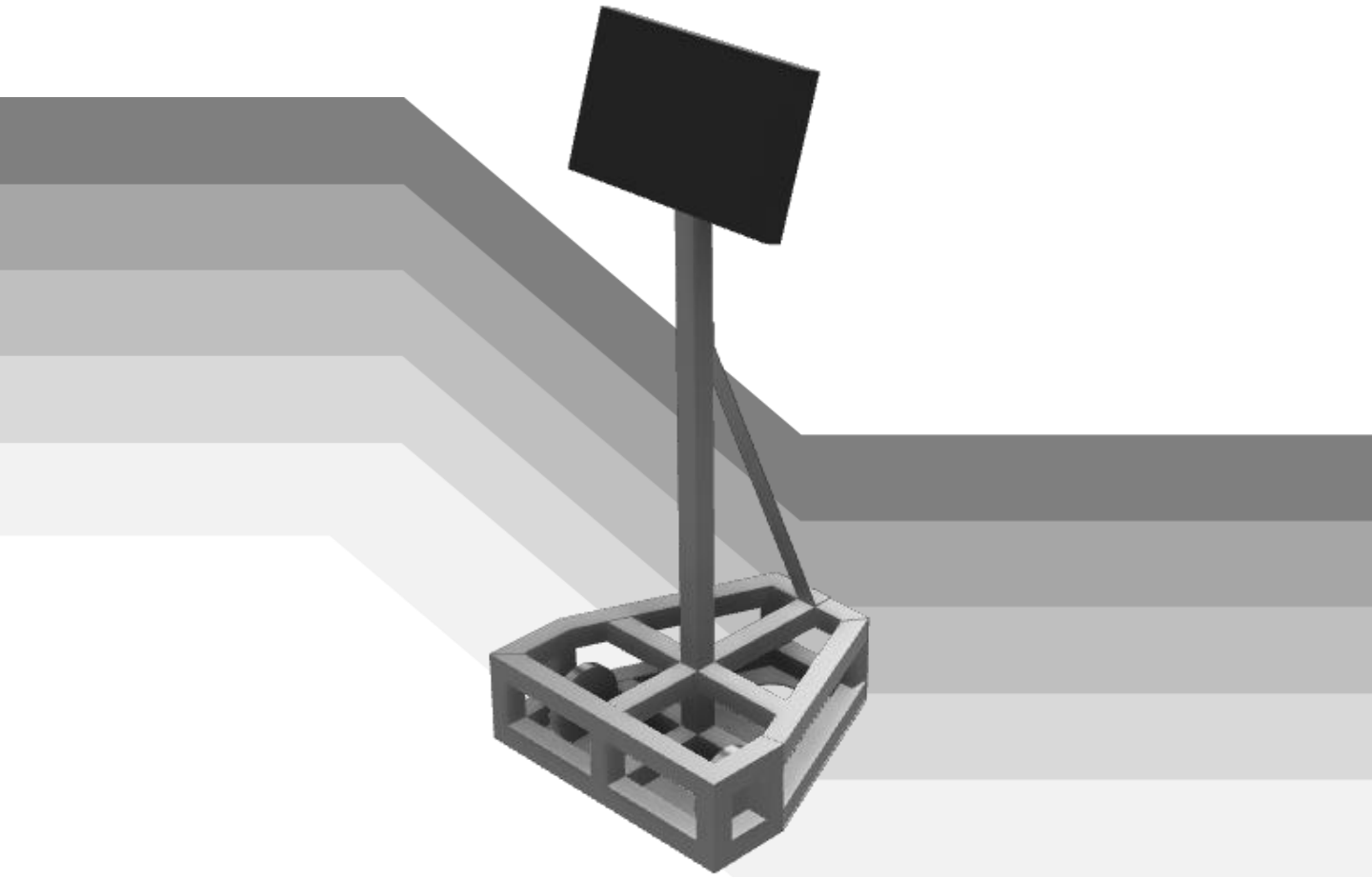
GRADE 11



Project File

Project Name: PAL Assistant - Software

Name: Abinav Pallathoor
Class & Sec: 11-C

Name: Aadil Abdul Jaleel
Class & Sec: 11-A

Academic Year: 2023-2024

Certificate

Board Roll No: _____

Certified that Master: _____

Grade 11 **Section** ___ has carried out practical work / Project work in Computer Science as prescribed by the Central Board of Secondary Education, New Delhi during the academic year 2021-2022.

Date: _____ Teacher in charge: _____

Acknowledgement

I extend my heartfelt appreciation to Mr. Yadav Singh for providing the platform and opportunity to undertake this computer science project. Your support and encouragement were crucial in fostering an environment conducive to learning and exploration.

I would like to express my gratitude to my classmates for their active collaboration and shared commitment to the project. Working together has not only deepened our understanding but also made the entire experience enjoyable and rewarding.

I also want to thank my friends and family for their unwavering support and understanding. Their encouragement kept me motivated during the ups and downs of the project.

In conclusion, the successful completion of this project is a testament to the collective efforts of all involved. Thank you for being part of this enriching journey.

Index

Introduction to Python

About Python

Python stands out as a powerful, high-level programming language renowned for its simplicity, readability, and versatility. Created by Guido van Rossum and first released in 1991, Python prioritizes code readability and developer-friendly syntax, which makes it an ideal choice for beginners and experienced programmers alike. Its elegant design emphasizes clear and concise code, reducing the cost of program maintenance and development time. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming, providing developers with the flexibility to choose the best approach for their projects.

One of Python's key strengths lies in its extensive ecosystem of libraries and frameworks, catering to a broad range of applications. From web development using frameworks like Django and Flask, to data science and machine learning with NumPy, pandas, and TensorFlow, Python has become a go-to language for diverse fields. Its adaptability extends to areas such as automation, artificial intelligence, network programming, and game development. Python's community-driven development model, supported by a vast and active community, ensures continuous growth, innovation, and a wealth of resources for developers worldwide. Overall, Python's simplicity, versatility, and community support make it an indispensable tool for a wide array of programming tasks and applications.

Introduction to Python

Features of Python

1. Readable and Simple Syntax

Python's clean and readable syntax uses whitespace and indentation to define code blocks, eliminating the need for braces or keywords. This simplifies the code, reducing the likelihood of syntax errors.

2. Extensive Standard Library

Python's extensive standard library minimizes reliance on external dependencies, providing built-in tools for tasks such as file handling, network communication, and data manipulation.

3. Dynamic Typing and Memory Management

Python's dynamic typing, determining variable types at runtime, couples with automatic memory management by the interpreter. This simplifies memory tasks, enabling developers to focus on program logic.

4. Object-Oriented Programming (OOP)

Python supports object-oriented programming, enabling code structuring with classes and objects for modular, reusable, and maintainable software.

5. Cross-Platform Compatibility

Python's cross-platform compatibility enables code to run seamlessly across operating systems, enhancing application portability. Ensuring broad compatibility for efficient software deployment.

Introduction to Python

Applications of Python

As python supports a wide variety of libraries it can be used for many applications such as:

1. **Web Development:** Django and Flask frameworks for building scalable and efficient web applications.
2. **Data Science:** NumPy, pandas, scikit-learn, TensorFlow, and PyTorch for data manipulation, analysis, and machine learning.
3. **Automation and Scripting:** Widely used for automating repetitive tasks and scripting, enhancing system administration and workflow efficiency.
4. **Scientific Computing:** Utilized in scientific research and computations, with libraries like SciPy providing tools for scientific and technical computing.
5. **Artificial Intelligence and Natural Language Processing:** Python serves as a primary language for AI and NLP applications, with frameworks such as NLTK and spaCy.
6. **Game Development:** Pygame library for developing 2D games, providing a straightforward interface for game designers and developers.
7. **Network Programming:** Python's socket programming and libraries like Twisted for creating network applications and protocols.
8. **Desktop GUI Applications:** Tkinter, PyQt, and Kivy for creating graphical user interfaces in desktop applications.
9. **Database Management:** Interface with various databases using libraries like SQLAlchemy, enabling efficient data storage and retrieval.
10. **Cybersecurity:** Python is widely used for cybersecurity tasks, scripting, and developing security tools.

Introduction to Python

Working with Python

Working with Python involves the use of a Python interpreter, also known as the Python shell, to write and execute Python programs. To initiate the interpreter, one must have Python installed on their computer or leverage an online Python interpreter. There are two primary execution modes: interactive mode and script mode. In interactive mode, individual statements can be executed instantaneously by typing them directly at the ">>>" prompt. This mode is useful for testing single-line code for immediate execution. However, a drawback is the inability to save statements for future use, requiring users to retype them for each run.

On the other hand, script mode allows users to write more extensive Python programs by saving multiple instructions in a Python source code file with the ".py" extension. Scripts can be executed using the interpreter by specifying the file name and path in the prompt or by using an Integrated Development and Learning Environment (IDLE). Script mode is advantageous for creating and executing more complex programs, as it allows users to save their work and reuse scripts for future runs. The output of the script appears on the shell, providing a comprehensive view of the program's execution results.

Problem Definition

In today's fast-paced and dynamic business environments, the demand for efficient and responsive customer service has never been higher. Traditional receptionist roles often struggle to keep up with the increasing volume of inquiries, resulting in longer wait times, frustrated customers, and potential business disruptions. Additionally, the rising cost of human labor and the need for 24/7 availability pose challenges to maintaining a consistently effective receptionist service.

To address these issues, there is a growing interest in implementing receptionist robots as a potential solution. However, several challenges need to be overcome to ensure the successful integration and adoption of these robotic systems. Key concerns include the need for seamless integration with existing communication systems, the ability to understand and respond appropriately to diverse customer queries, and ensuring a user-friendly interface that promotes positive interactions.

Our project was driven by the goal of developing an interactive and intelligent receptionist robot that not only answers questions in a smart manner but also prioritizes an approachable and friendly demeanor. We placed a strong emphasis on designing a user interface (UI) that is not only visually appealing but also ensures ease of use, aiming for a seamless and hassle-free interaction for users. By focusing on these aspects, our objective was to create a receptionist robot that not only fulfills its functional role effectively but also enhances the overall user experience through a combination of smart responses and a welcoming design.

Process To Solve The Problem

1. Problem Analysis

We began by thoroughly discussing the problem, analyzing its intricacies, and setting specific goals to guide our project's trajectory. This initial phase laid the groundwork for our focused approach, providing a clear direction for the project.

2. Algorithm Development

We then proceeded to break down the problem into smaller components and crafted tailored algorithms for each segment, ensuring a systematic and efficient approach to problem-solving. This method allowed us to address the intricacies of the problem with a structured mindset.

3. Integration

Following the development of individual algorithms, our next step involved combining them into a unified codebase. This process required careful consideration of interdependencies to ensure seamless coordination and functionality. We ensured the code had separate threads to prevent interruptions.

4. Testing and Bug Fixing

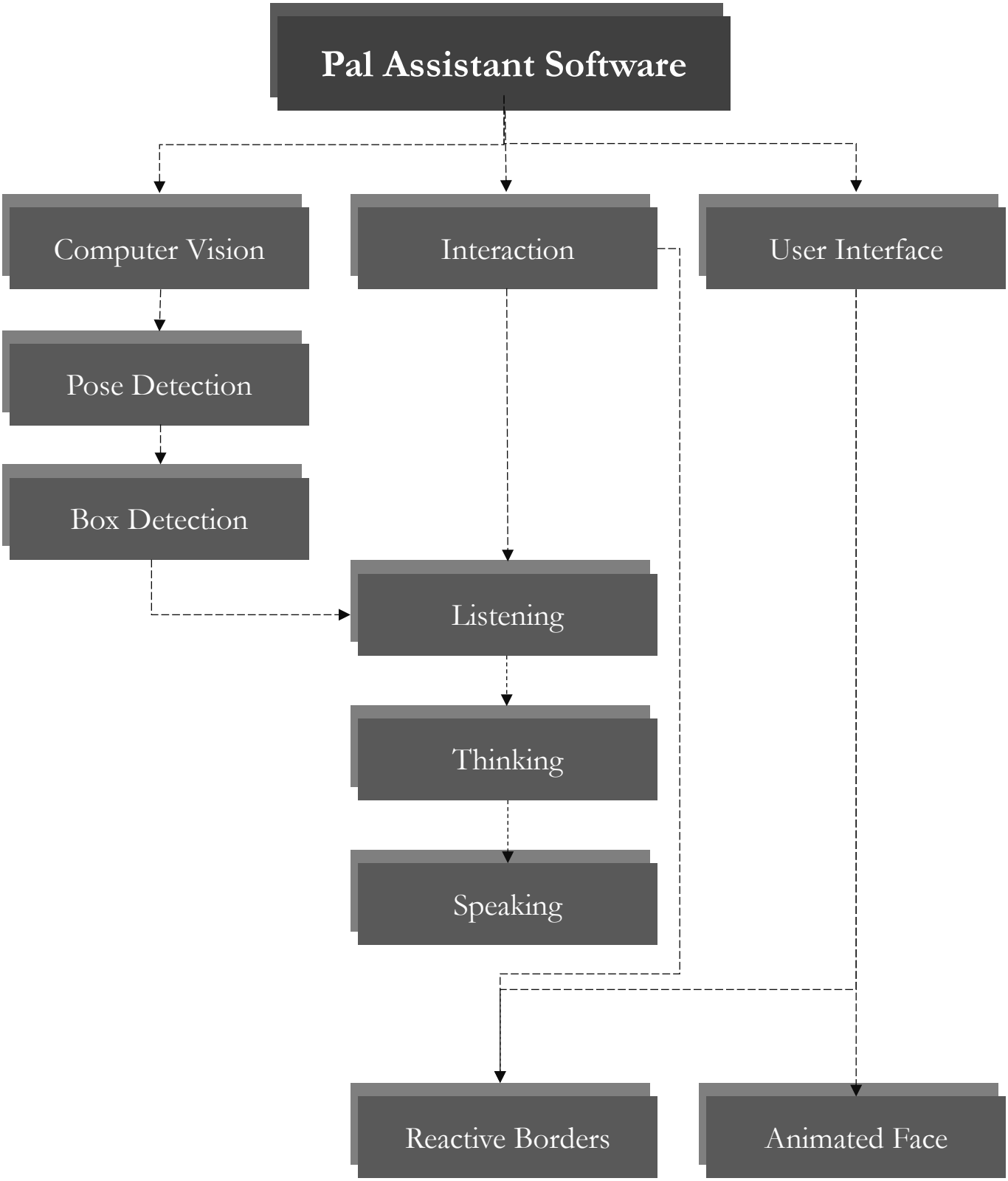
After integrating the code, we conducted thorough testing to ensure its bug-free performance, considering the need for extended operation and user interactions. This comprehensive testing approach aimed to guarantee the solution's reliability and effectiveness in real-world scenarios.

Algorithm Outline

The PAL software is structured into three integral components: computer vision, interaction, and the user interface, each running independently in separate threads. The computer vision segment relies on the Mediapipe library for efficient pose detection, determining whether an individual is positioned in front of the robot. Upon identifying a person, the system transitions into an active listening mode, initiating the thinking phase through the ChatGPT API. Subsequently, the generated text is spoken using the pyttsx module. To provide a visual representation of these processes, reactive borders dynamically change color, each hue corresponding to a distinct phase, with accompanying text to denote the current state.

Moreover, the software incorporates an animated face feature, implemented through the Pygame UI. This element adds a dynamic dimension to the robot's interactions, enabling it to display speaking animations during conversation and idle animations during non-interactive periods. The combination of these components ensures a seamless and engaging user experience, blending advanced functionalities with visual cues for effective communication.

Flowchart


Pal Assistant Software
Computer Vision
Interaction
User Interface
Pose Detection
Box Detection
Listening
Thinking
Speaking
Reactive Borders
Animated Face

Functions

1. SpeechRecognition()

This function enables a medium for Pal to first and foremost detect sound, and then can convert the detected sound to text which then allows the robot to think upon the instructions heard verbally. It is built with the help of a module called "speech_recognition".

2. SpeakText()

SpeakText() helps the robot converse with the user as it is able to translate the text received from the computer and then convert it to speech in order for the output to reach the user. We are able to do so using a module that supports Text-Speech, which is, "pyttsx3".

3. Response()

Once the instructions have been converted to text, it is then sent to Chat-GPT as it computes a suitable response suiting to the user's needs. We are able to communicate with Chat-GPT with its built in API feature which let's us use Chat-GPT tailored to our needs.

4. ReactiveBorder()

For ease of use and understanding for the users, we have implemented this function so that the user is able to infer the robot's emotion. For example, with the help of this function a different coloured border would surround the screen as it goes through the: Listening, Thinking, Speaking stages. We are able to display borders with the help of a module called, "pygame"

Functions

5. PalFace()

In order to give a more humanly presence for the robot, it is crucial to give it a pleasant face to be more welcoming to the guests. Therefore, with the help of a module called "pygame", we are able to display 2 circular eyes and a pleasant smile as it listens, thinks and speaks to the fellow user.

6. VoiceLoop()

This function will aid in the detection of potential guests as it roams around the reception. Once detected with the help of a camera and an auxiliary function, "CVLoop", it will run a greeting sequence followed by listening awaiting further instructions. This function is multi-threaded with "CVLoop", so both will run simultaneously for increased efficiency

7. CVLoop()

Similar to VoiceLoop(), CVLoop() will act as the eyes, as the camera pans around its surroundings to detect potential guests. It is able to detect people using the "cv2" module that can detect faces very quickly and efficiently.

8. UIFunctions()

As Pygame comes with limited number of functions we had to make custom gui functions. This includes a lerp function, custom rounded rectangles, and color changers.

Technical Documentation

Hardware Requirement

To run the program the hardware requirements are:

Operating System	64-bit Windows 7 or later
Processor	1.5GHz or faster
Memory	4GB (4,096MB) RAM
Free HDD Space	3GB

Software Requirement

To run the program the software requirements are:

1. Install of python (latest version)
2. Install of the program and executable file

User Documentation

1. Python Installation

Install the latest version of python from https://www.python.org/
Install python and pip using the installer and select all the options.

2. Install Libraries

Using pip and the following commands install the following libraries:

```
pip install pygame
pip install textwrap
pip install SpeechRecognition
pip install pyttsx3
pip install openai
pip install opencv-python
pip install cvzone
```

3. OpenAI API Key

Go to https://platform.openai.com/api-keys and create a new API key and copy the API key to the 'key.txt' file.

4. Running the Program

Double click the Pal_Software_V9.py file to run the program.
*The code requires wifi connection and new api key to run

5. Exiting the Program

As the file runs in full screen mode, press windows key to open taskbar, and quit the program using task manager, or simply closing the program on the taskbar.

Program Listing

```
# Libraries
# Multi Loop Library
import threading

# GUI Library
from pygame import *
import pygame.freetype
import pygame.gfxdraw
pygame.init()
from math import ceil
from textwrap import wrap
import random
import time

# Voice Library
import speech_recognition
import pyttsx3
import openai

# Computer Vision Library
import cv2
from cvzone.PoseModule import PoseDetector


# GPT API Key
# Getting API Key
key_file = open("key.txt", "r")
gpt_key = key_file.read()
print(gpt_key)

openai.api_key = str(gpt_key)
messages = [ {"role": "system", "content": "You are a intelligent
assistant."} ]

# GUI Variables
# Colours
BG_CLR = (0, 0, 0)
FT_CLR = (180, 180, 180)
LISTENING_CLR = (56, 116, 255)
THINKING_CLR = (59, 247, 160)
SPEAKING_CLR = (74, 255, 255)
IDLE_CLR = (255, 255, 255)
FACE_CLR = (255, 255, 255)
```

Program Listing

```
# Fonts
FT_FONT = pygame.freetype.SysFont('Roboto', 25)

# Computer Vision Functions
# Defining Variables
x_pos, width = 0, 0

def render_window(frame):
    cv2.imshow("Frame", frame)

    key = cv2.waitKey(10)
    if key == 27:
        exit()

# Voice Function
recognizer = speech_recognition.Recognizer()
def SpeechRecognition():
    try:
        with speech_recognition.Microphone() as source2:
            recognizer.adjust_for_ambient_noise(source2,
duration=0.2)
            audio2 = recognizer.listen(source2, timeout=1,
phrase_time_limit=3)
            user_inp = recognizer.recognize_google(audio2)

            print(f"User: {user_inp}")
            return user_inp.lower()

    except speech_recognition.RequestError as e:
        SpeakText("Sorry, I am not able to process the data")

    except:
        return

# Speak Text Function
def SpeakText(command):
    print(f"Pal: {command}")
    engine = pyttsx3.init()
    voice = engine.getProperty('voices')
    engine.setProperty('voice', voice[1].id)
    engine.say(command)
    engine.runAndWait()
```

Program Listing

```
# ChatGPT Function
def Response(user_inp, prompt):
    message = prompt + user_inp


    if message:
        messages.append(
            {"role": "user", "content": message},
        )
        chat = openai.ChatCompletion.create(
            model="gpt-3.5-turbo", messages=messages
        )
    reply = chat.choices[0].message.content
    messages.append({"role": "assistant", "content": reply})

    return reply

# GUI Functions
def AAfilledRoundedRect(surface,rect,color,radius=0.4):

    rect = Rect(rect)
    color = Color(*color)
    alpha = color.a
    color.a = 0
    pos = rect.topleft
    rect.topleft = 0,0
    rectangle = Surface(rect.size,SRCALPHA)

    circle = Surface([min(rect.size)*3]*2,SRCALPHA)
    draw.ellipse(circle,(0,0,0),circle.get_rect(),0)
    circle =
transform.smoothscale(circle,[int(min(rect.size)*radius)]*2)

    radius = rectangle.blit(circle,(0,0))
    radius.bottomright = rect.bottomright
    rectangle.blit(circle,radius)
    radius.topright = rect.topright
    rectangle.blit(circle,radius)
    radius.bottomleft = rect.bottomleft
    rectangle.blit(circle,radius)

    rectangle.fill((0,0,0),rect.inflate(-radius.w,0))
    rectangle.fill((0,0,0),rect.inflate(0,-radius.h))
```

Program Listing

```
    rectangle.fill(color,special_flags=BLEND_RGBA_MAX)
    rectangle.fill((255,255,255,alpha),special_flags=BLEND_RGBA_MIN
)

    return surface.blit(rectangle,pos)

# GUI Gradient Rectangle
def GradientRect(screen, left_colour, right_colour, target_rect ):
    colour_rect = pygame.Surface((2, 2))
    pygame.draw.line(colour_rect, left_colour,  (0,0), (0,1))
    pygame.draw.line(colour_rect, right_colour, (1,0), (1,1))
    colour_rect = pygame.transform.smoothscale(colour_rect,
(target_rect.width, target_rect.height))
    screen.blit(colour_rect, target_rect)

# GUI Centre Text
def RenderText(text_str, screen, font, font_colour, position):
    word_wrap = wrap(text_str, 70)

    if len(word_wrap) < 3:
        for i, word in enumerate(word_wrap[::-1]):
            text_rect = font.get_rect(word)
            text_rect.center = (position[0], position[1]-30*i)
            font.render_to(screen, text_rect.topleft, word,
font_colour)
    else:
        text_rect = font.get_rect("Speaking")
        text_rect.center = position
        font.render_to(screen, text_rect.topleft, "Speaking",
font_colour)

# GUI Colour Lerping Function
def LerpColor(start_color, end_color, progress):
    r = start_color[0] + int((end_color[0] - start_color[0]) *
progress)
    g = start_color[1] + int((end_color[1] - start_color[1]) *
progress)
    b = start_color[2] + int((end_color[2] - start_color[2]) *
progress)
    return (r, g, b)
```

Program Listing

```
# GUI Colour Transitioning Function
def ColorTransition(start_color, end_color, transition_duration,
start_time):
    elapsed_time = time.time() - start_time
    progress = min(elapsed_time / transition_duration, 1)

    current_color = LerpColor(start_color, end_color, progress)
    return current_color

# GUI Reactive Border
current_mode = mode = 1
transition_duration = 100
transition_start_time = time.time()
current_color = end_color = BG_CLR
def ReactiveBorder(mode, screen, border_width = 100):
    global current_color, end_color, transition_start_time,
current_mode

    if mode == 0: end_color = IDLE_CLR
    if mode == 1: end_color = LISTENING_CLR
    elif mode == 2: end_color = THINKING_CLR
    elif mode == 3: end_color = SPEAKING_CLR

    if mode != current_mode:
        current_mode = mode
        transition_start_time = time.time()

    current_color = ColorTransition(current_color, end_color, 30,
transition_start_time)

    GradientRect(screen, BG_CLR, current_color,
pygame.Rect(screen.get_width() - border_width, 0, border_width,
screen.get_height()))
    GradientRect(screen, current_color, BG_CLR, pygame.Rect(0, 0,
border_width, screen.get_height()))
```

Program Listing

```
# GUI Pal Face Function
def PalFace(mode, screen):
    pygame.gfxdraw.aacircle(screen, screen.get_width()//2,
screen.get_height()//2+10, 100, FACE_CLR)
    pygame.gfxdraw.filled_circle(screen, screen.get_width()//2,
screen.get_height()//2+10, 100, FACE_CLR)

    if mode == 3: mouth_offset = random.randint(0, 20)
    else: mouth_offset = 0

    pygame.draw.rect(screen, BG_CLR,
pygame.Rect(screen.get_width()//2 - 210//2,
screen.get_height()//2+10-200 + mouth_offset, 210, 200))


    pygame.gfxdraw.aacircle(screen, screen.get_width()//2 + 200,
screen.get_height()//2-100, 50, FACE_CLR)
    pygame.gfxdraw.filled_circle(screen, screen.get_width()//2 +
200, screen.get_height()//2-100, 50, FACE_CLR)

    pygame.gfxdraw.aacircle(screen, screen.get_width()//2 - 200,
screen.get_height()//2-100, 50, FACE_CLR)
    pygame.gfxdraw.filled_circle(screen, screen.get_width()//2 -
200, screen.get_height()//2-100, 50, FACE_CLR)

# Threading
string_text = "Hello, I am Pal"
prompt = "(You are a robot called pal made by Abinav Pallathoor and
Aadil Abdul Jaleel students of the school, you are a receptionist
for guests who enter the school(Our Own High School Al Warqa), you
are witty childlike robot, who is polite. The principal of our
school is Mrs. Anjali Murthy and the vice principal is Mr. Kapil.
You are a robot made by the RAISE club headed by our coach Udaya
Balakrishnan.  ): "
mode = 1
running = True
person_detected = False
```

Program Listing

```
# GUI Variables
# Colours
BG_CLR = (0, 0, 0)
FT_CLR = (180, 180, 180)
# Fonts
FT_FONT = pygame.freetype.SysFont('Roboto', 25)

# Setting Screen
screen = pygame.display.set_mode((0, 0), pygame.FULLSCREEN)
pygame.display.set_caption('Pal Screen')

# Main Loop
# Voice Thread
def VoiceLoop():
    global string_text, running, mode, person_detected
    detect_time_buffer = 10
    detect_time = current_time = time.time()
    greet_person = False
    greet_finished = True

    while running:
        current_time = time.time()
        if detect_time > current_time:
            if greet_person and  not greet_finished:
                mode = 3
                SpeakText("Hello, I am Pal")
                greet_person = False
                greet_finished = True

            print("Listening...")
            string_text = "Listening"
            mode = 1
            user_inp = SpeechRecognition()
            if user_inp == None or user_inp == "": continue

            print("Thinking...")
            string_text = "Thinking"
            mode = 2
            response = Response(user_inp, prompt)
```

Program Listing

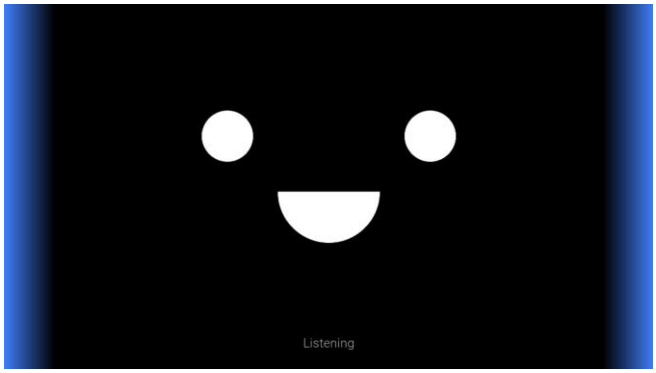
```
            print("Speaking...")
            string_text = response
            mode = 3
            SpeakText(response)

        if person_detected:
            detect_time = time.time() + detect_time_buffer
            if not greet_finished: greet_person = True
        else:
            mode = 0
            string_text = "Hello, I am Pal"
            greet_finished = False

# Computer Vision Thread
def CVLoop():
    global string_text, running, mode, person_detected

    detector = PoseDetector()
    capture = cv2.VideoCapture(0)

    while running:
        # Body Detection
        success, img = capture.read()
        img = detector.findPose(img)
        imlist, box = detector.findPosition(img)

        # Getting X & Width Value
        try:
            x_pos, width = box['center'][0] - 320,  box['bbox'][2]
            person_detected = True
        except:
            person_detected = False

        # Rendering Window
        render_window(img)

voice_thread = threading.Thread(target=VoiceLoop)
voice_thread.start()

cv_thread = threading.Thread(target=CVLoop)
cv_thread.start()
```

Program Listing

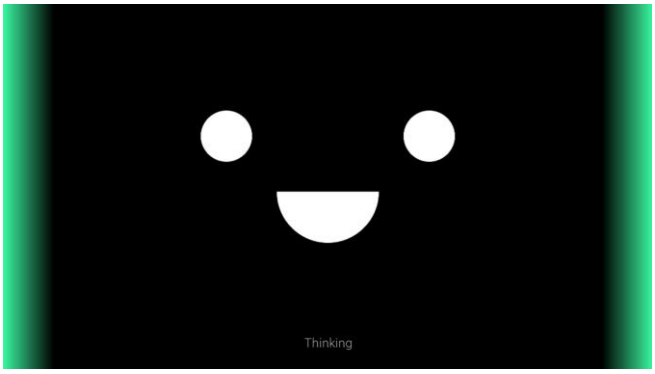
```
# GUI Loop
while running:
    # User Events
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Screen Rendering
    screen.fill(BG_CLR)
    ReactiveBorder(mode, screen)
    RenderText(string_text, screen, FT_FONT, FT_CLR,
(screen.get_width()/2, screen.get_height()-50))
    PalFace(mode, screen)

    pygame.display.flip()
```

Output

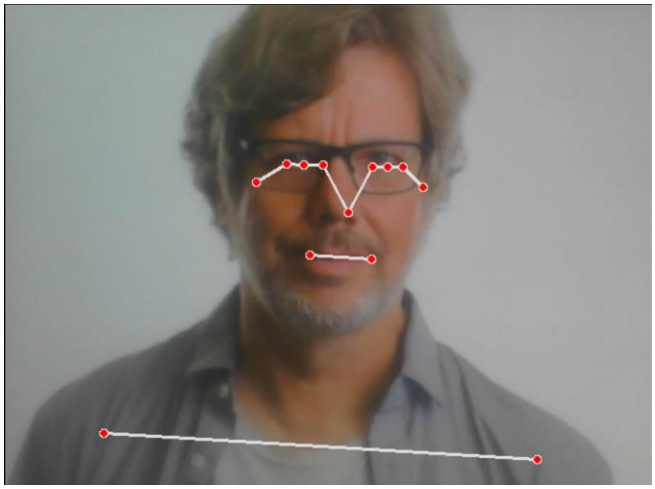
Pal Software UI, in idle state



Pal Software UI, in listening state



Pal Software UI, in thinking state



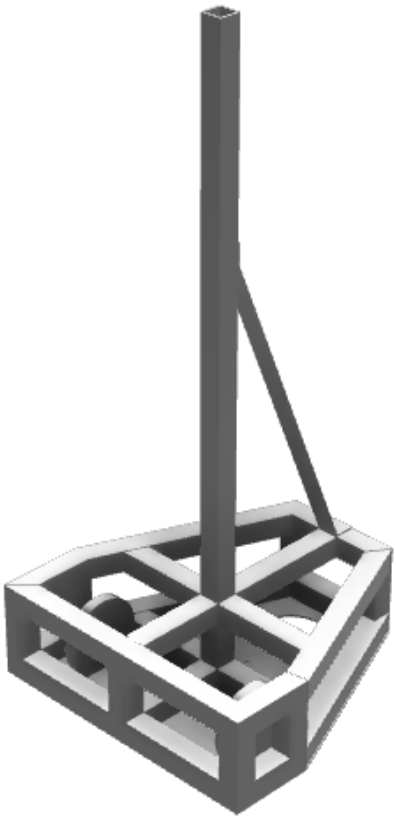
Pal Software Computer Vision

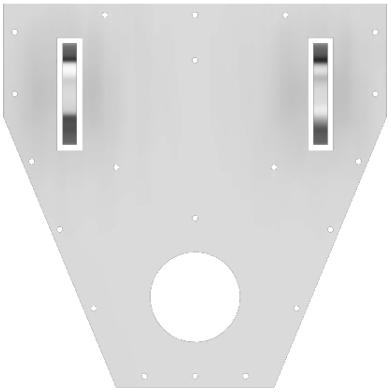
Robotics Documentation

Hardware

For the main chassis of the robot, we have designed a bespoke structure that we have fabricated. The material used is a magnesium alloy that was light but didn't compromise on strength. The total height comes just under 80cm with the weight without any components coming in a mere 2.7 kg.

We designed this using a CAD software called Fusion 360.





Facilitating the motion is with the help of 2 large motors situated at the top farthest corners. Instead of having 4 motors that takes more energy and is more difficult to maneuver in tight spaces, we rather elected to keep only a third wheel which is a castor wheel.

The castor wheel facilitates 360-degree free rotation which helps quick maneuvers and is more efficient compared to that of a 4-motor configuration.

Robotics Documentation

Electronics

As the main brain for Pal, we are using Pi-Top[4] which houses a modified raspberry-pi 4.
This device has a lot of computational features and helps us create a consistent connections between our motors or sensors with the computer.
It features a lot of ports opening vast amounts of modification making it the perfect choice for us.

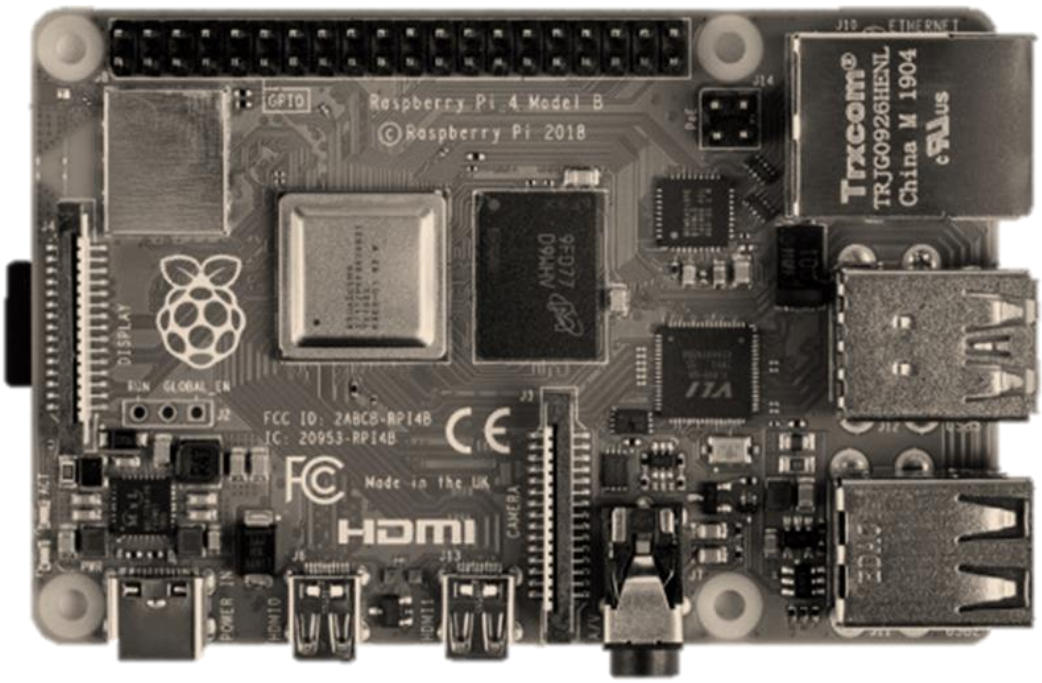
Helping us display the necessary information and the robot's emotion is a 11.6" touchscreen display. This is a special display built specifically for Pi-Top.
It delivers great resolution and great feedback to touch.

In order for the robot to comprehend the user, we need to equip it with a microphone.
It is important we use a uni-directional microphone rather than an omni-directional microphone, so it doesn't pick up unwanted noise, delivering consistent results.

Robotics Documentation

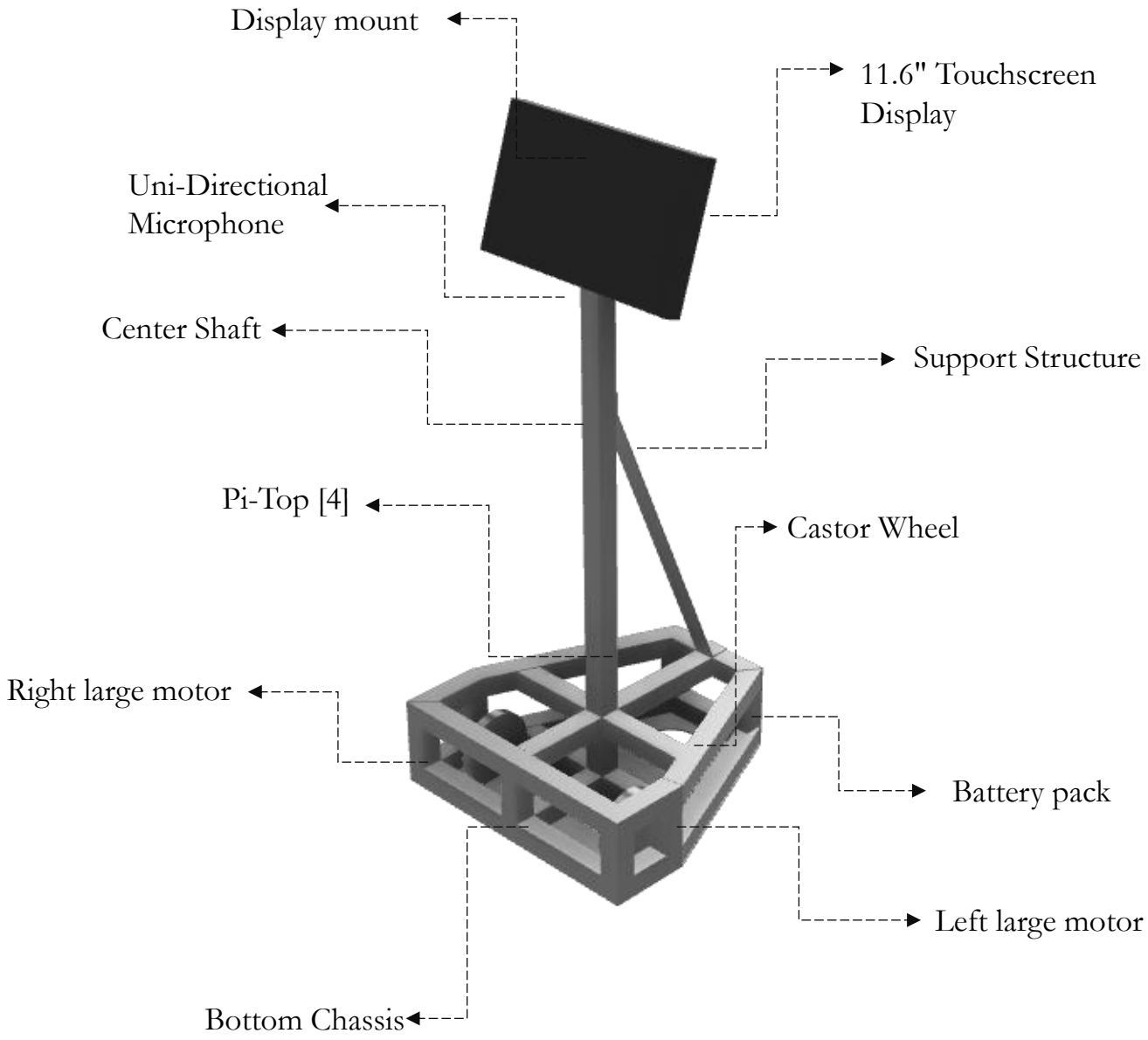
Electronics

On the software side of things, we have thought best to equip Pal with a raspberry pi 4. Since it delivers the best computational performance compared to that of other boards in the same field. We have not used Arduino or other microcontroller boards since it won't be able to compute a lot of information in a quick amount of time compared to a raspberry pi.
Another feature of a raspberry pi is it's vast choice of ports that really help in a consistent workflow.



Robotics Documentation

Robot Schematics



Limitations

Our project faced several challenges, including the code getting stuck in the listening phase, even without active speech input. To address this, we incorporated computer vision to detect a user's presence before initiating conversations. Although this mitigated the issue, it didn't completely resolve it, necessitating further improvements in accurately discerning user engagement.

Additionally, our ambition to enhance the robot's expressiveness through facial animation encountered complexities. Despite making progress, achieving a truly lifelike representation proved challenging. Moreover, the project's dependence on a strong Wi-Fi connection for optimal performance prompted us to explore local system execution. However, transitioning to a standalone system demanded significant computational power, highlighting the trade-offs between performance and hardware requirements. Continuous refinement efforts are crucial to overcome these limitations and improve the overall functionality and user experience of the system.

Suggestions for improvement

Our focus on improvement encompasses various crucial aspects of our project. Firstly, we aim to enhance the robot's emotional intelligence, enabling it to comprehend and respond to human emotions with greater sensitivity and respect. This involves refining the algorithm to accurately interpret emotional cues, fostering a more empathetic and responsive interaction between the robot and users.

Secondly, we aspire to bolster the robot's ability to identify and differentiate between individuals engaging with it. This involves a multi-modal approach, combining both voice recognition and computer vision technologies to create a more robust system for user identification. By seamlessly integrating these two modalities, we aim to provide a personalized and tailored experience, enriching the user-robot interaction.

Additionally, we are committed to stabilizing and improving the system's capability to remember people. While an initial implementation has been introduced, its stability requires further optimization. Finally, we are dedicated to optimizing response times, seeking ways to streamline the communication process and ensure prompt interactions. This involves fine-tuning the system architecture and algorithms to achieve faster and more efficient responses, ultimately contributing to a more seamless and engaging user experience.

Robotics Issues & Improvements

During the development of our project, Pal, we were faced with many challenges. First, we had an issue with implementing a media-pipe module and was forced to format the entire raspberry-pi which led us to going through the entire process once again. The text-speech would at times create faults and deliver inconsistent outputs. A silly issue yet created the most problems was a short USB type C-C display cable that was a bit short than the intended length. This prevented us from powering the motors since it was held in fully sealed enclosure.
Still, we were able to overcome all these issues and emerge victorious, delivering our most daring project so far: Pal.

Our project is strong, but to enhance its overall performance, we could address some areas of improvement. One would be the lack of arms taking on to more of a humanoid form. Secondly is the Speech model we are using. It's a very robotic voice and its lack of configurations gives us a limited set of options; at times it is difficult to comprehend Pal so it is an area we should improve on. An addition of a Lidar sensor would help in the 3D mapping of its surroundings enabling us to be fully confident on Pal's movements. But due to the limitations of the ultrasonic sensors equipped, Pal's movements have been necessary to be slow and careful. Lastly, developing a wireless charging dock would further enhance the user experience with Pal, rather the present situation that requires a human presence to plug a cable into the Pi-Top

Bibliography

Works Cited:

1. GeeksforGeeks
2. W3 Schools
3. Pygame Documentation
4. OpenAI
5. Stack overflow