



Nama: **Ashoka Tatang (122140051), Dimas Dharma (122140215), Yohanes Christian (122140217)**

Tugas Ke: **Final Project**

Mata Kuliah: **Sistem/Teknologi Multimedia (IF4021)**

Tanggal: 31 Mei 2025

1 Deskripsi Projek

Proyek ini adalah sebuah permainan tembak-menembak interaktif untuk dua pemain yang dikendalikan dengan kedipan mata dan gestur tangan menggunakan webcam. Pemain dapat menembakkan "peluru" dengan berkedip dan mengaktifkan "perisai" dengan gestur jempol.

2 Teknologi

Dalam proyek ini digunakan beberapa library dari python yang esensial dalam menjalankan program. Berikut merupakan library yang digunakan :

- cv2 (OpenCV): Digunakan untuk mengambil video dari webcam, memproses frame video (misalnya membalik atau mengubah ukuran), dan menampilkan antarmuka permainan.
- mediapipe: Kerangka kerja lintas platform untuk membangun alur kerja pembelajaran mesin, digunakan di sini untuk deteksi dan pelacakan landmark wajah (untuk mata dan hidung) dan tangan.
- numpy: Library untuk komputasi numerik, khususnya untuk operasi array, digunakan dalam perhitungan rasio aspek mata dan penanganan mask gambar.
- time: Library untuk fungsi terkait waktu, seperti mengukur cooldown kedipan dan durasi perisai.
- overlay (modul kustom): Modul ini berisi fungsi untuk menempatkan gambar (seperti sprite pemain, peluru, dan perisai) di atas frame video secara transparan.
- utils (modul kustom): Modul ini berisi fungsi-fungsi pembantu seperti perhitungan rasio aspek mata, deteksi gestur tangan, dan penggambaran healthbar.

3 Cara Kerja

Cara kerja pada proyek Final Project ini adalah sebagai berikut:

- Sistem menggunakan kamera webcam untuk menangkap video secara langsung.
- Kamera membagi menjadi 2 section untuk player 1 dan player 2 dan memberi batas agar user tidak melewati boundary line
- Sistem menggunakan library MediaPipe Face Mesh untuk mendeteksi landmark wajah, khususnya di sekitar mata, untuk menghitung Eye Aspect Ratio (EAR).

- Berdasarkan EAR, sistem mendeteksi kedipan mata pengguna. Jika mata terdeteksi berkedip dan memenuhi kondisi cooldown, proyektil ditembakkan dari posisi wajah pemain.
- Sistem juga menggunakan library MediaPipe Hands untuk mendeteksi tangan dan melacak posisi landmark jari.
- Berdasarkan posisi landmark tangan, sistem mendeteksi gestur tangan tertentu (misalnya, thumbs up).
- Jika gestur thumbs up terdeteksi, shield untuk pemain yang bersangkutan akan diaktifkan untuk durasi tertentu, memberikan perlindungan dari proyektil lawan.
- Posisi hidung dari deteksi face mesh digunakan untuk menentukan posisi horizontal dan vertikal karakter pemain di layar.
- Proyektil bergerak melintasi layar, dan sistem akan mendeteksi tabrakan antara proyektil dan pemain lawan. Jika terjadi tabrakan dan shield tidak aktif, health pemain akan berkurang.
- Healthbar pemain ditampilkan di layar untuk menunjukkan kondisi health mereka secara real-time.

4 Penjelasan Kode Program

Kode program terbagi menjadi tiga file: **main.py**, **overlay.py**, **menu-manager.py**, dan **utils.py**. Berikut adalah penjelasan dari masing-masing bagian kode:

4.1 main.py

main.py berisi kode yang menjadi fondasi utama program, pada kode ini program dijalankan kemudian akan memanggil fungsi yang dibutuhkan dari file pythona lainnya.

Kode 1: library pada Main.py

```
import cv2
import mediapipe as mp
import numpy as np
import threading
import time
from overlay import overlay_transparent
from utils import (
    eye_aspect_ratio,
    detect_hand_gesture,
    draw_healthbar,
)
import menu_manager
```

4.1.1 Penjelasan Library

- **cv2 (OpenCV-python)** : Digunakan untuk menangkap video dari *webcam*. OpenCV adalah library python yang umum digunakan dalam pemrosesan gambar dan video secara realtime.
- **mediapipe** : Dalam program **Evader** digunakan untuk facial landmark sebagai peletakan pesawat dan hand tracking untuk mendeteksi gestur tangan player.
- **numpy** : Digunakan untuk indexing landmark, dan perhitungan scale UI, Screen, Collision Box dan lain-lain.

- **threading** : Modul ini memungkinkan program untuk menjalankan beberapa bagian kode secara bersamaan dalam thread yang berbeda.
- **time** : Modul ini menyediakan berbagai fungsi terkait waktu. Dalam game ini, time digunakan untuk Mmengelola cooldown tembakan dan perisai, memastikan ada jeda waktu sebelum aksi yang sama bisa diulang dan juga mengukur durasi tampilan instruksi di awal game.
- **overlay** :
- **utils** :
- **menu manager** :

4.1.2 Fungsi rotate_image_alpha

Fungsi ini bertanggung jawab untuk memutar gambar sambil mempertahankan saluran alfa (transparansi). Ini sangat penting untuk gambar seperti pesawat ruang angkasa atau peluru yang memiliki area transparan dan perlu diputar tanpa kehilangan informasi transparansinya, sehingga dapat di-overlay dengan benar di atas latar belakang video.

Kode 2: rotate_image_alpha

```
def rotate_image_alpha(image, angle):
    h, w = image.shape[:2]
    center = (w // 2, h // 2)
    rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1.0)
    cos = np.abs(rotation_matrix[0, 0])
    sin = np.abs(rotation_matrix[0, 1])
    new_w = int((h * sin) + (w * cos))
    new_h = int((h * cos) + (w * sin))
    rotation_matrix[0, 2] += (new_w / 2) - center[0]
    rotation_matrix[1, 2] += (new_h / 2) - center[1]
    rotated = cv2.warpAffine(image, rotation_matrix, (new_w, new_h),
                             borderMode=cv2.BORDER_CONSTANT, borderValue=(0, 0, 0, 0))
    return rotated
```

4.1.3 Fungsi handle_mouse_event

Ini adalah fungsi callback untuk menangani event mouse di jendela OpenCV. Peran utamanya adalah mendeteksi kapan tombol mouse kiri diklik (cv2.EVENT__LBUTTONDOWN) dan memeriksa apakah klik tersebut terjadi di atas area tombol "REPLAY" atau "CLOSE" yang ditampilkan di layar. Jika ya, ia akan mengatur flag global (_restart_game_flag atau _exit_game_flag) yang kemudian akan diproses di main loop untuk mengubah status game atau keluar dari game.

Kode 3: handle_mouse_event

```
def handle_mouse_event(event, x, y, flags, param):
    global _restart_game_flag, _exit_game_flag, replay_button_rect, close_button_rect
    if event == cv2.EVENT_LBUTTONDOWN:
        # Check if replay button was clicked
        if replay_button_rect and \
            x >= replay_button_rect[0] and x <= replay_button_rect[0] + \
                replay_button_rect[2] and \
            y >= replay_button_rect[1] and y <= replay_button_rect[1] + \
                replay_button_rect[3]:
            _restart_game_flag = True
```

```
# Check if close button was clicked
if close_button_rect and \
    x >= close_button_rect[0] and x <= close_button_rect[0] + close_button_rect[2]
    and \
    y >= close_button_rect[1] and y <= close_button_rect[1] + close_button_rect[3]:
    _exit_game_flag = True
```

4.1.4 Fungsi activate__shield

Fungsi ini digunakan untuk mengaktifkan perisai (shield) untuk pemain tertentu. Ini memeriksa apakah perisai sudah aktif dan apakah cooldown perisai telah berlalu. Jika kondisi terpenuhi, ia akan mengatur flag shield_active_player1 atau shield_active_player2 menjadi True dan memulai thread baru untuk memanggil fungsi deactivate__shield setelah durasi tertentu.

Kode 4: activate__shield

```
def activate_shield(player_id):
    global shield_active_player1, shield_active_player2, \
        last_shield_deactivation_time_p1, last_shield_deactivation_time_p2
    current_time = time.time()
    if player_id == "Player 1":
        if not shield_active_player1 and \
            (current_time - last_shield_deactivation_time_p1 >= 5.0):
            shield_active_player1 = True
            threading.Thread(target=deactivate_shield, args=("Player 1",)).start()
    elif player_id == "Player 2":
        if not shield_active_player2 and \
            (current_time - last_shield_deactivation_time_p2 >= 5.0):
            shield_active_player2 = True
            threading.Thread(target=deactivate_shield, args=("Player 2",)).start()
```

4.1.5 Fungsi deactivate__shield

Fungsi ini adalah penonaktif perisai. Setelah jangka waktu yang ditentukan (3 detik), ia akan mengatur flag perisai pemain yang sesuai menjadi False dan mencatat waktu penonaktifan perisai terakhir. Ini penting untuk mengimplementasikan cooldown perisai.

Kode 5: deactivate__shield

```
def deactivate_shield(player_id):
    global shield_active_player1, shield_active_player2, \
        last_shield_deactivation_time_p1, last_shield_deactivation_time_p2
    current_time = time.time()
    if player_id == "Player 1":
        if not shield_active_player1 and \
            (current_time - last_shield_deactivation_time_p1 >= 5.0):
            shield_active_player1 = False
            last_shield_deactivation_time_p1 = current_time
            threading.Thread(target=activate_shield, args=("Player 1",)).start()
    elif player_id == "Player 2":
        if not shield_active_player2 and \
            (current_time - last_shield_deactivation_time_p2 >= 5.0):
            shield_active_player2 = False
            last_shield_deactivation_time_p2 = current_time
            threading.Thread(target=activate_shield, args=("Player 2",)).start()
```

4.1.6 Fungsi main

Ini adalah fungsi inti dari seluruh aplikasi game. Ini mengelola alur permainan secara keseluruhan. Berikut adalah peran-peran kuncinya:

- Inisialisasi Kamera dan Jendela: Memulai pengambilan video dari kamera dan membuat jendela display.
- Pengaturan Game Awal: Mengatur status game ke "PLAYING", menginisialisasi kesehatan pemain, status perisai, dan array untuk proyektil dan posisi pemain.
- Loop Game Utama: Ini adalah loop while True yang terus-menerus:
 - Logika Gameplay:
 - * Pergerakan Proyektil: Memperbarui posisi semua proyektil di layar.
 - * Deteksi Wajah dan Pemain: Mengidentifikasi wajah pemain, menentukan siapa Player 1 dan Player 2 berdasarkan posisi hidung relatif terhadap garis tengah layar.
 - * Kontrol Pemain (Kedipan): Menghitung Eye Aspect Ratio (EAR) untuk mendeteksi kedipan mata. Jika kedipan terdeteksi dan cooldown menembak sudah berlalu, proyektil baru akan ditambahkan ke layar.
 - * Deteksi Gerakan Tangan: Menggunakan `detect_hand_gesture` (dari `utils.py`) untuk mendeteksi gestur tangan (misalnya, jempol ke atas) dan memanggil `activate_shield` jika gestur yang sesuai terdeteksi.
 - * Deteksi Tabrakan: Memeriksa tabrakan antara proyektil dan pesawat pemain. Jika terjadi tabrakan dan perisai tidak aktif, kesehatan pemain yang terkena akan berkurang.
 - * Status Kemenangan: Memeriksa apakah kesehatan salah satu pemain telah mencapai nol, yang mengarah ke perubahan status game menjadi "WINNER".
 - Penggambaran Visual:
 - * Menggambar semua proyektil yang aktif.
 - * Menggambar perisai jika aktif di sekitar pemain.
 - * Menggambar pesawat pemain di posisi yang terdeteksi.
 - * Menggambar *health bar* untuk kedua pemain.
 - * Menggambar garis pembagi tengah di layar.
 - * Menampilkan instruksi awal permainan dengan efek *fade-out*.
 - Layar Pemenang: Jika `current_game_state` adalah `GAME_STATE_WINNER`:
 - * Menggambar gambar "WINNER" dan teks "X WINS!".
 - * Menggambar tombol "REPLAY" dan "CLOSE" dan menyimpan posisi mereka untuk interaksi mouse.
 - Input Pengguna: Memproses penekanan tombol keyboard (misalnya, ESC untuk keluar) dan flag yang diatur oleh `handle_mouse_event` untuk me-restart atau keluar dari game.
- Pembersihan: Melepaskan kamera dan menutup semua jendela OpenCV saat loop berakhir.

Kode 6: fungsi main

```
def main():
    global _restart_game_flag, _exit_game_flag, replay_button_rect, close_button_rect
    cap = cv2.VideoCapture(0)
    if not cap.isOpened():
```

```

    raise IOError("Tidak dapat membuka kamera.")
cv2.namedWindow("EVADER")
cv2.setMouseCallback("EVADER", handle_mouse_event)
current_game_state = GAME_STATE_PLAYING
winner_player_id = None
last_blink_time = [0, 0]
eye_ready_to_blink = [0, 0]
projectiles = []
global health_player1, health_player2, shield_active_player1, shield_active_player2
health_player1 = 100
health_player2 = 100
shield_active_player1 = False
shield_active_player2 = False
player_positions = [None, None]
show_instructions_start_time = time.time()
instruction_display_duration = 10
instruction_text = [
    "HOW TO PLAY :",
    "1. Area player dipisah dengan garis tengah.",
    "2. Player 1 (Kiri) aktif dengan jempol kiri.",
    "3. Player 2 (Kanan) aktif dengan jempol kanan.",
    "4. Berkedip untuk menembak.",
    "5. Gerakkan kepala untuk menghindari."
]

while True:
    ret, frame = cap.read()
    if not ret:
        break

    frame = cv2.flip(frame, 1)
    frame = cv2.resize(frame, RESIZED_FRAME_DIMENSIONS)
    ih, iw = frame.shape[:2]
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results_face = face_mesh.process(rgb_frame)
    results_hands = hands.process(rgb_frame)
    current_time = time.time()

    if current_game_state == GAME_STATE_PLAYING:
        # Gameplay logic
        new_projectiles = []
        for proj in projectiles:
            proj['x'] += 15 if proj['player'] == "Player 1" else -15
            if 0 <= proj['x'] + proj['w'] and proj['x'] <= iw:
                new_projectiles.append(proj)
        projectiles = new_projectiles

        if len(projectiles) > MAX_PROJECTILES:
            projectiles.pop(0)

        if results_face.multi_face_landmarks:
            detected_players = []
            for face_landmarks in results_face.multi_face_landmarks[:2]:
                nose = face_landmarks.landmark[NOSE_IDX]
                if nose.x < 0 or nose.x > 1 or nose.y < 0 or nose.y > 1:
                    continue

```

```

# Determine player ID based on nose position relative to the center line
player_id = 0 if nose.x * iw < SCREEN_CENTER_X else 1
target_player_w = PLAYER_TARGET_W
target_player_h = PLAYER1_TARGET_H if player_id == 0 else PLAYER2_TARGET_H

# Adjust X position for better placement
if player_id == 0: # Player 1 (left)
    player_x = int(nose.x * iw) - int(target_player_w * 0.7)
else: # Player 2 (right)
    player_x = int(nose.x * iw) - int(target_player_w * 0.3)
player_y = int(nose.y * ih) - target_player_h // 2

# Ensure only one player is detected per side if two faces are detected
if player_id not in [p["id"] for p in detected_players]:
    detected_players.append({
        "id": player_id,
        "x": player_x,
        "y": player_y,
        "landmarks": face_landmarks
    })

# Update player positions
for player in detected_players:
    player_id = player["id"]
    player_positions[player_id] = {"x": player["x"], "y": player["y"]}
    left_ear = eye_aspect_ratio(player["landmarks"].landmark, LEFT_EYE_IDX,
                                iw, ih)
    right_ear = eye_aspect_ratio(player["landmarks"].landmark, RIGHT_EYE_IDX,
                                iw, ih)
    avg_ear = (left_ear + right_ear) / 2.0
    blinking = avg_ear < BLINK_THRESHOLD
    eyes_open = avg_ear > OPEN_THRESHOLD

    if eyes_open:
        eye_ready_to_blink[player_id] = 1

    if blinking and eye_ready_to_blink[player_id] == 1 and current_time -
        last_blink_time[player_id] >= BLINK_COOLDOWN:
        ammo_rotation_angle = -90 if player_id == 0 else 90
        rotated_ammo = rotate_image_alpha(cv2.resize(ammo_img_raw, (AMMO_W,
            AMMO_H), interpolation=cv2.INTER_AREA), ammo_rotation_angle)
        rotated_ammo_h, rotated_ammo_w = rotated_ammo.shape[:2]

# Position the projectile at the tip of the spaceship
if player_id == 0:
    ammo_start_x = player["x"] + PLAYER_TARGET_W - (rotated_ammo_w // 2)
else: # Player 2 shooting to the left
    ammo_start_x = player["x"] - (rotated_ammo_w // 2)
ammo_start_y = player["y"] + (PLAYER1_TARGET_H if player_id == 0 else
    PLAYER2_TARGET_H) // 2 - (rotated_ammo_h // 2)

projectiles.append({
    'x': ammo_start_x,
    'y': ammo_start_y,
    'w': rotated_ammo_w,
    'h': rotated_ammo_h,
    'img': rotated_ammo,

```

```

        'player': f"Player {player_id + 1}"
    })
    last_blink_time[player_id] = current_time
    eye_ready_to_blink[player_id] = 0

# Detect hand gestures
if results_hands.multi_hand_landmarks:
    for hand_landmarks in results_hands.multi_hand_landmarks:
        gesture = detect_hand_gesture(hand_landmarks)
        hand_x_normalized = hand_landmarks.landmark[mp_hands.HandLandmark.WRIST].x
        if gesture == "thumbs_up":
            activate_shield("Player 1" if hand_x_normalized < 0.5 else "Player 2")

# Draw projectiles and detect collisions
for i in range(len(projectiles) - 1, -1, -1):
    proj = projectiles[i]
    frame = overlay_transparent(frame, proj['img'], proj['x'], proj['y'],
                                (proj['w'], proj['h']))
    collision_occurred = False

for idx, position in enumerate(player_positions):
    if position is not None:
        player_x, player_y = position["x"], position["y"]
        player_w_check = PLAYER_TARGET_W
        player_h_check = PLAYER1_TARGET_H if idx == 0 else PLAYER2_TARGET_H

        # Collision logic
        player_left = player_x
        player_top = player_y
        player_right = player_x + player_w_check
        player_bottom = player_y + player_h_check

        proj_left = proj['x']
        proj_top = proj['y']
        proj_right = proj['x'] + proj['w']
        proj_bottom = proj['y'] + proj['h']

        if (proj_left < player_right and proj_right > player_left and
            proj_top < player_bottom and proj_bottom > player_top):
            if proj['player'] != f"Player {idx + 1}":
                if (idx == 0 and shield_active_player1) or (idx == 1 and
                    shield_active_player2):
                    collision_occurred = True
                    break
            else:
                if idx == 0:
                    health_player1 -= 10
                else:
                    health_player2 -= 10
                collision_occurred = True
                if health_player1 <= 0:
                    current_game_state = GAME_STATE_WINNER
                    winner_player_id = "Player 2"
                elif health_player2 <= 0:
                    current_game_state = GAME_STATE_WINNER
                    winner_player_id = "Player 1"
                break

```



```

        if collision_occurred:
            projectiles.pop(i)

# Draw shields if active
for idx, position in enumerate(player_positions):
    if position is not None:
        player_x, player_y = position["x"], position["y"]
        player_w_shield = PLAYER_TARGET_W
        player_h_shield = PLAYER1_TARGET_H if idx == 0 else PLAYER2_TARGET_H

        shield_scale_factor = 1.3
        scaled_shield_w = int(player_w_shield * shield_scale_factor)
        scaled_shield_h = int(player_h_shield * shield_scale_factor)

        shield_draw_x = player_x - (scaled_shield_w - player_w_shield) // 2
        shield_draw_y = player_y - (scaled_shield_h - player_h_shield) // 2

        if idx == 0 and shield_active_player1:
            frame = overlay_transparent(frame, shield_img, shield_draw_x,
                                         shield_draw_y, (scaled_shield_w, scaled_shield_h))
        elif idx == 1 and shield_active_player2:
            frame = overlay_transparent(frame, shield_img, shield_draw_x,
                                         shield_draw_y, (scaled_shield_w, scaled_shield_h))

# Draw players and health bars
for idx, position in enumerate(player_positions):
    if position is not None:
        player_x, player_y = position["x"], position["y"]
        player_to_draw_img = player1_rotated if idx == 0 else player2_rotated
        player_w_draw = PLAYER_TARGET_W
        player_h_draw = PLAYER1_TARGET_H if idx == 0 else PLAYER2_TARGET_H
        frame = overlay_transparent(frame, player_to_draw_img, player_x, player_y,
                                     (player_w_draw, player_h_draw))
        health = health_player1 if idx == 0 else health_player2
        frame = draw_healthbar(frame, f"Player {idx + 1}", health, player_x,
                               player_y - 20)

# Draw dividing line between players
cv2.line(frame, (SCREEN_CENTER_X, 0), (SCREEN_CENTER_X, ih), (255, 255, 255), 2)

# Draw instructions with fade-out effect
elapsed_time = current_time - show_instructions_start_time
if current_game_state == GAME_STATE_PLAYING and elapsed_time <
    instruction_display_duration:
    alpha = 1.0
    if elapsed_time > instruction_display_duration - 2:
        alpha = 1.0 - (elapsed_time - (instruction_display_duration - 2)) / 2.0
        alpha = max(0, min(1, alpha))

    font = cv2.FONT_HERSHEY_SIMPLEX
    font_scale_base = ih / 480
    font_scale = font_scale_base * 0.5
    thickness = 1
    line_height = int(font_scale * 40)
    max_text_w = 0
    total_text_h = 0

```

```

for line in instruction_text:
    (text_w, text_h), baseline = cv2.getTextSize(line, font, font_scale,
        thickness)
    max_text_w = max(max_text_w, text_w)
    total_text_h += (text_h + baseline + 10)

start_x = (iw - max_text_w) // 2
start_y = (ih - total_text_h) // 2
y_offset = start_y

for line in instruction_text:
    (text_w, text_h), baseline = cv2.getTextSize(line, font, font_scale,
        thickness)
    x_pos = start_x + (max_text_w - text_w) // 2
    text_color = (255, 255, 255)
    text_color_alpha = (int(text_color[0] * alpha), int(text_color[1] * alpha),
        int(text_color[2] * alpha))
    cv2.putText(frame, line, (x_pos, y_offset), font, font_scale,
        text_color_alpha, thickness, cv2.LINE_AA)
    y_offset += (text_h + baseline + 10)

# Winner state display
if current_game_state == GAME_STATE_WINNER:
    projectiles.clear()
    winner_scale_factor = 0.5
    winner_display_w = int(winner_img_raw.shape[1] * winner_scale_factor)
    winner_display_h = int(winner_img_raw.shape[0] * winner_scale_factor)
    winner_scaled_img = cv2.resize(winner_img_raw, (winner_display_w,
        winner_display_h), interpolation=cv2.INTER_AREA)

    if winner_player_id == "Player 1":
        winner_x = (SCREEN_CENTER_X // 2) - (winner_display_w // 2)
    else:
        winner_x = SCREEN_CENTER_X + (SCREEN_CENTER_X // 2) - (winner_display_w // 2)

    winner_y = int(ih * 0.1)
    frame = overlay_transparent(frame, winner_scaled_img, winner_x, winner_y,
        (winner_display_w, winner_display_h))

    winner_text = f"{winner_player_id} WINS!"
    font = cv2.FONT_HERSHEY_SIMPLEX
    text_scale = ih / 480 * 1.0
    text_thickness = 2
    (text_w, text_h), baseline = cv2.getTextSize(winner_text, font, text_scale,
        text_thickness)
    text_x = winner_x + (winner_display_w // 2) - (text_w // 2)
    text_y = winner_y + winner_display_h + 30
    cv2.putText(frame, winner_text, (text_x, text_y), font, text_scale, (0, 255,
        255), text_thickness, cv2.LINE_AA)

# Draw REPLAY and CLOSE buttons
button_scale_factor = 0.3
replay_btn_resized = cv2.resize(replay_btn_img, (int(replay_btn_img.shape[1] *
    button_scale_factor), int(replay_btn_img.shape[0] * button_scale_factor)),
    interpolation=cv2.INTER_AREA)
close_btn_resized = cv2.resize(close_btn_img, (int(close_btn_img.shape[1] *

```

```

        button_scale_factor), int(close_btn_img.shape[0] * button_scale_factor)),
        interpolation=cv2.INTER_AREA)
    replay_btn_w, replay_btn_h = replay_btn_resized.shape[1],
    replay_btn_resized.shape[0]
    close_btn_w, close_btn_h = close_btn_resized.shape[1],
    close_btn_resized.shape[0]

    buttons_center_y = int(ih * 0.5)
    total_buttons_width = replay_btn_w + close_btn_w + 40
    start_x_buttons = (iw // 2) - (total_buttons_width // 2)
    replay_x = start_x_buttons
    replay_y = buttons_center_y - (replay_btn_h // 2)
    close_x = start_x_buttons + replay_btn_w + 40
    close_y = buttons_center_y - (close_btn_h // 2)

    frame = overlay_transparent(frame, replay_btn_resized, replay_x, replay_y)
    frame = overlay_transparent(frame, close_btn_resized, close_x, close_y)

    replay_button_rect = (replay_x, replay_y, replay_btn_w, replay_btn_h)
    close_button_rect = (close_x, close_y, close_btn_w, close_btn_h)

cv2.imshow("EVADER", frame)
key = cv2.waitKey(1) & 0xFF

# Global exit (ESC)
if key == 27:
    _exit_game_flag = True

# Handle button clicks via mouse callback flags
if _restart_game_flag:
    current_game_state = GAME_STATE_PLAYING
    winner_player_id = None
    health_player1 = 100
    health_player2 = 100
    shield_active_player1 = False
    shield_active_player2 = False
    projectiles.clear()
    player_positions = [None, None]
    last_blink_time = [0, 0]
    eye_ready_to_blink = [0, 0]
    show_instructions_start_time = time.time()
    _restart_game_flag = False
if _exit_game_flag:
    break

cap.release()
cv2.destroyAllWindows()

```

4.2 overlay.py dan Fungsi overlay_transparent

overlay.py berisi kode untuk menggabungkan gambar transparan (seperti pesawat ruang angkasa, peluru, perisai, atau tombol UI) ke dalam frame video secara mulus.

Kode 7: overlay_transparent

```
def overlay_transparent(bg, overlay, x, y, overlay_size=None):
```

```

if overlay_size:
    overlay = cv2.resize(overlay, overlay_size, interpolation=cv2.INTER_AREA)
    h, w = overlay.shape[:2]

# Ensure overlay fits within the background dimensions
if x + w > bg.shape[1] or y + h > bg.shape[0] or x < 0 or y < 0:
    return bg

# Extract overlay components
overlay_img = overlay[:, :, :3]
mask = overlay[:, :, 3] / 255.0
mask = np.stack([mask] * 3, axis=-1)

# Apply blending
roi = bg[y:y+h, x:x+w]
blended = (1.0 - mask) * roi + mask * overlay_img
bg[y:y+h, x:x+w] = blended.astype(np.uint8)

return bg

```

4.3 utils.py

File **utils.py**, berfungsi sebagai perpustakaan utilitas atau kumpulan fungsi-fungsi pembantu yang digunakan di berbagai bagian program utama (**main.py**). Daripada menulis ulang kode untuk tugas-tugas umum seperti menghitung rasio aspek mata, mendeteksi gestur tangan, atau menggambar health bar, fungsi-fungsi ini dienkapsulasi dalam **utils.py**

4.3.1 Fungsi eye_aspect_ratio

Fungsi ini menghitung Rasio Aspek Mata (Eye Aspect Ratio - EAR). EAR adalah metrik yang digunakan untuk menentukan seberapa terbuka mata seseorang. Ini sering digunakan dalam sistem deteksi kantuk atau, seperti dalam kasus ini, untuk mendeteksi kedipan mata.

Kode 8: eye_aspect_ratio

```

def eye_aspect_ratio(landmarks, eye_indices, img_w, img_h):
    p = np.array([(landmarks[i].x * img_w, landmarks[i].y * img_h) for i in eye_indices])
    A = np.linalg.norm(p[1] - p[5])
    B = np.linalg.norm(p[2] - p[4])
    C = np.linalg.norm(p[0] - p[3])
    ear = (A + B) / (2.0 * C)
    return ear

```

4.3.2 Fungsi detect_hand_gesture

Fungsi ini dirancang untuk mendeteksi gestur tangan spesifik, saat ini hanya gestur "jempol ke atas" (thumbs_up).

Kode 9: detect_hand_gesture

```

def detect_hand_gesture(hand_landmarks, thumb_threshold=0.1):
    thumb_tip = hand_landmarks.landmark[4]
    index_tip = hand_landmarks.landmark[8]
    middle_tip = hand_landmarks.landmark[12]

```

```

if (thumb_tip.y + thumb_threshold < index_tip.y) and (thumb_tip.y + thumb_threshold <
    middle_tip.y):
    return "thumbs_up"

return None

```

4.3.3 Fungsi draw_healthbar

Fungsi ini bertanggung jawab untuk menggambar *health bar* di atas frame video untuk pemain.

Kode 10: draw_healthbar

```

def draw_healthbar(frame, player_id, health, x, y, w=100, h=10, bg_color=(0, 0, 0),
    fg_color=(0, 255, 0)):
    cv2.rectangle(frame, (x, y), (x + w, y + h), bg_color, 2)

    fill_width = int((health / 100) * w)
    cv2.rectangle(frame, (x, y), (x + fill_width, y + h), fg_color, -1)

    cv2.putText(frame, player_id, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255,
        255), 1)
    return frame

```

4.4 menu_manager.py dan Fungsi run_menu

menu_manager mengelola dan menampilkan menu awal permainan. Ini berfungsi sebagai gerbang utama sebelum game EVADER dimulai. Tujuannya adalah untuk menyambut pemain, menampilkan judul game, dan memberikan opsi untuk memulai permainan.

Kode 11: menu_manager

```

def run_menu():
    # Load assets
    title_img = cv2.imread('assets/BUTTON TITLE/TITLE.png', cv2.IMREAD_UNCHANGED)
    start_btn_img = cv2.imread('assets/BUTTON TITLE/START.png', cv2.IMREAD_UNCHANGED)

    if title_img is None or start_btn_img is None:
        raise FileNotFoundError("Gambar tidak ditemukan. Pastikan semua aset ada di
            direktori yang benar.")
    if title_img.shape[2] < 4 or start_btn_img.shape[2] < 4:
        raise ValueError("Gambar tidak memiliki alpha channel.")

    # Constants
    RESIZED_FRAME_DIMENSIONS = (640, 480)
    SCREEN_CENTER_X = RESIZED_FRAME_DIMENSIONS[0] // 2

    # Initialize variables
    start_button_rect = None
    _start_game_flag = False

    def handle_mouse_event(event, x, y, flags, param):
        nonlocal _start_game_flag
        if event == cv2.EVENT_LBUTTONDOWN:
            # Check if start button was clicked
            if start_button_rect and \

```

```
x >= start_button_rect[0] and x <= start_button_rect[0] +
    start_button_rect[2] and \
y >= start_button_rect[1] and y <= start_button_rect[1] +
    start_button_rect[3]:
    _start_game_flag = True

# Main menu loop
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    raise IOError("Tidak dapat membuka kamera.")
cv2.namedWindow("EVADER")
cv2.setMouseCallback("EVADER", handle_mouse_event)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    frame = cv2.flip(frame, 1)
    frame = cv2.resize(frame, RESIZED_FRAME_DIMENSIONS)
    ih, iw = frame.shape[:2]

    # Draw title screen
    title_scaled = cv2.resize(title_img, (int(iw * 0.8), int(ih * 0.8)),
        interpolation=cv2.INTER_AREA)
    title_x = (iw - title_scaled.shape[1]) // 2
    title_y = int(ih * 0.2)
    frame = overlay_transparent(frame, title_scaled, title_x, title_y,
        (title_scaled.shape[1], title_scaled.shape[0]))

    # Draw start button
    start_btn_resized = cv2.resize(start_btn_img, (int(start_btn_img.shape[1] * 0.3),
        int(start_btn_img.shape[0] * 0.3)), interpolation=cv2.INTER_AREA)
    start_btn_w, start_btn_h = start_btn_resized.shape[1], start_btn_resized.shape[0]
    start_x = (iw // 2) - (start_btn_w // 2)
    start_y = int(ih * 0.7)
    frame = overlay_transparent(frame, start_btn_resized, start_x, start_y,
        (start_btn_w, start_btn_h))
    start_button_rect = (start_x, start_y, start_btn_w, start_btn_h)

    cv2.imshow("EVADER", frame)
    key = cv2.waitKey(1) & 0xFF

    # Exit on ESC
    if key == 27:
        break

    # Check if START button was clicked
    if _start_game_flag:
        break

cap.release()
cv2.destroyAllWindows()

return _start_game_flag
```

5 Setup Program

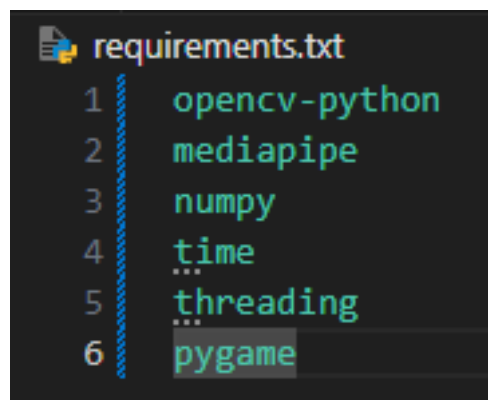
5.1 Versi Python

Pastikan versi python anda ada di rentang 3.10 - 3.12

```
C:\Users\LOQ>python --version
Python 3.12.9
```

5.2 Instalasi Dependencies

Install dependencies dengan `pip install -r requirements.txt`

A screenshot of a text editor window titled 'requirements.txt'. The file contains a list of Python dependencies, each on a new line, numbered 1 through 6. The dependencies are: opencv-python, mediapipe, numpy, time, threading, and pygame. The 'pygame' line is highlighted with a mouse cursor.

```
requirements.txt
1  opencv-python
2  mediapipe
3  numpy
4  time
5  threading
6  pygame
```

5.3 Run main.py

Jalankan file `main.py` untuk memulai game

```
PS C:\Kuliah\Semester_6\Mulmed\Evader> python main.py
```

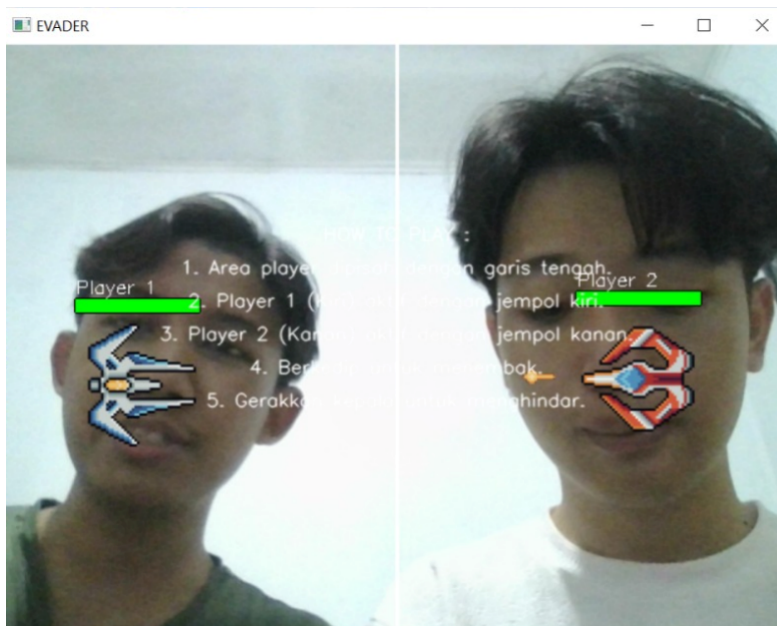
5.4 Mulai permainan

Akan muncul tampilan menu awal dari game. Tekan tombol start untuk memulai game



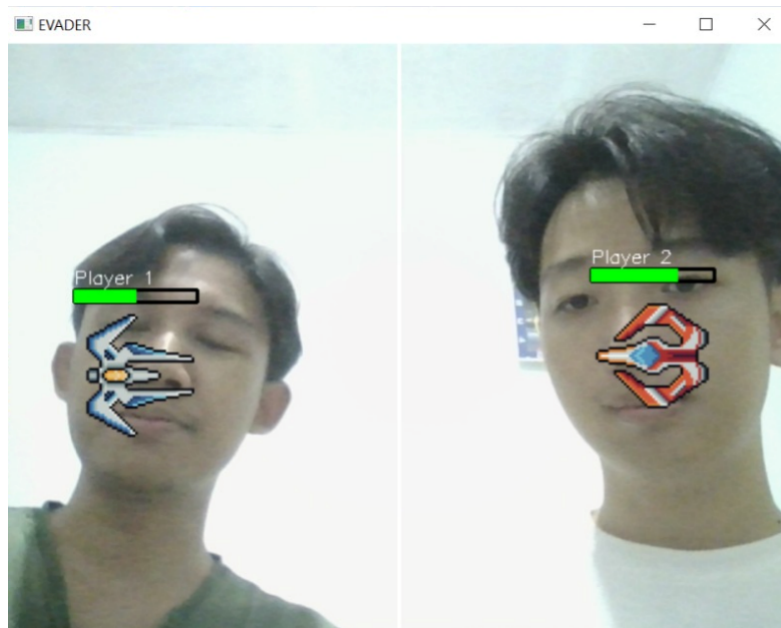
5.5 Instruksi permainan

Ditampilkan instruksi permainan untuk membantu player mengerti cara bermain



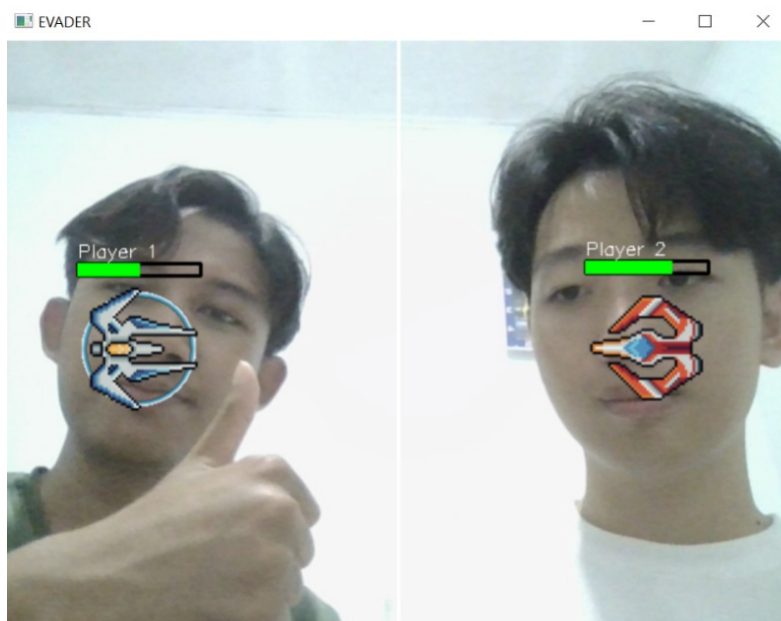
5.6 Permainan dimulai

Setiap player dapat menyerang dengan berkedip kemudian pesawat akan menembakkan peluru



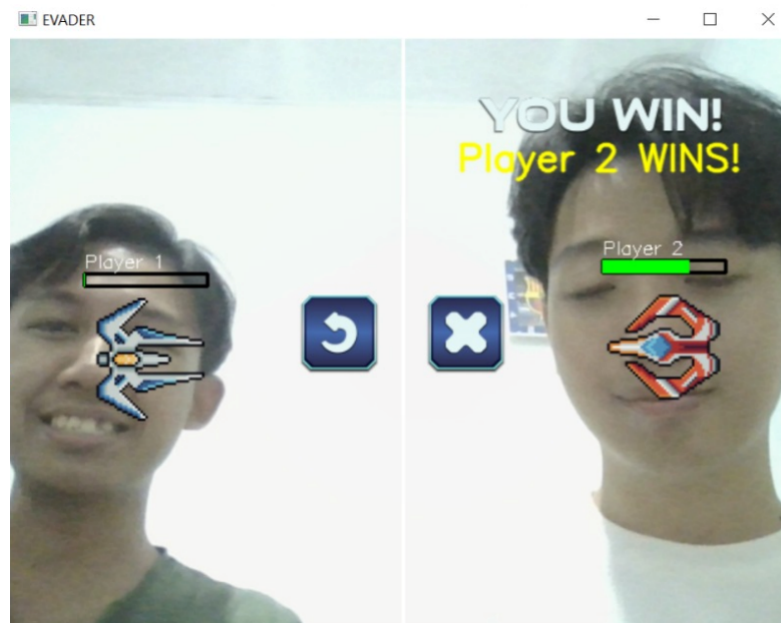
5.7 Mengaktifkan perisai

Ketika pemain membuat gestur jempol, akan mengaktifkan perisai pada pesawatnya



5.8 Kondisi Menang dan Kalah

Saat *health bar* pemain habis, maka akan dinyatakan kalah, dan pemain yang bertahan menjadi pemenangnya



References

- [1] Q. AI, “Multimedia project assistance,” 2025, <https://chat.qwen.ai/s/ab5cb85a-06ca-4323-ae86-1975675d5af8?fev=0.0.101>.
- [2] GeminiAI, “Debugging collision in a game,” 2025, <https://gemini.google.com/share/53eec50dd60b>.