EL-GY 6183 DIGITAL SIGNAL PROCESSING LAB

PROJECT REPORT

On

falalalalaaNG Christmas Box

A Python based game using Image, Audio and Video Processing

Submitted for grading towards the fulfilment of our degree of

MASTER OF SCIENCE

in

ELECTRICAL/COMPUTER ENGINEERING

By

VANSHIKA SACHDEV (vs2203)
KARTHIK SRIRAM (ks5189)

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NEW YORK UNIVERSITY

DECEMBER 2019

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Named after the acronym for the famed companies (FANG), and with the holiday season just around the corner(!), our project brings in the cheer and chill with a fun game, primarily using python, to share our joyous spirit! We hope it keeps our players engaged enough to leave the image and video processing, text to speech conversion, vision implementation, and audio embedding to us!

On a serious note, however, the program uses OpenCV and other python libraries such as pyttsx, pygame, and pyglet to provide an entertaining experience. The Tkinter GUI is an interactive pathway for the player/user to control the usage of the game.

CHAPTER 1
INTRODUCTION

The need for a break during the holidays to just relax and entertain ourselves as the weather starts getting cold, and the stress of taking final exams at school or meeting annual deadlines at work, was the inspiration behind the inception of the idea for this project.

Usage of modern artificial intelligence techniques, computer vision, speech processing, audio, image and video processing, have proven beneficial for the entertainment industry and this project only shows a small scope of the grand scale by which the future of media and gaming would aim to flourish.

Several hundreds to thousands of modern game algorithms have been developed across universities, gaming companies, AI-first industries, and start-ups using the above-mentioned concepts. The idea of hands-free, gesture recognition-based gaming has been at the forefront of the industry and has formed a trend since the success of gaming devices such as the Wii, Xbox Kinect and PlayStation.

The key role that image and video signal processing plays here is the part where the camera system of the gaming device/ computer uses what we have understood as frames to interpret what is available on the dataset such as parts of the hand, remote, etc. A lot of these frames are processed like how a signal processor would process an analog signal in parts. The frames are taken one at a time, iteratively, and matched to the dataset to recognize the image input. Noise in this case is background of the image which usually gets filtered through contouring or edge detection algorithms.

Audio processing comes into play when we want the game to have a background audio that has to say, change speed, or stop and start at different parts, or transform into different frequencies. This plays a key role in the experience a player/ user has while having fun playing the game. A good audio background keeps the players happy and engaged, and the rest is taken care of by the vision algorithms!

Python is the best choice, in our opinion, for a programming environment because of its flexibility and the amount of predefined available libraries that can be utilized or created for future modifications. In our DSP Lab class, the description of using Python has evolved from beginner to intermediate to semi-advanced, which we believe is thanks to the challenges and advancements that the lectures and labs provided.

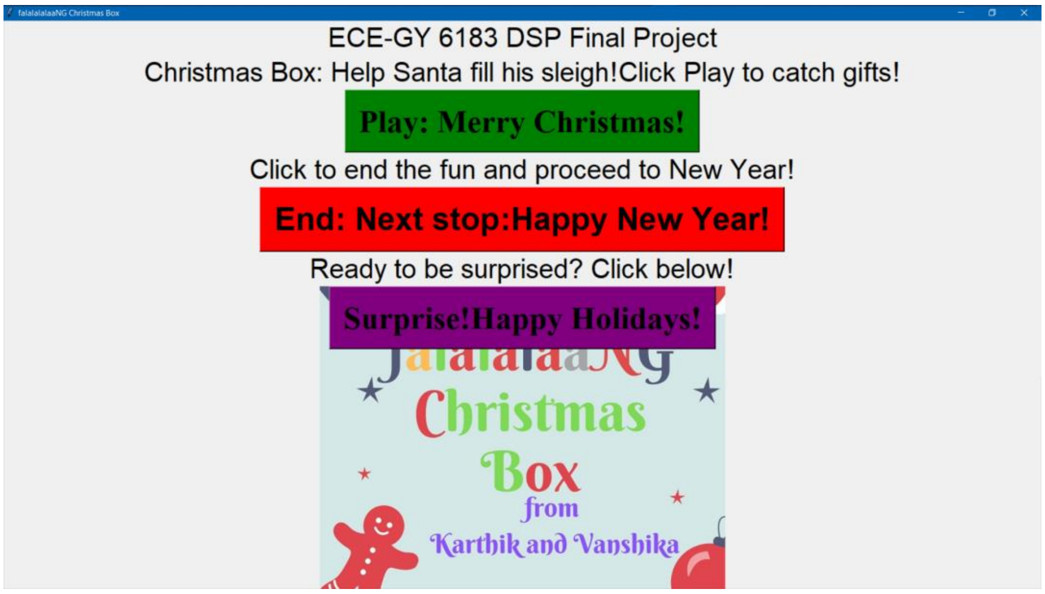
1.1 OVERVIEW OF OUR PROJECT AND ITS IMPLEMENTATION

The currently proposed project model incorporates the following concepts:

- Interactive GUI
- Image and Video processing using Computer vision
- Text to Speech processing
- Audio embedding and processing
- A SURPRISE!

1.1.1 Interactive GUI

The interactive GUI is responsible to do what it is clearly defined as- a graphical user interface- which provides a pathway between the controls required by the user and the graphical buttons which execute the necessary functions to perform the control action defined by the user.
The GUI uses a combination of text, buttons, and graphics (windows) to provide instructions or options that are available for the user to command with. In our project, we implemented this GUI by defining 3 buttons. The first button executes the 'Christmas' function in our python program, which is the function definition to play the game. The second button executes the 'NewYear' function, which is the function definition to exit the game. The third button which is labelled "Surprise" executes a surprise function defined in python which will be explained at the end of this report. Figure 1. shows the GUI implemented.



ECE-GY 6183 DSP Final Project

Christmas Box: Help Santa fill his sleigh!Click Play to catch gifts!

Click to end the fun and proceed to New Year!

Ready to be surprised? Click below!

Figure 1: Our Interactive GUI

1.1.2 Image and Video processing using Computer Vision

The core concept of our project is image and video processing. To implement this, we used the concept of computer vision which is the artificial intelligence algorithm that identifies parts of an image or video by matching it to an input dataset. Computer vision is primarily used in parallel to the concept of the human eye. The objective is to recognize and understand objects, people, parts, paths, etc.

In our project implementation we have used computer vision in a way whereas described in our introduction, we try to identify the part of our hand to match it to our inbuilt classifier/dataset. There are a lot of inbuilt algorithms that have been implemented as part of this image and video processing stage using computer vision such as DTCT and Gabor Wavelets. The inbuilt classifier is taken into the training stage using a local binary pattern. Figure 2. below shows the computer vision detection in our project.

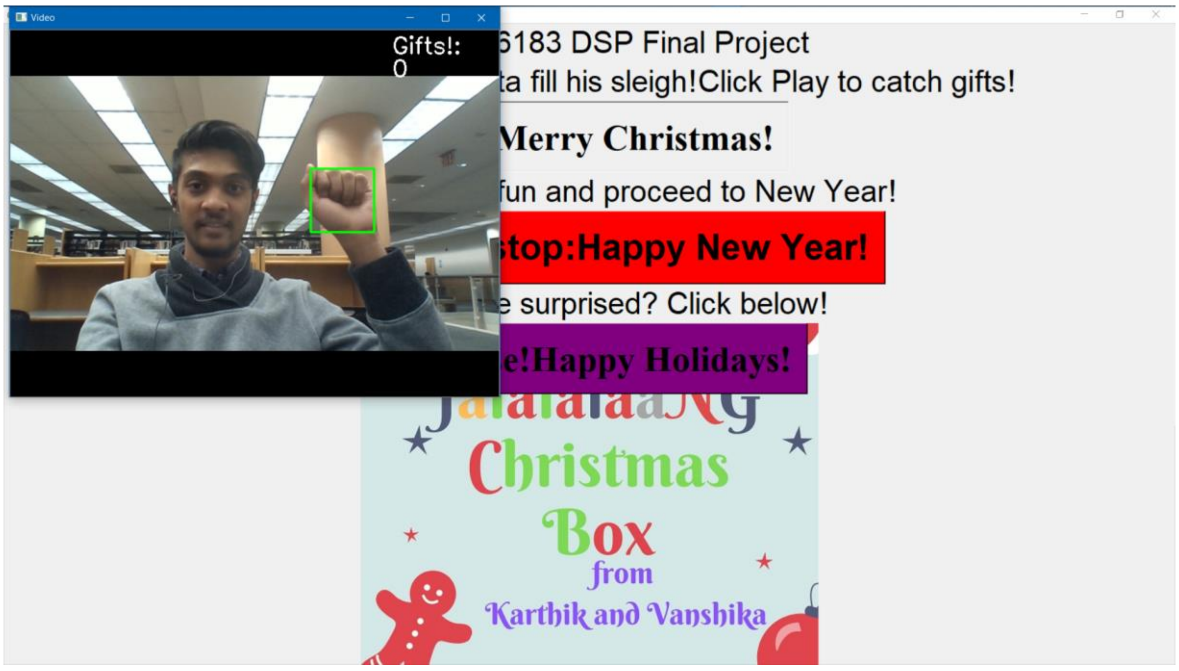


Figure 2. Computer Vision Detection

1.1.3 Text to Speech Processing

Another important concept that rules the world of AI is natural language processing. The whole idea of understanding what words are being used, in what context, and what the appropriate response would be, has strong alignment with the vision of major technology companies such as Facebook, Apple, Amazon, Netflix, Google etc. This has been part of the idea of consideration for us, because we believe that our project should be able to implement concepts that major companies are trying to establish in scale.

In our project idea, the original inception was to have a neural network identify speech input from the player, convert it to text, and provide feedback to play better. While this was not implemented, we successfully implemented the reverse, which is text to speech, where the number of gifts/items collected and other details that are available as text was said using the inbuilt python voice.

Figure 3 shows a block diagram for text to speech conversion. Once the text gets passed onto the analysis and past stage to the Linguistic block, we have 3 stages of processing there. We have phasing which converts the text into a phonetic context. Then we have the intonation which provides the required tone for the conversion. The last stage is the duration stage which determines the time parameters required for the speech signal. This gets passed to a waveform generator and then the speech wave gets synthesized.

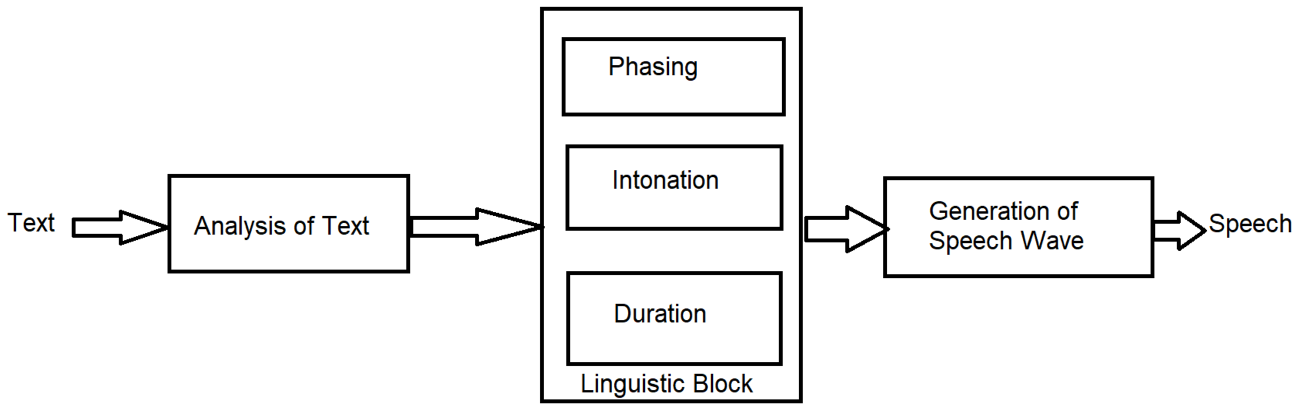


Figure 3. Text to Speech Block Diagram

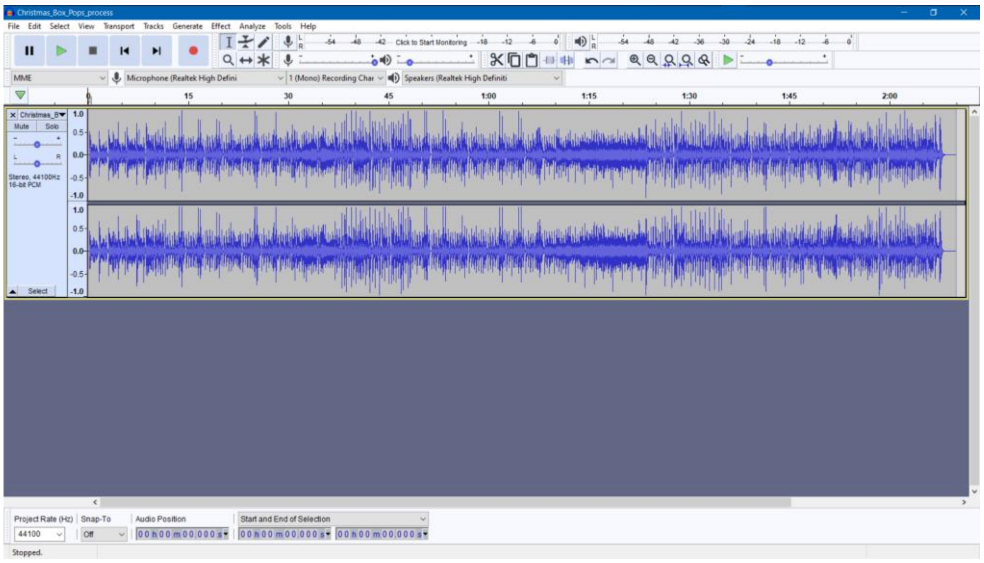
1.1.4 Audio Embedding and Processing

The Digital Signal Processing Lab has had a strong focus on audio processing techniques, and this formed the premise of using an audio file as background music in our python program to make the experience wholesome for the player/user. The general way of using audio in python is through the concept of the pyaudio library. However, in our case, since we focused on game development, we used pygame for audio which will be discussed later in this report.

The importance of audio is that in a game environment, the background score which undergoes changes in sound such as speed, frequency, and echo, convey different positions or emotions through the game. For example, a faster audio (higher tempo) would convey a change in speed in the game. Another example we have used is higher frequency, which conveys a greater challenge or a different level.

It has been understood in the gaming business that there is a direct correlation between the success of the game and the music that is played in the background. At scale, games like Mario by major companies like Nintendo have audio processing engineers do complex audio work on the music being embedded in their games.

The implementation of audio in our project was done by carefully analyzing a file and using the appropriate frequencies and tempo to appear at the correct level and speed of the game. We also used Audacity to perform a lot of analysis and format conversions. The wave file has the following representation in Audacity before conversion into mp3 and utilization.


Christmas_Box_Pops_process
File Edit Select View Transport Tracks Generate Effect Analyze Tools Help
Click to Start Monitoring
MME
Microphone (Realtek High Defini
1 (Mono) Recording Chan
Speakers (Realtek High Definiti
Christmas_B
Mute Solo
Select
Audio Position
Start and End of Selection
Stopped.

Stereo, 44100Hz 16-bit PCM

Project Rate (Hz) 44100

Snap-To Off

Figure 4. Audacity implementation of “Christmas_Box_Pops.wav”

1.1.5 SURPRISE!

You are at halfway…

Go to the end of the report to find out!

1.2 OUR PROPOSED GAME MODEL WORKFLOW AND FUNCTION

Every game or project needs a well-defined workflow and should be an upgrade by minimizing issues and bugs. Our proposed system model was designed by observing existing gesture-based games that used image and video processing and analyzing how they faced the at least two or more of the following challenges:

- Accuracy of detection.
- Latency
- Inadequate datasets and classifiers
- Complex algorithms
- Less interactive/ Less user-friendly
- Lack of end-to-end development. Usually presented only the skeletal output.
- Less sentimental and emotional value

To overcome the above challenges one after the other we defined the following solutions:

- Improve accuracy of detection by analyzing the video and images as individual frames and perform edge and boundary detection to filter out image noise. Utilize counter maps if necessary.
- Use algorithms with lower time and space complexity. If there is still a latency, pipeline it with outputs of previous stages. For example, while the main action gets executed, present outputs or print instructions.
- Usage of a haar classifier with additional number of samples in the dataset is preferred. Keeping the test input to a fist was recommended as it is easier to detect a fist than fingers.
- Instead of using one big program chunk that uses a complex, interlinked algorithm, we defined each operation as a separate algorithm and then interlinked them in the end as separate buttons on a GUI.
- This utilization of an interactive GUI makes it more user friendly for the player to choose what they want to do and when to do it. Increased degree of freedom=more fun.
- A full end-to-end game using image, audio and video processing, can tend to get complicated. However, the experience is enhanced when this challenge is

overcome. Therefore, our proposed system aims to bring in all key aspects of entertainment.

- Finally, we believe that anyone has the potential to program, do the calculations, understand the concepts, and implement them. It takes the rare few to make an experience have high sentimental and emotional value. Our proposed system wishes to connect the game to celebratory moments such as Christmas and New Year.

The block diagram of the modified proposed system is shown in Figure.5.

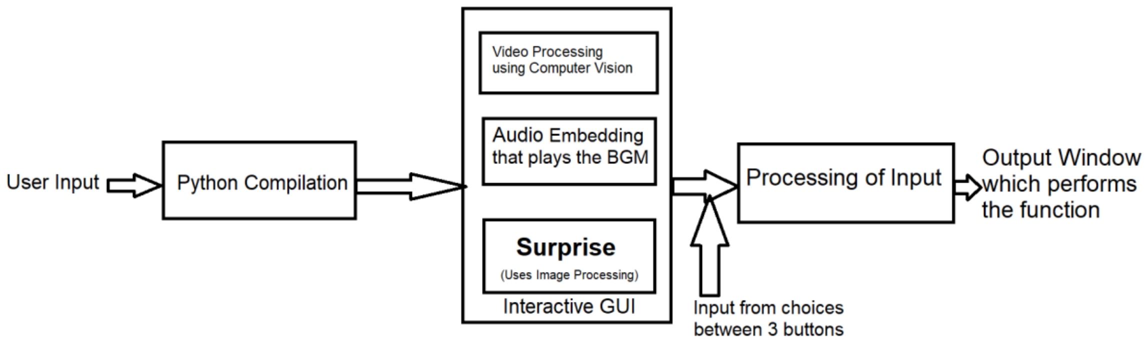

User Input
Python Compilation
Video Processing using Computer Vision
Audio Embedding that plays the BGM
Surprise
(Uses Image Processing)
Interactive GUI
Processing of Input
Input from choices between 3 buttons
Output Window which performs the function

Figure 5: Block Diagram

This block diagram describes how once the user is ready, they can compile the python program which will split the program at the GUI level into 3 different parts. Based on the input choice from the user again, the video and audio block come together for the game or the surprise block with image processing comes. An output window gets printed showing what corresponds to the desired user input.

CHAPTER 2

PYTHON LIBRARIES USED

2.1 OPENCV

The Computer Vision library called OpenCV is used in our project for its vast open source capability to perform vision tasks. The python import keyword is ‘cv2’ and our OpenCV version is 4.1.2. Since computer vision plays an important part in our project, OpenCV had great significance in performing functions such as video capture, frame by frame analysis, gesture recognition, matching, etc. OpenCV also plays a key role in making this project work in real-time due its capabilities of simultaneously acquiring and processing an image or video.

2.2 PYGAME

Another key python library used is pygame. This library is used in the integration of audio with video, and it proves resourceful in adding features during a python-based game development and execution stage. The import keyword is ‘pygame’ and our version is 1.9.6. In our project, pygame plays the role of initializing the audio file, mixing it, and playing/stopping whenever necessary.

2.3 PYTTSX

The python text-to-speech library is the library that has the functions necessary to carry out the text-to-speech processing explained. It has an engine that needs to be initialized, and the text to be spoken is given in “engine.say()”. Since tone and duration are important aspects as well, we need another function “engine.runAndWait()” that determines the duration of execution and the duration of delay between executions to make the speech sound more natural.

2.4 TIME

Time is a python library used to specify the latency or delays required for the input functions to work in an orderly manner. In our project we use a function under the library called ‘sleep’ to specify the number of time units we want between instructions or operations to execute. The purpose is to bring down the overall latency by using intermittent points at which time consumption can happen for parallel execution. This allows us to “pipeline” our design to execute all parts of our code and perform all the different processing techniques simultaneously. Therefore, this library plays an important role in defining the management and reliability of our real-time system.

2.5 SYS

In our python project interpreter, there are certain variables being used by functions that have established a direction connection to the interpreter strongly. To access these variables and functions, we use the sys (System-specific parameters and functions) library. The sys library also plays an important role when we require the program to exit the program when we click the “End” button in our GUI.

2.6 NUMPY

The numerical python library is one of the most used libraries for all number-based variables, arrays and functions. In our project, Numpy plays the role of performing the numerical operations on the variables defined such as the number of gifts, levels, etc.

2.7 MATH

The Math library is used for all mathematical functions to be carried out in python. The usage of the math library in our project was for time and speed calculation, latency calculation, and the applied mathematics involved in the image and video processing of the acquired frames.

2.8 TKINTER

TKinter is a python library that was discussed in detail in our Digital Signal Processing Lab. It is the library used to design and manage a graphical user interface (GUI) that involves steps to define the graphical appearance, provide instructions, and integrate the defined graphical elements such as buttons to the respective function that needs to be performed. In our project, Tkinter was the forefront of our design, as it is what we believe, is the heartbeat of the interaction between our players/users and our game. It provides the controls, and it defines the pathway between the user and the functionality provided by our system.

Tkinter has a 'root' definition which describes the GUI as a loop that keeps executing until a sufficient point of command has been provided by the user to end. This 'root' has the following function definitions used in our project:

- 'root.after()' in our check function to verify if a play/end/ or surprise function has been called.
- 'root.destroy()' to end the GUI (main loop) and exit back to the main program.
- 'root.mainloop()' to execute the GUI in the loop until a 'destroy' function is called.

Our GUI has a main heading, 3 instruction titles for 3 buttons. All of these were defined with the understanding of this library.

2.9 PYGLET and PIL

The Python imaging library (PIL) is a library used to acquire, modify, and store image files of different formats. From the formats of '.jpg' and '.png' for single image files, to our computer vision frame image formats to our "surprise", PIL plays an important role in the 3 mentioned image processing steps.

This concept of image format is the inspiration to what we took as a challenge to understand the working of images and processing them when they are of different formats. This is the idea which is the premise to what we are going to explain as our "surprise".

Pyglet was the library used to process our "surprise" file. It is a library used to perform visual effects and image development in python. This library is used in game development as well. In our project, under the "surprise" section, pyglet is used to take the visual file as input, get its parameters, process it, and print it.

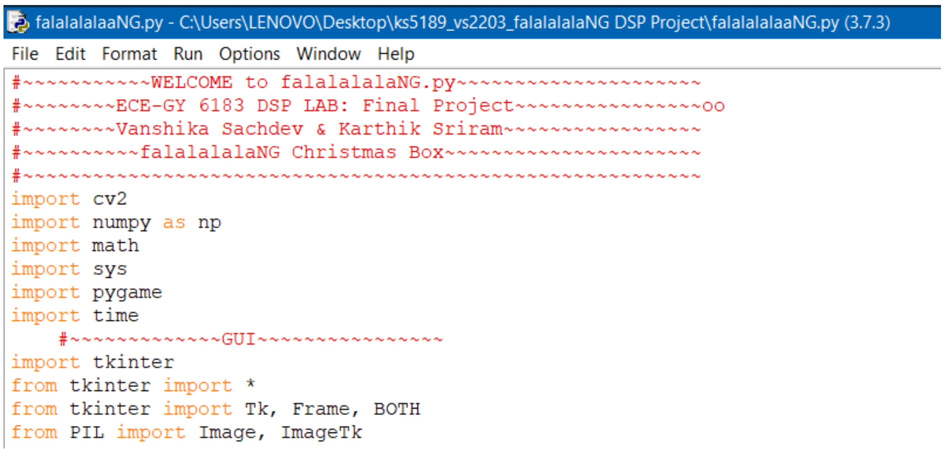
More on this in our surprise section of the report….

CHAPTER 3

WORKING PRINCIPLE AND SOFTWARE DESIGN FLOW

3.1 IMPORT OF LIBRARIES AND THEIR FUNCTIONS

The first step in our software development was to analyse the libraries meant to be imported and all the functions that each of those libraries provide to be defined. As mentioned in the previous section, every library needs to be installed, analysed, and utilized for successful program execution. Figure 6 below shows the top part of our code that has many of the main libraries defined and functions that are utilized by those libraries.


falalalalaaNG.py - C:\Users\LENOVO\Desktop\ks5189_vs2203_falalalalaNG DSP Project\falalalalaaNG.py (3.7.3)
File Edit Format Run Options Window Help

```
#~~~~~~~~~~~WELCOME to falalalalaNG.py~~~~~~~~~~~~~~~~~~~~~
#~~~~~~~~ECE-GY 6183 DSP LAB: Final Project~~~~~~~~~~~~~~~~oo
#~~~~~~~~Vanshika Sachdev & Karthik Sriram~~~~~~~~~~~~~~~~~
#~~~~~~~~~~falalalalaNG Christmas Box~~~~~~~~~~~~~~~~~~~~~~
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import cv2
import numpy as np
import math
import sys
import pygame
import time
    #~~~~~~~~~~~~~GUI~~~~~~~~~~~~~~~~
import tkinter
from tkinter import *
from tkinter import Tk, Frame, BOTH
from PIL import Image, ImageTk
```

Figure 6: Library Imports, Functions defined.

3.2 USER DEFINED FUNCTIONS

Our program invokes 3 functions and a “surprise” function defined by us, post assigning values to the variables defined for each of the functions that are inbuilt to the libraries imported. The three functions defined are:

- ‘Christmas()’ which is the main function to play the designed game.
- ‘NewYear()’ which is the function definition to end the execution of the game.
- ‘check()’ which is the function definition used to check if either ‘Christmas()’ or ‘NewYear()’ is called.

3.2.1 Christmas() function definition:

This function is the primary function definition of our project. The first part of this function has the text to speech conversion initialized using pyttsx3 and the second part is the embedding of our audio file using pygame. Then we use the default Cascade Haar Classifier in our file called “Game.xml” and we use “cv2.VideoCapture(0)” to acquire the video through the webcam.

We have initialized a set of variables such as the number of gifts, game level, victory level(current level), locking parameters for the rectangle, failure and loss acknowledgement, win acknowledgement, and the parameters that define the beginning and end of the loop.

The next step was to define the parameters of the font for the count of gifts, and to define margin and boundaries for the rectangle. Now, we may wonder what this rectangle is?

The most important part of our computer vision algorithm is the usage of “cv2.rectangle” which is fundamentally the definition of 2 rectangles, one in green and one in red, each to detect the part of the body in the dataset and the one to match it with respectively. We have four parameters for these rectangles, which are the x and y axis of their positions in the video, and the values of their height and width.

When the level has been changed to 1, the compiler provides instructions to display the part of the body. If the part of the body is detected, then the locking parameters latch themselves to the x and y values and the green rectangle stays around the detected part. This can be moved around the screen to match it with the red rectangle.

Iteratively, the program analyses the match and adds a value of 100 to the count of the number of gifts and uses “engine.say” to say the count value. This process repeats until a count of 800 and then the level gets incremented to 2, and then to 3 when the count is a value of 1400.

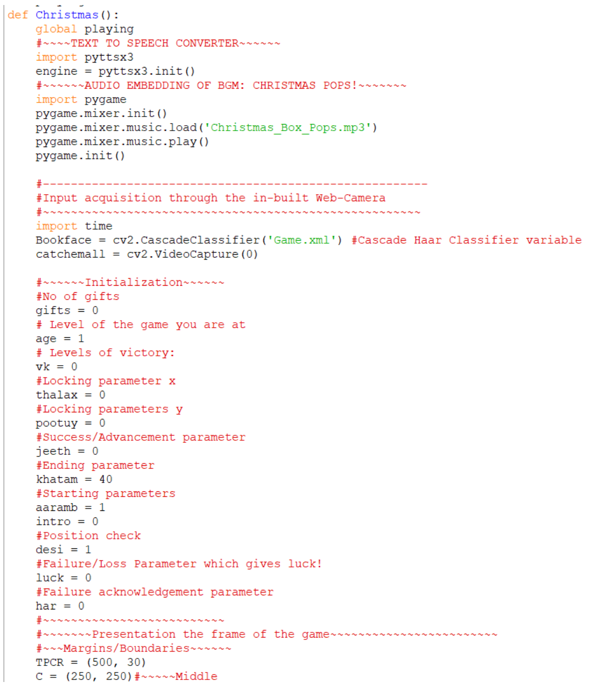
The conditional statements for example:

if x > 195 and x < 325 and y > 208 and y < 342 and w < 210:

define the x and y position of the red rectangle and the height and width of it. These values must match with the x and y values returned by the green rectangle.

The number of levels is 3, and if this is not reached, then you lose. If this is reached, you win. There are comments printed in between to display your performance.

The last part of this function include the conditional statements to come out of this iterative loop and stop playing the music. These include: winning, losing, ‘q’ for quit, and if the ‘End’ button is clicked in the GUI. Fig 7.1, 7.2 show the 2 important parts of the function. An additional feature to be noted is the change in speed of the game over time.

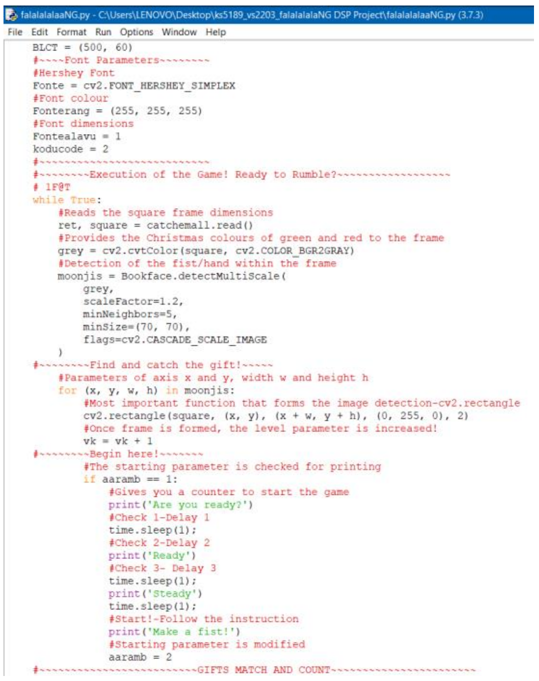


```
def Christmas():
    global playing
    #~~~~TEXT TO SPEECH CONVERTER~~~~~~
    import pyttsx3
    engine = pyttsx3.init()
    #~~~~~~AUDIO EMBEDDING OF BGM: CHRISTMAS POPS!~~~~~~~
    import pygame
    pygame.mixer.init()
    pygame.mixer.music.load('Christmas_Box_Pops.mp3')
    pygame.mixer.music.play()
    pygame.init()

    #-------------------------------------------------------
    #Input acquisition through the in-built Web-Camera
    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    import time
    Bookface = cv2.CascadeClassifier('Game.xml') #Cascade Haar Classifier variable
    catchemall = cv2.VideoCapture(0)

    #~~~~~~Initialization~~~~~~
    #No of gifts
    gifts = 0
    # Level of the game you are at
    age = 1
    # Levels of victory:
    vk = 0
    #Locking parameter x
    thalax = 0
    #Locking parameters y
    pootuy = 0
    #Success/Advancement parameter
    jeeth = 0
    #Ending parameter
    khatam = 40
    #Starting parameters
    aaramb = 1
    intro = 0
    #Position check
    desi = 1
    #Failure/Loss Parameter which gives luck!
    luck = 0
    #Failure acknowledgement parameter
    har = 0
    #~~~~~~~~~~~~~~~~~~~~~~~~~
    #~~~~~~~Presentation the frame of the game~~~~~~~~~~~~~~~~~~~~~~~~
    #~~~Margins/Boundaries~~~~~~
    TPCR = (500, 30)
    C = (250, 250)#~~~~~Middle
```

Figure 7.1 Christmas()-Part 1

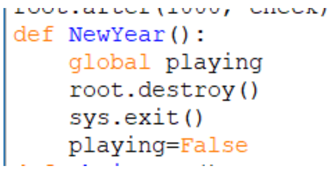

falalalalaaNG.py - C:\Users\LENOVO\Desktop\ks5189_vs2203_falalalalaNG DSP Project\falalalalaaNG.py (3.7.3)
File Edit Format Run Options Window Help

```
    BLCT = (500, 60)
    #~~~~Font Parameters~~~~~~~~
    #Hershey Font
    Fonte = cv2.FONT_HERSHEY_SIMPLEX
    #Font colour
    Fonterang = (255, 255, 255)
    #Font dimensions
    Fontealavu = 1
    kooducode = 2
    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    #~~~~~~~~Execution of the Game! Ready to Rumble?~~~~~~~~~~~~~~~~~~
    # 1F@T
    while True:
        #Reads the square frame dimensions
        ret, square = catchemall.read()
        #Provides the Christmas colours of green and red to the frame
        grey = cv2.cvtColor(square, cv2.COLOR_BGR2GRAY)
        #Detection of the fist/hand within the frame
        moonjis = Bookface.detectMultiScale(
            grey,
            scaleFactor=1.2,
            minNeighbors=5,
            minSize=(70, 70),
            flags=cv2.CASCADE_SCALE_IMAGE
        )
    #~~~~~~~~Find and catch the gift!~~~~~
        #Parameters of axis x and y, width w and height h
        for (x, y, w, h) in moonjis:
            #Most important function that forms the image detection-cv2.rectangle
            cv2.rectangle(square, (x, y), (x + w, y + h), (0, 255, 0), 2)
            #Once frame is formed, the level parameter is increased!
            vk = vk + 1
    #~~~~~~~~Begin here!~~~~~~~
            #The starting parameter is checked for printing
            if aaramb == 1:
                #Gives you a counter to start the game
                print('Are you ready?')
                #Check 1-Delay 1
                time.sleep(1);
                #Check 2-Delay 2
                print('Ready')
                #Check 3- Delay 3
                time.sleep(1);
                print('Steady')
                time.sleep(1);
                #Start!-Follow the instruction
                print('Make a fist!')
                #Starting parameter is modified
                aaramb = 2
    #~~~~~~~~~~~~~~~~~~~~~~~~GIFTS MATCH AND COUNT~~~~~~~~~~~~~~~~~~~~~~~
```

Figure 7.2 Christmas()-Part 2

3.2.2 <u>NewYear() function definition:</u>

This is the function that defines what the 'End' button does in the GUI. It uses the destroy to exit the loop of the GUI and 'sys.exit()' to escape from the loop of the main 'Christmas()' function when the value of a global flag 'playing' is true. Figure 9 shows this function.



```
def NewYear():
    global playing
    root.destroy()
    sys.exit()
    playing=False
```

Figure 8: NewYear()

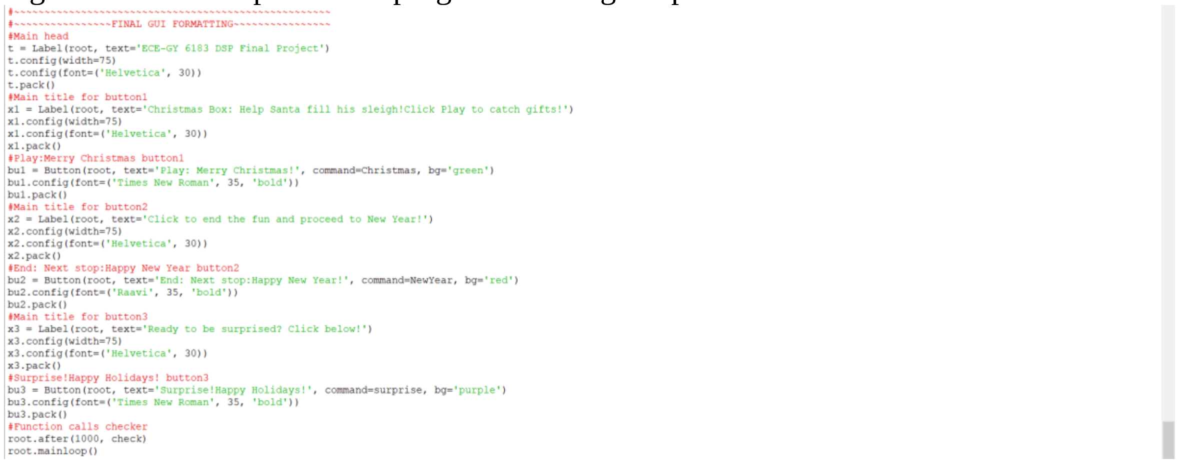
3.2.3 <u>check() function definition:</u>

This is the function that defines when the 'End' button executes in the GUI. It checks if the global flag 'playing' is true. If so, the 'Play' or 'Surprise' button can continue. If not, then the 'End' button can execute its function.

3.3 GRAPHICAL USER INTERFACE

The final part of the design flow is to provide the appearance of the GUI and the required matches to each of the functions to its corresponding buttons. 'Button 1' executes 'Christmas()', 'Button 2' executes 'NewYear', and 'Button 3' is the 'Surprise!'.

Figure 9 shows the part of the program defining the parameters for the GUI.



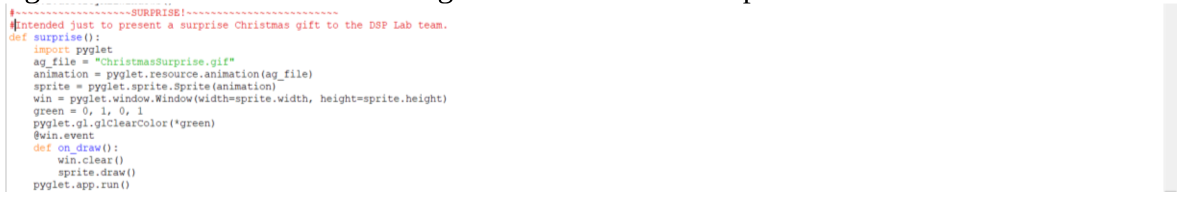
```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#~~~~~~~~~~~~~~~~FINAL GUI FORMATTING~~~~~~~~~~~~~~~~
#Main head
t = Label(root, text='ECE-GY 6183 DSP Final Project')
t.config(width=75)
t.config(font=('Helvetica', 30))
t.pack()
#Main title for button1
x1 = Label(root, text='Christmas Box: Help Santa fill his sleigh!Click Play to catch gifts!')
x1.config(width=75)
x1.config(font=('Helvetica', 30))
x1.pack()
#Play:Merry Christmas button1
bu1 = Button(root, text='Play: Merry Christmas!', command=Christmas, bg='green')
bu1.config(font=('Times New Roman', 35, 'bold'))
bu1.pack()
#Main title for button2
x2 = Label(root, text='Click to end the fun and proceed to New Year!')
x2.config(width=75)
x2.config(font=('Helvetica', 30))
x2.pack()
#End: Next stop:Happy New Year button2
bu2 = Button(root, text='End: Next stop:Happy New Year!', command=NewYear, bg='red')
bu2.config(font=('Raavi', 35, 'bold'))
bu2.pack()
#Main title for button3
x3 = Label(root, text='Ready to be surprised? Click below!')
x3.config(width=75)
x3.config(font=('Helvetica', 30))
x3.pack()
#Surprise!Happy Holidays! button3
bu3 = Button(root, text='Surprise!Happy Holidays!', command=surprise, bg='purple')
bu3.config(font=('Times New Roman', 35, 'bold'))
bu3.pack()
#Function calls checker
root.after(1000, check)
root.mainloop()
```

Figure 9. GUI Formatting

3.4 SURPRISE!

Alright…. If you have made it here… Thank you so much for reading through. This is the last part of our project and this was intended just to express our gratitude for the support from the Professor, the TAs and all the fellow students.

We used the pyglet library to process and edit a ‘gif’ format file as individual image frames and then print it as a collection of frames together in the same window. Pyglet does so by acquiring the width and height of the image frames in a sprite and using a function called ‘on_draw’ to draw the sprite onto its new frames.
Figure 10.1 shows the code and Figure 10.2 shows the output window.



```
#~~~~~~~~~~~~~~~~~~~~SURPRISE!~~~~~~~~~~~~~~~~~~~~~~~~~
#Intended just to present a surprise Christmas gift to the DSP Lab team.
def surprise():
    import pyglet
    ag_file = "ChristmasSurprise.gif"
    animation = pyglet.resource.animation(ag_file)
    sprite = pyglet.sprite.Sprite(animation)
    win = pyglet.window.Window(width=sprite.width, height=sprite.height)
    green = 0, 1, 0, 1
    pyglet.gl.glClearColor(*green)
    @win.event
    def on_draw():
        win.clear()
        sprite.draw()
    pyglet.app.run()
```

Figure 10.1: SURPRISE Code

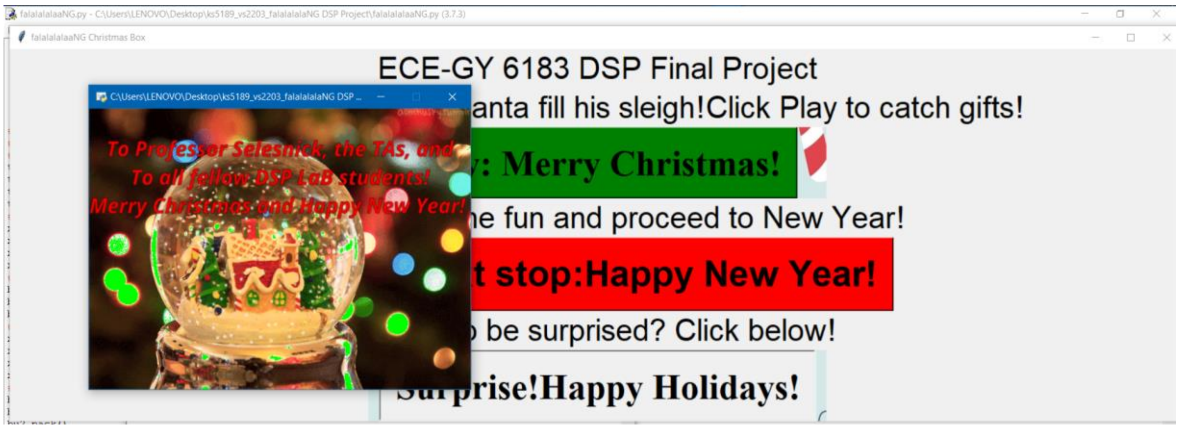

falalalalaaNG.py - C:\Users\LENOVO\Desktop\ks5189_vs2203_falalalalaNG DSP Project\falalalalaaNG.py (3.7.3)
falalalalaaNG Christmas Box
C:\Users\LENOVO\Desktop\ks5189_vs2203_falalalalaNG DSP ...
To Professor Selesnick, the TAs, and
To all fellow DSP LaB students!
Merry Christmas and Happy New Year!
ECE-GY 6183 DSP Final Project
anta fill his sleigh!Click Play to catch gifts!
: Merry Christmas!
he fun and proceed to New Year!
t stop:Happy New Year!
be surprised? Click below!
rprise!Happy Holidays!

Figure 10.2 SURPRISE Output Window

MERRY CHRISTMAS!
AND
HAPPY NEW YEAR!

CHAPTER 4
CONCLUSION

The project thus combines computer vision from OpenCv with interactive elements from TKinter, adds in music from pygame(), visual graphics (GIF) in pyglet() and text-to-speech from pyttsx() to develop an interesting real time interactive Christmas game programmed in python. The game gets trickier as you go from one level to another keeping the user engaged. The end-to-end model using image, audio and video processing increases its entertainment value.

REFERENCES

1. GIF Function; https://www.daniweb.com/programming/software-development/code/217060/show-animated-gif-files-python

2. TKinter; https://www.geeksforgeeks.org/python-gui-tkinter/

3. Computer Vision; Programming Computer Vision with Python- http://programmingcomputervision.com/downloads/ProgrammingComputerVision_CCdraft.pdf