Nama	:	1. **Aziz Kurniawan (122140097)** 2. **Muhammad Yusuf (122140193)** 3. **Harisya Miranti (122140049)**
Mata Kuliah	:	**Sistem Teknologi Multimedia (IF25-40305)**
Tugas	:	**Shadow Boxing**
Dosen Pengampu	:	**Martin Clinton Tosima Manullang, Ph.D.**
Tanggal	:	**December 11, 2025**

IMPLEMENTASI GAME SHADOW BOXING INTERAKTIF BERBASIS COMPUTER VISION MENGGUNAKAN MEDIAPIPE

1 Pendahuluan

1.1 Latar Belakang

Perkembangan teknologi computer vision dan machine learning saat ini memungkinkan interaksi manusia dengan komputer menjadi lebih intuitif dan menarik. Salah satu aplikasi yang berkembang pesat adalah game interaktif berbasis deteksi gerakan real-time, di mana pengguna dapat berinteraksi dengan sistem menggunakan gerakan tubuh tanpa memerlukan controller fisik.

Shadow boxing adalah teknik latihan tinju yang dilakukan tanpa lawan, di mana praktisi melakukan gerakan tinju di udara untuk melatih teknik, kecepatan, dan stamina. Dalam project ini, kami mengimplementasikan sebuah game boxing interaktif yang dapat mendeteksi gerakan tinju dan pertahanan pengguna secara real-time menggunakan teknologi MediaPipe. Game ini menciptakan pengalaman bermain yang immersive dengan sistem combat phase-based, difficulty levels, dan feedback visual-audio yang responsif.

Game Shadow Boxing ini menerapkan konsep computer vision untuk deteksi fist punching, defense blocking, dan dodge mechanism. Sistem menggunakan MediaPipe untuk mendeteksi hand landmarks (21 titik per tangan), pose landmarks (33 titik tubuh), dan face mesh (468 titik wajah). Dengan deteksi multi-landmark ini, game dapat membedakan antara gerakan menyerang (fist punch) dan bertahan (fingertips covering eyes).

1.2 Rumusan Masalah

1. Bagaimana mengimplementasikan sistem deteksi fist punching yang akurat menggunakan MediaPipe hand landmarks?
2. Bagaimana merancang combat system dengan phase-based gameplay (player attack vs enemy attack)?
3. Bagaimana mengimplementasikan sequential hitbox spawning dengan position-based placement dan exclusion zones?
4. Bagaimana mengimplementasikan defense system yang dapat mendeteksi blocking dan dodging?
5. Bagaimana merancang difficulty levels yang seimbang untuk berbagai tingkat pemain?

1.3 Tujuan

1. Mengimplementasikan game boxing interaktif dengan MediaPipe untuk deteksi gerakan real-time
2. Mengembangkan combat system dengan phase management (player attack, enemy attack warning, enemy attack)
3. Mengimplementasikan hitbox system dengan sequential spawning dan collision detection
4. Mengimplementasikan defense system untuk blocking (fingertips near eyes) dan dodging (head movement)
5. Merancang tiga difficulty levels (Easy, Medium, Hard) dengan parameter yang berbeda
6. Mengintegrasikan audio system dengan LUFS normalization untuk consistent sound levels

1.4 Batasan Masalah

1. Program dikembangkan menggunakan bahasa Python 3.10+
2. Input berupa video stream dari webcam real-time
3. Deteksi gerakan dibatasi pada gerakan tinju dasar (JAB, CROSS, HOOK)
4. Game mode: Player vs Enemy (single player)
5. Jumlah ronde: 3 ronde dengan rest period
6. Program berjalan pada kondisi pencahayaan yang memadai untuk deteksi MediaPipe optimal
7. Resolution: 1280x720 pixels (720p)

2 Landasan Teori

2.1 MediaPipe Framework

MediaPipe adalah framework open-source yang dikembangkan oleh Google untuk membangun pipeline pemrosesan media multimodal. Framework ini menyediakan berbagai solusi machine learning yang telah dilatih sebelumnya untuk berbagai task seperti pose estimation, hand tracking, face detection, dan lain-lain.

MediaPipe menggunakan model machine learning yang efisien dan dapat berjalan real-time bahkan pada device dengan spesifikasi menengah. Framework ini mendukung berbagai platform termasuk desktop, mobile, dan web, menjadikannya pilihan ideal untuk aplikasi computer vision interaktif.

2.2 Hand Landmark Detection

MediaPipe Hands dapat mendeteksi 21 landmark pada setiap tangan dengan presisi tinggi dalam kondisi real-time. Setiap landmark memiliki koordinat x, y, dan z yang merepresentasikan posisi dalam ruang 3D normalized (0.0 - 1.0).

Landmark yang paling penting untuk deteksi fist dalam game ini adalah:

- **Landmark 0 (Wrist)**: Basis untuk menghitung posisi tangan
- **Landmark 8, 12, 16, 20**: Fingertips untuk deteksi fist dan defense
- **Landmark 9 (Middle Finger MCP)**: Palm center untuk distance calculation

2.3 Pose Landmark Detection

MediaPipe Pose dapat mendeteksi 33 landmark pada tubuh manusia dengan akurasi tinggi. Dalam konteks game shadow boxing, landmark yang relevan adalah:

- **Landmarks 1-6**: Eye landmarks untuk defense detection
- **Landmarks 11-12**: Shoulders untuk exclusion zones
- **Landmarks 23-24**: Hips untuk body exclusion zones

2.4 Face Mesh Detection

MediaPipe Face Mesh mendeteksi 468 landmark pada wajah dengan detail tinggi. Dalam game ini, face mesh digunakan untuk:

- Menentukan face bounding box untuk exclusion zone
- Deteksi target area untuk enemy attacks
- Dodge detection berdasarkan head movement

2.5 Fist Detection Algorithm

Deteksi fist menggunakan pendekatan geometrik dengan menganalisis:

1. **Finger Angles**: Sudut antara tip, middle, dan base joint setiap jari. Jari tertekuk (fist) memiliki sudut $<$ threshold (biasanya 90°)
2. **Distance to Palm**: Jarak euclidean dari fingertips ke palm center (landmark 9). Fist memiliki distance $<$ threshold
3. **Combined Threshold**: Rata-rata angle $< 90°$ AND rata-rata distance < 0.15 (normalized)

Formula deteksi fist:

$$is\_fist = \left(\frac{1}{n}\sum_{i=1}^{n}\theta_i < \theta_{threshold}\right) \wedge \left(\frac{1}{n}\sum_{i=1}^{n} d_i < d_{threshold}\right) \tag{1}$$

Di mana θ_i adalah sudut jari ke-i, d_i adalah jarak fingertip ke palm, dan n adalah jumlah jari (4, tidak termasuk thumb).

3 Game Design dan Metodologi

3.1 Konsep Game

Shadow Boxing adalah game boxing interaktif one-on-one combat antara player dan enemy AI. Game menggunakan phase-based combat system dengan pergantian antara player attack phase dan enemy attack phase. Player menyerang dengan memukul hitbox yang muncul secara sekuensial, sementara enemy menyerang dengan combo attacks yang harus diblock atau di-dodge oleh player.

Game Loop:

1. **Main Menu**: Player memilih difficulty (Easy/Medium/Hard)
2. **Round Start**: Splash screen "ROUND X"
3. **Combat Phase**: Player dan enemy bergantian menyerang
4. **Rest Period**: Jeda antar ronde (10 detik)
5. **Game Over**: Tampil hasil akhir (Win/Lose)

3.2 Phase System

Game menggunakan tiga phase utama yang bergantian:

1. Player Attack Phase (3.0-3.5 detik)

- Player menyerang hitbox yang muncul secara sekuensial
- Sequential spawning: hitbox berikutnya spawn setelah hitbox sebelumnya dipukul
- Position-based: JAB spawn di kiri (0-50% width), CROSS/HOOK spawn di kanan (50-100% width)
- Damage berdasarkan difficulty dan punch type (random range)

2. Enemy Attack Warning Phase (0.7-1.5 detik)

- Visual indicator menunjukkan posisi serangan enemy
- Warning time berbeda per difficulty (HARD=0.7s, MEDIUM=1.0s, EASY=1.5s)
- Player mempersiapkan defense (block atau dodge)

3. Enemy Attack Phase

- Enemy melakukan 2-3 combo attacks berturut-turut
- Delay 400ms antar serangan dalam combo
- Target prioritas: Face detection, fallback ke pose landmarks 0-10
- Damage calculation: base × difficulty multiplier

3.3 Combat Mechanics

Damage System:

Player damage menggunakan random range per difficulty:

- **EASY**: JAB(12-13), HOOK(12-14), CROSS(12-15)
- **MEDIUM**: JAB(8-10), HOOK(8-11), CROSS(8-12)
- **HARD**: JAB(5-6), HOOK(5-7), CROSS(5-8)

Last hit bonus: 1.1× multiplier untuk serangan terakhir dalam combo.
Enemy damage: Random 10-20 × difficulty multiplier (EASY=0.7×, MEDIUM=1.0×, HARD=1.3×)
Defense Mechanisms:

1. **Block**: Fingertips (landmarks 8,12,16,20) dalam radius 150px dari eye area (pose landmarks 1-6) → Damage reduction 80%
2. **Dodge**: Head position keluar dari target area → 100% avoid damage

Win Conditions:

- **Victory**: Health tertinggi setelah 3 ronde
- **Defeat**: Health mencapai 0 (KO)
- **Perfect Win**: Tidak menerima damage sama sekali

Tabel 1: Parameter Difficulty Levels

Parameter	EASY	MEDIUM	HARD
Enemy Cooldown	3.0-5.0s	2.0-3.5s	1.5-2.5s
Damage Multiplier	0.7×	1.0×	1.3×
Warning Time	1.5s	1.0s	0.7s
Player Attack Time	3.5s	3.0s	2.5s

3.4 Difficulty Levels

3.5 Arsitektur Sistem

Sistem game menggunakan arsitektur modular dengan separation of concerns:

1. **core**/: Config, constants, math utilities
2. **systems**/: Vision, audio, render, input processing
3. **entities**/: Player dan enemy state management
4. **game**/: Game state, hitbox system, round manager
5. **ui**/: Menu, HUD, overlays, result screen
6. **assets**/: Fonts, sprites, audio (sfx & music)

3.6 Teknologi dan Library yang Digunakan

Project Shadow Boxing ini menggunakan beberapa library Python yang memiliki peran spesifik dalam implementasi sistem. Berikut adalah penjelasan detail mengenai setiap library yang digunakan beserta fungsinya:

3.6.1 OpenCV (opencv-python 4.10.0.84)

OpenCV (Open Source Computer Vision Library) merupakan library open-source untuk computer vision dan machine learning. Dalam project ini, OpenCV memiliki fungsi sebagai:

1. Mengambil input video dari webcam secara real-time dengan resolusi 1280x720 pixels menggunakan `cv2.VideoCapture()`
2. Melakukan konversi color space dari BGR ke RGB untuk kompatibilitas dengan MediaPipe menggunakan `cv2.cvtColor()`
3. Menggambar shapes, text, dan overlay pada frame video seperti health bar, hitbox, dan visual effects dengan functions seperti `cv2.rectangle()`, `cv2.circle()`, `cv2.line()`
4. Mirror effect untuk pengalaman yang lebih intuitif menggunakan `cv2.flip()`
5. Menampilkan output video dalam window dengan frame rate 30 FPS (meskipun rendering utama menggunakan Pygame)

3.6.2 MediaPipe (mediapipe 0.10.14)

MediaPipe adalah framework machine learning yang dikembangkan oleh Google untuk building perception pipelines. Framework ini menyediakan pre-trained models untuk berbagai computer vision tasks. Dalam project ini, MediaPipe digunakan untuk:

MediaPipe Hands:

1. Mendeteksi hingga 21 hand landmarks pada setiap tangan (maksimum 2 tangan)
2. Landmark detection untuk fist recognition untuk bagian wrist, fingertips, dan middle finger MCP
3. Digunakan untuk Fist detection algorithm, punch type classification, dan defense blocking detection

MediaPipe Pose:

1. Mendeteksi 33 body landmarks untuk tracking posisi tubuh pemain
2. Key landmarks pada bagian Eyes (1-6), shoulders (11-12), hips (23-24), dan knees (25-26)
3. Digunakan untuk Body exclusion zones, eye position untuk defense detection, dan posture analysis

MediaPipe Face Mesh:

1. Mendeteksi 468 facial landmarks dengan detail tinggi
2. Digunakan untuk Face bounding box calculation, target area untuk enemy attacks, face exclusion zone untuk hitbox placement

Real-time Processing:

1. Memproses semua deteksi hands, pose, dan face secara bersamaan dengan latency $<$50ms

3.6.3 NumPy (numpy 1.26.4)

NumPy adalah library fundamental untuk scientific computing dalam Python. Dalam project ini Numpy berfungsi untuk:

1. Menyimpan dan mengolah posisi landmark (titik-titik tangan dan tubuh) dalam bentuk array untuk pemrosesan yang lebih cepat
2. Menghitung jarak antara dua titik menggunakan rumus Euclidean dengan fungsi `np.linalg.norm()` untuk mendeteksi kecepatan gerakan
3. Menghitung jarak antara tangan pemain dengan hitbox untuk deteksi tabrakan
4. Menghitung sudut tekukan jari untuk mengetahui apakah tangan sedang mengepal (fist)
5. Menghitung arah dan lintasan gerakan pukulan berdasarkan perpindahan posisi tangan dari waktu ke waktu
6. Menghitung rata-rata sudut jari dan rata-rata jarak ujung jari ke telapak tangan untuk algoritma deteksi kepalan tangan
7. Melakukan perhitungan secara bersamaan (paralel) tanpa perlu menggunakan perulangan satu per satu, sehingga game dapat berjalan dengan lancar (real-time) di 25-30 FPS

Contoh penggunaan dalam project:

```python
# Distance calculation
distance = np.sqrt((x2-x1)**2 + (y2-y1)**2)

# Average angle calculation untuk fist detection
avg_angle = np.mean(angles) if angles else 180

# Array slicing untuk landmark filtering
eye_landmarks = pose_landmarks[1:7]
```

3.6.4 Pygame (pygame 2.5.2)

Pygame adalah library untuk pengembangan game dan multimedia applications. Dalam project ini Pygame berfungsi sebagai:

Audio System Management:

1. pygame.mixer yaitu sistem audio dengan multi-channel support untuk simultaneous sound playback
2. Music Playback untuk Background music dengan loop, volume control, dan fade effects
3. Sound Effects berfungsi dalam Loading dan playing sound effects untuk punch, hit, KO, round transitions
4. Audio Channels yang terdiri dari 8 channels untuk memastikan sounds tidak saling interrupt
5. LUFS Normalization, dimana semua audio dinormalisasi ke -16 LUFS untuk consistent volume levels

Game Display & Rendering:

1. Membuat tampilan game fullscreen dengan resolusi 1280x720 pixels (layar penuh tanpa border)
2. Mengubah hasil tangkapan kamera dari format OpenCV menjadi format Pygame agar bisa ditampilkan di layar game
3. Menggambar elemen-elemen HUD seperti bar health, timer waktu, dan penghitung combo di atas layar permainan
4. Membuat overlay (lapisan) semi-transparan untuk menu dan efek transisi antar scene, sehingga latar belakang masih terlihat samar

Input Handling:

1. Mendeteksi tombol keyboard untuk kontrol game seperti SPACE (mulai game), ESC (pause/resume), Q (keluar), dan tombol panah (navigasi menu)
2. Menggunakan fungsi `pygame.event.get()` untuk menangani banyak event (kejadian) yang terjadi dalam satu frame, seperti klik mouse atau penekanan tombol secara bersamaan
3. Menggunakan `pygame.key.get_pressed()` untuk mengecek apakah tombol sedang ditekan secara terus-menerus (tidak hanya saat pertama kali ditekan), berguna untuk pergerakan kontinu

3.6.5 Library Pendukung Lainnya

Selain library utama di atas, project ini juga menggunakan built-in Python libraries:

1. time
2. random
3. math
4. os

3.6.6 Integrasi dan Dependency Management

Semua library dikelola menggunakan `requirements.txt` dengan version pinning untuk reproducibility:

```
opencv-python==4.10.0.84
mediapipe==0.10.14
numpy==1.26.4
pygame==2.5.2
```

Launcher scripts (`shadow_boxing.bat` dan `shadow_boxing.sh`) secara otomatis memeriksa dan menginstall dependencies sebelum menjalankan game, memastikan seamless user experience.

4 Implementasi

4.1 Struktur Project

Project ini disusun dengan struktur modular yang mengikuti prinsip separation of concerns:

```
shadow-boxing/
|-- main.py                    # Entry point & game loop
|-- requirements.txt           # Dependencies
|-- core/                      # Core utilities
|   |-- config.py              # Game configuration
|   |-- constants.py           # Game constants
|   |-- math_utils.py          # Math helpers
|   |-- utils.py               # Font & image utilities
|-- systems/                   # Game systems
|   |-- vision_system.py       # MediaPipe integration
|   |-- audio_system.py        # Sound manager
|   |-- render_system.py       # Graphics rendering
|   |-- input_processor.py     # Input detection
|-- entities/                  # Game entities
|   |-- player/                # Player state
|   |-- enemy/                 # Enemy AI
|-- game/                      # Game logic
|   |-- game_state.py          # State management
|   |-- hit_box_system.py      # Hitbox generation
|   |-- round_manager.py       # Round system
|   |-- damage_system.py       # Damage calculation
|-- ui/                        # User interface
|   |-- menu_system.py         # Main menu
|   |-- hud_renderer.py        # HUD elements
|   |-- fight_overlay.py       # Round transitions
|   |-- result_screen.py       # Results
|-- assets/                    # Game assets
    |-- font/                  # PressStart2P.ttf
    |-- sprites/               # Game sprites
    |-- sfx/                   # Sound effects
    |-- music/                 # Background music
```

Kode 1: Struktur Direktori Project

4.2 Vision System - MediaPipe Integration

Kode 2 menunjukkan implementasi vision system yang mengintegrasikan MediaPipe Hands, Pose, dan Face Mesh.

```
import cv2
import mediapipe as mp

class VisionSystem:
```

```python
    def __init__(self, game_config):
        self.config = game_config
        # Initialize camera
        self.cap = cv2.VideoCapture(self.config.CAMERA_INDEX)
        self.cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
        self.cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

        # Initialize MediaPipe solutions
        self.mp_hands = mp.solutions.hands
        self.mp_pose = mp.solutions.pose
        self.mp_face_mesh = mp.solutions.face_mesh

        # Configure Hands detector
        self.hands = self.mp_hands.Hands(
            static_image_mode=False,
            max_num_hands=2,
            min_detection_confidence=0.7,
            min_tracking_confidence=0.5
        )

        # Configure Pose detector
        self.pose = self.mp_pose.Pose(
            static_image_mode=False,
            model_complexity=0,  # Lightweight model
            min_detection_confidence=0.5,
            min_tracking_confidence=0.5
        )

        # Configure Face Mesh detector
        self.face_mesh = self.mp_face_mesh.FaceMesh(
            static_image_mode=False,
            max_num_faces=1,
            refine_landmarks=False,
            min_detection_confidence=0.5,
            min_tracking_confidence=0.5
        )

    def get_frame(self):
        """Get frame with all detections"""
        success, frame = self.cap.read()
        if not success:
            return None

        # Flip for mirror effect
        frame = cv2.flip(frame, 1)

        # Convert to RGB for MediaPipe
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Process all detections
        hand_results = self.hands.process(rgb_frame)
        pose_results = self.pose.process(rgb_frame)
        face_results = self.face_mesh.process(rgb_frame)

        return {
            'frame': frame,
            'hands': hand_results,
            'pose': pose_results,
            'face': face_results
        }
```

Kode 2: Vision System Implementation

Penjelasan: menunjukkan implementasi vision system yang mengintegrasikan MediaPipe Hands, Pose, dan Face Mesh. Kelas VisionSystem berfungsi sebagai bridge antara input kamera dan proses deteksi computer vision. Sistem ini menginisialisasi tiga modul MediaPipe secara bersamaan dengan konfigurasi yang disesuaikan untuk performa optimal: Hands detector untuk mendeteksi maksimal 2 tangan dengan 21 landmark per tangan, Pose detector dengan model ringan (complexity=0) untuk tracking 33 titik tubuh, dan Face Mesh untuk mendeteksi 468 landmark wajah. Fungsi get_frame() mengambil frame dari webcam, melakukan mirror flip agar pengalaman lebih intuitif, mengonversi color space dari BGR ke RGB untuk kompatibilitas MediaPipe, kemudian memproses frame tersebut melalui ketiga detector secara simultan dan mengembalikan hasil deteksi beserta frame asli dalam bentuk dictionary untuk digunakan oleh sistem lain dalam game.

4.3 Input Processor - Fist Detection

Kode 3 menunjukkan algoritma deteksi fist menggunakan angle dan distance calculation.

```python
import numpy as np
from core.math_utils import calculate_angle, distance

class InputProcessor:
    def __init__(self, game_config):
        self.config = game_config
        self.fist_angle_threshold = 90  # degrees
        self.fist_distance_threshold = 0.15  # normalized

    def _is_fist(self, landmarks):
        """Detect if hand is making a fist"""
        # Calculate angles for fingers (exclude thumb)
        angles = []
        for tip_id in [8, 12, 16, 20]:  # Index, middle, ring, pinky
            base_id = tip_id - 2
            mid_id = tip_id - 1

            angle = calculate_angle(
                (landmarks[tip_id].x, landmarks[tip_id].y),
                (landmarks[mid_id].x, landmarks[mid_id].y),
                (landmarks[base_id].x, landmarks[base_id].y)
            )
            angles.append(angle)

        avg_angle = np.mean(angles) if angles else 180

        # Calculate distances from fingertips to palm (landmark 9)
        palm = (landmarks[9].x, landmarks[9].y)
        distances = []
        for tip_id in [8, 12, 16, 20]:
            tip = (landmarks[tip_id].x, landmarks[tip_id].y)
            distances.append(distance(tip, palm))

        avg_dist = np.mean(distances) if distances else 1.0

        # Fist detected if both conditions met
        return (avg_angle < self.fist_angle_threshold and
                avg_dist < self.fist_distance_threshold)
```

Kode 3: Fist Detection Algorithm

Penjelasan: Kelas InputProcessor mengimplementasikan algoritma deteksi kepalan tangan menggunakan dua metrik utama: sudut tekukan jari dan jarak ujung jari ke telapak tangan. Sistem menghitung sudut antara tiga titik pada setiap jari (ujung, tengah, pangkal) untuk keempat jari (telunjuk, tengah, manis, kelingking) menggunakan fungsi calculate_angle(). Rata-rata sudut di bawah

90 derajat mengindikasikan jari tertekuk. Selanjutnya, sistem menghitung jarak Euclidean dari setiap ujung jari ke pusat telapak tangan (landmark 9). Jarak rata-rata di bawah 0.15 (dalam koordinat normalized) menunjukkan jari berada dekat dengan telapak. Kepalan tangan terdeteksi ketika kedua kondisi terpenuhi secara bersamaan, menghasilkan deteksi yang akurat dengan false positive rate rendah karena membutuhkan konfirmasi ganda dari geometri tangan.

4.4 Hitbox System - Sequential Spawning

Kode 4 menunjukkan implementasi sequential hitbox spawning dengan position-based placement dan exclusion zones.

```python
import random
import math

class HitBoxSystem:
    def __init__(self, game_config):
        self.config = game_config
        self.hitbox_radius = 65
        self.active_hitboxes = []
        self.sequential_mode = True

    def generate_hitboxes(self, combo_sequence, face_bbox,
                          pose_landmarks):
        """Generate sequential hitboxes dengan exclusion zones"""
        # Define face exclusion zone (expanded 100px)
        face_zone = None
        if face_bbox:
            fx, fy, fw, fh = face_bbox
            face_zone = {
                'x1': max(0, fx - 100),
                'y1': max(0, fy - 100),
                'x2': min(1280, fx + fw + 100),
                'y2': min(720, fy + fh + 100)
            }

        # Define body exclusion zones (shoulders & hips)
        body_zones = []
        if pose_landmarks:
            # Landmarks 11,12 (shoulders), 23,24 (hips)
            for idx in [11, 12, 23, 24]:
                if idx < len(pose_landmarks):
                    lm = pose_landmarks[idx]
                    x = int(lm.x * 1280)
                    y = int(lm.y * 720)
                    body_zones.append({
                        'cx': x, 'cy': y, 'radius': 120
                    })

        # Generate hitboxes
        for i, punch_type in enumerate(combo_sequence):
            placed = False
            attempts = 0

            while not placed and attempts < 100:
                attempts += 1

                # Position-based placement
                if punch_type == "JAB":
                    # Left side (0-50% width)
                    x = random.randint(50, 640 - 130)
                else:  # CROSS or HOOK
```

```
                    # Right side (50-100% width)
                    x = random.randint(640, 1280 - 180)

                y = random.randint(200, 720 - 180)
                center_x = x + 65
                center_y = y + 65

                # Check exclusion zones
                if self._in_exclusion_zone(center_x, center_y,
                                           face_zone, body_zones):
                    continue

                # Check overlap with existing hitboxes
                if self._check_overlap(center_x, center_y):
                    continue

                # Place hitbox
                self.active_hitboxes.append({
                    'id': i,
                    'x': x, 'y': y,
                    'center_x': center_x,
                    'center_y': center_y,
                    'punch_type': punch_type,
                    'active': (i == 0)  # Only first active
                })
                placed = True
```

Kode 4: Hitbox Generation dengan Exclusion Zones

Penjelasan: Kelas HitBoxSystem mengelola munculnya target pukulan secara berurutan dengan mempertimbangkan zona terlarang dan penempatan berbasis posisi. Sistem membuat dua jenis exclusion zone, yaitu zona wajah yang diperluas 100 piksel dari bounding box wajah untuk menghindari target menutupi area wajah pemain, dan zona tubuh dengan radius 120 piksel di sekitar bahu (landmarks 11-12) dan pinggul (landmarks 23-24) untuk menghindari area tubuh natural. Setiap target ditempatkan berdasarkan jenis pukulan, dengan JAB spawn di sisi kiri layar (0-50% lebar) dan CROSS/HOOK di sisi kanan (50-100% lebar). Sistem melakukan hingga 100 percobaan untuk menemukan posisi valid yang tidak bertabrakan dengan exclusion zones atau target lain yang sudah ada. Target dibuat dalam mode sequential di mana hanya target pertama yang aktif, dan target berikutnya akan diaktifkan setelah target sebelumnya dipukul, menciptakan gameplay yang terstruktur dan progressive.

4.5 Defense System

Kode 5 menunjukkan implementasi defense detection menggunakan fingertips proximity ke eye area.

```
class InputProcessor:
    def _check_defense(self, game_state):
        """Check defense: fingertips near eye area"""
        # Get eye landmarks (pose landmarks 1-6)
        eye_landmarks_idx = [1, 2, 3, 4, 5, 6]
        eye_positions = []

        for idx in eye_landmarks_idx:
            if idx < len(game_state.pose_landmarks):
                lm = game_state.pose_landmarks[idx]
                eye_x = int(lm.x * 1280)
                eye_y = int(lm.y * 720)
                eye_positions.append((eye_x, eye_y))

        if not eye_positions:
            return

```

```
        # Calculate average eye position
        avg_eye_x = sum(p[0] for p in eye_positions) // len(eye_positions)
        avg_eye_y = sum(p[1] for p in eye_positions) // len(eye_positions)

        # Defense threshold: 150px radius
        defense_threshold = 150

        # Check fingertips (landmarks 8,12,16,20) from both hands
        defending_hands = 0

        for hand_state in [self.hand_states['Left'],
                           self.hand_states['Right']]:
            if hand_state['fingertips']:
                for tip_pos in hand_state['fingertips']:
                    dx = tip_pos[0] - avg_eye_x
                    dy = tip_pos[1] - avg_eye_y
                    dist = (dx**2 + dy**2)**0.5

                    if dist < defense_threshold:
                        defending_hands += 1
                        break

        # Defense active if at least one hand covering
        self.defense_active = (defending_hands >= 1)
```

Kode 5: Defense Detection Algorithm

Penjelasan: Metode _check_defense() mengimplementasikan sistem deteksi pertahanan pemain dengan mengukur kedekatan ujung jari ke area mata. Sistem pertama-tama mengumpulkan posisi eye landmarks (pose landmarks 1-6) dan menghitung rata-rata posisi mata sebagai titik referensi. Dengan threshold jarak 150 piksel, sistem memeriksa apakah ujung jari dari kedua tangan (landmarks 8, 12, 16, 20) berada dalam radius ini dari posisi rata-rata mata. Perhitungan menggunakan jarak Euclidean antara setiap ujung jari dan posisi mata, dengan formula $(dx^2 + dy^2)^{0.5}$. Pertahanan dianggap aktif ketika minimal satu tangan memiliki ujung jari yang berada dalam threshold, yang mengindikasikan pemain sedang menutup wajah untuk blocking. Pendekatan ini lebih akurat dibanding palm detection karena fokus pada fingertips yang secara natural berada dekat mata saat pemain melakukan blocking stance, menghasilkan false positive rate di bawah 5%.

5 Hasil dan Pembahasan

5.1 Hasil Implementasi

Game Shadow Boxing berhasil diimplementasikan dengan fitur-fitur lengkap:

1. **Core Features:**

   Game ini menggunakan teknologi MediaPipe untuk mendeteksi gerakan tangan, tubuh, dan wajah pemain secara real-time (langsung tanpa jeda). Sistem pertarungan dirancang dengan tiga fase yang bergantian: fase pemain menyerang (Player Attack), fase peringatan serangan musuh (Enemy Attack Warning), dan fase musuh menyerang (Enemy Attack). Target pukulan (hitbox) muncul secara berurutan satu per satu, di mana target berikutnya baru muncul setelah target sebelumnya berhasil dipukul (hit-to-spawn mechanism).

   Pemain dapat memilih tiga tingkat kesulitan (mudah, sedang, sulit) dengan parameter yang berbeda-beda untuk setiap level. Setiap permainan terdiri dari 3 ronde pertarungan dengan periode istirahat di antara ronde. Untuk bertahan dari serangan musuh, pemain bisa melakukan blocking (menutup wajah dengan tangan) atau dodging (menghindar dengan menggerakkan

kepala). Seluruh suara efek game telah dinormalisasi ke standar -16 LUFS sehingga volume suara konsisten dan tidak ada yang terlalu keras atau pelan.

2. **Game Mechanics:**

   Sistem mendeteksi kepalan tangan (fist) dengan menganalisis dua hal: sudut tekukan jari dan jarak ujung jari ke telapak tangan. Ketika pemain memukul, sistem menghitung apakah tangan mengenai target menggunakan perhitungan jarak berbentuk lingkaran. Target pukulan (hitbox) ditempatkan berdasarkan jenis pukulan: JAB muncul di sisi kiri layar, sedangkan CROSS dan HOOK muncul di sisi kanan.

   Untuk memastikan target tidak menutupi wajah atau tubuh pemain, sistem membuat zona terlarang di area wajah (100 piksel) dan bagian tubuh penting seperti bahu dan pinggul (120 piksel). Jika target gagal ditempatkan karena zona terlarang terlalu banyak, sistem memiliki tiga tingkat cadangan: pertama dengan aturan lengkap, kedua dengan aturan yang lebih longgar, dan ketiga memaksa target muncul dengan aturan minimal. Pertahanan pemain dideteksi ketika ujung jari berada dalam jarak 150 piksel dari area mata. Damage yang diberikan bervariasi secara acak tergantung tingkat kesulitan dan jenis pukulan, dengan bonus 1.1× untuk pukulan terakhir dalam kombo.

3. **User Interface:**

   Antarmuka game dimulai dengan menu utama yang memungkinkan pemain memilih tingkat kesulitan sebelum bermain. Selama pertarungan berlangsung, layar menampilkan HUD (Heads-Up Display) yang berisi bar kesehatan untuk melihat sisa nyawa pemain dan musuh, timer untuk menghitung waktu ronde, penghitung kombo untuk melacak jumlah pukulan beruntun, dan indikator fase untuk menunjukkan giliran siapa yang sedang menyerang. Saat transisi antar ronde, muncul overlay bertuliskan "ROUND X" atau "FIGHT!" untuk memberi jeda visual. Setelah pertandingan selesai, layar hasil menampilkan statistik lengkap termasuk jumlah pukulan yang berhasil mengenai target (hits landed) dan total damage yang diberikan ke musuh, sehingga pemain dapat melihat performa mereka.

5.2 Technical Achievements

1. **Hitbox Generation Success Rate: 100%**

   Sistem berhasil mencapai tingkat keberhasilan 100% dalam menempatkan target pukulan (hitbox) dengan menggunakan mekanisme tiga tingkat cadangan. Tingkat pertama menggunakan aturan lengkap dengan zona terlarang penuh dan jarak minimal 30 piksel antar target. Jika gagal, tingkat kedua mencoba dengan aturan yang lebih longgar: zona wajah diperkecil menjadi 50 piksel dan zona tubuh dihilangkan, dengan jarak minimal 20 piksel. Jika masih gagal, tingkat ketiga memaksa target muncul dengan pemeriksaan minimal untuk memastikan game tetap bisa dimainkan.

   Sebelum sistem cadangan ini diterapkan, tingkat keberhasilan hanya sekitar 70% untuk kombinasi 3 pukulan atau lebih, artinya sering kali target tidak bisa muncul dan permainan terganggu. Setelah implementasi sistem tiga tingkat ini, semua kombinasi pukulan (dari 2 hingga 4 target) berhasil ditempatkan 100%, sehingga game selalu berjalan lancar tanpa ada target yang gagal muncul.

2. **Defense Detection Accuracy**

   Defense system menggunakan fingertips (bukan palm) untuk mengurangi false positives:

   - **Block detection**: Fingertips dalam 150px dari eye area
   - **Damage reduction**: 80% (20% damage diterima)

- **False positive rate**: $< 5\%$ (berdasarkan testing manual)

3. **Performance Optimization**

Tabel 2: Performance Metrics

Metric	Value
Average FPS	25-30 fps
Detection Latency	<50ms
Input-to-Effect Latency	<100ms
Memory Usage	500MB

5.3 Game Balance Analysis

Testing dilakukan pada tiga difficulty levels:

Tabel 3: Difficulty Balance Testing

Metric	EASY	MEDIUM	HARD
Avg Hits to KO Enemy	7-8	10-12	15-18
Avg Time per Round	30-40s	40-50s	50-60s
Player Damage Taken	20-30	40-60	70-90
Win Rate (Casual)	90%	60%	30%

Observations:

- **EASY**: Cocok untuk pemula, enemy mudah di-KO dengan damage tinggi (12-15)
- **MEDIUM**: Balanced gameplay, membutuhkan strategi defense
- **HARD**: Challenging, membutuhkan timing dan defense yang baik

5.4 Tantangan dan Solusi

Challenge 1: Hitbox Generation Failures

Problem: Sequential hitboxes gagal generate karena constraint terlalu ketat (face zone, body zones, overlap, last position). Later hitboxes (index 2-3) sering gagal mencari posisi valid.

Solution:

(a) Reset attempts counter per-hitbox (bukan global)

(b) Implementasi 3-tier fallback dengan relaxed constraints

(c) Force placement sebagai last resort

Result: Hitbox generation success rate meningkat dari 70% menjadi 100%.

Challenge 2: Defense False Positives

Problem: Defense terdeteksi saat tangan tidak benar-benar menutup wajah (palm detection terlalu luas).

Solution:

(a) Ganti dari palm detection menjadi fingertips detection

(b) Gunakan landmarks 8,12,16,20 (fingertips only)

(c) Threshold 150px dari average eye position

Result: False positive rate turun dari 20% menjadi <5%.

Challenge 3: Body Landmark Collisions

Problem: Hitbox sering spawn di area di mana tangan naturally idle (dekat shoulders, hips).

Solution:

(a) Tambahkan body exclusion zones

(b) Landmarks 11,12 (shoulders), 23,24 (hips)

(c) Radius 120px per landmark

Result: Hitbox placement menjadi lebih natural dan comfortable untuk dipukul.

Challenge 4: Performance dengan Multiple MediaPipe Models

Problem: Running Hands + Pose + Face Mesh simultaneously menurunkan FPS drastis (dari 30fps ke 15fps).

Solution:

(a) Gunakan model_complexity=0 untuk Pose (lightweight)

(b) Disable refine_landmarks untuk Face Mesh

(c) Set buffer size=1 untuk camera capture

Result: FPS stabil di 25-30fps dengan all detections active.

5.5 Instalasi dan Cara Penggunaan Program

5.5.1 Persyaratan Sistem

Sebelum menjalankan game Shadow Boxing, pastikan sistem memenuhi persyaratan berikut:

- **Sistem Operasi**: Windows 10/11, Linux (Ubuntu 20.04+), atau macOS 10.15+
- **Python**: Versi 3.10 atau lebih baru
- **Webcam**: Resolusi minimal 720p dengan frame rate 30 FPS
- **RAM**: Minimal 4GB (direkomendasikan 8GB)
- **Storage**: Minimal 500MB ruang kosong
- **Pencahayaan**: Ruangan dengan pencahayaan yang cukup untuk deteksi MediaPipe optimal

5.5.2 Instalasi Manual

Untuk instalasi manual, ikuti langkah-langkah berikut:

1. Clone Repository

```
git clone https://github.com/Aziz097/shadow-boxing.git
cd shadow-boxing
```

2. Install Dependencies

```
pip install -r requirements.txt
```

File `requirements.txt` berisi:

```
opencv-python==4.10.0.84
mediapipe==0.10.14
numpy==1.26.4
pygame==2.5.2
```

3. Jalankan Game

```
python -u main.py
```

5.5.3 Instalasi Menggunakan Launcher Scripts

Untuk kemudahan, project ini menyediakan launcher scripts yang secara otomatis memeriksa dan menginstall dependencies:

Windows:

```
shadow_boxing.bat
```

Linux/macOS:

```
chmod +x shadow_boxing.sh
./shadow_boxing.sh
```

Launcher scripts akan:

(a) Memeriksa apakah Python terinstall
(b) Memeriksa ketersediaan library yang diperlukan (opencv, mediapipe, pygame)
(c) Menginstall library yang belum tersedia secara otomatis
(d) Menjalankan game setelah semua dependencies terpenuhi

5.5.4 Cara Penggunaan

1. Main Menu

(a) Setelah game dimulai, pemain akan melihat main menu dengan latar belakang feed kamera
(b) Gunakan tombol **Arrow Up/Down** untuk navigasi menu
(c) Pilih tingkat kesulitan: **Easy**, **Medium**, atau **Hard**
(d) Tekan **ENTER** untuk memulai game
(e) Tekan **Q** untuk keluar

2. Gameplay

(a) **Player Attack Phase**:
  i. Target pukulan (hitbox) akan muncul di layar
  ii. Kepalkan tangan dan pukul target yang muncul
  iii. Target baru akan muncul setelah target sebelumnya dipukul
  iv. Waktu terbatas (2.5-3.5 detik tergantung difficulty)

(b) **Enemy Attack Phase**:
  i. Peringatan visual akan muncul sebelum enemy menyerang
  ii. **Block**: Tutup wajah dengan kedua tangan untuk mengurangi damage 80%
  iii. **Dodge**: Gerakkan kepala keluar dari area target untuk menghindari damage 100%

(c) **Kontrol Game**:

i. **Q**: Keluar ke menu utama

3. Tips Bermain

(a) Pastikan posisi badan menghadap kamera dengan pencahayaan yang cukup

(b) Berdirilah pada jarak 1-2 meter dari kamera

(c) Untuk deteksi fist yang optimal, kepalkan tangan dengan kuat

(d) Perhatikan indikator fase untuk mengetahui kapan giliran menyerang atau bertahan

(e) Manfaatkan warning phase untuk mempersiapkan defense

(f) Kombinasi block dan dodge untuk meminimalkan damage yang diterima

6 Kesimpulan dan Saran

6.1 Kesimpulan

(a) Game Shadow Boxing interaktif berhasil diimplementasikan dengan memanfaatkan MediaPipe untuk deteksi real-time hand, pose, dan face landmarks

(b) Combat system dengan phase-based gameplay (player attack vs enemy attack) berfungsi dengan baik dan menciptakan gameplay yang engaging

(c) Sequential hitbox spawning dengan position-based placement dan exclusion zones berhasil memberikan pengalaman gameplay yang natural

(d) Defense system dengan fingertips proximity detection mampu mendeteksi blocking dengan akurasi tinggi (false positive $<5\%$)

(e) Tiga difficulty levels (Easy, Medium, Hard) telah dibalance dengan baik untuk berbagai tingkat pemain

(f) Hitbox generation dengan 3-tier fallback mechanism mencapai success rate 100%

(g) Performance game stabil di 25-30 FPS dengan multiple MediaPipe models berjalan simultaneously

(h) Audio system dengan LUFS normalization memberikan consistent sound levels untuk better player experience

6.2 Saran Pengembangan

(a) **Machine Learning Enhancement**: Implementasi ML model untuk klasifikasi punch type yang lebih akurat berdasarkan trajectory dan velocity patterns

(b) **Combo Detection Advanced**: Mendeteksi player combo patterns dan memberikan bonus damage untuk execution yang baik

(c) **Training Mode**: Mode latihan dengan feedback untuk kualitas teknik (punch speed, form, timing)

(d) **Performance Optimization**:

- Multi-threading untuk MediaPipe processing
- GPU acceleration untuk pose estimation
- Adaptive resolution berdasarkan device capability

(e) **Multiplayer Mode**: Local co-op atau online multiplayer dengan synchronization

(f) **Advanced AI**:

- Adaptive difficulty berdasarkan player skill
- Pattern recognition untuk counter player strategies
- Multiple enemy types dengan attack patterns berbeda

(g) **Gameplay Features**:

- Replay system dengan slow motion
- Achievement dan progression system
- Character customization
- Tournament mode dengan bracket

(h) **Cross-Platform**: Port ke mobile (Android/iOS) dan web (WebAssembly)

(i) **Accessibility**:

- Configurable MediaPipe confidence thresholds
- Adjustable difficulty parameters
- Colorblind-friendly UI

Kolaborasi Tim:

(a) **Brainstorming**: Semua anggota berpartisipasi dalam design decisions dan problem solving

(b) **Code Review**: Cross-review untuk maintain code quality

(c) **Testing**: Kolaboratif testing untuk identify bugs dan balance issues

(d) **Documentation**: Shared responsibility untuk README dan technical documentation

7 Development Logbook

Project dikembangkan secara iteratif dengan milestone-milestone berikut:

Tabel 4: Timeline Pengembangan

Tanggal	**Kegiatan**	**Hasil / Progress**
10/28/2024	Project Initialization	Repository setup, team coordination, scope definition
11/09/2024	Asset Selection & Flow Design	Fixed assets (sound, sprites, font), game flow wireframe, UI mockups
11/10/2024	Core Implementation	**Systems:** MediaPipe integration, audio manager, visual effects. **Mechanics:** Punch detection, defense system, hitbox generation, combo patterns. **UI:** Menu system, HUD renderer, transitions. **Features:** 3-level difficulty, text caching, fullscreen support
11/17/2024	Audio Processing & Polish	**Audio:** WAV conversion (44100 Hz, mono, 16-bit PCM), LUFS normalization (-16 dB). **UI Polish:** Font size adjustments, menu alignment fixes. **Automation:** Launcher scripts (bat/sh), auto dependency check
11/18/2024	Code Refactoring	Flatten folder structure, remove dead code, module docstrings, import optimization, clean code principles (SRP, DRY), clear naming conventions
11/24/2024	Combat System Overhaul	**Major Changes:** Timing-based → sequence system, hit-to-spawn mechanism, immediate damage, last hit 1.1× bonus, player attack 4s. **Bug Fixes:** Phase transition, sequential hitbox consistency, phantom outline rendering
11/25/2024	Advanced Features & Bug Fixes	**Improvements:** Hit detection expanded (landmarks 3-20), position-based hitbox, body exclusion zones, attack duration limit (4s max). **Bug Fixes:** Hitbox generation failures, 3-tier fallback, per-hitbox counter, collision spacing (50px → 30px). **Cleanup:** Unused variables removed, config audit, debug logging

8 Pembagian Tugas

Tabel 5: Pembagian Tugas Anggota Kelompok

Nama	**Kontribusi**
Aziz Kurniawan	Lead Developer: Game logic implementation, MediaPipe integration dan tuning, hitbox system development, combat mechanics, sequential spawning algorithm, bug fixing dan optimization, quality assurance testing
Harisya Miranti	UI/UX Designer: Visual effects implementation, HUD renderer design, menu system, fight overlay animations, sprite integration, asset creation dan curation, user interface design, testing dan user experience evaluation
Muhammad Yusuf	Systems Architect: MediaPipe integration, game flow architecture, audio system implementation, vision system optimization, input processing, systems integration, documentation, laporan dan presentasi

Referensi

(a) Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M. G., Lee, J., Chang, W.-T., Hua, W., Georg, M., & Grundmann, M. (2019). *MediaPipe: A Framework for Building Perception Pipelines*. arXiv preprint arXiv:1906.08172. https://arxiv.org/abs/1906.08172

(b) Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L., & Grundmann, M. (2020). *MediaPipe Hands: On-device Real-time Hand Tracking*. arXiv preprint arXiv:2006.10214. https://arxiv.org/abs/2006.10214

(c) Bazarevsky, V., Grishchenko, I., Raveendran, K., Zhu, T., Zhang, F., & Grundmann, M. (2020). *BlazePose: On-device Real-time Body Pose Tracking*. arXiv preprint arXiv:2006.10204. https://arxiv.org/abs/2006.10204

(d) OpenCV Team. (2024). *OpenCV: Open Source Computer Vision Library*. https://opencv.org/

(e) Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., & Oliphant, T. E. (2020). *Array programming with NumPy*. Nature, 585(7825), 357-362. https://doi.org/10.1038/s41586-020-2649-2

(f) Pygame Community. (2024). *Pygame: Python Game Development*. https://www.pygame.org/

(g) Pisharady, P. K., & Saerbeck, M. (2015). *Recent methods and databases in vision-based hand gesture recognition: A review*. Computer Vision and Image Understanding, 141, 152-165. https://doi.org/10.1016/j.cviu.2015.08.004

(h) Rautaray, S. S., & Agrawal, A. (2015). *Vision based hand gesture recognition for human computer interaction: a survey*. Artificial Intelligence Review, 43(1), 1-54. doi:10.1007/s10462-012-9356-9

(i) Maggioni, E., Cobden, R., & Obrist, M. (2020). *Design and development challenges in multimodal gesture recognition*. ACM Transactions on Computer-Human Interaction (TOCHI), 27(4), 1-43. https://doi.org/10.1145/3397869