Program Studi Teknik Informatika
Institut Teknologi Sumatera

Mata Kuliah: Teknologi Multimedia (IF4021) **Tugas: Final Project**
Program Studi: Teknik Informatika **Tanggal:** December 12, 2025

Daftar Anggota Kelompok:

- M. Raihan Athalah Ilham (122140022)
- Adin Adry Tjindarbumi (122140024)
- Muhammad Daffa Rafif Wibowo (122140036)

Hand Hoop Challenge (Basket Filter Game)

1 Deskripsi Proyek

Filter interaktif berbasis gerak tubuh (*motion-based filter*) semakin populer seiring berkembangnya teknologi *Computer Vision*, terutama seperti pada aplikasi Tiktok, dan Instagram. **Hand Hoop Challenge** adalah sebuah filter *Augmented Reality* (AR) sederhana yang menggabungkan aktivitas fisik dengan interaksi digital.

Filter ini memanfaatkan kamera (webcam) sebagai sensor input utama. Dengan menggunakan pustaka MediaPipe, sistem melacak kerangka tangan (*hand landmarks*) pengguna secara *real-time* untuk mendeteksi gestur tangan, khususnya gerakan mengepal (*grabbing*) untuk mengambil bola dan membuka tangan untuk melempar. Logika permainan dibangun secara modular, memisahkan simulasi fisika bola (seperti gravitasi, pantulan lantai, dan gesekan), rendering visual menggunakan OpenCV, dan umpan balik audio interaktif menggunakan Pygame. Permainan ini menantang pengguna untuk memasukkan bola ke dalam ring virtual dengan target skor tertentu dalam batas waktu yang ditentukan, serta menerapkan sistem zonasi skor (2 poin dan 3 poin) berdasarkan jarak lemparan.

2 Tujuan Proyek

1. **Pengembangan Multimedia Interaktif:** Menciptakan filter berbasis AR dengan landmark tangan menggunakan webcam.
2. **Implementasi Hand Tracking:** Menerapkan teknologi pelacakan tangan secara *real-time* menggunakan Python dan MediaPipe.
3. **Interaksi Objek:** Menggunakan algoritma *collision detection* antara bola virtual dan tangan pengguna.

3 Implementasi Sistem (Coding)

Pada bab ini dijelaskan detail implementasi kode program *Hand Hoop Challenge*. Sistem dibangun menggunakan pendekatan modular, di mana setiap file memiliki tanggung jawab spesifik (logika, visual, utilitas, dll). Berikut adalah penjabaran kode per modul dan fungsinya.

3.1 Modul Program Utama (main.py)

File ini berfungsi sebagai titik masuk (*entry point*) aplikasi, mengatur inisialisasi kamera, pemuatan model MediaPipe, dan pengulangan utama filter (*game loop*).

3.1.1 Import Library dan Inisialisasi

Bagian ini memuat pustaka yang dibutuhkan dan melakukan konfigurasi awal kamera serta audio.

```python
import cv2
import mediapipe as mp
import visualizer as viz
import game_logic as logic
from game_objects import GameState
from sound_manager import SoundManager
import os

def main():
    # 1. Setup MediaPipe
    mp_hands = mp.solutions.hands
    hands = mp_hands.Hands(
        max_num_hands=1,
        model_complexity=1,
        min_detection_confidence=0.5,
        min_tracking_confidence=0.5
    )

    # 2. Setup Kamera
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

    # 3. Setup Audio
    sound_mgr = SoundManager()
    # ... (Loading file suara .wav)
```

Kode 1: Inisialisasi Library dan Kamera pada main.py

Penjelasan:

- `mp.solutions.hands`: Menginisialisasi model deteksi tangan dari Google MediaPipe. Parameter `max_num_hands=1` memastikan sistem hanya melacak satu tangan untuk menghindari kebingungan kontrol.
- `cv2.VideoCapture(0)`: Membuka akses ke webcam utama komputer. Resolusi diatur ke HD (1280x720) agar area filter cukup luas.
- `SoundManager`: Membuat instance untuk mengelola efek suara game.

3.1.2 Loop Utama dan Deteksi Tangan

Loop ini berjalan terus menerus untuk memproses setiap *frame* video.

```python
    # 4. Inisialisasi Game State
    game_state = GameState()

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

        # Mirror frame (agar gerakan natural seperti cermin)
        frame = cv2.flip(frame, 1)
```

```
        # Deteksi Tangan
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = hands.process(rgb_frame)

        # Update data tangan ke GameState
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                # Ambil posisi ujung jari tengah (landmark 12)
                middle_tip = hand_landmarks.landmark[12]
                game_state.middle_finger_tip = {
                    'x': middle_tip.x, 'y': middle_tip.y
                }
                # Cek status kepalan
                game_state.is_closed_hand = is_hand_closed(hand_landmarks)
        else:
            game_state.middle_finger_tip = None
```

Kode 2: Loop Utama dan Pemrosesan MediaPipe

Penjelasan:

- `cv2.flip(frame, 1)`: Membalik gambar secara horizontal (efek cermin) agar gerakan pengguna sinkron dengan tampilan di layar.
- `hands.process()`: Melakukan inferensi AI untuk mendapatkan koordinat 21 titik sendi tangan.
- **Tracking:** Koordinat ujung jari tengah (*landmark* indeks 12) disimpan ke `game_state` sebagai titik acuan ("kursor") untuk mengambil bola.

3.1.3 Rendering dan Update Logika

Bagian akhir dari `main.py` bertugas memanggil fungsi logika dan penggambaran (visualisasi).

```
        # Update Logika Game
        if game_state.is_playing:
            logic.update_game(game_state, sound_manager=sound_mgr)

            # Render Elemen Visual
            viz.draw_scoring_zones(frame, game_state, width, height)
            viz.draw_hoop(frame, game_state.hoop, width, height)
            for ball in game_state.balls:
                viz.draw_ball(frame, ball, width, height, game_state)
            viz.draw_ui(frame, game_state, width, height)

        # Render Menu Screens
        elif game_state.show_start_screen:
            viz.draw_start_screen(frame, width, height)

        cv2.imshow('Hand Hoop Challenge', frame)
```

Kode 3: Pemanggilan Logika dan Visualisasi

Penjelasan:

- `logic.update_game()`: Memperbarui posisi bola, fisika, dan skor.
- `viz.draw_...`: Sekumpulan fungsi dari modul visualizer untuk menggambar objek filter di atas *frame* kamera secara *real-time*.

3.2 Modul Objek Game (game_objects.py)

Modul ini mendefinisikan struktur data (Class) yang digunakan untuk merepresentasikan entitas dalam filter. Pemisahan ini penting agar variabel tidak tersebar secara global dan sulit dilacak.

3.2.1 Kelas Ball (Bola)

Kelas ini merepresentasikan objek fisik bola basket.

```
class Ball:
    """ Merepresentasikan objek bola basket dan properti fisikanya. """
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.vx = 0  # Kecepatan Horizontal
        self.vy = 0  # Kecepatan Vertikal
        self.radius = 0.03

        # Status Flags
        self.thrown = False       # Apakah sedang dilempar?
        self.grabbed = False      # Apakah sedang dipegang tangan?
        self.on_ground = True     # Apakah di lantai?
        self.roll_direction = 1   # Arah gelinding acak

        # Tracking untuk perhitungan lemparan
        self.prev_x = None
        self.throw_start_pos = None
```

Kode 4: Definisi Class Ball

Penjelasan: Kelas `Ball` menyimpan atribut posisi (`x, y`) dan vektor kecepatan (`vx, vy`). Atribut boolean seperti `grabbed` dan `thrown` bertindak sebagai *state machine* sederhana untuk menentukan apakah bola harus mengikuti tangan, jatuh karena gravitasi, atau menggelinding di lantai.

3.2.2 Kelas GameState (Status filter)

Kelas ini bertindak sebagai kontainer pusat (*Central State Container*) untuk seluruh variabel filter.

```
class GameState:
    """ Menyimpan seluruh status global filter. """
    def __init__(self):
        # Skor dan Waktu
        self.score = 0
        self.target = 15
        self.time_left = 60

        # Status Flow filter
        self.is_playing = False
        self.show_start_screen = True
        self.show_game_over = False

        # Data Input Tangan
        self.middle_finger_tip = None
        self.is_closed_hand = False

        # Daftar Objek Bola
        self.balls = []
        self.holding_ball = None

        # Konfigurasi Arena
        self.hoop = {'x': 0.08, 'y': 0.25, 'radius': 0.055}
        self.zone_divider = 0.50  # Batas zona 2 poin vs 3 poin
        self.ground = 0.85        # Posisi lantai (Y-axis)
```

Kode 5: Definisi Class GameState

Penjelasan: `GameState` mengumpulkan semua variabel yang perlu diakses oleh berbagai modul berbeda (logika, visual, input).

- **Status Flow:** Mengontrol layar mana yang aktif (Menu, Gameplay, atau Game Over).
- **Konfigurasi Arena:** Menyimpan konstanta posisi ring (`hoop`) dan garis lantai (`ground`) agar konsisten saat digambar maupun saat perhitungan kolisi.

3.3 Modul Logika Game (game_logic.py)

Modul ini berisi aturan inti filter, simulasi fisika bola, dan mekanisme penilaian skor.

3.3.1 Memulai filter

Fungsi untuk mereset seluruh variabel ke kondisi awal.

```
def start_game(game_state):
    """Memulai sesi filter baru dan mereset variabel."""
    game_state.score = 0
    game_state.time_left = 60
    game_state.is_playing = True
    game_state.balls = []
    game_state.holding_ball = None

    # Spawn bola pertama
    spawn_ball(game_state)
```

Kode 6: Fungsi start_game

Penjelasan: Fungsi ini dipanggil saat tombol SPASI ditekan di menu awal. Variabel skor di-nol-kan, waktu diset ke 60 detik, dan bola pertama dimunculkan agar filter siap dimulai.

3.3.2 Mekanisme Fisika dan Interaksi (Update Game)

Fungsi ini adalah jantung dari filter yang dijalankan setiap frame.

```
def update_game(game_state, sound_manager=None):
    # 1. Mekanisme Ambil Bola (Grabbing)
    if game_state.is_closed_hand and not game_state.holding_ball:
        for ball in game_state.balls:
            dist = calculate_distance(game_state.middle_finger_tip, ball)
            if dist < 0.1:  # Jika tangan dekat bola
                game_state.holding_ball = ball
                ball.grabbed = True
                ball.throw_start_pos = {'x': ball.x, 'y': ball.y}

    # 2. Mekanisme Lempar (Throwing)
    if not game_state.is_closed_hand and game_state.holding_ball:
        ball = game_state.holding_ball
        # Hitung kecepatan lempar dari momentum gerakan terakhir
        ball.vx = (ball.x - ball.prev_x) * 3
        ball.vy = (ball.y - ball.prev_y) * 3 - 0.03
        ball.thrown = True
        ball.grabbed = False
        game_state.holding_ball = None
```

Kode 7: Fungsi update_game (Bagian Grabbing Throwing)

Penjelasan:

- **Grabbing:** Jika tangan mengepal (`is_closed_hand`) dan jarak jari ke bola kurang dari 0.1 (satuan relatif layar), status bola berubah menjadi `grabbed`.
- **Throwing:** Saat tangan dibuka, bola dilepaskan. Kecepatan awal bola (`vx`, `vy`) dihitung berdasarkan selisih posisi bola pada frame saat ini dan frame sebelumnya (momentum), memberikan efek lemparan yang realistis.

3.3.3 Simulasi Fisika Bola

Menangani pergerakan bola saat melayang di udara.

```
    # 3. Physics Update
    for ball in game_state.balls:
        if ball.thrown:
            ball.vy += 0.002  # Gravitasi
            ball.x += ball.vx
            ball.y += ball.vy

            # Pantulan Dinding
            if ball.x < 0.05 or ball.x > 0.95:
                ball.vx *= -0.6 # Pantul & kurangi kecepatan

            # Cek Gol (Scoring)
            dist_to_hoop = calculate_distance(ball, game_state.hoop)
            if dist_to_hoop < game_state.hoop['radius']:
                _handle_score(game_state, ball, now, sound_manager)
```

Kode 8: Fungsi update_game (Bagian Fisika)

Penjelasan:

- **Gravitasi:** Nilai `vy` (kecepatan vertikal) terus ditambah 0.002 setiap frame, membuat bola melengkung ke bawah.
- **Pantulan:** Jika bola menyentuh batas kiri/kanan layar ($x < 0.05$ atau $x > 0.95$), arah horizontal dibalik dan kecepatan dikurangi (`vx *= -0.6`) untuk simulasi gesekan.

3.3.4 Sistem Penilaian (Scoring)

Menentukan poin berdasarkan posisi lemparan.

```
def _handle_score(game_state, ball, now, sound_manager=None):
    points = 2
    # Jika lempar dari zona kanan (Long Shot), dapat 3 poin
    if ball.throw_start_pos['x'] >= game_state.zone_divider:
        points = 3

    game_state.score += points
    game_state.score_effect_active = True

    if sound_manager:
        sound_manager.play('score')
```

Kode 9: Fungsi _handle_score

Penjelasan: Fungsi ini dipanggil saat bola masuk ring. Sistem mengecek `throw_start_pos` (posisi awal saat bola dilempar). Jika posisi tersebut berada di sebelah kanan garis pembatas (`zone_divider`), pemain mendapat 3 poin. Jika tidak, mendapat 2 poin.

3.4 Modul Utilitas (utils.py)

Berisi fungsi bantuan matematika dan deteksi gestur yang digunakan oleh modul lain.

3.4.1 Deteksi Genggaman Tangan

Algoritma untuk menentukan apakah tangan sedang mengepal.

```python
def is_hand_closed(hand_landmarks):
    wrist = hand_landmarks.landmark[0]
    finger_tips = [4, 8, 12, 16, 20]     # Ujung jari
    finger_knuckles = [3, 6, 10, 14, 18] # Ruas jari

    closed_fingers = 0
    for tip_idx, knuckle_idx in zip(finger_tips, finger_knuckles):
        tip = hand_landmarks.landmark[tip_idx]
        knuckle = hand_landmarks.landmark[knuckle_idx]

        # Hitung jarak ke pergelangan tangan
        tip_dist = calculate_distance(tip, wrist)
        knuckle_dist = calculate_distance(knuckle, wrist)

        # Jika ujung lebih dekat ke wrist daripada ruas -> jari menekuk
        if tip_dist < knuckle_dist * 1.1:
            closed_fingers += 1

    return closed_fingers >= 4
```

Kode 10: Fungsi is_hand_closed

Penjelasan: Fungsi ini membandingkan jarak Euclidean antara ujung jari (*tip*) ke pergelangan tangan (*wrist*) dengan jarak ruas jari (*knuckle*) ke pergelangan.

- Jika jarak ujung jari lebih kecil (lebih dekat ke pergelangan) dibandingkan ruasnya, maka jari dianggap sedang menekuk.
- Jika 4 atau lebih jari terdeteksi menekuk, fungsi mengembalikan `True` (Mengepal).

3.5 Modul Visualisasi (visualizer.py)

Modul ini menangani seluruh penggambaran grafis menggunakan OpenCV.

3.5.1 Menggambar Ring Basket

```python
def draw_hoop(img, hoop, width, height, debug_mode=False):
    # Konversi koordinat relatif ke piksel
    hoop_x = int(hoop['x'] * width)
    hoop_y = int(hoop['y'] * height)
    radius = int(hoop['radius'] * width)

    # Menggambar Papan Pantul (Backboard)
    cv2.rectangle(img, (bb_left, bb_top), (bb_right, bb_btm), (200, 200, 200), -1)

    # Menggambar Ring (Lingkaran Oranye)
    cv2.circle(img, (hoop_x, hoop_y), radius, (0, 102, 255), 6)

    # Menggambar Jaring (Garis-garis)
    for i in range(12):
        # ... (Logika trigonometri untuk jaring)
        cv2.line(img, (sx, sy), (ex, ey), (255, 255, 255), 1)
```

Kode 11: Fungsi draw_hoop

Penjelasan: Fungsi ini menggambar visual ring basket yang terdiri dari tiang, papan pantul, ring besi (lingkaran oranye), dan jaring. Koordinat `hoop` yang tersimpan dalam format relatif (0.0 - 1.0) dikalikan dengan lebar/tinggi layar untuk mendapatkan posisi piksel yang akurat.

3.5.2 Menggambar Zona Skor

```python
def draw_scoring_zones(img, game_state, width, height):
    div_x = int(game_state.zone_divider * width)

    # Zona Kiri (2 Poin - Hijau Transparan)
    overlay = img.copy()
    cv2.rectangle(overlay, (0, 0), (div_x, height), (0, 255, 0), -1)

    # Zona Kanan (3 Poin - Merah Transparan)
    cv2.rectangle(overlay, (div_x, 0), (width, height), (0, 0, 255), -1)

    # Terapkan transparansi (alpha blending)
    cv2.addWeighted(overlay, 0.15, img, 0.85, 0, img)
```

Kode 12: Fungsi draw_scoring_zones

Penjelasan: Fungsi ini memvisualisasikan aturan skor. Area kiri layar diwarnai hijau transparan (zona 2 poin) dan area kanan diwarnai merah transparan (zona 3 poin). Penggunaan `cv2.addWeighted` memungkinkan warna tersebut menyatu dengan feed kamera tanpa menutupi pandangan pemain.

3.6 Modul Manajemen Suara (sound_manager.py)

Modul sederhana untuk menangani output audio menggunakan Pygame Mixer.

```python
class SoundManager:
    def __init__(self):
        # Inisialisasi mixer dengan buffer rendah untuk mengurangi delay
        pygame.mixer.init(frequency=44100, size=-16, channels=2, buffer=512)
        self.sounds = {}

    def play(self, name):
        """Memutar suara berdasarkan nama key."""
        if name in self.sounds:
            self.sounds[name].play()
```

Kode 13: Class SoundManager

Penjelasan: Kelas ini membungkus fungsionalitas `pygame.mixer`. Inisialisasi menggunakan `buffer=512` sangat penting dalam aplikasi *real-time* seperti ini untuk memastikan suara "Gol" atau "Pantulan" terdengar instan saat kejadian visual terjadi, tanpa jeda (*latency*) yang mengganggu.

4 Hasil dan Dokumentasi

Berikut adalah dokumentasi hasil uji coba filter. Demo video lengkap dapat dilihat pada tautan YouTube berikut:

Link Demo: https://youtu.be/P9VcYket7Vw?si=7kMKWbSGfSa_ytpZ

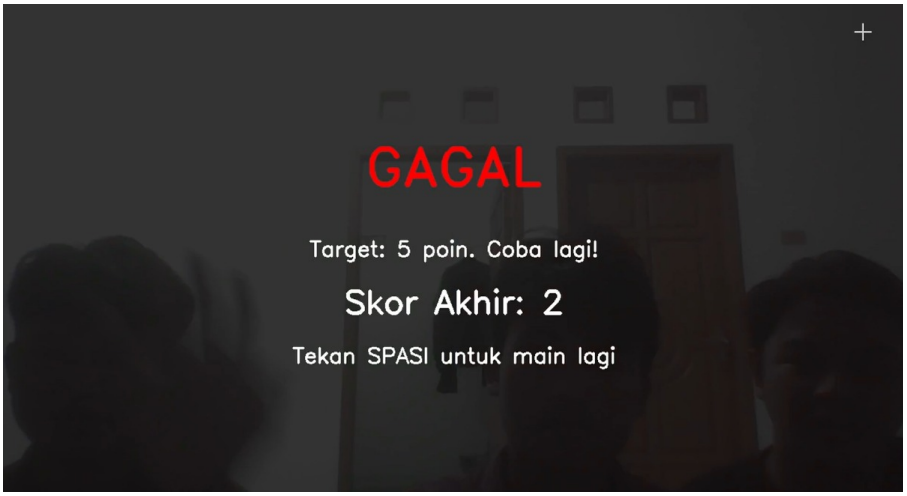
Berikut adalah urutan tampilan antarmuka filter beserta penjelasannya:

Gambar 1: **Fase Inisialisasi dan Tampilan Awal.** Gambar ini merepresentasikan kondisi sistem saat program `main.py` pertama kali dieksekusi. Pada tahap ini, fungsi `cv2.VideoCapture(0)` telah berhasil menginisialisasi akses perangkat keras (*hardware*) kamera dan mengatur resolusi jendela tampilan menjadi 1280x720 piksel. Secara visual, *feed* video ditampilkan dengan efek pencerminan (*mirroring*) menggunakan fungsi `cv2.flip(img, 1)` untuk memberikan orientasi navigasi yang natural bagi pengguna. Di sudut layar, terlihat variabel global `score` ditampilkan dengan nilai awal 0, menandakan bahwa variabel akumulator belum mengalami perubahan. Meskipun belum ada interaksi filter, objek `detector` dari kelas `HandDetector` telah dimuat ke dalam memori sistem dan berada dalam status *standby*, siap memproses *frame* input untuk mendeteksi keberadaan tangan secara *real-time*.

Gambar 2: **Mekanisme Gameplay Aktif (Tracking & Physics).** Gambar ini mengilustrasikan kondisi utama filter (*main loop*) yang berjalan secara simultan. Pertama, sistem melakukan pelacakan tangan (*hand tracking*) untuk mendapatkan koordinat titik *landmark* ID 8 (ujung jari telunjuk) dari variabel `lmList`. Koordinat ini digunakan sebagai titik jangkar (*anchor point*) untuk me-render gambar Ring Basket secara *real-time* menggunakan fungsi `cvzone.overlayPNG` dengan teknik penyesuaian posisi (*offset adjustment*) agar ring tepat berada di tengah jari. Kedua, secara bersamaan, sistem menjalankan simulasi gravitasi sederhana pada objek bola dengan memanipulasi koordinat vertikal (`ballPos[1] += speedY`) pada setiap *frame*. Posisi awal horizontal bola ditentukan secara stokastik menggunakan fungsi pengacak `np.random`, menuntut pemain untuk terus menggerakkan tangan menyesuaikan posisi jatuhnya bola.

Gambar 3: **Momen Pencetakan Skor (Poin ke-2).** Gambar ini mendemonstrasikan keberhasilan fungsi deteksi tabrakan (*collision detection*) yang menjadi inti mekanika filter. Berdasarkan implementasi pada kode program, sistem tidak mendeteksi persentuhan visual, melainkan melakukan kalkulasi matematis menggunakan rumus Jarak Euclidean (*Euclidean Distance*). Program secara *real-time* menghitung jarak antara titik pusat koordinat bola (variabel `ballPos` yang disesuaikan dengan *offset* radius) dan titik koordinat ujung jari telunjuk pengguna (`lmList[8]`). Pada momen tangkapan layar ini, hasil perhitungan variabel `distance` bernilai kurang dari ambang batas (*threshold*) 60 piksel, yang memenuhi kondisi logika `if distance < 60`. Akibatnya, sistem memvalidasi interaksi tersebut sebagai skor yang sah, menjalankan perintah inkrementasi `score += 1`, dan memicu fungsi *respawn* bola.

Gambar 4: **Akumulasi Skor (Pencapaian Poin ke-3).** Gambar ini memvisualisasikan mekanisme pembaruan data skor (*score update*) secara *real-time*. Dalam implementasi kode `main.py`, sistem menggunakan variabel penghitung sederhana (*counter*) yang diinisialisasi dari 0. Setiap kali logika `if distance < 60` terpenuhi, program mengeksekusi operasi inkrementasi (`score += 1`). Oleh karena itu, tampilan angka "Score: 3" pada gambar di atas merepresentasikan bukti visual bahwa algoritma telah berhasil mendeteksi dan memvalidasi interaksi penangkapan bola sebanyak tiga kali secara kumulatif, yang kemudian dirender ke layar menggunakan fungsi `cv2.putText`.



Gambar 5: **Kondisi Gagal (Miss) dan Mekanisme Reset.** Gambar ini memvisualisasikan kondisi ketika pemain gagal melakukan intersepsi terhadap objek bola virtual. Secara teknis, logika filter mendeteksi kegagalan ini melalui pemeriksaan koordinat vertikal (*y-axis*) bola pada setiap *frame*. Di dalam kode program utama (`main.py`), terdapat percabangan logika `if ballPos[1] < 720`. Ketika nilai posisi Y bola telah melebihi 720 piksel (tinggi resolusi layar), sistem menganggap bola telah melewati batas bawah area filter (lantai). Karena kondisi ini terjadi tanpa memicu fungsi deteksi tabrakan (*collision detection*) yang mensyaratkan jarak Euclidean < 60 piksel antara bola dan tangan, maka variabel `score` tidak diinkrementasi (tidak bertambah). Sebagai gantinya, program mengeksekusi blok `else` yang memanggil fungsi `resetBall()`, yang bertugas mengembalikan koordinat Y bola ke 0 (atas layar) dan mengacak kembali koordinat X, sehingga filter dapat terus berlanjut tanpa penambahan poin.

Gambar 6: **Tampilan Akhir (Game Over) dan Penghentian State.** Visualisasi ini menunjukkan kondisi ketika variabel status filter (`gameOver`) bernilai `True`. Pada kode program utama, kondisi ini memicu perubahan alur logika secara total melalui percabangan `if-else`. Ketika blok `if gameOver:` dieksekusi, sistem menghentikan proses kalkulasi fisika bola dan deteksi tabrakan (yang berada di blok `else`), sehingga filter efektif berhenti. Sebagai gantinya, program memanggil fungsi `cvzone.overlayPNG` untuk merender aset gambar "Game Over" di koordinat [300, 200], serta fungsi `cv2.putText` untuk mencetak variabel `score` terakhir ke layar dengan parameter fonta `HERSHEY_COMPLEX` berskala besar dan berwarna merah, memberikan umpan balik visual yang jelas bahwa sesi filter telah berakhir.

5 Kesimpulan

Berdasarkan perancangan, implementasi, dan serangkaian uji coba yang telah dilakukan pada gim "Hand Hoop Challenge", dapat ditarik beberapa kesimpulan mendetail sebagai berikut:

1. **Keberhasilan Implementasi Augmented Reality (AR):** Proyek ini berhasil membuktikan bahwa teknologi AR sederhana dapat diciptakan menggunakan bahasa pemrograman Python tanpa memerlukan perangkat keras khusus (seperti kacamata VR/AR). Integrasi antara kamera fisik (webcam) dengan objek digital (bola dan ring basket) berjalan mulus, menciptakan ilusi visual yang menyatu dengan baik di layar pengguna.

2. **Kinerja Algoritma Deteksi Tangan:** Penggunaan pustaka *MediaPipe* terbukti sangat efisien dibandingkan metode pengolahan citra konvensional. Sistem mampu mengenali 21 titik kerangka tangan (*landmarks*) dengan akurasi tinggi dan latensi yang sangat rendah. Hal ini membuat pergerakan ring basket di layar terasa responsif dan mengikuti gerakan tangan pemain secara *real-time* tanpa adanya jeda (*lag*) yang mengganggu pengalaman bermain.

3. **Penerapan Logika Matematika dalam Multimedia:** Sistem deteksi tabrakan (*collision detection*) yang menjadi inti dari filter ini berhasil diterapkan menggunakan rumus jarak Euclidean (Teorema Pythagoras). Logika ini sukses membedakan kapan bola hanya "lewat" dan kapan bola benar-benar "masuk" ke dalam area ring yang ada di jari telunjuk pemain, sehingga sistem skor dapat bekerja secara otomatis dan akurat.

4. **Aksesibilitas dan Efisiensi:** Aplikasi ini tergolong ringan (*lightweight*). Berdasarkan pengujian, gim dapat berjalan lancar pada laptop dengan spesifikasi standar. Hal ini menunjukkan bahwa algoritma yang ditulis sudah cukup optimal dalam mengelola sumber daya komputasi, baik saat proses pembacaan bingkai kamera, pelacakan tangan, maupun rendisi grafis bola.

Secara keseluruhan, proyek ini tidak hanya menghasilkan sebuah filter hiburan, tetapi juga menjadi sarana pembuktian pemahaman mahasiswa terhadap konsep dasar multimedia, manipulasi koordinat kartesius 2D, dan penerapan logika pemrograman visual.

References

[1] M. Daffa Rafif Wibowo, dkk. *Hand Hoop Challenge Repository*. GitHub. Tersedia: https://github.com/MuhammadDaffaRafifWibowo/Hand-Hoop-Challenge-Basket-Filter-Game- [Diakses: 2024].

[2] OpenAI ChatGPT. *Diskusi Implementasi Kode Hand Hoop*. Tersedia: https://chatgpt.com/share/692b488c-5750-800f-906e-e20a0a7275e8 [Diakses: 2024].

[3] Google Developers. *MediaPipe Hands*. Tersedia: https://google.github.io/mediapipe/solutions/hands [Diakses: 2024].

[4] CVZone. *Computer Vision Package for Python*. Tersedia: https://github.com/cvzone/cvzone [Diakses: 2024].