



---

Nama:

1. **Jonathan Jethro (122140213)**
2. **M. Rizqullah Bimo (122140220)**
3. **Felix Ignasius Sinaga (121140146)**

Mata Kuliah: **Multimedia (IF4021)**

**Tugas Besar**

Tanggal: 31/05/25

---

## 1 Pendahuluan

Perkembangan teknologi multimedia telah membuka berbagai peluang inovatif dalam menciptakan pengalaman interaktif antara pengguna dan sistem digital. Salah satu bentuk interaksi yang semakin populer dan menarik perhatian adalah pemanfaatan gerakan tubuh, khususnya gerakan tangan, sebagai media kontrol. Interaksi berbasis gerakan tangan tidak hanya meningkatkan keterlibatan pengguna, tetapi juga memberikan pengalaman yang lebih alami dan intuitif dalam penggunaan aplikasi multimedia. "Hand Movement" merujuk pada sebuah sistem berbasis filter multimedia yang menggunakan deteksi gerakan tangan sebagai metode kontrol utama. Dalam sistem ini, posisi tangan pengguna dideteksi secara real-time dan pengguna diharuskan menggerakkan tangan mereka untuk meniru atau menyesuaikan bentuk tangan dengan rintangan virtual yang ditampilkan di layar. Hanya dengan menyamakan bentuk tangan secara tepat, pengguna dapat melewati rintangan yang ada. Pendekatan ini menghadirkan konsep permainan atau interaksi visual yang menggabungkan antara teknologi deteksi gerakan dengan elemen hiburan. Implementasi sistem ini tidak hanya menguji ketepatan gerakan pengguna, tetapi juga melatih koordinasi motorik dan konsentrasi, sehingga berpotensi digunakan untuk berbagai tujuan, baik hiburan, edukasi, maupun pelatihan berbasis multimedia.

## 2 Cara Kerja Filter

Filter ini merupakan game gesture interaktif berbasis Python ini menggabungkan pustaka Ursina untuk tampilan grafik 3D dan OpenCV untuk mendeteksi gesture tangan secara real-time melalui webcam. Sistem dirancang agar pengguna dapat berinteraksi langsung dengan permainan menggunakan gerakan tangan, menciptakan pengalaman multimedia yang imersif dan responsif. Proses dimulai dengan inisialisasi elemen-elemen utama seperti jendela 3D, entitas pemain, musuh, jalur permainan, kamera, serta panel tampilan untuk video webcam. Modul HandDetector, yang merupakan bagian terpisah dari sistem, digunakan untuk mengenali berbagai bentuk gesture tangan melalui video yang diambil dari webcam.

Pengolahan video webcam dilakukan dalam thread terpisah agar deteksi gesture tidak mengganggu jalannya game utama. Setiap frame yang tertangkap diproses untuk mengenali gesture tangan, dan hasilnya ditampilkan di panel dalam antarmuka game. Musuh dalam permainan ini digambarkan sebagai entitas yang masing-masing mewakili satu jenis gesture, seperti "peace", "stop", atau "thumbs up". Musuh akan bergerak mendekati pemain secara bertahap, dan pemain diharuskan menirukan gesture yang sesuai dengan musuh yang sedang mendekat.

Ketika musuh mencapai jarak tertentu dari pemain, sistem akan membandingkan gesture yang dikenali kamera dengan gesture yang dituntut oleh musuh. Jika cocok, musuh akan menghilang dan

permainan berlanjut ke musuh berikutnya. Jika gesture salah atau tidak dikenali dalam waktu yang ditentukan, permainan akan diulang dari awal, dan pesan instruksi akan ditampilkan untuk memberi tahu pengguna. Apabila semua musuh berhasil dilewati dengan gesture yang benar, pemain akan memenangkan permainan dan menerima pesan keberhasilan.

Secara keseluruhan, game ini dirancang tidak hanya sebagai hiburan, tetapi juga sebagai media pelatihan kognitif dan motorik. Pengguna dilatih untuk merespons visual dengan gerakan fisik yang tepat dan cepat, menjadikannya sebuah pendekatan inovatif dalam pemanfaatan teknologi gesture recognition dalam dunia multimedia interaktif.

### 3 Penjelasan Kode

#### 3.1 Import Library

```
1 from ursina import *
2 import cv2
3 import threading
4 from hand_detection import HandDetector
5 from game_controls import apply_gesture_effect
6 import numpy as np
7 from PIL import Image
8 import time
9 import random
10
11
```

Kode 1: Import Library

- Menggunakan Ursina (engine 3D Python) untuk membuat window game dan objek 3D.
- OpenCV (cv2) untuk akses webcam dan frame video.
- Threading untuk menjalankan pengambilan frame webcam di latar belakang.
- HandDetector untuk mendeteksi gesture tangan (modul eksternal).
- `apply_gesture_effect` untuk memberikan efek pada pemain berdasarkan gesture (modul eksternal).

#### 3.2 Inisialisasi Game

```
1 app = Ursina()
2 window.title = 'Gesture Game'
3 window.borderless = False
4 window.fullscreen = False
5 window.exit_button.visible = False
6 window.fps_counter.enabled = True
7 camera.rotation_x = 5
8 camera.rotation_y = -20
9 camera.position = (4, 3, -12)
10
```

Kode 2: Inisialisasi Game

- `app = Ursina()` Membuat dan menginisialisasi aplikasi game berbasis Ursina Engine.
- `window.title = 'Gesture Game'` Mengatur judul jendela aplikasi menjadi "Gesture Game".

- `window.borderless = False` Jendela aplikasi memiliki batas (tidak borderless).
- `window.fullscreen = False` Jendela aplikasi tidak menggunakan mode layar penuh.
- `window.exit_button.visible = False` Tombol keluar (`exit`) pada jendela game disembunyikan.
- `window.fps_counter.enabled = True` Menampilkan penghitung frame per second (FPS) di layar.
- `camera.rotation_x = 5` Mengatur sudut rotasi kamera pada sumbu X sebesar 5 derajat.
- `camera.rotation_y = -20` Mengatur sudut rotasi kamera pada sumbu Y sebesar -20 derajat.
- `camera.position = (4, 3, -12)` Mengatur posisi kamera pada koordinat (4, 3, -12) di ruang 3D.

### 3.3 Webcam dan Hand Detector

```

1  detector = HandDetector()
2  cap = cv2.VideoCapture(0)
3  if not cap.isOpened():
4      print("Error: Tidak dapat membuka webcam")
5      application.quit()
6  cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
7  cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)
8  gesture_result = None
9  gesture_frame = None
10

```

Kode 3: Webcam dan Hand Detector

- `detector = HandDetector()` Membuat objek deteksi tangan untuk mengenali gesture dari webcam.
- `cap = cv2.VideoCapture(0)` Membuka akses ke webcam utama (biasanya webcam laptop/PC).
- `if not cap.isOpened():` Mengecek apakah webcam berhasil dibuka. Jika gagal, tampilkan pesan error dan keluar dari aplikasi.
- `cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)` Mengatur lebar (resolusi) frame video webcam menjadi 320 piksel (lebih ringan prosesnya).
- `cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)` Mengatur tinggi (resolusi) frame video webcam menjadi 240 piksel.
- `gesture_result = None` Variabel untuk menyimpan hasil deteksi gesture (awal kosong).
- `gesture_frame = None` Variabel untuk menyimpan frame (gambar) hasil deteksi gesture (awal kosong).

### 3.4 Panel Tampilan Webcam

```

11 empty_frame = np.zeros((240, 320, 3), dtype=np.uint8)
12 empty_img = Image.fromarray(empty_frame)
13 webcam_texture = Texture(empty_img)
14 webcam_panel = Entity(
15     parent=camera.ui,
16     model='quad',
17     texture=webcam_texture,
18     scale=(0.4, 0.3),
19     position=(-0.7, -0.4)
20 )
21

```

Kode 4: Panel Tampilan Webcam

- Membuat frame kosong sebagai placeholder awal untuk panel webcam di UI.
- Mengonversi frame kosong ke objek gambar dan tekstur.
- Membuat entity (panel) di UI untuk menampilkan frame dari webcam.

### 3.5 Fungsi Pengambil Frame Webcam (Thread)

```

22 def webcam_loop():
23     global gesture_result, gesture_frame, webcam_texture, webcam_panel
24     while True:
25         ret, frame = cap.read()
26         if not ret:
27             continue
28         gesture, _ = detector.get_hand_data(frame)
29         gesture_result = gesture
30         gesture_frame = annotated
31         rgb_frame = cv2.cvtColor(annotated, cv2.COLOR_BGR2RGB)
32         rgb_frame = cv2.flip(rgb_frame, 1)
33         img = Image.fromarray(rgb_frame)
34         webcam_panel.texture = Texture(img)
35         time.sleep(1/20) # Batasi frame rate ~20 fps
36
37     threading.Thread(target=webcam_loop, daemon=True).start()
38

```

Kode 5: Fungsi Pengambil Frame Webcam

- Membuat fungsi `webcam_loop()` untuk mengambil frame webcam secara terus-menerus.
- Setiap frame dideteksi gesture-nya, hasilnya disimpan.
- Frame diubah ke format RGB, dibalik horizontal, lalu diupdate ke panel UI.
- Fungsi ini dijalankan di thread terpisah agar tidak mengganggu jalannya game.

### 3.6 Data Gesture, Musuh dan Jalan

```

39 gesture_textures = {
40     "peace": 'assets/two.png',
41     "stop": 'assets/stop.png',
42     "one_finger_up": 'assets/one.png',
43     "fist": 'assets/fist.png',

```

```

44     "thumbs_up": 'assets/thumb.png'
45 }
46 gesture_sequence = ["peace", "stop", "one_finger_up", "fist", "thumbs_up"]
47 colors = [color.green, color.blue, color.yellow, color.orange, color.pink]
48 musuh_posisi = [(0, 0, 100), (0, 0, 200), (0, 0, 300), (0, 0, 400), (0, 0, 500)]
49 roads = [
50     Entity(model='source/g.glb', scale=(5, 0.1, 50), position=(0, -2, 10), color=color.gray),
51     Entity(model='source/g.glb', scale=(5, 0.1, 50), position=(0, -2, 60), color=color.gray)
52 ]
53

```

Kode 6: Data Gesture; Musuh; Jalan

- Mendefinisikan gambar/textures untuk setiap gesture.
- Membuat urutan gesture musuh dan warna terkait.
- Menentukan posisi awal musuh di depan pemain.
- Membuat dua entitas jalan sebagai latar belakang yang bergerak.

### 3.7 Buat Pemain dan Musuh

```

54 player = Entity(
55     model='sphere',
56     scale=2,
57     position=(0, 0, 0),
58     collider='box',
59     color=color.white
60 )
61 enemies = []
62 for i, gesture in enumerate(gesture_sequence):
63     enemy = Entity(
64         model='quad',
65         texture=gesture_textures[gesture],
66         scale=4,
67         position=musuh_posisi[i],
68         collider='box',
69         name=gesture
70     )
71     enemies.append(enemy)
72

```

Kode 7: Buat Pemain dan Musuh

- Membuat entity pemain berupa bola putih di posisi awal.
- Membuat daftar musuh (entity) dengan gambar gesture masing-masing, ditempatkan berurutan di jalur yang sama.

### 3.8 UI Text

```

73 instruction_text = Text(
74     text="Tiru gesture musuh!",
75     parent=camera.ui,
76     position=(0, 0.4),
77     origin=(0, 0),
78     scale=1.5,
79     color=color.white

```

```

80 )
81 info_text = Text(
82     text="Tekan Q untuk Quit | R untuk Replay",
83     parent=camera.ui,
84     position=(0, 0.33),
85     origin=(0, 0),
86     scale=1,
87     color=color.light_gray
88 )
89 win_text = Text(
90     text="",
91     parent=camera.ui,
92     position=(0, 0),
93     scale=3,
94     color=color.green,
95     enabled=False
96 )
97

```

Kode 8: UI Text

- Membuat teks instruksi utama di layar (gesture yang harus ditiru).
- Membuat teks info tombol (Quit dan Replay) di layar.
- Membuat teks kemenangan yang muncul saat pemain menang.

### 3.9 Variabel Kontrol Game

```

98 current_enemy_index = 0
99 player_speed = 0.01
100 world_speed = 0.2
101 win = False
102 check_gesture = False
103 gesture_cooldown = 0
104 wrong_gesture = False
105 wrong_gesture_timer = 0
106 gesture_detection_timer = 0
107 gesture_detection_limit = 30 # batas waktu (frame) deteksi gesture
108

```

Kode 9: Variabel dan Kontrol Game

- Menyimpan status dan parameter seperti index musuh saat ini, kecepatan pemain/dunia, status menang, cooldown gesture, timer gesture salah, dan timer deteksi gesture.

### 3.10 Fungsi Reset dan Update Game

```

109 def reset_game():
110     global current_enemy_index, win, check_gesture, gesture_cooldown
111     global wrong_gesture, wrong_gesture_timer, gesture_detection_timer
112     player.position = (0, 0, 0)
113     player.color = color.white
114     .....
115
116 def update():
117     global current_enemy_index, win, check_gesture, gesture_cooldown
118     global wrong_gesture, wrong_gesture_timer, gesture_detection_timer
119     for road in roads:

```

```

120 .....
121

```

Kode 10: Fungsi Reset dan Update

- `reset_game()` Mengatur ulang semua status untuk mulai/restart game.
- `update()` Fungsi utama yang dijalankan setiap frame untuk mengatur logika game, cek gesture, menang/kalah, dll.

### 3.11 game\_controls.py

```

122 from ursina import color
123 import random
124
125 colors = [color.green, color.blue, color.yellow, color.orange, color.pink]
126
127 def apply_gesture_effect(entity, gesture):
128
129     .....
130

```

Kode 11: game\_controls.py

Kode ini berisi fungsi `apply_gesture_effect` yang memberikan efek tertentu pada sebuah entitas (seperti player) di game berdasarkan gesture (gerakan tangan) yang terdeteksi. Misalnya, jika gesture "peace" terdeteksi, warna entitas akan berubah secara acak dan entitas akan berotasi. Gesture lain seperti "one\_finger\_up", "stop", "fist", dan "thumbs\_up" juga memiliki efek berbeda pada posisi, warna, rotasi, atau ukuran entitas.

### 3.12 hand\_detection.py

```

131 import cv2
132 import mediapipe as mp
133 import numpy as np
134 from collections import Counter
135
136 class HandDetector:
137     ...
138     def __init__(self, max_hands=1, detection_confidence=0.7, tracking_confidence=0.7):
139         self.mp_hands = mp.solutions.hands
140         self.hands = self.mp_hands.Hands(
141             .....
142         def get_finger_state(self, landmarks, finger_tips, finger_pips, finger_dips=None, finger_mcps=
None):
143
144             extended = []
145             for i, (tip, pip) in enumerate(zip(finger_tips, finger_pips)):
146                 ....
147         def get_hand_data(self, frame):
148
149             rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
150             results = self.hands.process(rgb)
151
152             if results.multi_hand_landmarks:
153                 hand = results.multi_hand_landmarks[0]
154                 ....
155         def release(self):
156             self.hands.close()

```

## Kode 12: hand\_detection.py

- Inisialisasi (`_init_`)  
Membuat objek MediaPipe Hands. Menyediakan buffer dan variabel untuk stabilisasi gesture agar hasil deteksi tidak cepat berubah.
- Fungsi `get_finger_state`  
Mendeteksi apakah jari dalam keadaan terentang atau tidak, berdasarkan posisi landmark pada jari. Untuk ibu jari menggunakan posisi x, sedangkan jari lain menggunakan posisi y.
- Fungsi `get_hand_data`  
Mengubah frame ke format RGB, lalu memproses dengan mediapipe untuk mendapatkan landmark tangan. Mengecek setiap posisi jari (ibu jari, telunjuk, tengah, manis, kelingking) apakah terentang atau tidak. Menentukan gesture berdasarkan status lima jari. Melakukan stabilisasi gesture (gesture harus sama beberapa frame sebelum dianggap valid). Mengembalikan gesture yang stabil, posisi tengah tangan, dan frame dengan anotasi gambar tangan.
- Fungsi Release  
Menutup resource mediapipe saat sudah tidak digunakan.

## 4 Asset

Berikut ini adalah contoh cara untuk memuat multi-gambar.



(a) Two



(b) Thumb



(c) Stop



(d) Fist



(e) One

Gambar 1: Asset Bentuk Tangan

## 5 Alur Implementasi

1. Pengambilan Gambar dari Webcam
  - Menggunakan OpenCV untuk menangkap frame secara real-time dari kamera.



## 2. Deteksi Gesture Tangan

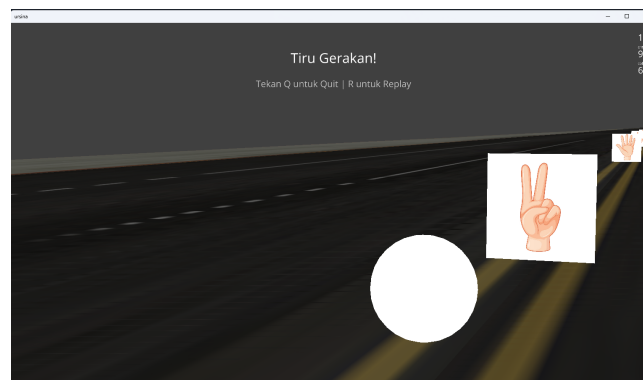
- Menggunakan MediaPipe (melalui kelas HandDetector di hand\_detection.py) untuk mendeteksi posisi tangan dan gesture (misal: peace, stop, thumbs up, fist, one finger up). - Setiap gesture dikenali berdasarkan posisi jari yang terdeteksi dari landmark tangan.

## 3. Penghubung Gesture ke Efek Visual

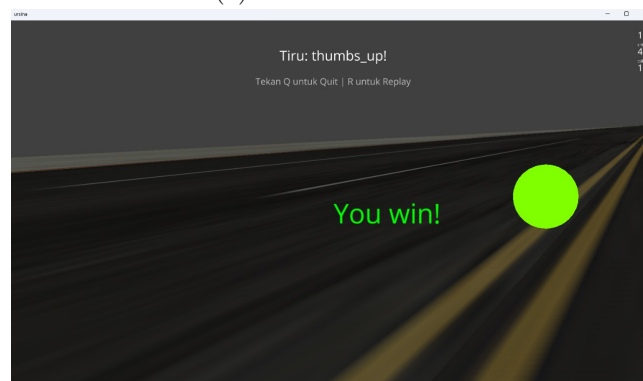
- Gesture yang terdeteksi akan dikirim ke fungsi pengendali (seperti apply\_gesture\_effect di game\_controls.py). - Fungsi ini mengubah properti suatu entitas (misal: player di game Ursina) seperti posisi, warna, rotasi, atau skala sesuai gesture.

## 4. Visualisasi di Game

- Menggunakan Ursina Engine untuk menampilkan objek 3D yang bisa bergerak/berubah tergantung gesture yang dikenali.



(a) Awal Permainan



(b) Akhir Permainan