar , mute/unmute , volumebar, visible.**
w. **Paused() :- this function is used for pausing audio.**
x. **Resumed () :- this function is used to resume audio from paused. And paused is also likened to it.**
y. **volumeDown ():- used to decrease volume. IF(volume>25) decrease by 10 unit otherwise by 2 unit**
z. **volumeup ():- used to increase volume. IF(volume<75) increase by 10 unit otherwise by 2 unit**
aa. **mute() :- This function is used to mute the audio.**
bb. **unmute() :- This function is used to unmute the audio.And mute is also linked with it as same button to access.**
cc. **Stop():- this function is used to stop the audio**
dd. **Lengthup():- this function is used to forward the music tie by 10 unit.**
ee. **Lengthdown():- this function is used to backward the music tie by 10 unit.**
ff. **Library important:- mutagen.mp3 to access mp3 files, Pygame mixer file to control audio.tkinter.ttk progressbar to make progress bar**

4. MIRROR

a. **release() :- this function releases the picture that has been taken**
b. **imshow() :- function to show the created frame**
c. **Library :- Cv2 to access web came and video playing**
d. **Cv2.VideoCapture : - to Capture the video**
e. **Vid.read :- To read the web came files to display**
f. **Cv2.waitkey() :- to make video access to control at each millisecond**
g. **Cv2.destroyAllWindow() : -to end the window which shoeing web came video.**

Source Code

//GUI mainpage(i)

```
From  second import *
from tkinter import *
from PIL import ImageTk,Image
root = Tk()
root.title("JOY HUB")
#root.iconbitmap("icon.jpg")
frame1 = LabelFrame(root, text="audio player")
frame2 = LabelFrame(root, text="mirror")
frame3 = LabelFrame(root, text="Tetris")
frame4 = LabelFrame(root, text="Pong")
frame5 = LabelFrame(root, text="frame5",padx=50,pady=169)
frame1.grid(row=0,column=0)
frame2.grid(row=0,column=1)
frame3.grid(row=1,column=0)
frame4.grid(row=1,column=1)
frame5.grid(row=0,column=2,rowspan=8,sticky=E+W)image_pong_ =
ImageTk.PhotoImage(Image.open("C:/Users/ARSHI/PycharmProjects/pygames/pong_game1.jpg"))
image_tetrus_= ImageTk.PhotoImage(Image.open("C:/Users/ARSHI/PycharmProjects/pygames/tetris_game.JPG"))
image_music_ = ImageTk.PhotoImage(Image.open("C:/Users/ARSHI/PycharmProjects/pygames/pong_icon.PNG"))
image_vedio_= ImageTk.PhotoImage(Image.open("C:/Users/ARSHI/PycharmProjects/pygames/pong_icon.PNG"))
login_id = Entry(frame5,width=50,bg="black", fg="white", borderwidth=5)
paswrd = Entry(frame5,width=50,bg="black",fg="red",borderwidth=5)
submit = Button(frame5,text="LogIn",padx=30,pady=5,bg="midnightblue",fg="white",command=allgame)
image_pong = Label(frame4,image= image_pong_,width=250,height=220)
image_tetrus = Label(frame3,image= image_tetrus_,width=250,height=220)
image_music = Label(frame2,image= image_music_,width=250,height=220)
image_vedio = Label(frame1,image= image_vedio_,width=250,height=220)
image_music.grid(row=0,column=0)
image_pong.grid(row=0,column=1)
image_tetrus.grid(row=0,column=2)
image_vedio.grid(row=0,column=4)
user = Label(frame5,text="User Id : ")
user.grid(row=0,column=0)
password = Label(frame5, text = "Password : ")
password.grid(row=2,column=0)
space = Label(frame5, text= "      ")
space.grid(row=1,column=1)
login_id.grid(row=0,column=1)
paswrd.grid(row=2,column=1)
space1 = Label(frame5, text= "      ")
space1.grid(row=3,column=1)
submit.grid(row=4,column=1)
root,mainloop()
```

//GUI MAINPAGE(ii)[second.py]

```
def allgame():
  def audioplayer():
    stream = open("main.py")
    read_file = stream.read()
    exec(read_file)
  def pong_game_extra():
    stream = open("main.py")
    read_file = stream.read()
    exec(read_file)
  def tetris():
    stream = open("main.py")
    read_file = stream.read()
    exec(read_file)
  def mirror():
    stream = open("R.py")
    read_file = stream.read()
    exec(read_file)
  #from tkinter import filedialog
  from tkinter import *
  import tkinter as tk
  from itertools import cycle
  from PIL import ImageTk
  import time
  #from main import *
  class slider:
    def __init__(self,sp):
      self.sp=sp
      #self.sp.title("slider")
      #self.sp.geometry("1350x700+0+0")
      #==#
      self.image1=ImageTk.PhotoImage(file="musicpagef.png")
      self.image2 = ImageTk.PhotoImage(file="tetrisf.png")
      self.image3 = ImageTk.PhotoImage(file="pongf.png")
      self.image4 = ImageTk.PhotoImage(file="mirrorf.png")

      #la
      Frame_slider=Frame(self.sp)
      Frame_slider.grid(row=1, column=0,columnspan=3,rowspan=6)
      self.lbl1=Label(Frame_slider,image=self.image1,bg='black', width = 1090,height=200,relief=tk.RIDGE, bd=5)#,bg='black',
  width = 1000,height=200,relief=tk.RIDGE, bd=5
      self.lbl1.grid(row=1, column=0,columnspan=3,rowspan=6)
      self.lbl2 = Label(Frame_slider, image=self.image2,bg='black', width = 1090,height=200,relief=tk.RIDGE, bd=5)
      self.lbl2.grid(row=1, column=0,columnspan=3,rowspan=6)
      self.lbl3 = Label(Frame_slider, image=self.image3, bg='black', width=1090, height=200, relief=tk.RIDGE, bd=5)
      self.lbl3.grid(row=1, column=0, columnspan=3, rowspan=6)
      self.lbl4 = Label(Frame_slider, image=self.image4, bg='black', width=1090, height=200, relief=tk.RIDGE, bd=5)
      self.lbl4.grid(row=1, column=0, columnspan=3, rowspan=6)
      self.x=180
      self.slider_func()

    def slider_func(self):
```

```
        self.x-=1
        if self.x==0:
            self.x=180
            time.sleep(3)
            #swap
            self.new_im=self.image1
            self.image1=self.image2
            self.image2 = self.image3
            self.image3 = self.image4
            self.image4=self.new_im

            self.lbl1.config(image=self.image1)
            self.lbl2.config(image=self.image2)
            self.lbl3.config(image=self.image3)
            self.lbl4.config(image=self.image4)
        #self.lbl2.place(x=self.x,y=0)
        self.lbl2.after(40,self.slider_func)


sp = Tk()
sp.geometry('1100x625+0+0')
sp.resizable(False,False)
implay = PhotoImage(file='playf.png')
immusicplayer = PhotoImage(file='musicpagef.png')
immirror = PhotoImage(file='mirrorf.png')
imtetris = PhotoImage(file='tetrisf.png')
impong = PhotoImage(file='pongf.png')
imlogo = PhotoImage(file='logof.png')
sp.configure(bg='dimgray')
    #image button

logo = Label(sp, text='JOY HUB',bg='medium spring green', font=('arial', 13, 'italic bold'),relief=tk.RIDGE, width=36,
bd=10,borderwidth = 5)
logo.grid(row=0, column=0)
user = Label(sp, text='user_name',bg='medium spring green', font=('arial', 13, 'italic bold'), width=35,relief=tk.RIDGE, bd=5)
user.grid(row=0, column=1)
madeby = Label(sp, text='@PygammersProduction',bg='medium spring green', font=('arial', 13, 'italic bold'),
width=35,relief=tk.RIDGE, bd=5)
madeby.grid(row=0, column=2)


#slider2 = Label(sp,image=immirror, font=('arial', 13, 'italic bold'),bg='black', width = 1000,height=200,relief=tk.RIDGE, bd=5)
#slider2.grid(row=1, column=0,columnspan=3,rowspan=6)
obj=slider(sp)

Game1 = Button(sp, image = immusicplayer,text='Music Player', width=350,bg='black', activebackground='purple',
bd=5,command=musicplayer)
Game1.grid(row=7, column=0)
Game2 = Button(sp, image = imtetris, width=350,bg='black',  activebackground='purple', bd=5,command=tetris)
Game2.grid(row=7, column=2)
logoima = Label(sp,  image = imlogo, width=350,height=250,bg='black',   activebackground='purple', bd=5)
logoima.grid(row=7, column=1,rowspan=8,pady=50)
Game3 = Button(sp, image = impong, width=350,bg='black',   activebackground='purple', bd=5,command=pong)
```

```
    Game3.grid(row=9, column=0)
    Game4 = Button(sp, image = immirror,  width=350,bg='black', activebackground='purple', bd=5,command=mirror)
    Game4.grid(row=9, column=2)


    # get a series of gif images you have in the working folder
    # or use full path, or set directory to where the images are
    image_files = [
    'mirrorf.png',
    'pongf.png',
    'tetrisf.png',
    'musicpagef.png'
    ]

    sp.mainloop()
```

//audio player

```
def lengthdown():
    ml = mixer.music.get_pos() // 1000
    print(ml)
    mixer.music.set_pos(ml - 5.0)
    tl = mixer.music.get_pos() // 1000
    print(tl)
    CurrentSongLength = mixer.music.get_pos() // 1000

    progressbarmusicstartLabel.configure(text='{}'.format(str(datetime.timedelta(seconds=CurrentSongLength))))
    progressbarmusic['value'] = CurrentSongLength
def lengthup():
    ml = mixer.music.get_pos() // 1000
    print(ml)
    mixer.music.set_pos(ml + 5.0)
    tl = mixer.music.get_pos() // 1000
    print(tl)
    CurrentSongLength = mixer.music.get_pos() // 1000

    progressbarmusicstartLabel.configure(text='{}'.format(str(datetime.timedelta(seconds=CurrentSongLength))))
    progressbarmusic['value'] = CurrentSongLength

def unmute():
    global currentvol
    mp.unmutebutton.grid_remove()
    mp.mutebutton.grid()
    mixer.music.set_volume(currentvol)
    audiostatuslabel.configure(text='Playing...')
def mute():
    global currentvol
    mp.mutebutton.grid_remove()
    mp.unmutebutton.grid()
    currentvol = mixer.music.get_volume()
    mixer.music.set_volume(0)
    audiostatuslabel.configure(text='Muted...')
```

```
def resume():
    mp.ResumeButton.grid_remove()
    mp.PauseButton.grid()
    mixer.music.unpause()
    audiostatuslabel.configure(text='Playing...')
def stop():
    mixer.music.stop()
    audiostatuslabel.configure(text='Stopped...')


def volumeup():
    vol = mixer.music.get_volume()
    if(vol*100<=75):
        mixer.music.set_volume(vol+0.1)
    else:
        mixer.music.set_volume(vol + 0.01)
    ProgressbarVolumeLabel.configure(text='{}%'.format(int(mixer.music.get_volume()*100)))
    ProgressbarVolume['value'] = mixer.music.get_volume()*100

def volumedown():
    vol = mixer.music.get_volume()
    if (vol * 100 >= 25):
        mixer.music.set_volume(vol - 0.1)
    else:
        mixer.music.set_volume(vol - 0.01)
    ProgressbarVolumeLabel.configure(text='{}%'.format(int(mixer.music.get_volume() * 100)))
    ProgressbarVolume['value'] = mixer.music.get_volume() * 100
def pause():
    mixer.music.pause()
    mp.PauseButton.grid_remove()
    mp.ResumeButton.grid()
    audiostatuslabel.configure(text='Paused...')
def playmusic():
    ad = audiotrack.get()
    mixer.music.load(ad)
    progressbarLabel.grid()
    progressbarmusicLabel.grid()
    mp.mutebutton.grid()
    mixer.music.set_volume(0.6)
    mixer.music.play()
    audiostatuslabel.configure(text='Playing...')

    Song = MP3(ad)
    totalsonglength = int(Song.info.length)
    progressbarmusic['maximum'] = totalsonglength
    progressbarmusicendLabel.configure(text='{}'.format(str(datetime.timedelta(seconds=totalsonglength))))
    def progressbarmusictick():



        CurrentSongLength = mixer.music.get_pos() // 1000
        if(CurrentSongLength!=-1):
            progressbarmusic['value'] = CurrentSongLength
```

```
            progressbarmusicstartLabel.configure(text='{}'.format(str(datetime.timedelta(seconds=CurrentSongLength))))
            progressbarmusic.after(2, progressbarmusictick)
        else:
            CurrentSongLength = 0
            progressbarmusic['value'] = CurrentSongLength
            progressbarmusicstartLabel.configure(text='{}'.format(str(datetime.timedelta(seconds=CurrentSongLength))))
            progressbarmusic.after(2, progressbarmusictick)
    progressbarmusictick()

def musicurl():
    try:
        dd = filedialog.askopenfilename(initialdir='C:/Users/mridu/Downloads',
                                        title='Select Audio File',
                                        filetype=(('MP3','*.mp3'),('WAV','*.wav')))
    except:
        dd = filedialog.askopenfilename(title='Select Audio File',
                                        filetype=(('MP3','*.mp3'),('WAV','*.wav')))
    audiotrack.set(dd)


def createlabel():
    global
audiostatuslabel,ProgressbarVolumeLabel,ProgressbarVolume,progressbarLabel,progressbarmusicendLabel,progressbarmusic,
progressbarmusicstartLabel,progressbarmusicLabel
    global implay,imsearch,imstop,impause,immute,imunmute,imvolup,imvoldow
    #image button
    implay = PhotoImage(file='playf.png')
    imsearch = PhotoImage(file='search.png')
    imstop = PhotoImage(file='stopf.png')
    impause = PhotoImage(file='pausef.png')
    immute = PhotoImage(file='mutef.png')
    imunmute = PhotoImage(file='unmutef.png')
    imvolup = PhotoImage(file='volumeupf.png')
    imvoldow = PhotoImage(file='volumedownf.png')


    #size of button
    implay = implay.subsample(1,1)
    #lable
    TrackLabel = Label(mp,text='Select Audio Track : ', background='CadetBlue1',font=('arial',15,'italic bold'))
    TrackLabel.grid(row=0,column=0,padx=20,pady=20)

    audiostatuslabel = Label(mp,text='', background='red2',font=('arial',15,'italic bold'),width=20)
    audiostatuslabel.grid(row=2,column=1)
    #entry box
    Tracklableentry = Entry(mp,font=('arial',16,'italic bold'),width=35,textvariable=audiotrack)
    Tracklableentry.grid(row=0,column=1,padx=20,pady=20)
    #button
    #searchbutton
    BrowseButton= Button(mp,text='Search',bg='cyan2',font=('arial',13,'italic bold'),width=100,bd=0,
                activebackground='purple4',command=musicurl,image=imsearch,compound=RIGHT)
    BrowseButton.grid(row=0,column=2,padx=20,pady=20)
    #playbutton
```

```
    mp.PlayButton= Button(mp,text='Play  ',  bg='orange', font=('arial', 13, 'italic bold'), width=100, bd=0,
                      activebackground='purple4',command=playmusic,image =implay,compound=RIGHT)
    mp.PlayButton.grid(row=1, column=0, padx=20, pady=20)
    #pausebutton
    mp.PauseButton = Button(mp, text='Pause  ', bg='cyan2', font=('arial', 13, 'italic bold'), width=100, bd=0,
                    activebackground='purple4',command=pause,image =impause,compound=RIGHT)
    mp.PauseButton.grid(row=1, column=1, padx=20, pady=20)
    # resumebutton
    mp.ResumeButton = Button(mp, text='Resume  ', bg='orange', font=('arial', 13, 'italic bold'), width=100, bd=5,
                     activebackground='purple4', command=resume,image =implay,compound=RIGHT)
    mp.ResumeButton.grid(row=1, column=1, padx=20, pady=20)
    mp.ResumeButton.grid_remove()
    #mutebutton
    mp.mutebutton=Button(mp,text='Mute  ',width=80,bg='orange',activebackground='purple',bd=5,command=mute,image
=immute,compound=RIGHT)
    mp.mutebutton.grid(row=3,column=3)
    mp.mutebutton.grid_remove()
    # unmutebutton
    mp.unmutebutton = Button(mp, text='Unmute  ', width=80, bg='orange', activebackground='purple',
bd=5,command=unmute,image =imunmute,compound=RIGHT)
    mp.unmutebutton.grid(row=3, column=3)
    mp.unmutebutton.grid_remove()
    #volumeup
    VolumeupButton = Button(mp, text='Volume Up ', bg='orange', font=('arial', 13, 'italic bold'), width=140, bd=5,
                    activebackground='purple4',command=volumeup,image =imvolup,compound=RIGHT)
    VolumeupButton.grid(row=1, column=2, padx=20, pady=20)
    # stopbutton
    StopButton = Button(mp, text='Stop ', bg='orange', font=('arial', 13, 'italic bold'), width=100, bd=0,
                    activebackground='purple4',command=stop,image =imstop,compound=RIGHT)
    StopButton.grid(row=2, column=0, padx=20, pady=20)
    # volumedown
    VolumeDownButton = Button(mp, text='Volume Down ', bg='orange', font=('arial', 13, 'italic bold'), width=140, bd=5,
                      activebackground='purple4',command=volumedown,image =imvoldow,compound=RIGHT)
    VolumeDownButton.grid(row=2, column=2, padx=20, pady=20)
#progressbar volume
    progressbarLabel = Label(mp,text='',bg='red')
    progressbarLabel.grid(row=0,column=3,rowspan=3,padx=20,pady=20)
    progressbarLabel.grid_remove()
    ProgressbarVolume = Progressbar(progressbarLabel,orient=VERTICAL,mode='determinate',
                          value=60,length=190)
    ProgressbarVolume.grid(row=0,column=0,ipadx=5)

    ProgressbarVolumeLabel = Label(progressbarLabel,text='60%',bg='lightgray',width=3)
    ProgressbarVolumeLabel.grid(row=0,column=0)
#Progresbarmusic
    progressbarmusicLabel =Label(mp,text='',bg='red')
    progressbarmusicLabel.grid(row=4,column=0,columnspan=3,padx=20,pady=20)
    progressbarmusicLabel.grid_remove()

    progressbarmusicstartLabel = Label(progressbarmusicLabel, text='0.00.0', bg='red',width=10)
    progressbarmusicstartLabel.grid(row=0, column=0)

    progressbarmusic = Progressbar(progressbarmusicLabel,orient=HORIZONTAL,mode='determinate',value=0)
```

```
    progressbarmusic.grid(row=0,column=1,ipadx=370)


    progressbarmusicendLabel = Label(progressbarmusicLabel, text='0.00.0', bg='red')
    progressbarmusicendLabel.grid(row=0, column=2)
#musiclengthincresing
    musiclengthinc = Button(mp, text='>>>', bg='cyan2', font=('arial', 13, 'italic bold'), width=20, bd=5,
                            activebackground='purple4', command=lengthup)
    musiclengthinc.grid(row=3, column=1, padx=20, pady=20)
#musiclengthdecresing
    musiclengthdec = Button(mp, text='<<<', bg='cyan2', font=('arial', 13, 'italic bold'), width=20, bd=5,
                            activebackground='purple4', command=lengthdown)
    musiclengthdec.grid(row=3, column=0, padx=20, pady=20)
#
from tkinter import *
#from PIL import Image,ImageTk
from tkinter import filedialog
from pygame import mixer
from tkinter.ttk import Progressbar
import datetime
from mutagen.mp3 import MP3
mp = Tk()
mp.geometry('1100x500+100+100')
mp.title('Music Player')
mp.iconbitmap('music.ico')
mp.resizable(False,False)
mp.configure(bg='black')
#
c=0
audiotrack = StringVar()
currentvol = 0
totalsonglength =0
ss = ' Joy Hub '
count = 0

#slider

text = ''
SliderLabel= Label(mp,text=ss,bg='gray50',font=('arial', 40, 'italic bold'))
SliderLabel.grid(row=6,column=1,padx=20,pady=20)
def IntroLabel():
    global count,text
    if(count>=len(ss)):
        count = -1
        text = ''
        SliderLabel.configure(text=text)
    else:
        text = text+ss[count]
        SliderLabel.configure(text=text)
    count += 1
    SliderLabel.after(200,IntroLabel)
IntroLabel()
mixer.init()
```

```
createlabel()
mp.mainloop()
```

//PONG GAME[pong_game_extra]

```
import pygame
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)

pygame.init()

# Initializing the display window
size = (800, 600)
screen = pygame.display.set_mode(size)
pygame.display.set_caption("pong")

# Starting coordinates of the paddle
rect_x = 400
rect_y = 580

# initial speed of the paddle
rect_change_x = 0
rect_change_y = 0

# initial position of the ball
ball_x = 50
ball_y = 50

# speed of the ball
ball_change_x = 5
ball_change_y = 5

score = 0


# draws the paddle. Also restricts its movement between the edges
# of the window.
def drawrect(screen, x, y):
    if x <= 0:
        x = 0
    if x >= 699:
        x = 699
    pygame.draw.rect(screen, RED, [x, y, 100, 20])


'''def drawball(screen,x,y):
    if x<0:
        x=0
        ball_change_x = ball_change_x * -1
    elif x>785:
        x=785
```

```
        ball_change_x = ball_change_x * -1
    elif y<0:
        y=0
        ball_change_y = ball_change_y * -1
    elif x>rect_x and x<rect_x+100 and y==565:
        ball_change_y = ball_change_y * -1
    elif y>600:
        ball_change_y = ball_change_y * -1
    pygame.draw.rect(screen,WHITE,[x,y,15,15])'''

# game's main loop
done = False
clock = pygame.time.Clock()
while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                rect_change_x = -6
            elif event.key == pygame.K_RIGHT:
                rect_change_x = 6
            # elif event.key == pygame.K_UP:
            # rect_change_y = -6
            # elif event.key == pygame.K_DOWN:
            # rect_change_y = 6'''
        elif event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT or event.key == pygame.K_RIGHT:
                rect_change_x = 0
            elif event.key == pygame.K_UP or event.key == pygame.K_DOWN:
                rect_change_y = 0
    screen.fill(BLACK)
    rect_x += rect_change_x
    rect_y += rect_change_y

    ball_x += ball_change_x
    ball_y += ball_change_y

    # this handles the movement of the ball.
    if ball_x < 0:
        ball_x = 0
        ball_change_x = ball_change_x * -1
    elif ball_x > 785:
        ball_x = 785
        ball_change_x = ball_change_x * -1
    elif ball_y < 0:
        ball_y = 0
        ball_change_y = ball_change_y * -1
    elif ball_x > rect_x and ball_x < rect_x + 100 and ball_y == 565:
        ball_change_y = ball_change_y * -1
        score = score + 1
    elif ball_y > 600:
        ball_change_y = ball_change_y * -1
```

```
        score = 0
    pygame.draw.rect(screen, WHITE, [ball_x, ball_y, 15, 15])

    # drawball(screen,ball_x,ball_y)
    drawrect(screen, rect_x, rect_y)

    # score board
    font = pygame.font.SysFont('Calibri', 24, False, False)
    text = font.render("Score = " + str(score), True, WHITE)
    screen.blit(text, [600, 100])

    pygame.display.flip()
    clock.tick(60)

pygame.quit()
```

//TETRIS GAME

```
import pygame
import random

"""
10 x 20 square grid
shapes: S, Z, I, O, J, L, T
represented in order by 0 - 6
"""

pygame.font.init()

# GLOBALS VARS
s_width = 800
s_height = 700
play_width = 300  # meaning 300 // 10 = 30 width per block
play_height = 600  # meaning 600 // 20 = 20 height per blo ck
block_size = 30

top_left_x = (s_width - play_width) // 2
top_left_y = s_height - play_height


# SHAPE FORMATS

S = [['.....',
      '.....',
      '..00.',
      '.00..',
      '.....'],
     ['.....',
      '..0..',
      '..00.',
      '...0.',
      '.....']]
```

```
Z = [['.....',
      '.....',
      '.00..',
      '..00.',
      '.....'],
     ['.....',
      '..0..',
      '.00..',
      '.0...',
      '.....']]

I = [['..0..',
      '..0..',
      '..0..',
      '..0..',
      '.....'],
     ['.....',
      '0000.',
      '.....',
      '.....',
      '.....']]

O = [['.....',
      '.....',
      '.00..',
      '.00..',
      '.....']]

J = [['.....',
      '.0...',
      '.000.',
      '.....',
      '.....'],
     ['.....',
      '..00.',
      '..0..',
      '..0..',
      '.....'],
     ['.....',
      '.....',
      '.000.',
      '...0.',
      '.....'],
     ['.....',
      '..0..',
      '..0..',
      '.00..',
      '.....']]

L = [['.....',
      '...0.',
      '.000.',
      '.....',
      '.....'],
     ['.....',
      '..0..',
      '..0..',
      '..00.',
```

```
      '.....'],
     ['.....',
      '.....',
      '.000.',
      '.0...',
      '.....'],
     ['.....',
      '.00..',
      '..0..',
      '..0..',
      '.....']]

T = [['.....',
      '..0..',
      '.000.',
      '.....',
      '.....'],
     ['.....',
      '..0..',
      '..00.',
      '..0..',
      '.....'],
     ['.....',
      '.....',
      '.000.',
      '..0..',
      '.....'],
     ['.....',
      '..0..',
      '.00..',
      '..0..',
      '.....']]

shapes = [S, Z, I, O, J, L, T]
shape_colors = [(0, 255, 0), (255, 0, 0), (0, 255, 255), (255, 255, 0), (255, 165, 0), (0, 0, 255), (128, 0, 128)]
# index 0 - 6 represent shape


class Piece(object):
    rows = 20  # y
    columns = 10  # x

    def __init__(self, column, row, shape):
        self.x = column
        self.y = row
        self.shape = shape
        self.color = shape_colors[shapes.index(shape)]
        self.rotation = 0  # number from 0-3


def create_grid(locked_positions={}):
    grid = [[(0,0,0) for x in range(10)] for x in range(20)]

    for i in range(len(grid)):
        for j in range(len(grid[i])):
            if (j,i) in locked_positions:
                c = locked_positions[(j,i)]
                grid[i][j] = c
```

```
    return grid


def convert_shape_format(shape):
    positions = []
    format = shape.shape[shape.rotation % len(shape.shape)]

    for i, line in enumerate(format):
        row = list(line)
        for j, column in enumerate(row):
            if column == '0':
                positions.append((shape.x + j, shape.y + i))

    for i, pos in enumerate(positions):
        positions[i] = (pos[0] - 2, pos[1] - 4)

    return positions


def valid_space(shape, grid):
    accepted_positions = [[(j, i) for j in range(10) if grid[i][j] == (0,0,0)] for i in range(20)]
    accepted_positions = [j for sub in accepted_positions for j in sub]
    formatted = convert_shape_format(shape)

    for pos in formatted:
        if pos not in accepted_positions:
            if pos[1] > -1:
                return False

    return True


def check_lost(positions):
    for pos in positions:
        x, y = pos
        if y < 1:
            return True
    return False


def get_shape():
    global shapes, shape_colors

    return Piece(5, 0, random.choice(shapes))


def draw_text_middle(text, size, color, surface):
    font = pygame.font.SysFont('comicsans', size, bold=True)
    label = font.render(text, 1, color)

    surface.blit(label, (top_left_x + play_width/2 - (label.get_width() / 2), top_left_y + play_height/2 - label.get_height()/2))


def draw_grid(surface, row, col):
    sx = top_left_x
    sy = top_left_y
    for i in range(row):
        pygame.draw.line(surface, (128,128,128), (sx, sy+ i*30), (sx + play_width, sy + i * 30))  # horizontal lines
```

```
        for j in range(col):
            pygame.draw.line(surface, (128,128,128), (sx + j * 30, sy), (sx + j * 30, sy + play_height))  # vertical lines


def clear_rows(grid, locked):
    # need to see if row is clear the shift every other row above down one

    inc = 0
    for i in range(len(grid)-1,-1,-1):
        row = grid[i]
        if (0, 0, 0) not in row:
            inc += 1
            # add positions to remove from locked
            ind = i
            for j in range(len(row)):
                try:
                    del locked[(j, i)]
                except:
                    continue
    if inc > 0:
        for key in sorted(list(locked), key=lambda x: x[1])[::-1]:
            x, y = key
            if y < ind:
                newKey = (x, y + inc)
                locked[newKey] = locked.pop(key)


def draw_next_shape(shape, surface):
    font = pygame.font.SysFont('comicsans', 30)
    label = font.render('Next Shape', 1, (255,255,255))

    sx = top_left_x + play_width + 50
    sy = top_left_y + play_height/2 - 100
    format = shape.shape[shape.rotation % len(shape.shape)]

    for i, line in enumerate(format):
        row = list(line)
        for j, column in enumerate(row):
            if column == '0':
                pygame.draw.rect(surface, shape.color, (sx + j*30, sy + i*30, 30, 30), 0)

    surface.blit(label, (sx + 10, sy- 30))


def draw_window(surface):
    surface.fill((0,0,0))
    # Tetris Title
    font = pygame.font.SysFont('comicsans', 60)
    label = font.render('TETRIS', 1, (255,255,255))

    surface.blit(label, (top_left_x + play_width / 2 - (label.get_width() / 2), 30))

    for i in range(len(grid)):
        for j in range(len(grid[i])):
            pygame.draw.rect(surface, grid[i][j], (top_left_x + j* 30, top_left_y + i * 30, 30, 30), 0)

    # draw grid and border
    draw_grid(surface, 20, 10)
```

```
    pygame.draw.rect(surface, (255, 0, 0), (top_left_x, top_left_y, play_width, play_height), 5)
    # pygame.display.update()


def main():
    global grid

    locked_positions = {}  # (x,y):(255,0,0)
    grid = create_grid(locked_positions)

    change_piece = False
    run = True
    current_piece = get_shape()
    next_piece = get_shape()
    clock = pygame.time.Clock()
    fall_time = 0

    while run:
        fall_speed = 0.27

        grid = create_grid(locked_positions)
        fall_time += clock.get_rawtime()
        clock.tick()

        # PIECE FALLING CODE
        if fall_time/1000 >= fall_speed:
            fall_time = 0
            current_piece.y += 1
            if not (valid_space(current_piece, grid)) and current_piece.y > 0:
                current_piece.y -= 1
                change_piece = True

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
                pygame.display.quit()
                quit()

            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_LEFT:
                    current_piece.x -= 1
                    if not valid_space(current_piece, grid):
                        current_piece.x += 1

                elif event.key == pygame.K_RIGHT:
                    current_piece.x += 1
                    if not valid_space(current_piece, grid):
                        current_piece.x -= 1
                elif event.key == pygame.K_UP:
                    # rotate shape
                    current_piece.rotation = current_piece.rotation + 1 % len(current_piece.shape)
                    if not valid_space(current_piece, grid):
                        current_piece.rotation = current_piece.rotation - 1 % len(current_piece.shape)

                if event.key == pygame.K_DOWN:
                    # move shape down
                    current_piece.y += 1
                    if not valid_space(current_piece, grid):
```

```
                current_piece.y -= 1

            '''if event.key == pygame.K_SPACE:
              while valid_space(current_piece, grid):
                current_piece.y += 1
              current_piece.y -= 1
              print(convert_shape_format(current_piece))'''  # todo fix

        shape_pos = convert_shape_format(current_piece)

        # add piece to the grid for drawing
        for i in range(len(shape_pos)):
            x, y = shape_pos[i]
            if y > -1:
                grid[y][x] = current_piece.color

        # IF PIECE HIT GROUND
        if change_piece:
            for pos in shape_pos:
                p = (pos[0], pos[1])
                locked_positions[p] = current_piece.color
            current_piece = next_piece
            next_piece = get_shape()
            change_piece = False

            # call four times to check for multiple clear rows
            clear_rows(grid, locked_positions)

        draw_window(win)
        draw_next_shape(next_piece, win)
        pygame.display.update()

        # Check if user lost
        if check_lost(locked_positions):
            run = False

    draw_text_middle("You Lost", 40, (255,255,255), win)
    pygame.display.update()
    pygame.time.delay(2000)


def main_menu():
    run = True
    while run:
        win.fill((0,0,0))
        draw_text_middle('Press any key to begin.', 60, (255, 255, 255), win)
        pygame.display.update()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False

            if event.type == pygame.KEYDOWN:
                main()
    pygame.quit()


win = pygame.display.set_mode((s_width, s_height))
pygame.display.set_caption('Tetris')
```

```
main_menu()  # start game
```

//MIRROR

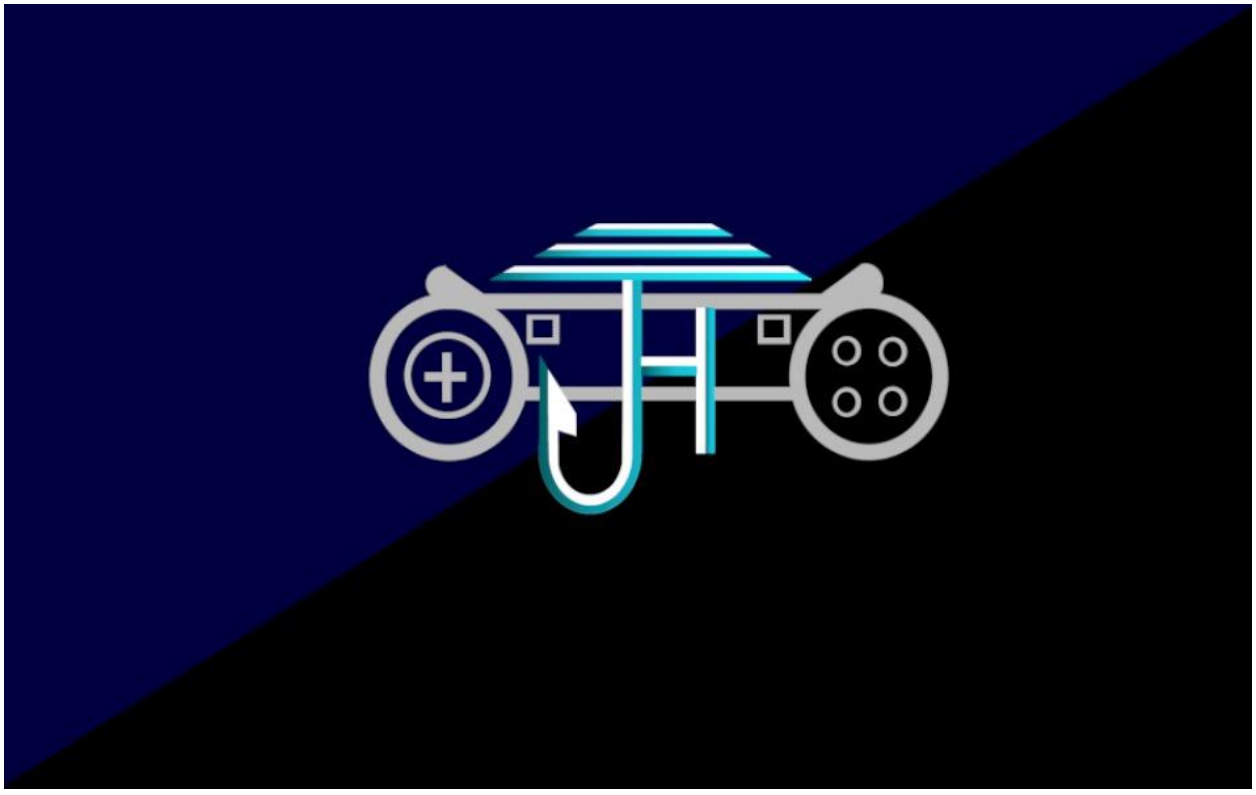
```
# import the opencv library
import cv2

# define a video capture object
vid = cv2.VideoCapture(0)

while (True):

    # Capture the video frame
    # by frame
    ret, frame = vid.read()

    # Display the resulting frame
    cv2.imshow('frame', frame)

    # the 'q' button is set as the
    # quitting button you may use any
    # desired button of your choice
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# After the loop release the cap object
vid.release()
# Destroy all the windows
cv2.destroyAllWindows()
```

Results

We finally got the end product as the 'JOY HUB' an interactive interface that is made to entertain the user with various different interfaces i.e. Tetris, Pong game, Audio player and mirror, which includes all the above mentioned modules. We learnt how to make a GUI using Tkinter in Python and also learnt to implement database connectivity using SQL . Appart from the tkinter we have also used pygame in this project to produce a interactive and attractive interface for the end user.

- ⇨ AUDIO PLAYER is preffered for the playing different types of audio
- ⇨ PONG GAME and TETRIS are created so that the user could entertain themselves for a duration of time .
- ⇨ MIRROR is designed for sharing the camera .

#LOGO



References

- https://www.python-course.eu/index.php
- www.blog.pythonlibrary.org
- www.anaconda.com
- www.stacksoverflow.com
- www.geeksforgeeks.com