TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

NATIONAL COLLEGE OF ENGINEERING

A

PROJECT PROPOSAL

ON

"RESPONSIVE POMODORO TIMER WITH AMBIENT ANIMATION AND AUDIO FEEDBACK USING PYGAME"

SUBMITTED BY:

PRABHAT RAWAL (NCE079BCT022)

SUBMITTED TO:

ER. RAJAN KUSI

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

LALITPUR, NEPAL

1 AUG, 2024

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to our dedicated subject teacher, Er Rajan Kusi. His guidance and support have been significant in assigning us this project, allowing to bridge the gap between academia and practical application. His insights and encouragement have significantly enriched my learning experience.

The project executed provides an opportunity to apply previous knowledge, view real examples set in a worldly framework, and have the opportunity to be involved with real applications of computer Graphics. All of this promotes a deeper understanding of the subject being studied and an improved ability to recall the information and experiences later, as needed.

Furthermore, I owe a debt of gratitude to the Department of Electronics and Computer Engineering of National College of Engineering for their invaluable assistance throughout this voyage. The provision of essential documents and a conducive platform for my work has been instrumental in shaping the trajectory of our project.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CG	Computer Graphics
RGB	Red, Green, Blue (color model)
UI	User Interface
2D	Two-Dimensional
SDL	Simple DirectMedia Layer
Pygame	Python library for 2D games and graphical applications
JPG	Joint Photographic Experts Group
ESC	Escape
PNG	Portable Network Graphic

1. INTRODUCTION

1.1 Background

Computer graphics is the use of computers to create and display images and visuals. It includes drawing shapes, images, animations, and 3D models on a screen. Basic elements of computer graphics involve pixels (the tiny dots that make up an image), colors (usually represented using the RGB color model), and coordinates to place objects. It is used in games, movies, design, simulations, and user interfaces. Simple graphics include lines, circles, and rectangles, which are drawn using algorithms. Tools like Pygame, OpenGL, and Unity help developers create graphics easily. Computer graphics make it possible to turn ideas into visual content using technology.

Pygame is a free and open-source Python library used for making 2D games and simple graphical applications. It provides modules for drawing shapes, handling images, playing sounds, managing user input , and controlling the game loop. Pygame makes it easy to work with graphics by offering built-in functions to draw lines, circles, rectangles, and display images on the screen. It is built on top of the SDL (Simple DirectMedia Layer) library, which handles low-level multimedia tasks. Pygame is popular among beginners because it's simple to use and works well for learning game development, animations, and basic computer graphics in Python.

This project presents a responsive Pomodoro timer developed using Python and the Pygame library, designed to enhance focus and productivity through a combination of ambient animation and audio feedback. The timer operates in full-screen mode and dynamically adapts to various screen resolutions by calculating UI element positions based on the device's width and height, ensuring a consistent visual experience across displays. Ambient background themes—such as campfires, snowy landscapes, and nighttime cityscapes—are animated frame-by-frame and accompanied by looped audio, creating an immersive and calming work environment. Motivational quotes and real-time clocks further reinforce mindfulness during study or work sessions. The interface includes clearly styled start and reset buttons, and the background themes automatically transition when the current audio ends. This project is ideal for users seeking a focused,

aesthetically pleasing productivity tool and offers flexibility for developers to customize positioning or themes as needed.

1.2 Objectives

List out the specific objectives to solve your problem. Do not write general objectives.

- To build a responsive Pomodoro timer using Pygame.
- To enhance user focus with ambient animations and audio feedback.
- To provide a clean, customizable, and resolution-independent interface.

2. METHODOLOGY

2.1 Block Diagram

The block flow diagram outlines the execution process of the Pomodoro timer system, detailing its operational structure across four key stages. The initialization phase begins by setting up the PyGame mixer and font, defining the screen's height and width, and loading themes including folders and music, while also initializing a list of quotes and selecting a random folder and theme for a 25-minute duration. Subsequently, the load theme stage involves loading animation frames in JPG format, scaling them to fit the screen size, playing the theme music, and choosing a random quote to display. The main loop constitutes the core operational phase, where the system handles random events, manages mouse clicks for start/pause functionality, updates the timer when not paused, refreshes animation and background frames, displays the current time and random quote, checks for music end to trigger theme changes, and updates the display at 60 frames per second. Finally, the exit stage concludes the process by stopping PyGame and safely terminating the system. This structured flow ensures a seamless and efficient operation of the Pomodoro timer.

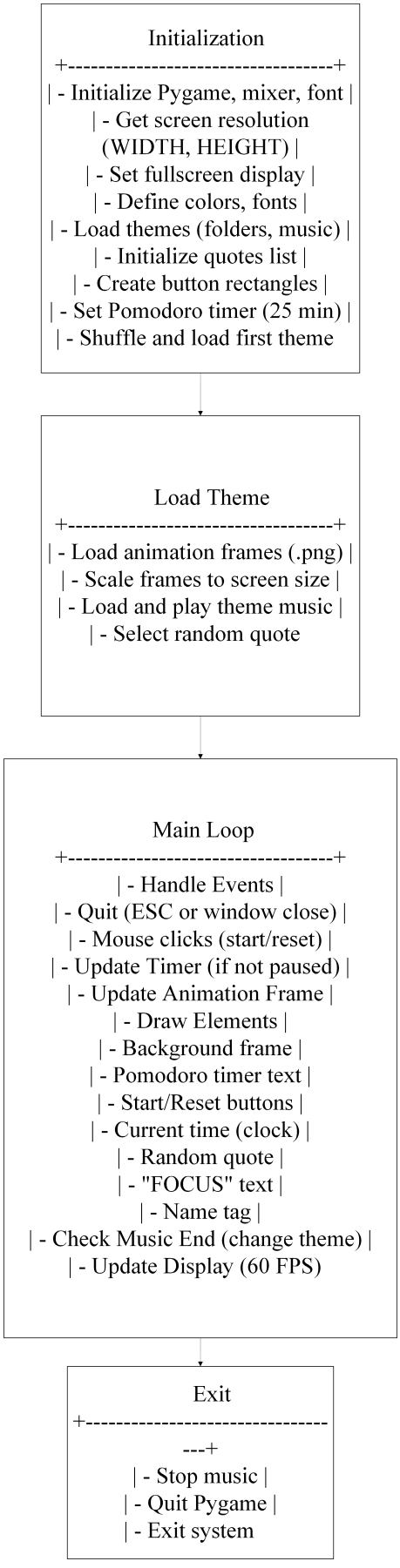

Figure 2.1: Block Diagram of Execution of code flow

2.2 Syntax

Table 2.1: Common Pygame and Python functions used in the Pomodoro Timer project

Function / Method	**Usage / Purpose**
`pygame.init()`	Initializes all imported Pygame modules.
`pygame.mixer.init()`	Initializes the mixer module for handling audio playback.
`pygame.font.init()`	Initializes the font module for rendering text.
`pygame.display.Info()`	Retrieves current display resolution (width and height).
`pygame.display.set_mode()`	Creates a window/screen of specified size (fullscreen in this case).
`pygame.display.set_caption()`	Sets the title of the game window.
`pygame.font.SysFont()`	Loads a system font for rendering text on screen.
`pygame.Rect()`	Defines rectangular areas (used for buttons).
`pygame.draw.rect()`	Draws rectangles on the screen (used for button backgrounds).
`font.render()`	Renders text into an image (Surface) that can be displayed.
`screen.blit()`	Draws a surface (image or text) onto the screen.
`pygame.display.flip()`	Updates the full display surface to the screen.
`pygame.time.get_ticks()`	Gets the number of milliseconds since Pygame was initialized (used for animation).
`clock.tick()`	Limits the frame rate of the game loop (60 FPS here).
`pygame.event.get()`	Returns a list of all user input events (keyboard, mouse, etc.).
`pygame.mixer.music.load()`	Loads a music file for playback.
`pygame.mixer.music.play()`	Starts playing the loaded music file.
`pygame.mixer.music.get_busy()`	Checks if music is currently playing.
`pygame.quit()`	Shuts down all Pygame modules.

`sys.exit()`	Exits the program safely.
`os.listdir()`	Lists all files in a folder (used to load image frames).
`pygame.image.load()`	Loads an image file into a Pygame surface.
`pygame.transform.scale()`	Resizes images to fit the screen dimensions.
`random.choice()`	Selects a random item from a list (used for quotes).
`random.shuffle()`	Randomizes the order of themes.
`time.time()`	Returns the current time in seconds (used for countdown logic).
`datetime.datetime.now()`	Gets the current system date and time (for displaying clock).

3. EXPECTED OUTPUT

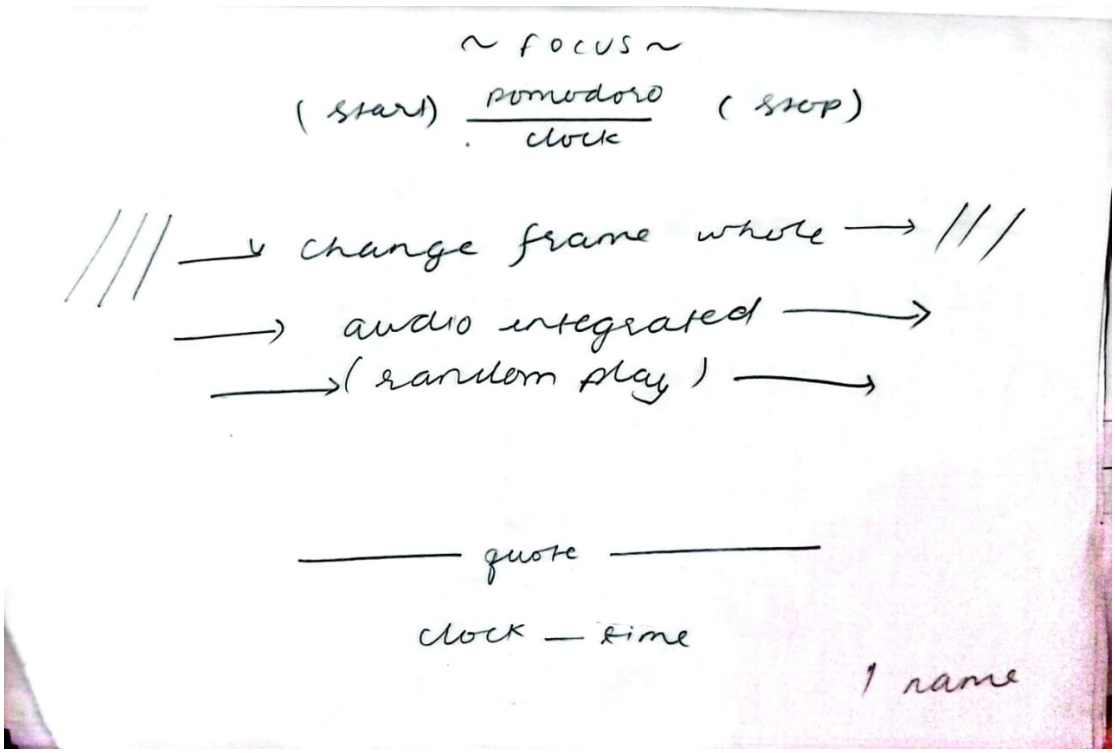

Figure 3.1: Expected output from the code execution

4. OUTPUT

When the program runs, it opens a fullscreen window displaying an animated background based on the selected theme. Overlaid on this background is a large Pomodoro timer showing the countdown in minutes and seconds. Below or near the timer are two rounded buttons labeled ”start” and ”restart” to control the timer. A digital clock showing the current system time is displayed on the screen. An inspirational quote related to focus and productivity appears near the bottom of the screen, updating with each theme change. Additionally, a decorative label ” FOCUS ” and a name tag with ”| Prabhat D Rawal” are shown as part of the interface. Background music corresponding to the theme plays continuously and switches automatically when the music ends. The entire layout adapts dynamically to fit any screen resolution.

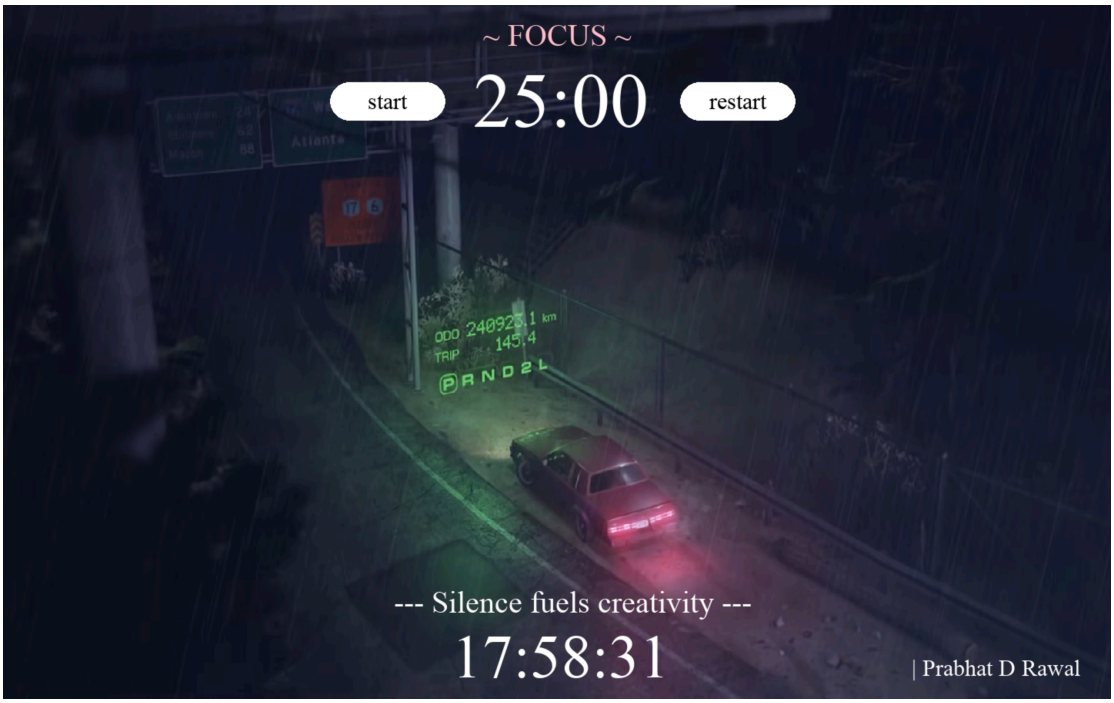

Figure 4.1: Output screen of frame - rainy frame

5. APPENDIX

```
import pygame
import sys
import time
import os
import datetime
import random

# --- Initialization ---
pygame.init()
pygame.mixer.init()
pygame.font.init()

# Screen
info = pygame.display.Info()
WIDTH, HEIGHT = info.current_w, info.current_h # getting infor of
                                    width and height
screen = pygame.display.set_mode((WIDTH, HEIGHT), pygame.FULLSCREEN)
pygame.display.set_caption("Pomodoro + Campfire Themes") # display
                                    caption

# Colors
WHITE = (255, 255, 255)
BUTTON_TEXT = (0, 0, 0)

# Fonts
font_large = pygame.font.SysFont("Times new Roman", 100)
font_button = pygame.font.SysFont("Times new Roman", 30)
clock_font = pygame.font.SysFont("Times new Roman", 80)
quote_font = pygame.font.SysFont("Times new Roman", 40)
focus_font = pygame.font.SysFont("Times new Roman",40)
name_font = pygame.font.SysFont("Times new Roman",30)


# --- Themes ---
themes = [
```

```
    {"folder": "campfire", "music": "camp_fire.mp3"},
    {"folder": "midnight_lake", "music": "panda.mp3"},
    {"folder": "gaming_brother", "music": "Spirited.mp3"},
      {"folder": "rainy_highway", "music": "Sparkle.mp3"},
    {"folder": "snow_fall", "music": "lie.mp3"},
    {"folder": "warm_nights", "music": "naruto.mp3"},
    {"folder": "cyber_cafe", "music": "roses.mp3"},
    {"folder": "bustling_city", "music": "lie2.mp3"}
]

# --- Quotes ---
quotes = [
    "--- A warm fire warms the heart ---",
    "--- Focus is a form of respect ---",
    "--- Silence fuels creativity ---",
    "--- Every second counts ---",
    "--- Burnout happens without balance ---",
    "--- Let the flames carry your thoughts ---",
    "--- Be still like the 'fires glow ---",
    "--- Deep breaths, steady mind ---",
    "--- Flow with focus, not force ---",
    "--- Let stillness sharpen your clarity ---",
    "--- Create slowly, burn brightly ---",
    "--- Rest is part of the rhythm ---",
    "--- The mind clears where the fire crackles ---",
    "--- Calm is your power ---",
    "--- Let your thoughts simmer, not boil ---",
    "--- Drift into focus, like ash in wind ---",
    "--- One moment, fully lived ---",
    "--- Light the fire, not the stress ---",
    "--- In silence, productivity grows ---",
    "--- The ember of effort lights success ---"
]

quote_text = random.choice(quotes)  # initial quote

# Buttons
buttons = {
```

```
    "start": pygame.Rect(WIDTH - 1015, HEIGHT - 800, 150, 50), #
                                        width , height , length
                                        breadth
    "reset": pygame.Rect(WIDTH - 560, HEIGHT - 800, 150, 50)
}

def draw_rounded_button(rect, text, active=False):
    bg_color = (255, 255, 255) if not active else (230, 230, 255)
    pygame.draw.rect(screen, bg_color, rect, border_radius=25)
    text_surface = font_button.render(text, True, BUTTON_TEXT)
    text_rect = text_surface.get_rect(center=rect.center)
    screen.blit(text_surface, text_rect)

# --- Timer ---
POMODORO_TIME = 25 * 60
time_left = POMODORO_TIME
paused = True
last_tick = time.time()

# --- Animation ---
frame_delay = 200
last_update = pygame.time.get_ticks()
current_frame = 0
frames = []
frame_count = 0
theme_index = 0

def load_theme(theme):
    global frames, frame_count, current_frame, quote_text
    folder = theme["folder"]
    music = theme["music"]
    frames = []

    try:
        for f in sorted(os.listdir(folder)):
            if f.endswith('.png'):
                img = pygame.image.load(os.path.join(folder, f)).
                                                    convert_alpha()
                img = pygame.transform.scale(img, (WIDTH, HEIGHT))
```

```
                frames.append(img)
    except FileNotFoundError:
        print(f"[ERROR] Frame folder '{folder}' not found.")
        pygame.quit()
        sys.exit()

    if not frames:
        print(f"[ERROR] No frames in '{folder}'.")
        pygame.quit()
        sys.exit()

    frame_count = len(frames)
    current_frame = 0

    if os.path.isfile(music):
        try:
            pygame.mixer.music.load(music)
            pygame.mixer.music.play()
        except pygame.error as e:
            print(f"[ERROR] Music load failed: {e}")
    else:
        print(f"[ERROR] Music file '{music}' not found.")

    # New quote for each theme
    quote_text = random.choice(quotes) # random choose quotes

# Shuffle and load first theme
random.shuffle(themes)
load_theme(themes[theme_index])

# --- Main Loop ---
clock = pygame.time.Clock()
running = True
while running:
    # --- Events ---
    for event in pygame.event.get():
        if event.type == pygame.QUIT or (
            event.type == pygame.KEYDOWN and event.key == pygame.
                                                    K_ESCAPE
```

```
        ):
            running = False
        elif event.type == pygame.MOUSEBUTTONDOWN and event.button ==
                                                1:
            mouse_pos = pygame.mouse.get_pos()
            if buttons["start"].collidepoint(mouse_pos):
                if time_left <= 0:
                    time_left = POMODORO_TIME
                paused = False
                last_tick = time.time()
            elif buttons["reset"].collidepoint(mouse_pos):
                time_left = POMODORO_TIME
                paused = True

    # --- Timer Update ---
    if not paused:
        current_time = time.time()
        delta = current_time - last_tick
        time_left -= delta
        last_tick = current_time
        if time_left <= 0:
            time_left = POMODORO_TIME
            paused = True

    # --- Animation Frame Update ---
    now = pygame.time.get_ticks()
    if now - last_update > frame_delay:
        current_frame = (current_frame + 1) % frame_count
        last_update = now

    # --- Draw ---
    screen.blit(frames[current_frame], (0, 0))

    # Pomodoro Timer
    minutes = int(time_left) // 60
    seconds = int(time_left) % 60
    time_str = f"{minutes:02}:{seconds:02}"
    timer_surface = font_large.render(time_str, True, WHITE)
```

```
screen.blit(timer_surface, timer_surface.get_rect(center=(WIDTH -
                                    715, HEIGHT - 775)))

# Buttons
draw_rounded_button(buttons["start"], "start") # buttons
                                    development - start
draw_rounded_button(buttons["reset"], "restart") # buttons
                                    development - restart

# Clock
current_time_str = datetime.datetime.now().strftime("%H:%M:%S") #
                                     clock display format
time_surface = clock_font.render(current_time_str, True, WHITE)
screen.blit(time_surface, (WIDTH - 857, HEIGHT - 100)) # clock
                                    location placement

# Quote
quote_surface = quote_font.render(quote_text, True, WHITE)
quote_rect = quote_surface.get_rect(center=(WIDTH - 700, HEIGHT -
                                     125)) # quote placement
screen.blit(quote_surface, quote_rect)

  # focus
focus_surface = focus_font.render("~ FOCUS ~", True, (255,192,203
                                    )) # 255,192,203 - gold shade
                                    type color code
focus_rect = focus_surface.get_rect(center=(WIDTH - 720, HEIGHT -
                                     860)) # original place 720 ,
                                    860
screen.blit(focus_surface, focus_rect)

  # name tag on the surface
name_surface = name_font.render("| Prabhat D Rawal", True, WHITE)
name_rect = name_surface.get_rect(center=(WIDTH - 150, HEIGHT -
                                    40)) # original place 720 ,
                                    860
screen.blit(name_surface, name_rect) # plotting on the screen

pygame.display.flip()
```

```
    clock.tick(60)

    # --- Theme Change on Music End ---
    if not pygame.mixer.music.get_busy():
        theme_index = (theme_index + 1) % len(themes)
        load_theme(themes[theme_index])

# --- Exit ---
pygame.mixer.music.stop()
pygame.quit()
sys.exit()
```