Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama Anggota:

1. **Arsyadana Estu Aziz (121140068)**
2. **Fatur Arkan Syawalva (121140229)**
3. **Alfath Elnandra (121140157)**

Tugas Besar: Final Project

Tanggal: 22 Desember 2024

Mata Kuliah: **Multimedia (IF4021)**

1 Deskripsi Proyek

Proyek ini mengembangkan permainan interaktif menggunakan Pygame yang dikendalikan melalui input visual dari webcam dan input suara dari mikrofon secara real-time. Sistem memproses citra visual menggunakan OpenCV (cv2) untuk mendeteksi gerakan atau isyarat, sementara tingkat kekerasan suara (desibel) dari mikrofon digunakan sebagai parameter untuk mengontrol aksi karakter dalam permainan, seperti melompat atau bergerak, menciptakan pengalaman bermain yang lebih dinamis dan responsif. Lalu diakhir pengerjaan proyek akan ada laporan Tugas Akhir Multimedia yang dibuat menggunakan LateX. [1]

2 Latar Belakang

Kemajuan teknologi digital telah memberikan pengaruh besar pada berbagai bidang, termasuk multimedia dan informatika. Salah satu inovasi yang terus berkembang adalah pengolahan citra digital, yang mencakup manipulasi visual seperti pembuatan filter. Filter visual telah menjadi elemen penting dalam berbagai aplikasi modern, mulai dari media sosial hingga perangkat lunak pengeditan gambar dan video.

Multimedia, yang mencakup teks, gambar, audio, dan video, berfungsi sebagai sarana komunikasi dan ekspresi kreatif. Dalam konteks pengolahan citra, filter digunakan untuk meningkatkan estetika, memberikan efek khusus, atau bahkan membantu analisis data visual. Sebagai contoh, filter dapat digunakan untuk mengubah suasana gambar menjadi lebih dramatis dengan efek warna tertentu, atau membantu meningkatkan visibilitas dalam kondisi gambar yang kurang optimal.

Di sisi lain, informatika menyediakan landasan teknis untuk pengolahan data multimedia. Dalam pembuatan filter, bahasa pemrograman seperti Python menjadi alat yang sangat relevan karena kemudahan penggunaannya dan ketersediaan pustaka pendukung seperti OpenCV, Pillow, dan NumPy. Dengan Python, pengembang dapat menciptakan filter yang beragam, mulai dari efek sederhana seperti grayscale dan blur hingga efek yang kompleks seperti edge detection dan transformasi warna adaptif.

Namun, penerapan teknologi ini juga menghadirkan tantangan, seperti efisiensi algoritma pengolahan citra, kompatibilitas dengan berbagai format media, serta optimalisasi performa untuk aplikasi real-time. Untuk mengatasi tantangan tersebut, diperlukan pendekatan yang mengintegrasikan konsep multimedia dengan kemampuan analitis dan algoritmik dari informatika.

Pembuatan filter dengan Python membuka peluang untuk mengeksplorasi inovasi di bidang pengolahan citra, baik untuk keperluan kreatif maupun fungsional. Dengan memanfaatkan sinergi antara multimedia dan informatika, pengembang dapat menghasilkan solusi yang tidak hanya menarik secara visual tetapi juga efisien secara teknis, sehingga dapat memenuhi kebutuhan yang diinginkan.

3 Teknologi

Teknologi yang digunakan dalam mengerjakan *Final Project* ini yaitu sebagai berikut :

- Python adalah bahasa pemrograman yang ditafsirkan, tingkat tinggi, dan serbaguna. Filosofi desain Python menekankan keterbacaan kode dengan penggunaan spasi yang signifikan. Bahasa ini digunakan sebagai fondasi utama dalam pengembangan logika game dan integrasi berbagai pustaka [2].
- Pygame adalah pustaka Python yang digunakan untuk mengembangkan game 2D. Pustaka ini menangani grafis, input keyboard, suara, dan tampilan antarmuka game, memungkinkan pengembangan sprite karakter, animasi pemain, platform, dan elemen lingkungan secara dinamis [3].
- OpenCV (Open Source Computer Vision Library) adalah pustaka visi komputer sumber terbuka yang digunakan untuk menangkap dan memproses video dari kamera. OpenCV memfasilitasi manipulasi frame video dan integrasi latar belakang visual secara real-time dalam game [4].
- NumPy adalah pustaka Python yang digunakan untuk komputasi ilmiah. Dalam proyek ini, NumPy digunakan untuk memproses data audio dengan efisien dan menghitung volume suara secara real-time.
- PyAudio adalah pustaka Python yang digunakan untuk menangkap dan memproses input audio secara langsung dari mikrofon. Teknologi ini memungkinkan kontrol permainan menggunakan suara.
- Wave adalah pustaka Python yang digunakan untuk merekam dan menyimpan data audio dalam format .wav. Teknologi ini memastikan rekaman audio dapat disimpan dengan format yang mudah diproses.
- Threading adalah teknik pemrograman yang memungkinkan eksekusi beberapa proses secara bersamaan. Dalam proyek ini, threading digunakan untuk menangani rekaman audio dan gameplay secara bersamaan untuk menghindari jeda.
- MoviePy adalah pustaka Python yang digunakan untuk pengeditan video. Teknologi ini memungkinkan penggabungan video dan audio dalam satu file dengan codec libx264 dan audio codec aac untuk hasil akhir yang optimal.
- Time adalah pustaka bawaan Python yang digunakan untuk mengatur durasi dalam berbagai aspek permainan, seperti waktu invincibility karakter dan tampilan pesan di layar.

4 Cara Kerja

Cara kerja pada projek Ambario adalah sebagai berikut :

- Sistem menggunakan perangkat input seperti microphone sebagai media untuk mendeteksi suara dalam permainan.
- Sistem menampilkan lingkungan permainan dalam bentuk dunia 2D dengan tampilan platform.
- Pemain akan bergerak dengan mekanisme pergerakan karakter dari kiri ke kanan menggunakan event handling dari Pygame.
- Program mendeteksi decibel inputan suara dan kemudian mengatur kecepatan vertikal karakter untuk membuat efek lompatan.
- Pemain harus menghindari block plant agar nyawa karakter tidak berkurang atau habis. Jika nyawa karakter habis maka permainan berakhir.
- Pemain harus melompat dengan mengeluarkan suara untuk bisa berpindah platform dan tidak terjatuh. Jikakarakter terjatuh dari platform maka permainan berakhir.
- Permainan akan selesai ketika karakter mencapai kastil di platform terakhir.

5 Penjelasan Kode Program

Kode program dibuat menjadi 1 bagian kode yaitu pada **ambario.py**, Berikut adalah penjelasannya yaitu :

```
1 import pygame
2 import cv2
3 import numpy as np
4 import pyaudio
5 import wave
6 import threading
7 import time
8 from moviepy import VideoFileClip, AudioFileClip
```

Kode 1: Library

- Pygame adalah library Python untuk membuat aplikasi multimedia seperti game atau pemutar audio dan video.
- OpenCV (Open Source Computer Vision Library) digunakan untuk pengolahan gambar dan video.
- Numpy digunakan untuk pengolahan data yang melibatkan perhitungan matematis atau transformasi data dalam bentuk matriks.
- Pyaudio digunakan dalam aplikasi yang membutuhkan input atau output audio langsung, seperti perekaman atau pemutaran suara.
- wave digunakan untuk manipulasi file audio sederhana.
- threading berguna dalam menjalankan beberapa operasi secara paralel, seperti pemrosesan audio dan video secara bersamaan.
- time digunakan bersama dengan threading untuk kontrol waktu.
- MoviePy adalah library untuk manipulasi video dan audio.
- VideoFileClip: Untuk memuat dan memanipulasi file video.

- AudioFileClip: Untuk memuat dan memanipulasi file audio.

```
def combine_audio_video(video_path, audio_path, output_path):
    video = VideoFileClip(video_path)
    audio = AudioFileClip(audio_path)
    final_video = video.with_audio(audio)
    final_video.write_videofile(output_path, codec="libx264", audio_codec="aac")
```

Kode 2: Function File Combine Audio Video

Function ini digunakan untuk menggabungkan file video dan audio menjadi satu file video yang dihasilkan sistem setelah permainan berakhir. Function ini akan dijalankan / digunakan pada diakhir proses sistem dan menyimpan file video dengan nama final-output.avi.

```
class AudioRecorder(threading.Thread):
    def __init__(self, filename="output.wav", rate=44100, frames_per_buffer=1048):
        super(AudioRecorder, self).__init__()
        self.filename = filename
        self.rate = rate
        self.frames_per_buffer = frames_per_buffer
        self.audio_frames = []
        self.volume = 0
        self.running = False

        self.p = pyaudio.PyAudio()
        self.stream = self.p.open(format=pyaudio.paInt16,
                                  channels=1,
                                  rate=self.rate,
                                  input=True,
                                  frames_per_buffer=self.frames_per_buffer)

    def run(self):
        self.running = True
        while self.running:
            try:
                data = self.stream.read(self.frames_per_buffer, exception_on_overflow=False)
                self.audio_frames.append(data)

                # Calculate volume
                audio_data = np.frombuffer(data, dtype=np.int16)
                if len(audio_data) == 0 or np.all(audio_data == 0):
                    self.volume = 0
                else:
                    self.volume = np.linalg.norm(audio_data) / np.sqrt(len(audio_data))

            except Exception as e:
                print("Audio recording error:", e)

    def stop(self):
        self.running = False
        self.stream.stop_stream()
        self.stream.close()
        self.p.terminate()

    def save(self):
        with wave.open(self.filename, 'wb') as wavefile:
            wavefile.setnchannels(1)
            wavefile.setsampwidth(self.p.get_sample_size(pyaudio.paInt16))
            wavefile.setframerate(self.rate)
            wavefile.writeframes(b''.join(self.audio_frames))
```

Kode 3: Class Audio Recorder

- def init : Fungsi ini adalah konstruktor yang diinisialisasi saat objek dari kelas Audio Recorder dibuat. Digunakan untuk menyimpan parameter atibut kelas, menginisiasi PyAudio, dan membuka stream audio untuk perekaman dengan konfigurasi format, kanal, rate, dan ukuran buffer.
- def run : Fungsi ini dipanggil saat thread dijalankan menggunakan metode .start() untuk melakukan perekaman audio secara terus menerus hingga dihentikan (self.running diset menjadi False), membaca audio dari input, menyimpannya di self.audio-frames, dan menghitung volume audio secara real time.
- def stop : Menghentikan proses perekaman dengan mengatur self.running = False, menutup stream audio, dan menghentikan penggunaan PyAudio.
- def save : Menyimpan data audio yang telah direkam ke file WAV.

```
class Block(pygame.sprite.Sprite):
    def __init__(self, x, y):
        super().__init__()
        self.images = [
            pygame.image.load("Model/piranha_frame_1.png").convert_alpha(),
            pygame.image.load("Model/piranha_frame_2.png").convert_alpha()
        ]
        self.image = self.images[0]
        self.rect = self.image.get_rect(midbottom=(x, y))
        self.index = 0

    def update(self):
        self.index += 0.1
        if self.index >= len(self.images):
            self.index = 0
        self.image = self.images[int(self.index)]
```

Kode 4: Class Block Wall

- Kelas Block digunakan untuk membuat objek animasi berbasis gambar dengan dua frame animasi.
- Animasi dicapai dengan secara berkala mengubah gambar berdasarkan nilai index.
- Sprite dapat dipindahkan dan diperbarui menggunakan mekanisme sprite di Pygame, seperti yang dilakukan dengan fungsi update() dan draw() dalam game loop.

```

class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.player_walk = [
            pygame.image.load('Model/Mario - Walk1.gif').convert_alpha(),
            pygame.image.load('Model/Mario - Walk2.gif').convert_alpha(),
            pygame.image.load('Model/Mario - Walk3.gif').convert_alpha()
        ]
        self.player_jump = pygame.image.load("Model/Mario - Jump.gif").convert_alpha()
        self.image = self.player_walk[0]
        self.rect = self.image.get_rect(midbottom=(100, 350))
        self.gravity = 0
        self.player_index = 0
        self.on_ground = True
        self.dead = False
        self.invincible = False
        self.invincible_duration = 2  # seconds
        self.last_hit_time = 0
```

```
20
21      def apply_gravity(self):
22          self.gravity += 1
23          self.rect.y += self.gravity
24
25      def jump(self, jump_force):
26          if self.on_ground:
27              self.gravity = jump_force
28
29      def die(self):
30          self.dead = True
31          self.gravity = -15  # Initial jump force for death animation
32
33      def hit(self):
34          self.invincible = True
35          self.last_hit_time = time.time()
36
37      def update(self):
38          if not self.dead:
39              self.apply_gravity()
40              self.animation_state()
41              if self.invincible and (time.time() - self.last_hit_time) > self.invincible_duration:
42                  self.invincible = False
43          else:
44              self.apply_gravity()
45              self.image = self.player_death
46
47      def animation_state(self):
48          if not self.on_ground:
49              self.image = self.player_jump
50          else:
51              self.player_index += 0.1
52              if self.player_index >= len(self.player_walk):
53                  self.player_index = 0
54              self.image = self.player_walk[int(self.player_index)]
55          self.rect.x += 1  # Move the player to the right
```

Kode 5: Class Player

- Gravitasi: Pemain jatuh secara alami berdasar tinggi lompatan.
- Lompat : Pemain dapat melompat dengan ketinggian yang ditentukan volume suara.
- Mati : Pemain dapat mati setelah terkena serangan.
- Invicible : Pemain juga bisa menjadi invincible sementara setelah terkena serangan, yang memberi perlindungan sementara.
- Animasi: Pemain memiliki animasi untuk berjalan dan melompat. Indeks animasi berjalan diubah setiap frame untuk membuat gerakan halus.
- Pergerakan: Pemain bergerak ke kanan untuk setiap frame.

```
1
2   class Game:
3       def __init__(self):
4           pygame.init()
5           self.screen = pygame.display.set_mode((640, 480))
6           pygame.display.set_caption("Jumping Game with Camera Background")
7           self.clock = pygame.time.Clock()
8           self.running = True
```

```
        self.show_congratulations = False
        self.show_game_over = False
        self.message_start_time = 0
        self.message_duration = 3  # seconds
        self.score = 0
        self.lives = 3

        # Load and resize the platform image
        self.platform_image = pygame.image.load("Model/ground.png").convert_alpha()
        self.platform_image = pygame.transform.scale(self.platform_image, (200, 50))  # Resize to (width, height)
        self.player = pygame.sprite.GroupSingle(Player())

        ## Pipe Image
        self.pipe_image = pygame.image.load("Model/pipe.gif").convert_alpha()
        self.pipe_image = pygame.transform.scale(self.pipe_image, (120, 400))

        ## Ocean image
        self.ocean = pygame.image.load("Model/ocean-1.png").convert_alpha()
        self.ocean = pygame.transform.scale(self.ocean, (640, 480))
        self.ocean_rect = self.ocean.get_rect(topleft=(0, 165))

        ## Castle Image
        self.castle_image = pygame.image.load("Model/Castle.png").convert_alpha()
        self.castle_image = pygame.transform.scale(self.castle_image, (100, 100))


        self.video_cap = cv2.VideoCapture(0)
        self.fourcc = cv2.VideoWriter_fourcc(*"XVID")
        self.out = cv2.VideoWriter("output.avi", self.fourcc, 15, (640, 480))
        self.audio_recorder = AudioRecorder()


        ## Bottom Limit for the Platforms is aroudn 400 since we have Wave that will block the view of the platforms
        self.platform_layouts = [
            {
                "platforms": [(100, 350), (400, 300), (700, 250), (1000, 300), (1300, 250), (1700, 175), (1900, 300), (2220, 350)],
                "blocks": [(450, 300), (780, 250), (1010, 300)]  # Example block positions
            }
        ]

        self.platforms = []
        self.pipes = []
        for layout in self.platform_layouts:
            for pos in layout["platforms"]:
                platform_rect = self.platform_image.get_rect(topleft=pos)
                self.platforms.append(platform_rect)
                pipe_rect = self.pipe_image.get_rect(midtop=(pos[0] + platform_rect.width // 2, pos[1] + platform_rect.height))
                self.pipes.append(pipe_rect)

        self.blocks = pygame.sprite.Group()
        for layout in self.platform_layouts:
            for pos in layout["blocks"]:
                self.blocks.add(Block(pos[0], pos[1]))

        # Place the castle at the end of the last platform
        last_platform = self.platforms[-1]
        self.castle_rect = self.castle_image.get_rect(midbottom=(last_platform.left + last_platform.width // 2, last_platform.top + 5))
```

```
66          self.platform_speed = 5
67
68      def detect_scream(self, volume, threshold=500):
69          if volume > threshold:
70              jump_force = min(-5 - (volume - threshold) / 250, -15)
71              return jump_force
72          return 0
73
74      def overlay_text(self, text, size, color, position):
75          """Overlay text on the screen."""
76          font = pygame.font.Font(None, size)
77          text_surface = font.render(text, True, color)
78          text_rect = text_surface.get_rect(center=position)
79          self.screen.blit(text_surface, text_rect)
80
81      def update_hud(self, volume = 0):
82          """Update the HUD with the current volume, score, and lives."""
83          font = pygame.font.Font(None, 36)
84          volume_text = font.render(f"Volume: {int(volume)}", True, (255, 255, 255))
85          score_text = font.render(f"Score: {self.score}", True, (255, 255, 255))
86          lives_text = font.render(f"Lives: {self.lives}", True, (255, 255, 255))
87
88          self.screen.blit(volume_text, (10, 10))
89          self.screen.blit(score_text, (10, 50))
90          self.screen.blit(lives_text, (10, 90))
91
92
93      def run(self):
94          self.audio_recorder.start()
95
96          countdown_seconds = 3
97          countdown_start_time = time.time()
98          current_volume = 0  # Initialize current_volume
99
100         while self.running:
101             current_time = time.time()
102             elapsed_time = current_time - countdown_start_time
103
104             for event in pygame.event.get():
105                 if event.type == pygame.QUIT:
106                     self.running = False
107
108             ret, frame = self.video_cap.read()
109             if ret:
110                 frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
111                 frame = np.rot90(frame)
112                 self.screen.fill([0, 0, 0])
113                 frame_surface = pygame.surfarray.make_surface(frame)
114                 self.screen.blit(frame_surface, (0, 0))
115
116                 if elapsed_time < countdown_seconds:
117                     self.overlay_text(str(countdown_seconds - int(elapsed_time)), 74, (255, 255,
    255), (320, 240))
118                 else:
119                     current_volume = self.audio_recorder.volume
120                     jump_force = self.detect_scream(current_volume)
121                     if jump_force:
122                         self.player.sprite.jump(jump_force)
123
124                     self.player.update()
125                     self.blocks.update()
126
```

```
                    for platform in self.platforms:
                        platform.x -= self.platform_speed

                    for pipe in self.pipes:
                        pipe.x -= self.platform_speed

                    for block in self.blocks:
                        block.rect.x -= self.platform_speed

                    self.castle_rect.x -= self.platform_speed

                    player_rect = self.player.sprite.rect
                    self.player.sprite.on_ground = False
                    for platform in self.platforms:
                        if player_rect.colliderect(platform):
                            if player_rect.bottom > platform.top and player_rect.top < platform.top
:
                                player_rect.bottom = platform.top
                                self.player.sprite.on_ground = True
                                self.player.sprite.gravity = 0
                            elif player_rect.top < platform.bottom and player_rect.bottom >
platform.bottom:
                                player_rect.top = platform.bottom
                            elif player_rect.right > platform.left and player_rect.left < platform.
left:
                                player_rect.right = platform.left
                            elif player_rect.left < platform.right and player_rect.right > platform
.right:
                                player_rect.left = platform.right

                    # Check collision with blocks
                    for block in self.blocks:
                        if player_rect.colliderect(block.rect):
                            print("Collision with block!")
                            if not self.player.sprite.invincible:
                                self.player.sprite.hit()
                                self.lives -= 1
                                if self.lives <= 0:
                                    self.player.sprite.die()
                                    self.show_game_over = True
                                    self.message_start_time = time.time()
                                    self.running = False

                    # Check if player falls off the screen
                    if player_rect.top > self.screen.get_height():
                        print("Player fell off the screen!")
                        self.player.sprite.die()
                        self.lives -= 1
                        self.show_game_over = True
                        self.message_start_time = time.time()
                        self.running = False

                    self.player.draw(self.screen)
                    for platform in self.platforms:
                        self.screen.blit(self.platform_image, platform)
                    for pipe in self.pipes:
                        self.screen.blit(self.pipe_image, pipe)
                    self.blocks.draw(self.screen)

                    # Draw the castle
                    self.screen.blit(self.castle_image, self.castle_rect)

```

```
                    # Draw the ocean
                    self.screen.blit(self.ocean, self.ocean_rect)

                    # Check collision with castle
                    if player_rect.colliderect(self.castle_rect):
                        print("Congratulations! You've reached the castle!")
                        self.show_congratulations = True
                        self.message_start_time = time.time()
                        self.running = False

                    # Update the score based on distance traveled
                    self.score += 1

                if self.show_congratulations:
                    self.overlay_text("Congratulations!", 74, (255, 255, 255), (320, 240))
                    if (current_time - self.message_start_time) > self.message_duration:
                        self.show_congratulations = False

                if self.show_game_over:
                    self.overlay_text("Game Over", 74, (255, 0, 0), (320, 240))
                    if (current_time - self.message_start_time) > self.message_duration:
                        self.show_game_over = False

                # Update the HUD
                self.update_hud(current_volume)

                pygame.display.update()

                frame_for_video = np.array(pygame.surfarray.pixels3d(self.screen))
                frame_for_video = np.transpose(frame_for_video, (1, 0, 2))
                frame_for_video = cv2.cvtColor(frame_for_video, cv2.COLOR_RGB2BGR)
                self.out.write(frame_for_video)

                self.clock.tick(15)

        # Ensure the final message is displayed for the specified duration
        end_time = time.time()
        while (time.time() - end_time) < self.message_duration:
            if self.show_congratulations:
                self.overlay_text("Congratulations!", 74, (255, 255, 255), (320, 240))
            if self.show_game_over:
                self.overlay_text("Game Over", 74, (255, 0, 0), (320, 240))
            pygame.display.update()
            frame_for_video = np.array(pygame.surfarray.pixels3d(self.screen))
            frame_for_video = np.transpose(frame_for_video, (1, 0, 2))
            frame_for_video = cv2.cvtColor(frame_for_video, cv2.COLOR_RGB2BGR)
            self.out.write(frame_for_video)
            self.clock.tick(15)

        self.audio_recorder.stop()
        self.audio_recorder.save()
        self.video_cap.release()
        self.out.release()

        # Combine audio and video
        combine_audio_video("output.avi", "output.wav", "final_output.avi")

        pygame.quit()

if __name__ == "__main__":
    game = Game()
    game.run()
```

```
247
248
```

Kode 6: Class Game

6 Hasil Analisis

6.1 Keakuratan

Keakuratan filter dalam mendeteksi perubahan audio dan mengubahnya menjadi aksi dalam video (misalnya, memicu animasi lompatan pada pemain) cukup tinggi. Pengukuran volume audio yang dilakukan oleh filter berhasil memproses variasi suara dengan tepat, menunjukkan respons yang sesuai pada setiap perubahan intensitas suara. Namun, terdapat sedikit penurunan keakuratan pada suara dengan frekuensi atau volume yang sangat rendah, yang terkadang menyebabkan ketidakakuratan dalam interpretasi aksi yang dihasilkan.

6.2 Proses Data

Proses data berjalan dengan lancar, namun membutuhkan perhatian khusus terhadap pemrosesan audio dan video secara bersamaan. Pengolahan audio dilakukan dalam waktu nyata dengan menggunakan perangkat keras yang memadai, yang memungkinkan filter untuk membaca dan mengonversi data audio dengan cepat. Proses penggabungan data audio dan video menggunakan MoviePy juga berjalan sesuai rencana, dengan hasil akhir berupa video yang menyertakan suara dan visual yang disinkronkan dengan baik.

6.3 Performa Sistem

Performa sistem dapat dikatakan baik, dengan tingkat responsivitas yang tinggi terhadap perubahan input, baik pada data audio maupun video. Sistem mampu memproses audio dengan real-time feedback, menghasilkan video yang terupdate dengan setiap perubahan volume audio

6.4 Kendala

Meskipun filter berhasil berfungsi dengan baik, beberapa kendala muncul selama pengujian dan implementasi. Salah satu kendala terbesar adalah keterbatasan dalam mengelola dua media (audio dan video) secara bersamaan dalam waktu nyata. Penggabungan audio dan video menggunakan MoviePy berjalan dengan baik, namun untuk aplikasi lebih lanjut, terutama dengan kualitas video tinggi, beberapa masalah seperti lag dan desinkronisasi terkadang muncul.

7 Kesimpulan

Berdasarkan hasil dan analisis yang dilakukan, filter yang dikembangkan berhasil memproses data audio dan video yang sesuai dengan tujuan awal. Filter ini mampu memberikan dua output berupa audio dan video, yang kemudian berhasil digabungkan menggunakan library MoviePy. Proses penggabungan ini menunjukkan bahwa sistem mampu mendukung workflow yang melibatkan data dengan format yang berbeda.

Kinerja filter menunjukkan respons yang sangat sensitif terhadap perubahan data input. Hal ini terlihat dari kemampuan filter untuk mendeteksi perubahan volume audio secara real-time dan menyesuaikan animasi dengan cepat. Responsivitas ini menjadi salah satu keunggulan utama filter.

Namun, terdapat beberapa tantangan yang dihadapi selama pengembangan. Salah satu upaya untuk mencoba metode atau pendekatan lain ternyata sulit direalisasikan, baik karena keterbatasan teknis maupun karena ketidaksesuaian dengan framework yang digunakan. Meskipun demikian, keterbatasan

ini menjadi pelajaran berharga untuk mempertimbangkan alternatif teknologi atau metode yang lebih sesuai di masa depan.

Secara keseluruhan, filter ini menunjukkan potensi yang baik untuk diterapkan dalam proyek-proyek berbasis multimedia interaktif dan memiliki ruang untuk pengembangan lebih lanjut. Optimalisasi tambahan, terutama pada proses penggabungan data, dapat dilakukan untuk meningkatkan efisiensi dan kualitas output.

References

[1] Y. Name, "Tutorial belajar LaTeX dasar untuk pemula," https://www.sains.web.id/2018/12/tutorial-belajar-latex.html, 18 Dec. 2018, accessed: 2024-12-21.

[2] R. S. A Bogdanchikov1, M Zhaparov1, "Python to learn programming," https://iopscience.iop.org/article/10.1088/1742-6596/423/1/012027/pdf, 2013, accessed: 2024-12-21.

[3] S. G. C. G. N. P. M. R. K. Piyush N Shinde1, Yash J Chavan2, "Pygame: Develop games using python," https://www.academia.edu/download/100115021/fileserve.pdf, Jun. 2021, accessed: 2024-12-21.

[4] D. J. S. Ujjwal Sharma*, Tanya Goel, "Real-time image processing usingdeeplearningwithopencvand-python," https://pnrjournal.com/index.php/home/article/view/8539, 2023, accessed: 2024-12-21.