Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama: **Ferdana Al-Hakim**
Tugas Ke: **Final Project**
Mata Kuliah: **Sistem/Teknologi Multimedia (IF25-40305)**
Tanggal: Desember 2024

HANDBEATS: GESTURE RHYTHM GAME

Game Ritme Berbasis Deteksi Gesture Tangan

1 Pendahuluan

HandBeats adalah game interaktif berbasis kamera yang mengintegrasikan tiga komponen pemrosesan multimedia: *image processing*, *video processing*, dan *audio processing*. Game ini menampilkan zona instrumen musik seperti "Kick", "Snare", dan "Hi-Hat" di layar, di mana pemain harus mengetuk area tersebut menggunakan gesture tangan dengan timing yang tepat mengikuti irama musik.

1.1 Latar Belakang

Perkembangan teknologi *computer vision* dan *machine learning* memungkinkan deteksi gesture tangan secara real-time dengan akurasi tinggi. Teknologi ini dapat dimanfaatkan untuk menciptakan pengalaman interaktif yang menarik dalam bentuk rhythm game yang tidak memerlukan perangkat input tambahan selain webcam.

1.2 Tujuan

Tujuan dari pembuatan project ini adalah:

- Mengimplementasikan pemrosesan image untuk deteksi tangan menggunakan MediaPipe Hands
- Mengintegrasikan pemrosesan video real-time dengan overlay grafis game
- Menerapkan pemrosesan audio untuk musik loop seamless dan sound effects
- Menciptakan gameplay yang responsif dengan sistem scoring dan feedback visual

2 Komponen Multimedia

Project HandBeats mengintegrasikan tiga komponen utama pemrosesan multimedia yang bekerja secara bersamaan untuk menciptakan pengalaman bermain yang interaktif.

2.1 Image Processing

Komponen image processing bertanggung jawab untuk mendeteksi dan melacak posisi tangan pemain secara real-time menggunakan MediaPipe Hands.

2.1.1 MediaPipe Hands Detection

MediaPipe Hands adalah solusi machine learning untuk deteksi tangan yang dapat mendeteksi hingga 21 landmark pada setiap tangan. Implementasi pada HandBeats menggunakan confidence threshold 0.7 untuk memastikan deteksi yang akurat.

```
class HandTracker:
    def __init__(self):
        self.mp_hands = mp.solutions.hands
        self.hands = self.mp_hands.Hands(
            static_image_mode=False,
            max_num_hands=2,
            min_detection_confidence=0.7,
            min_tracking_confidence=0.7
        )
        self.mp_drawing = mp.solutions.drawing_utils
```

Kode 1: Inisialisasi MediaPipe Hands

2.1.2 Bounding Box Calculation

Setelah landmark terdeteksi, sistem menghitung bounding box untuk setiap tangan yang digunakan dalam collision detection dengan zona target.

```
def calculate_hand_bbox(self, hand_landmarks, width, height):
    x_coords = [lm.x for lm in hand_landmarks.landmark]
    y_coords = [lm.y for lm in hand_landmarks.landmark]

    min_x = int(min(x_coords) * width)
    max_x = int(max(x_coords) * width)
    min_y = int(min(y_coords) * height)
    max_y = int(max(y_coords) * height)

    return {
        'x': min_x,
        'y': min_y,
        'width': max_x - min_x,
        'height': max_y - min_y
    }
```

Kode 2: Kalkulasi Bounding Box Tangan

2.2 Video Processing

Video processing menangani capture frame dari webcam, konversi color space, dan rendering overlay grafis game ke layar.

2.2.1 Camera Capture

Sistem menggunakan OpenCV untuk capture video dari webcam dengan resolusi 1280x720 (HD 720p) untuk memastikan kualitas deteksi yang optimal.

```
self.camera = cv2.VideoCapture(0)
self.camera.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
self.camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
```

Kode 3: Inisialisasi Kamera

2.2.2 Frame Processing dan Overlay

Setiap frame video yang ditangkap diproses untuk deteksi tangan, kemudian di-composite dengan elemen grafis game seperti falling objects, score, dan visual feedback.

```
def _render_camera_feed(self, frame):
    # Konversi BGR ke RGB
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Resize dengan aspect fill
    scale = max(target_width / src_w, target_height / src_h)
    new_w = int(src_w * scale)
    new_h = int(src_h * scale)

    frame_resized = cv2.resize(frame_rgb, (new_w, new_h))

    # Center crop
    frame_cropped = frame_resized[crop_y:crop_y + target_height,
                                  crop_x:crop_x + target_width]

    # Convert ke pygame surface
    frame_transposed = np.transpose(frame_cropped, (1, 0, 2))
    frame_surface = pygame.surfarray.make_surface(frame_transposed)

    self.screen.blit(frame_surface, (0, 0))
```

Kode 4: Pemrosesan Video Frame

2.3 Audio Processing

Audio processing mengelola musik background loop dan sound effects instrumen yang dimainkan saat pemain berhasil mengenai target.

2.3.1 Seamless Music Looping

Musik background di-loop secara seamless tanpa jeda menggunakan pygame.mixer dengan beat duration 9 detik.

```
def play_main_beat(self, loops=-1):
    if not self.music_playing:
        pygame.mixer.music.play(loops)  # -1 untuk infinite loop
        self.music_playing = True
        print("Main beat started (looping)")
```

Kode 5: Music Looping

2.3.2 Polyphonic Sound Effects

Sistem audio mendukung polyphonic playback, memungkinkan multiple sound effects dimainkan bersamaan saat pemain mengenai multiple targets secara simultan.

```
def play_instrument(self, instrument: str):
    if instrument in self.sounds and self.sounds[instrument]:
        self.sounds[instrument].play()

def play_hit_sound(self, hit_type: str = 'good'):
    volume = {
        'perfect': 1.0,
        'good': 0.8,
        'ok': 0.6
    }.get(hit_type, 0.8)
```

```
    sound = self.sounds['hit']
    sound.set_volume(volume)
    sound.play()
```

Kode 6: Instrument Sound Playback

3 Implementasi Sistem

3.1 Arsitektur Sistem

HandBeats dibangun dengan arsitektur modular yang memisahkan setiap komponen ke dalam module terpisah untuk maintainability dan scalability. Struktur project dapat dilihat pada kode 7.

```
handbeats-rhythm-game/
          main.py                  # Entry point
          config/
                constants.py         # Konstanta warna, zona, path
                settings.py          # Pengaturan difficulty
                beatmap.py           # Generator pattern
          src/
                game_manager.py      # Orchestrator utama
                audio_manager.py     # Audio processing
                hand_tracker.py      # Image processing
                lane.py              # Target zones
                falling_object.py    # Rhythm notes
                collision.py         # Hit detection
                score_manager.py     # Scoring system
          ui/
                menu_screen.py       # Main menu
                game_screen.py       # Game UI
                result_screen.py     # Result stats
          assets/
              audio/               # Music dan SFX
              image/               # Icon instrumen
```

Kode 7: Struktur Project

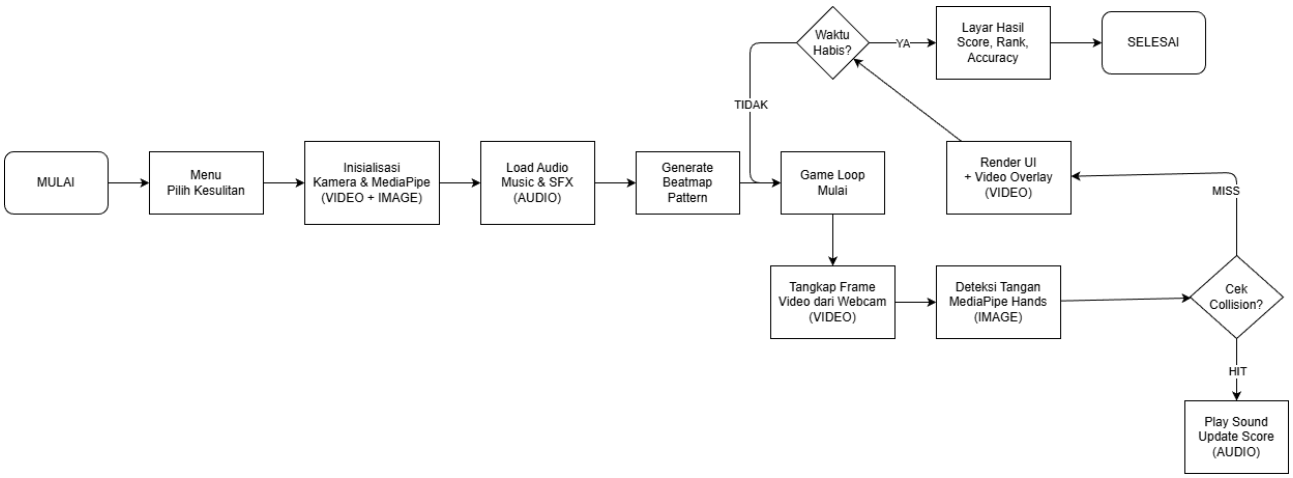
3.2 Alur Sistem

Alur kerja sistem HandBeats dapat dilihat pada flowchart di Gambar 1. Sistem dimulai dari menu pemilihan difficulty, kemudian melakukan inisialisasi komponen multimedia, masuk ke game loop utama, dan diakhiri dengan layar hasil.

3.3 Game Loop

Game loop merupakan inti dari sistem yang mengintegrasikan ketiga komponen multimedia secara real-time.

```
def update_gameplay(self, dt: float):
    self.game_time += dt

    # VIDEO PROCESSING: Capture frame
    ret, frame = self.camera.read()

    # IMAGE PROCESSING: Detect hands
    frame, hand_results, face_results = self.hand_tracker.process_frame(frame)

    # Get positions untuk collision
```



Gambar 1: Flowchart Alur Sistem HandBeats

```
    fingertip_positions = self.hand_tracker.get_fingertip_positions(
        hand_results, SCREEN_WIDTH, SCREEN_HEIGHT
    )

    # Calculate velocities untuk gesture detection
    fingertip_velocities = self.hand_tracker.calculate_velocity(
        fingertip_positions,
        self.hand_tracker.prev_fingertip_positions
    )

    # Update lanes dengan velocity check
    active_lanes = self.lane_manager.check_collisions_with_velocity(
        fingertip_positions,
        chin_position,
        fingertip_velocities,
        chin_velocity,
        self.hand_tracker.velocity_threshold
    )

    # COLLISION DETECTION
    objects_in_zone = self.falling_objects.get_objects_in_hit_zone()
    hit_results = self.collision_detector.check_multiple_objects(
        objects_in_zone, active_lanes
    )

    # Process hit results
    for hit_result in hit_results:
        self.score_manager.add_hit(hit_result)

        # AUDIO PROCESSING: Play sounds
        self.audio_manager.play_hit_sound(hit_result.rating.lower())
        self.audio_manager.play_instrument(hit_result.instrument)
```

Kode 8: Main Game Loop

3.4 Collision Detection

Sistem collision detection menggunakan velocity-based gesture recognition untuk mencegah "idle farming" - pemain harus benar-benar melakukan gerakan cepat untuk mengenai target.

```
def check_collisions_with_velocity(self, fingertip_positions,
                                   fingertip_velocities,
```

```
                                    velocity_threshold):
    active_lanes = {}

    for lane in self.lanes:
        lane_active = False

        for hand_label, fingertip_zone in fingertip_positions.items():
            if lane.check_hand_collision(fingertip_zone):
                # Check velocity - harus bergerak cukup cepat!
                if (hand_label in fingertip_velocities and
                    fingertip_velocities[hand_label] >= velocity_threshold):
                    lane_active = True
                    active_lanes[lane.instrument] = hand_label
                    break

        lane.activate(lane_active)

    return active_lanes
```

Kode 9: Velocity-Based Collision Detection

3.5 Difficulty System

Game menyediakan tiga tingkat kesulitan yang mempengaruhi kecepatan falling objects, timing window, dan kompleksitas pattern.

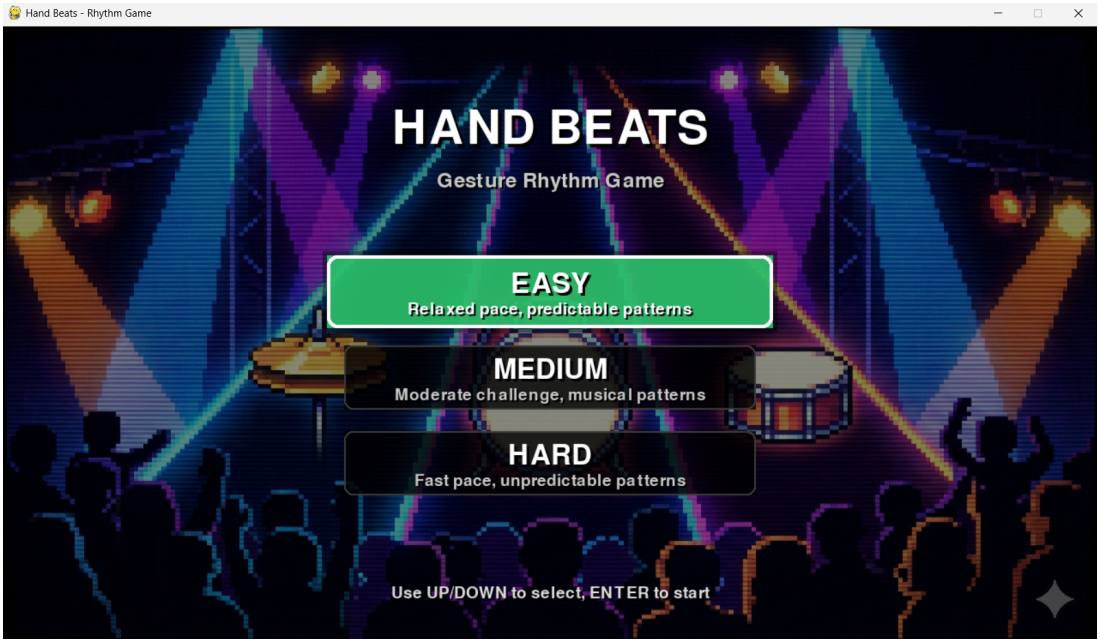
```
EASY = {
    'falling_speed': 5.0,
    'pattern_type': 'simple',
    'perfect_window': 120,  #  120ms
    'good_window': 200,     #  200ms
    'ok_window': 300        #  300ms
}

MEDIUM = {
    'falling_speed': 7.0,
    'pattern_type': 'smart',
    'perfect_window': 80,
    'good_window': 150,
    'ok_window': 230
}

HARD = {
    'falling_speed': 9.5,
    'pattern_type': 'complex',
    'perfect_window': 60,
    'good_window': 120,
    'ok_window': 180
}
```

Kode 10: Difficulty Settings

4 User Interface

4.1 Menu Screen

Menu screen menampilkan pilihan difficulty dengan visual yang menarik dan mudah dipahami. Background menggunakan pixel art style dengan overlay semi-transparent untuk keterbacaan teks yang optimal (Gambar 2).



Gambar 2: Menu Screen dengan Difficulty Selection

4.2 Gameplay Screen

Layar gameplay menampilkan video feed dari kamera sebagai background fullscreen dengan overlay elemen game seperti score, combo, timer, falling objects, dan visual feedback (Gambar 3).

Elemen-elemen UI pada gameplay:

- **Score**: Menampilkan skor saat ini di pojok kiri atas
- **Combo**: Menampilkan combo count di tengah atas dengan multiplier
- **Timer**: Countdown waktu permainan di pojok kanan atas
- **Lanes**: Tiga zona target (Kick, Snare, Hi-Hat) dengan icon instrumen
- **Falling Objects**: Notes yang jatuh dari atas ke zona target
- **Hand Indicators**: Lingkaran biru/oranye menunjukkan posisi fingertip

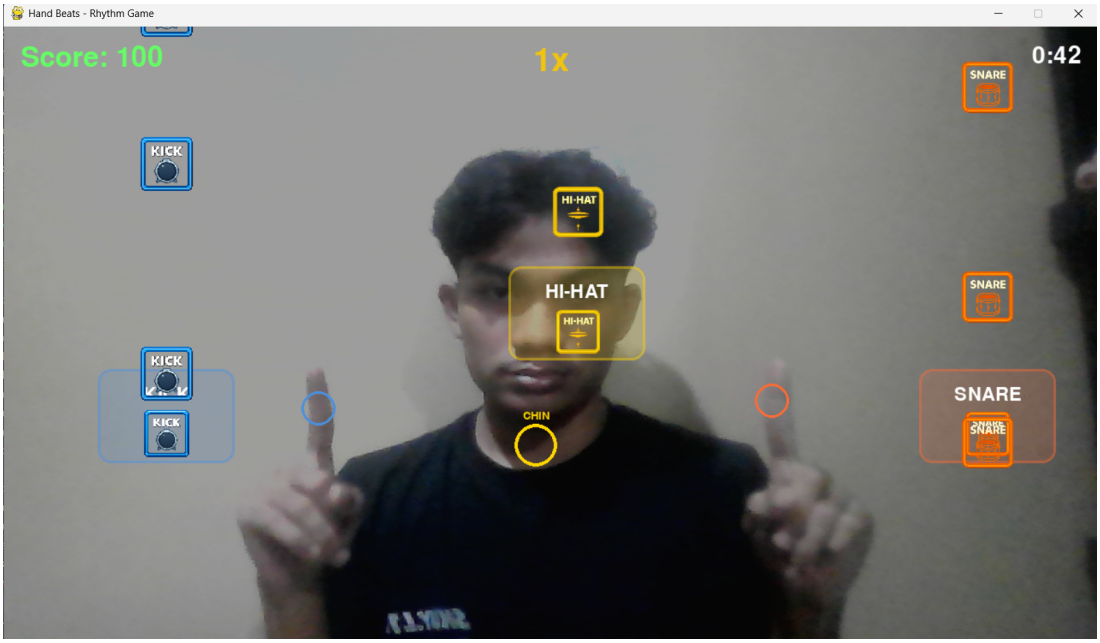
4.3 Result Screen

Result screen menampilkan statistik lengkap performa pemain setelah game berakhir, termasuk score, accuracy, rank, combo, dan breakdown hit counts (Gambar 4).

5 Sistem Scoring

5.1 Timing Windows

Sistem scoring menggunakan timing windows untuk menentukan akurasi hit. Semakin dekat timing dengan target, semakin tinggi poin yang didapat.



Gambar 3: Gameplay Screen dengan Video Overlay

Tabel 1: Timing Windows dan Point Values

Rating	Easy	Medium	Hard
PERFECT	±120ms (100 pts)	±80ms (100 pts)	±60ms (100 pts)
GOOD	±200ms (50 pts)	±150ms (50 pts)	±120ms (50 pts)
OK	±300ms (25 pts)	±230ms (25 pts)	±180ms (25 pts)
MISS	Outside window (0 pts)	Outside window (0 pts)	Outside window (0 pts)

5.2 Combo Multiplier

Sistem combo memberikan multiplier bonus untuk consecutive hits yang berhasil:

- 10x combo: 1.2x multiplier
- 20x combo: 1.5x multiplier
- 30x combo: 2.0x multiplier
- 50x combo: 2.5x multiplier

5.3 Ranking System

Setelah game berakhir, sistem memberikan rank berdasarkan accuracy:

- S Rank: 95% - 100%
- A Rank: 85% - 94%
- B Rank: 75% - 84%
- C Rank: 65% - 74%
- D Rank: $< 65\%$

Gambar 4: Result Screen dengan Performance Stats

6 Teknologi dan Tools

6.1 Library dan Framework

Project ini menggunakan beberapa library Python untuk implementasi:

- **Pygame**: Game engine dan audio processing
- **OpenCV**: Video capture dan frame processing
- **MediaPipe**: Hand detection dan tracking
- **NumPy**: Array manipulation untuk image processing

6.2 Development Tools

Tools yang digunakan dalam development:

- **Python 3.11**: Programming language
- **VS Code**: Code editor
- **Git**: Version control
- **Draw.io**: Flowchart design

7 Kesimpulan

HandBeats berhasil mengintegrasikan tiga komponen multimedia (image, video, audio processing) menjadi sebuah rhythm game interaktif yang responsif dan engaging. Penggunaan MediaPipe untuk hand detection memberikan akurasi tinggi, sementara sistem collision detection berbasis velocity memastikan gameplay yang fair dan challenging.

7.1 Pencapaian

- Implementasi real-time hand tracking dengan MediaPipe
- Integrasi seamless video overlay dengan gameplay elements
- Audio system dengan polyphonic playback dan infinite looping
- Tiga tingkat kesulitan dengan dynamic pattern generation
- Scoring system dengan combo multiplier dan ranking

7.2 Pengembangan Masa Depan

Beberapa fitur yang dapat dikembangkan:

- Online leaderboard untuk competitive play
- Custom song import dengan auto-beatmap generation
- Multiplayer mode untuk head-to-head competition
- Achievement system dan progression tracking
- Mobile platform support menggunakan Kivy atau similar framework

Lampiran

AI Assistant Tools

Dalam pengembangan project ini, beberapa AI assistant digunakan untuk membantu problem solving, code optimization, dan debugging:

- **ChatGPT**: Untuk konsultasi arsitektur sistem, debugging, dan optimisasi algoritma
  - Link: https://chatgpt.com/share/68ff6cbb-d63c-800f-871e-9593a1298534
- **Google Gemini**: Untuk optimisasi collision detection dan performance tuning
  - Link: https://gemini.google.com/share/beadc25d28a9
- **Claude Code**: Untuk code refactoring, documentation, dan implementation assistance

Asset Credits

- Background Music: Original composition (9-second loop, 120 BPM)
- Sound Effects: Pixabay Free Sound Library
  - Kick drum sample
  - Snare drum sample
  - Hi-hat sample
- Menu Background: Custom pixel art design
- Instrument Icons: Custom designed PNG assets

Repository

Source code project HandBeats tersedia secara lengkap di GitHub:
https://github.com/luciferdana/handbeats-rhythm-game