



Nama Tim: **King-Astro**

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

Anggota Kelompok:

1. **Hagai Kopusi Sinulingga (122140059)**
 2. **Muklis Mustaqim (122140115)**
 3. **Akhwan Adib Al Hakim (122140149)**
-

Tugas Besar Teknologi Multimedia

Tanggal: 28 Mei 2025

1 Pendahuluan

1.1 Nama Proyek

Game Filter Pesawat Luar Angkasa King Astro (Space Shooter Face Filter Game).

1.2 Kategori Proyek

Proyek "King-Astro: Game Filter Pesawat Luar Angkasa" termasuk dalam kategori Aplikasi Komputer Visi Interaktif Real-Time. Proyek ini memanfaatkan teknologi deteksi wajah dan deteksi titik penanda wajah (facial landmark detection) yang diimplementasikan menggunakan pustaka MediaPipe. Secara umum, deteksi wajah digunakan untuk mengidentifikasi keberadaan pengguna di depan kamera secara terus-menerus, yang menjadi syarat utama agar game bisa dijalankan. Setelah wajah terdeteksi, sistem akan menggunakan posisi titik-titik penanda di wajah khususnya bagian hidung sebagai kontrol utama untuk menggerakkan pesawat. Selain itu, gerakan membuka mulut juga dideteksi dan digunakan untuk menjalankan fungsi menembak dalam game. Seluruh proses, mulai dari pengambilan gambar kamera, deteksi wajah, hingga penerjemahan gerakan wajah menjadi aksi dalam game, dilakukan secara real-time [1]. Hal ini menunjukkan kemampuan MediaPipe dalam membangun sistem pemrosesan visual yang cepat dan efisien. Konsep game ini mirip dengan aplikasi augmented reality (AR) atau filter wajah, di mana wajah pengguna dijadikan sebagai antarmuka (interface) utama untuk mengontrol elemen dalam permainan.

1.3 Deskripsi Proyek

Game Filter Pesawat Luar Angkasa adalah sebuah game 2D side-scrolling shooter yang dikembangkan menggunakan Python[2]. Inti dari game ini adalah interaksi pemain melalui webcam. Menggunakan library MediaPipe, game mendeteksi landmark wajah pemain untuk mengontrol gerakan pesawat (berdasarkan posisi hidung) dan untuk menembak (berdasarkan bukaan mulut). Permainan terdiri dari dua fase utama. Fase pertama adalah fase bertahan dan mengumpulkan skor, di mana pemain harus menghindari rintangan berupa meteor dan mengumpulkan item energi. Setelah durasi tertentu, game memasuki fase kedua, yaitu pertarungan melawan bos. Bos memiliki pola serangan sendiri, sistem nyawa, dan periode kekebalan setelah terkena serangan. Pemain harus mengalahkan bos untuk memenangkan permainan. Proyek ini dirancang dengan struktur kode modular, memisahkan konfigurasi (config.py), logika inti game (game_manager.py), dan titik masuk aplikasi (main.py). Aset grafis dan audio disimpan dalam folder terpisah (assets/) untuk kerapian. Fitur tambahan seperti musik latar, tampilan UI yang informatif (skor, progress bar, nyawa bos, instruksi), dan penyesuaian ukuran jendela tampilan juga diimplementasikan untuk meningkatkan pengalaman pengguna[3].

1.4 Tujuan Proyek

Tujuan proyek "King-Astro: Game Filter Pesawat Luar Angkasa" adalah untuk mengembangkan sebuah game interaktif yang memanfaatkan teknologi komputer visi, khususnya deteksi wajah dan titik penanda wajah dari MediaPipe, guna menciptakan pengalaman bermain yang unik dan imersif melalui kontrol gestur wajah, sambil tetap mempertahankan struktur kode yang modular dan pengalaman pengguna yang optimal.

2 Methods

2.1 Cara Kerja Proyek

Alur kerja proyek "Game Filter Pesawat Luar Angkasa" adalah sebagai berikut:

2.1.1 Inisialisasi Sistem

Saat main.py dijalankan, objek GameManager dibuat. Konstruktor GameManager akan:

- Menginisialisasi webcam menggunakan OpenCV (cv2.VideoCapture).
- Menginisialisasi model MediaPipe Face Mesh untuk deteksi landmark wajah.
- Memuat semua asset grafis (gambar pesawat, rintangan, energi, bos) dan audio (musik latar) dari folder assets. Path aset dan parameter lainnya diambil dari config.py.
- Menginisialisasi Pygame mixer untuk pemutaran musik latar.
- Memanggil metode reset() untuk mengatur semua variabel status game ke kondisi awal (skor 0, game belum berakhir, boss belum muncul, dll.).

2.1.2 Loop Utama Game

Metode run() pada GameManager memulai loop utama. Di setiap iterasi:

- Pengambilan Frame: Frame video diambil dari webcam, kemudian di-flip secara horizontal.
- Deteksi Wajah: Frame dikonversi ke format RGB dan diproses oleh MediaPipe Face Mesh untuk mendapatkan landmark wajah pemain.
- Update Status Game (if not self.game_over):
 - Update Pemain: Posisi pesawat pemain diperbarui berdasarkan posisi landmark hidung. Mekanisme tembakan diaktifkan jika pemain membuka mulut (jarak landmark bibir atas dan bawah melebihi ambang batas) dan cooldown tembakan terpenuhi. Peluru pemain ditambahkan ke daftar.
 - Update Peluru Pemain: Posisi semua peluru pemain diperbarui. Dilakukan pengecekan tabrakan antara peluru pemain dengan bos. Jika kena dan bos tidak kebal, nyawa bos berkurang, bos menjadi kebal sementara, dan peluru dihapus. Jika nyawa bos habis, pemain menang.
 - Transisi Fase: Waktu permainan dihitung. Jika melebihi GAME_DURATION_UNTIL_BOSS dan bos belum muncul, status diubah, bos diinisialisasi, dan semua rintangan/energi fase awal dibersihkan.
 - Logika Fase:

- * Jika bos belum muncul (elif not self.musuh_muncul): Rintangan meteor dan item energi di-spawn secara periodik. Dilakukan pengecekan tabrakan pesawat dengan energi (menambah skor).
 - * Jika bos sudah muncul (if self.musuh_muncul): Logika pergerakan bos, pola serangan bos (menembakkan peluru), dan status kekebalan bos diperbarui.
 - Update Rintangan: Posisi semua rintangan (meteor dan peluru bos) diperbarui. Dilakukan pengecekan tabrakan pesawat dengan rintangan (menyebabkan game over jika bukan kemenangan).
- Penggambaran (Drawing):
 - Semua objek game yang aktif (pesawat, peluru pemain, rintangan, energi, bos) digambar ke frame menggunakan gambar asset yang sesuai. Peluru pemain digambar sebagai persegi panjang.
 - Efek visual seperti bos berkedip saat kebal diimplementasikan.
 - Antarmuka Pengguna (UI) digambar, termasuk progress bar menuju bos, skor, instruksi tembakan (saat bos aktif), dan bilah nyawa bos.
 - Jika game over atau menang, layar akhir yang sesuai ditampilkan.
 - Tampilan Frame: Frame yang telah digambari semua elemen kemudian diubah ukurannya sesuai DISPLAY_WIDTH dan DISPLAY_HEIGHT dari config.py dan ditampilkan menggunakan cv2.imshow().
 - Input Keyboard: Menangani input tombol 'q' untuk keluar dan 'r' untuk mereset game jika sudah berakhir.

2.1.3 Restart Program

Setelah loop utama berakhir (pemain menekan 'q'), webcam dilepaskan, musik dihentikan, mixer Pygame ditutup, dan semua jendela OpenCV dihancurkan.

2.2 Requirements Project

2.2.1 Perangkat Keras:

- Komputer atau Laptop dengan sistem operasi Windows, macOS, atau Linux.
- Webcam yang terpasang dan berfungsi.
- Speaker atau headphone untuk output audio.

2.2.2 Perangkat Lunak:

- Python versi 3.8 atau yang lebih baru.
- pip (Python package installer).
- Library Python (dapat diinstal melalui pip install -r requirements.txt):
 - opencv-python: Untuk pemrosesan gambar, video, interaksi webcam, dan fungsi gambar dasar.
 - numpy: Untuk operasi numerik efisien, terutama digunakan oleh OpenCV dan untuk kalkulasi jarak antar landmark.

- mediapipe: Untuk deteksi wajah dan landmark wajah secara real-time.
 - pygame: Khususnya modul pygame.mixer untuk memutar musik latar.
- Aset Tambahan:
 - File gambar (format PNG dengan transparansi disarankan) untuk pesawat, rintangan, energi, dan bos, disimpan dalam folder assets/.
 - File musik latar (format MP3), misalnya undertale_megalovania.mp3, disimpan dalam folder assets/.

2.3 Langkah-Langkah penggerjaan proyek

Proyek dikembangkan melalui serangkaian iterasi, dengan setiap iterasi menambahkan fungsionalitas baru atau menyempurnakan yang sudah ada. Tahapan umumnya adalah sebagai berikut:

2.3.1 Setup Dasar & Kontrol Pemain

Menginisialisasi pengambilan video dari webcam, mengintegrasikan MediaPipe Face Mesh untuk mendeteksi landmark wajah, dan mengimplementasikan kontrol dasar pesawat menggunakan gerakan kepala (posisi hidung).

2.3.2 Mekanisme Game Awal

Menambahkan objek rintangan (meteor) dan item energi yang bergerak secara acak dari kanan ke kiri layar. Mengimplementasikan deteksi tabrakan antara pesawat pemain dengan rintangan (menyebabkan game over) dan dengan item energi (menambah skor). Membuat sistem skor sederhana.

2.3.3 Pengembangan Bos

Merancang dan mengimplementasikan kemunculan bos setelah durasi tertentu. Memberikan bos kemampuan untuk bergerak secara acak dalam area terbatas di sisi kanan layar. Mengembangkan pola serangan beruntun untuk bos (menembakkan beberapa projektil dengan jeda).

2.3.4 Sistem Nyawa & Kekebalan Bos

Mengimplementasikan sistem nyawa untuk bos (misalnya, 3 nyawa). Setelah bos terkena serangan, ia akan menjadi kebal selama beberapa detik (misalnya, 3 detik), yang ditandai dengan efek visual berkedip. Membuat bilah nyawa (health bar) untuk bos.

2.3.5 Mekanisme Tembakan Pemain

Mengubah cara pemain menyerang bos. Awalnya mungkin dengan menabrakkan pesawat, kemudian ditingkatkan menjadi mekanisme menembak projektil dengan cara membuka mulut (dideteksi dari jarak antar landmark bibir). Mengimplementasikan cooldown untuk tembakan pemain.

2.3.6 UI & UX (User Interface & User Experience)

Menambahkan elemen UI seperti progress bar yang menunjukkan waktu hingga bos muncul, teks skor yang jelas, instruksi cara menembak saat bos muncul ("Buka Mulut untuk Menembak!"). Mengimplementasikan layar "Game Over" dan "Victory" dengan opsi untuk memulai ulang permainan ('R') atau keluar ('Q'). Memberikan opsi untuk menyesuaikan ukuran jendela tampilan game.

2.3.7 Audio

Mengintegrasikan library Pygame untuk menambahkan musik latar yang diputar secara berulang selama permainan.

2.3.8 Refactoring & Dokumentasi

Merapikan struktur kode dengan memisahkannya menjadi tiga file utama (config.py, game_manager.py, main.py). Membuat folder assets/ untuk semua sumber daya eksternal. Menambahkan docstring pada setiap fungsi dan kelas untuk menjelaskan tujuan, parameter, dan nilai kembalinya. Membuat file requirements.txt untuk memudahkan manajemen dependensi.

3 Results and Analysis

3.1 Code Program

Kode program proyek ini terorganisir dalam tiga file Python utama dan sebuah folder asset untuk sumber daya eksternal, memastikan modularitas dan kemudahan pengelolaan.

3.1.1 Code Program game_manager.py

File game_manager.py berisi kelas GameManager yang merupakan inti dari logika permainan. Kelas ini bertanggung jawab atas:

- Inisialisasi: Menyiapkan webcam, model deteksi wajah MediaPipe, memuat aset grafis dan audio, serta menginisialisasi semua variabel status permainan.
- Loop Utama Permainan: Metode run() mengontrol alur eksekusi utama, mulai dari menangkap frame, memproses input pemain melalui deteksi wajah, memperbarui semua status objek game (pemain, musuh, projektil, rintangan), hingga menggambar semua elemen ke layar.
- Manajemen Status: Mengelola variabel-variabel penting seperti skor, status game_over, victory, kemunculan musuh, nyawa bos, dan status kekebalan bos.
- Logika Pembaruan Objek: Serangkaian metode privat (_update_player, _update_player_projectiles, _update_pre_boss_phase, _update_boss_phase, _update_obstacles) yang dipanggil dalam setiap iterasi loop untuk mengubah keadaan objek berdasarkan interaksi dan waktu.
- Logika Penggambaran: Metode-metode privat (_draw_ui, _draw_boss_health_bar, _overlay_image) yang menangani aspek visual, menampilkan semua objek dan informasi kepada pemain.
- Metode Bantu: Fungsi-fungsi pendukung untuk tugas-tugas seperti pembuatan objek baru (_create_rintangan, _create_energi) dan penentuan target gerakan bos (_get_new_musuh_target).

Penggunaan kelas GameManager mengenkapsulasi seluruh logika dan data game, sehingga interaksi dari main.py menjadi sangat sederhana.

```
1 import cv2
2 import numpy as np
3 import mediapipe as mp
4 import random
5 import time
6 from config import *
7 import pygame
8
```

```
10 class GameManager:
11     """
12     Kelas utama yang mengelola semua logika, status, dan alur permainan.
13
14     Kelas ini bertanggung jawab untuk inisialisasi, menjalankan loop utama,
15     memperbarui status semua objek game, menggambar di layar, dan menangani
16     input dari pemain.
17     """
18     def __init__(self):
19         """
20         Menginisialisasi GameManager.
21
22         Mempersiapkan webcam, model MediaPipe, memuat semua asset gambar,
23         dan mengatur ulang status game ke kondisi awal.
24         """
25         self._init_camera_and_mediapipe()
26         self._load_assets()
27         # --- BARU: Inisialisasi Pygame Mixer dan putar musik ---
28         pygame.mixer.init()
29         try:
30             pygame.mixer.music.load(BACKGROUND_MUSIC_PATH)
31             pygame.mixer.music.set_volume(0.5) # Atur volume (0.0 hingga 1.0)
32             pygame.mixer.music.play(-1) # Angka -1 berarti musik akan diulang terus-menerus
33         except pygame.error as e:
34             print(f"Error memuat atau memutar musik: {e}")
35         # --- AKHIR BLOK BARU ---
36         self.reset()
37
38     def _init_camera_and_mediapipe(self):
39         """Inisialisasi webcam dan model MediaPipe Face Mesh."""
40         self.cap = cv2.VideoCapture(0)
41         if not self.cap.isOpened():
42             print("Error: Tidak bisa membuka kamera.")
43             exit()
44
45         self.original_width = int(self.cap.get(cv2.CAP_PROP_FRAME_WIDTH))
46         self.original_height = int(self.cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
47
48         self.mp_face_mesh = mp.solutions.face_mesh
49         self.face_mesh = self.mp_face_mesh.FaceMesh(
50             max_num_faces=1,
51             min_detection_confidence=0.5,
52             min_tracking_confidence=0.5
53         )
54
55     def _load_assets(self):
56         """Memuat semua asset gambar yang dibutuhkan untuk game."""
57         try:
58             self.pesawat_img = self._load_single_asset(PESAWAT_PATH)
59             self.rintangan_img = self._load_single_asset(RINTANGAN_PATH)
60             self.energi_img = self._load_single_asset(ENERGI_PATH)
61             self.musuh_img = self._load_single_asset(MUSUH_PATH)
62         except (FileNotFoundException, TypeError) as e:
63             print(f"Error: {e}"); exit()
64
65     def _load_single_asset(self, path):
66         """
67             Memuat satu file gambar dan memvalidasinya.
68
69             Args:
70                 path (str): Path menuju file gambar.
71
```

```
72     Returns:
73         np.ndarray: Gambar yang dimuat oleh OpenCV.
74
75     Raises:
76         FileNotFoundError: Jika file tidak ditemukan di path yang diberikan.
77         TypeError: Jika gambar tidak memiliki alpha channel (bukan format transparan).
78     """
79     img = cv2.imread(path, cv2.IMREAD_UNCHANGED)
80     if img is None: raise FileNotFoundError(f"File tidak ditemukan di path: {path}")
81     if img.shape[2] != 4: raise TypeError(f"Gambar '{path}' harus memiliki alpha channel.")
82     return img
83
84 def reset(self):
85     """Mereset semua status permainan ke kondisi awal untuk memulai game baru."""
86     self.score = 0
87     self.game_over = False
88     self.victory = False
89     self.start_time = time.time()
90
91     self.peluru_pemain_list = []
92     self.rintangan_list = []
93     self.energi_list = []
94
95     self.musuh_muncul = False
96     self.musuh_obj = None
97     self.musuh_target_pos = None
98     self.musuh_boundaries = get_musuh_boundaries(self.original_width, self.original_height)
99
100    self.is_in_sequence = False
101    self.shots_fired_in_sequence = 0
102    self.last_shot_time = 0
103    self.last_sequence_time = 0
104
105    self.boss_health = BOSS_MAX_HEALTH
106    self.boss_is_invincible = False
107    self.boss_hit_time = 0
108
109    self.last_pemain_shot_time = 0
110
111    self.last_rintangan_spawn = time.time()
112    self.last_energi_spawn = time.time()
113
114 def _update_player(self, results):
115     """
116         Memperbarui posisi pesawat pemain dan menangani input tembakan.
117
118     Args:
119         results: Objek hasil deteksi dari MediaPipe Face Mesh.
120     """
121     self.pesawat_pos = {'x': 0, 'y': 0, 'w': PESAWAT_W, 'h': PESAWAT_H}
122     if results.multi_face_landmarks:
123         face_landmarks = results.multi_face_landmarks[0].landmark
124         hidung = face_landmarks[1]
125         self.pesawat_pos['x'] = int(hidung.x * self.original_width) - (self.pesawat_pos['w'] // 2)
126         self.pesawat_pos['y'] = int(hidung.y * self.original_height) - (self.pesawat_pos['h'] // 2)
127
128         lip_atas, lip_bawah = face_landmarks[13], face_landmarks[14]
129         lip_distance = abs(lip_atas.y - lip_bawah.y)
130         if lip_distance > MOUTH_OPEN_THRESHOLD and time.time() - self.last_pemain_shot_time >
PEMAIN_SHOOT_COOLDOWN:
```

```

131         peluru_baru = {'x': self.pesawat_pos['x'] + self.pesawat_pos['w'], 'y': self.
132             pesawat_pos['y'] + self.pesawat_pos['h'] // 2 - PEMAIN_PELURU_H // 2}
133             self.peluru_pemain_list.append(peluru_baru)
134             self.last_pemain_shot_time = time.time()
135
136     def _update_player_projectiles(self):
137         """Memperbarui posisi peluru pemain dan mendeteksi tabrakan dengan bos."""
138         for p in self.peluru_pemain_list[:]:
139             p['x'] += PEMAIN_PELURU_SPEED
140             if p['x'] > self.original_width:
141                 self.peluru_pemain_list.remove(p)
142                 continue
143
144             if self.musuh_muncul and self.boss_health > 0 and not self.boss_is_invincible:
145                 if (p['x'] < self.musuh_obj['x'] + self.musuh_obj['w'] and
146                     p['x'] + PEMAIN_PELURU_W > self.musuh_obj['x'] and
147                     p['y'] < self.musuh_obj['y'] + self.musuh_obj['h'] and
148                     p['y'] + PEMAIN_PELURU_H > self.musuh_obj['y']):
149
150                     self.boss_health -= 1
151                     self.boss_is_invincible = True
152                     self.boss_hit_time = time.time()
153                     self.peluru_pemain_list.remove(p)
154
155                     if self.boss_health <= 0:
156                         self.victory, self.game_over = True, True
157                         self.rintangan_list.clear()
158                         break
159
160     def _update_pre_boss_phase(self):
161         """Menangani logika permainan sebelum bos muncul (spawn rintangan & energi)."""
162         if time.time() - self.last_rintangan_spawn > random.uniform(1.5, 3):
163             self._create_rintangan(); self.last_rintangan_spawn = time.time()
164         if time.time() - self.last_energi_spawn > random.uniform(1, 2.5):
165             self._create_energi(); self.last_energi_spawn = time.time()
166
167         for e in self.energi_list[:]:
168             e['x'] -= 4
169             if (self.pesawat_pos['x'] < e['x'] + e['w'] and self.pesawat_pos['x'] + self.
170                 pesawat_pos['w'] > e['x'] and
171                 self.pesawat_pos['y'] < e['y'] + e['h'] and self.pesawat_pos['y'] + self.
172                 pesawat_pos['h'] > e['y']):
173                     self.score += SCORE_PER_ENERGI; self.energi_list.remove(e)
174             if e['x'] < -e['w']: self.energi_list.remove(e)
175
176     def _update_boss_phase(self):
177         """Menangani semua logika yang terkait dengan bos (gerakan, serangan, status)."""
178         if self.musuh_target_pos:
179             dx, dy = self.musuh_target_pos['x'] - self.musuh_obj['x'], self.musuh_target_pos['y'] -
180             self.musuh_obj['y']
181             distance = np.sqrt(dx**2 + dy**2)
182             if distance < MUSUH_MOVE_SPEED: self.musuh_target_pos = self._get_new_musuh_target()
183             else: self.musuh_obj['x'] += int(dx / distance * MUSUH_MOVE_SPEED); self.musuh_obj['y'] +
184             += int(dy / distance * MUSUH_MOVE_SPEED)
185
186             self.musuh_obj['x'] = max(self.musuh_boundaries['min_x'], min(self.musuh_obj['x'], self.
187                 musuh_boundaries['max_x']))
188             self.musuh_obj['y'] = max(self.musuh_boundaries['min_y'], min(self.musuh_obj['y'], self.
189                 musuh_boundaries['max_y']))
190
191             if self.boss_is_invincible and time.time() - self.boss_hit_time >
192             BOSS_INVINCIBILITY_DURATION:

```

```

185         self.boss_is_invincible = False
186
187     if self.boss_health > 0:
188         if not self.is_in_sequence:
189             if time.time() - self.last_sequence_time > SEQUENCE_COOLDOWN:
190                 self.is_in_sequence, self.shots_fired_in_sequence, self.last_shot_time = True,
191                 0, time.time() - SHOT_INTERVAL
192             else:
193                 if self.shots_fired_in_sequence < SHOTS_PER_SEQUENCE:
194                     if time.time() - self.last_shot_time > SHOT_INTERVAL:
195                         pos_y = random.randint(0, self.original_height - RINTANGAN_MUSUH_H)
196                         self.rintangan_list.append({'x': self.musuh_obj['x'], 'y': pos_y, 'w':
197 RINTANGAN_MUSUH_W, 'h': RINTANGAN_MUSUH_H, 'type': 'musuh_bullet'})
198                         self.shots_fired_in_sequence, self.last_shot_time = self.
199                         shots_fired_in_sequence + 1, time.time()
200                     else: self.is_in_sequence, self.last_sequence_time = False, time.time()
201
202     def _update_obstacles(self):
203         """Memperbarui posisi rintangan dan mendekripsi tabrakan dengan pemain."""
204         for r in self.rintangan_list[:]:
205             r['x'] -= 7 if r.get('type') == 'musuh_bullet' else 5
206             if (self.pesawat_pos['x'] < r['x'] + r['w'] and self.pesawat_pos['x'] + self.
207             pesawat_pos['w'] > r['x'] and
208                 self.pesawat_pos['y'] < r['y'] + r['h'] and self.pesawat_pos['y'] + self.
209                 pesawat_pos['h'] > r['y']):
210                 if not self.victory: self.game_over = True
211                 if r['x'] < -r['w']: self.rintangan_list.remove(r)
212
213     def _draw_ui(self, frame):
214         """
215             Menggambar semua elemen User Interface (UI) ke layar.
216
217         Args:
218             frame (np.ndarray): Frame game tempat UI akan digambar.
219
220             elapsed_time = time.time() - self.start_time
221             progress_to_boss = min(1.0, elapsed_time / GAME_DURATION_UNTIL_BOSS)
222             bar_width_val = int(progress_to_boss * (self.original_width - 40))
223             bar_color = COLOR_BOSS_BAR_BG if self.musuh_muncul else COLOR_BOSS_BAR_FG
224             cv2.rectangle(frame, (20, 20), (20 + bar_width_val, 40), bar_color, -1)
225             cv2.rectangle(frame, (20, 20), (self.original_width - 20, 40), COLOR_GREY, 2)
226             text_bar = "LAWAN BOSS!" if self.musuh_muncul else "Menuju Boss"
227             cv2.putText(frame, text_bar, (self.original_width // 2 - 70, 35), cv2.FONT_HERSHEY_SIMPLEX,
228             0.7, COLOR_BLACK, 2)
229
230             cv2.putText(frame, f"SCORE: {self.score}", (20, 70), cv2.FONT_HERSHEY_SIMPLEX, 1,
231             COLOR_YELLOW, 2)
232             if self.musuh_muncul and not self.victory:
233                 cv2.putText(frame, "Buka Mulut untuk Menembak!", (20, 110), cv2.FONT_HERSHEY_SIMPLEX,
234                 0.8, COLOR_WHITE, 2)
235
236             if self.game_over:
237                 overlay = frame.copy()
238                 cv2.rectangle(overlay, (0, self.original_height // 2 - 100), (self.original_width, self.
239                 original_height // 2 + 100), COLOR_BLACK, -1)
240                 cv2.addWeighted(overlay, 0.6, frame, 0.4, 0, frame)
241
242                 end_text, end_color = ("VICTORY!", COLOR_GREEN) if self.victory else ("GAME OVER",
243                 COLOR_RED)
244                 cv2.putText(frame, end_text, (self.original_width // 2 - 180, self.original_height // 2
245                 - 50), cv2.FONT_HERSHEY_TRIPLEX, 2, end_color, 3)
246                 cv2.putText(frame, f"Final Score: {self.score}", (self.original_width // 2 - 150, self.

```

```

236     original_height // 2 + 20), cv2.FONT_HERSHEY_SIMPLEX, 1.5, COLOR_WHITE, 3)
237         cv2.putText(frame, "Tekan 'R' untuk Mulai Lagi", (self.original_width // 2 - 220, self.
238 original_height - 50), cv2.FONT_HERSHEY_SIMPLEX, 1, COLOR_WHITE, 2)
239
240     def _draw_boss_health_bar(self, frame):
241         """
242             Menggambar bilah nyawa (health bar) milik bos.
243
244             Args:
245                 frame (np.ndarray): Frame game tempat bilah nyawa akan digambar.
246             """
247             if self.musuh_muncul and self.boss_health > 0:
248                 bar_w, bar_h, bar_x, bar_y = MUSUH_W, 15, self.musuh_obj['x'], self.musuh_obj['y'] - 15
249 - 5
250                 cv2.rectangle(frame, (bar_x, bar_y), (bar_x + bar_w, bar_y + bar_h), COLOR_RED, -1)
251                 current_health_w = int(bar_w * (self.boss_health / BOSS_MAX_HEALTH))
252                 cv2.rectangle(frame, (bar_x, bar_y), (bar_x + current_health_w, bar_y + bar_h),
253 COLOR_GREEN, -1)
254                 cv2.rectangle(frame, (bar_x, bar_y), (bar_x + bar_w, bar_y + bar_h), COLOR_WHITE, 2)
255
256     def run(self):
257         """Loop utama permainan."""
258         while self.cap.isOpened():
259             success, frame = self.cap.read()
260             if not success: break
261
262             frame = cv2.flip(frame, 1)
263             results = self.face_mesh.process(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
264
265             if not self.game_over:
266                 self._update_player(results)
267                 self._update_player_projectiles()
268
269                 elapsed_time = time.time() - self.start_time
270                 if not self.musuh_muncul and elapsed_time > GAME_DURATION_UNTIL_BOSS:
271                     self.musuh_muncul = True
272                     self.musuh_obj = {'x': self.musuh_boundaries['max_x'], 'y': self.
273 original_height // 2 - MUSUH_H // 2, 'w': MUSUH_W, 'h': MUSUH_H}
274                     self.musuh_target_pos = self._get_new_musuh_target()
275                     self.rintangan_list.clear(); self.energi_list.clear()
276
277                     if self.musuh_muncul: self._update_boss_phase()
278                     else: self._update_pre_boss_phase()
279
280                     self._update_obstacles()
281
282                     # --- Menggambar semua elemen ke frame ---
283                     for p in self.peluru_pemain_list:
284                         cv2.rectangle(frame, (p['x'], p['y']), (p['x'] + PEMAIN_PELURU_W, p['y'] +
285 PEMAIN_PELURU_H), COLOR_BLUE, -1)
286                         for e in self.energi_list:
287                             self._overlay_image(frame, self.energi_img, e['x'], e['y'], e['w'], e['h'])
288                         for r in self.rintangan_list:
289                             self._overlay_image(frame, self.rintangan_img, r['x'], r['y'], r['w'], r['h'])
290
291                         self._overlay_image(frame, self.pesawat_img, self.pesawat_pos['x'], self.pesawat_pos['y'],
292 self.pesawat_pos['w'], self.pesawat_pos['h'])
293
294                         if self.musuh_muncul and self.musuh_obj:
295                             draw_boss = not (self.boss_is_invincible and int(time.time() * 10) % 2 == 0)
296                             if draw_boss and self.boss_health > 0:
297                                 self._overlay_image(frame, self.musuh_img, self.musuh_obj['x'], self.musuh_obj['y'])

```

```

'y'], self.musuh_obj['w'], self.musuh_obj['h'])
        self._draw_boss_health_bar(frame)

    self._draw_ui(frame)

    frame_tampilan = cv2.resize(frame, (DISPLAY_WIDTH, DISPLAY_HEIGHT), interpolation=cv2.INTER_LINEAR)
    cv2.imshow("Game Filter Pesawat", frame_tampilan)

    key = cv2.waitKey(5) & 0xFF
    if key == ord('q'): break
    if self.game_over and key == ord('r'): self.reset()

# --- BARU: Hentikan musik dan mixer saat game ditutup ---
pygame.mixer.music.stop()
pygame.mixer.quit()
# --- AKHIR BLOK BARU ---
self.cap.release()
cv2.destroyAllWindows()

def _overlay_image(self, background, overlay, x, y, w, h):
    """Menempelkan gambar transparan (overlay) ke atas gambar latar (background)."""
    # (Implementasi detail tetap sama)
    if w <= 0 or h <= 0: return
    overlay_resized = cv2.resize(overlay, (w, h))
    if overlay_resized.shape[2] < 4: return
    alpha_channel = overlay_resized[:, :, 3] / 255.0
    y1, y2, x1, x2 = max(0, y), min(background.shape[0], y + h), max(0, x), min(background.shape[1], x + w)
    overlay_roi_h, overlay_roi_w = y2 - y1, x2 - x1
    if overlay_roi_h <= 0 or overlay_roi_w <= 0: return
    background_roi = background[y1:y2, x1:x2]
    overlay_rgb_roi = overlay_resized[0:overlay_roi_h, 0:overlay_roi_w, 0:3]
    alpha = alpha_channel[0:overlay_roi_h, 0:overlay_roi_w]
    for c in range(3):
        background_roi[:, :, c] = (1.0 - alpha) * background_roi[:, :, c] + alpha * overlay_rgb_roi[:, :, c]

def _create_rintangan(self):
    """Membuat objek rintangan baru di luar layar sebelah kanan."""
    min_y, max_y = 75, self.original_height - 75 - RINTANGAN_H
    y_pos = random.randint(min_y, max_y) if min_y < max_y else self.original_height // 2
    self.rintangan_list.append({'x': self.original_width, 'y': y_pos, 'w': RINTANGAN_W, 'h': RINTANGAN_H, 'type': 'normal'})

def _create_energi(self):
    """Membuat objek energi baru di luar layar sebelah kanan."""
    min_y, max_y = 75, self.original_height - 75 - ENERGI_H
    y_pos = random.randint(min_y, max_y) if min_y < max_y else self.original_height // 2
    self.energi_list.append({'x': self.original_width, 'y': y_pos, 'w': ENERGI_W, 'h': ENERGI_H})

def _get_new_musuh_target(self):
    """Menghasilkan posisi target acak baru untuk pergerakan bos."""
    target_x = random.randint(self.musuh_boundaries['min_x'], self.musuh_boundaries['max_x'])
    target_y = random.randint(self.musuh_boundaries['min_y'], self.musuh_boundaries['max_y'])
    return {'x': target_x, 'y': target_y}

```

Kode 1: game_manager.py

3.1.2 Code Program config.py

File config.py berfungsi sebagai pusat konfigurasi untuk semua parameter permainan. Ini mencakup:

- Pengaturan Tampilan: Dimensi jendela game yang diinginkan (DISPLAY_WIDTH, DISPLAY_HEIGHT).
- Pengaturan Game Umum: Durasi hingga bos muncul, skor per item energi.
- Path Aset: Lokasi file gambar dan musik di dalam folder assets.
- Ukuran Objek: Dimensi (lebar dan tinggi) untuk pesawat, rintangan, energi, bos, dan proyektil.
- Mekanik Boss dan Pemain: Parameter seperti jumlah nyawa bos, durasi kekebalan, pola serangan (cooldown, jumlah tembakan, interval), kecepatan gerakan bos, cooldown tembakan pemain, ambang batas bukaan mulut, dan kecepatan peluru pemain.
- Batas Gerakan Musuh: Fungsi get_musuh_boundaries untuk menghitung area pergerakan bos secara dinamis berdasarkan resolusi asli webcam.
- Pengaturan Warna: Definisi konstanta warna yang digunakan untuk menggambar elemen UI dan objek.

Pemisahan konfigurasi ini memungkinkan penyesuaian parameter game dengan mudah tanpa menyentuh logika inti di game_manager.py.

```

1 """
2 """
3 File Konfigurasi Pusat untuk Game Filter Pesawat.
4
5 File ini berisi semua variabel global dan pengaturan yang digunakan di seluruh
6 proyek. Tujuannya adalah untuk memudahkan penyesuaian (tweaking) parameter game
7 seperti ukuran objek, kecepatan, durasi, path aset, dan warna tanpa harus
8 mengubah kode logika utama di 'game_manager.py'.
9 """
10
11 # --- PENGATURAN TAMPILAN ---
12 DISPLAY_WIDTH = 960
13 DISPLAY_HEIGHT = 720
14
15 # --- PENGATURAN GAME ---
16 GAME_DURATION_UNTIL_BOSS = 30
17 SCORE_PER_ENERGI = 10
18
19 # --- PATH ASET GAMBAR ---
20 # Pastikan semua gambar berada di dalam folder 'assets'
21 PESAWAT_PATH = 'assets/pesawat.png'
22 RINTANGAN_PATH = 'assets/rintangan.png'
23 ENERGI_PATH = 'assets/energi.png'
24 MUSUH_PATH = 'assets/musuh.png'
25 BACKGROUND_MUSIC_PATH = 'assets/undertale_megalovania.mp3'
26
27 # --- PENGATURAN UKURAN OBJEK ---
28 PESAWAT_W, PESAWAT_H = 78, 40
29 RINTANGAN_W, RINTANGAN_H = 60, 60
30 ENERGI_W, ENERGI_H = 30, 54
31 MUSUH_W, MUSUH_H = 150, 150
32 RINTANGAN_MUSUH_W, RINTANGAN_MUSUH_H = RINTANGAN_W // 2, RINTANGAN_H // 2
33 PEMAIN_PELRUW_W, PEMAIN_PELRUW_H = 25, 10
34
35 # --- PENGATURAN MEKANIK BOSS ---
36 BOSS_MAX_HEALTH = 3

```

```

37 BOSS_INVINCIBILITY_DURATION = 3.0
38 SEQUENCE_COOLDOWN = 1.5
39 SHOTS_PER_SEQUENCE = 10
40 SHOT_INTERVAL = 0.5
41 MUSUH_MOVE_SPEED = 3
42
43 # --- PENGATURAN MEKANIK PEMAIN ---
44 PEMAIN_SHOOT_COOLDOWN = 0.75
45 MOUTH_OPEN_THRESHOLD = 0.05
46 PEMAIN_PELURU_SPEED = 15
47
48 # --- PENGATURAN BATAS GERAKAN MUSUH (tergantung ukuran frame asli) ---
49 def get_musuh_boundaries(width, height):
50     """Menghitung dan mengembalikan batas pergerakan untuk boss.
51
52     Args:
53         width (int): Lebar frame asli dari webcam.
54         height (int): Tinggi frame asli dari webcam.
55
56     Returns:
57         dict: Dictionary berisi batas 'min_x', 'max_x', 'min_y', 'max_y'.
58     """
59     return {
60         'min_x': int(width * 0.70),
61         'max_x': width - MUSUH_W - 20,
62         'min_y': 20,
63         'max_y': height - MUSUH_H - 20
64     }
65
66 # --- PENGATURAN WARNA (B, G, R) ---
67 COLOR_WHITE = (255, 255, 255)
68 COLOR_BLACK = (0, 0, 0)
69 COLOR_RED = (0, 0, 255)
70 COLOR_GREEN = (0, 255, 0)
71 COLOR_BLUE = (255, 100, 0)
72 COLOR_YELLOW = (0, 255, 255)
73 COLOR_GREY = (128, 128, 128)
74 COLOR_BOSS_BAR_BG = (200, 0, 0)
75 COLOR_BOSS_BAR_FG = (0, 180, 0)

```

Kode 2: config.py

3.1.3 Code Program main.py

File main.py adalah titik masuk (entry point) utama untuk menjalankan aplikasi game. Kode di dalamnya sangat ringkas:

- Mengimpor kelas GameManager dari game_manager.py.
- Dalam blok kondisional if __name__ == "__main__": (standar praktik untuk skrip Python yang dapat dieksekusi), sebuah instance dari GameManager dibuat.
- Metode run() dari instance GameManager tersebut kemudian dipanggil untuk memulai dan menjalankan loop utama permainan.

```

1 """
2 """
3 Titik Masuk Utama (Main Entry Point) untuk Game Filter Pesawat.
4
5 File ini bertanggung jawab untuk membuat instance dari kelas GameManager

```

```
6 dan menjalankan loop utama permainan. Untuk menjalankan game, eksekusi file ini
7 dari terminal: `python main.py`
8 """
9
10 from game_manager import GameManager
11
12 if __name__ == "__main__":
13     # Membuat instance dari kelas utama game
14     game = GameManager()
15
16     # Memulai dan menjalankan loop utama game
17     game.run()
```

Kode 3: main.py

3.2 Penjelasan Code Program

Class GameManager: Kelas utama yang mengatur seluruh logika, status, dan alur permainan. Bertanggung jawab untuk inisialisasi, menjalankan loop utama, memperbarui status semua objek game, menggambar di layar, dan menangani input dari pemain.

3.2.1 __init__(self)

- Penjelasan: Ini adalah konstruktor kelas. Dipanggil secara otomatis saat objek GameManager dibuat.
- Tugas Utama:
 - Memanggil `_init_camera_and_mediapipe()` untuk menyiapkan webcam dan model deteksi wajah MediaPipe.
 - Memanggil `_load_assets()` untuk memuat semua gambar yang dibutuhkan (pesawat, rintangan, dll.).
 - BARU: Menginisialisasi pygame.mixer untuk audio.
 - BARU: Mencoba memuat file musik latar (`BACKGROUND_MUSIC_PATH`), mengatur volume, dan memainkannya secara berulang (-1). Jika gagal, akan mencetak pesan error.
 - Memanggil `self.reset()` untuk mengatur ulang semua variabel game ke kondisi awal.

3.2.2 _init_camera_and_mediapipe(self)

- Penjelasan: Fungsi internal (ditandai dengan `_`) yang bertugas menginisialisasi sumber input video (webcam) dan model MediaPipe Face Mesh
- Tugas Utama:
 - Membuka koneksi ke webcam default (`cv2.VideoCapture(0)`). Jika gagal, program akan keluar.
 - Mendapatkan resolusi (lebar dan tinggi) asli dari frame webcam.
 - Menginisialisasi objek FaceMesh dari `mp.solutions.face_mesh`. Model ini dikonfigurasi untuk mendeteksi maksimal 1 wajah dengan tingkat kepercayaan deteksi dan pelacakan minimal 0.5.

3.2.3 `_load_assets(self)`

- Penjelasan: Fungsi internal untuk memuat semua asset gambar yang diperlukan oleh game.
- Tugas Utama:
 - Memanggil `_load_single_asset()` untuk setiap aset gambar (pesawat, rintangan, energi, musuh) berdasarkan path yang didefinisikan di config.py (misalnya PESAWAT_PATH).
 - Menggunakan blok try-except untuk menangani error jika file tidak ditemukan (FileNotFoundException) atau jika gambar tidak memiliki format yang benar (TypeError, khususnya tidak ada alpha channel). Jika error terjadi, program akan keluar.

3.2.4 `_load_single_asset(self, path)`

- Penjelasan: Fungsi internal pembantu yang memuat satu file gambar dan melakukan validasi dasar.
- Parameter:
 - `path (str)`: Lokasi file gambar.
- Tugas Utama:
 - Membaca file gambar menggunakan `cv2.imread(path, cv2.IMREAD_UNCHANGED)`. IMREAD_UNCHANGED memastikan alpha channel (untuk transparansi) ikut dimuat jika ada.
 - Mengecek apakah gambar berhasil dimuat. Jika tidak (`img is None`), akan memunculkan FileNotFoundError.
 - Mengecek apakah gambar memiliki 4 channel (RGB + Alpha). Jika tidak (`img.shape[2] != 4`), akan memunculkan TypeError karena game ini memerlukan gambar dengan transparansi.
- Mengembalikan: Gambar yang dimuat sebagai array NumPy jika valid.

3.2.5 `reset(self)`

- Penjelasan: Mengatur ulang semua variabel status game ke nilai awalnya. Ini berguna untuk memulai game baru atau setelah game over.
- Tugas Utama:
 - Mengatur skor (`self.score`) ke 0.
 - Mengatur status `game_over` dan `victory` ke False.
 - Mencatat waktu mulai (`self.start_time`).
 - Mengosongkan daftar peluru pemain, rintangan, dan energi.
 - Mengatur ulang status terkait musuh/bos (belum muncul, objek musuh kosong, target posisi kosong).
 - Mengatur ulang batas pergerakan musuh berdasarkan resolusi layar.
 - Mengatur ulang variabel terkait sekuens tembakan musuh dan cooldown-nya.
 - Mengatur ulang nyawa bos (`self.boss_health`) ke maksimum, status tak terkalahkan (`self.boss_is_invincible`) dan waktu bos terkena serangan.
 - Mengatur ulang cooldown tembakan pemain dan waktu spawn terakhir untuk rintangan dan energi.

3.2.6 `_update_player(self, results)`

- Penjelasan: Fungsi internal untuk memperbarui posisi pesawat pemain berdasarkan deteksi wajah dan menangani aksi menembak pemain.
- Parameter:
 - `results`: Objek hasil deteksi dari MediaPipe Face Mesh yang berisi landmark wajah.
- Tugas Utama:
 - Menginisialisasi posisi default pesawat.
 - Jika landmark wajah terdeteksi:
 - * Mengambil koordinat titik hidung (landmark ke-1) untuk menentukan posisi X dan Y pesawat di tengah hidung.
 - * Mengambil koordinat bibir atas (landmark ke-13) dan bibir bawah (landmark ke-14).
 - * Menghitung jarak vertikal antara bibir atas dan bawah.
 - * Jika jarak bibir melebihi ambang batas (`MOUTH_OPEN_THRESHOLD`) dan cooldown tembakan pemain (`PEMAIN_SHOOT_COOLDOWN`) telah terpenuhi:
 - Membuat objek peluru baru di depan pesawat.
 - Menambahkan peluru tersebut ke `self.peluru_pemain_list`.
 - Memperbarui `self.last_pemain_shot_time` untuk cooldown.

3.2.7 `_update_player_projectiles(self)`

- Penjelasan: Fungsi internal untuk memperbarui posisi semua peluru yang ditembakkan oleh pemain dan mendeteksi tabrakan dengan bos.
- Tugas Utama:
 - Melakukan iterasi pada salinan daftar peluru pemain (`self.peluru_pemain_list[:]`) agar aman untuk menghapus item saat iterasi.
 - Menggerakkan setiap peluru ke kanan dengan kecepatan `PEMAIN_PELURU_SPEED`.
 - Jika peluru keluar layar (melebihi lebar layar), peluru akan dihapus dari daftar.
 - Jika bos sudah muncul (`self.musuh_muncul`), nyawa bos lebih dari 0, dan bos tidak sedang tak terkalahkan (`not self.boss_is_invincible`):
 - * Mengecek tabrakan antara peluru pemain dan area bos.
 - * Jika terjadi tabrakan:
 - Mengurangi nyawa bos (`self.boss_health`).
 - Mengaktifkan status tak terkalahkan untuk bos (`self.boss_is_invincible`) untuk sementara.
 - Mencatat waktu bos terkena serangan (`self.boss_hit_time`).
 - Menghapus peluru pemain yang mengenai bos.
 - Jika nyawa bos habis (`self.boss_health <= 0`):
 - Set `self.victory` dan `self.game_over` menjadi True.
 - Menghapus semua rintangan yang mungkin masih ada.
 - Keluar dari loop (karena bos sudah dikalahkan).
- Mengembalikan: Gambar yang dimuat sebagai array NumPy jika valid.

3.2.8 `_update_pre_boss_phase(self)`

- Penjelasan: Fungsi internal yang menangani logika permainan sebelum bos muncul. Ini termasuk memunculkan rintangan dan item energi.
- Tugas Utama:
 - Memunculkan rintangan baru (`_create_rintangan()`) secara berkala berdasarkan interval acak (`random.uniform(1.5, 3)` detik).
 - Memunculkan item energi baru (`_create_energi()`) secara berkala berdasarkan interval acak (`random.uniform(1, 2.5)` detik).
 - Memperbarui posisi item energi:
 - * Menggerakkan setiap item energi ke kiri.
 - * Mengecek tabrakan antara pesawat pemain dan item energi. Jika bertabrakan, skor pemain bertambah (`SCORE_PER_ENERGI`) dan item energi dihapus.
 - * Jika item energi keluar layar, item tersebut dihapus.
 - Mengembalikan: Gambar yang dimuat sebagai array NumPy jika valid.

3.2.9 `_update_boss_phase(self)`

- Penjelasan: Fungsi internal yang menangani semua logika yang terkait dengan bos, termasuk pergerakan, pola serangan, dan status tak terkalahkan.
- Tugas Utama:
 - Pergerakan Bos:
 - * Jika ada target posisi (`self.musuh_target_pos`):
 - Hitung vektor arah dan jarak ke target.
 - Jika jarak sudah sangat dekat, dapatkan target posisi baru (`_get_new_musuh_target()`).
 - Jika tidak, gerakkan bos ke arah target dengan kecepatan `MUSUH_MOVE_SPEED`.
 - * Pastikan posisi bos selalu berada dalam batas yang telah ditentukan (`self.musuh_boundaries`).
 - Status Tak Terkalahkan (Invincibility):
 - * Jika bos sedang tak terkalahkan dan durasi tak terkalahkan (`BOSS_INVINCIBILITY_DURATION`) sudah lewat, set `self.boss_is_invincible` kembali ke `False`.
 - Pola Serangan Bos (Jika nyawa > 0):
 - * Jika tidak sedang dalam sekuens serangan (`not self.is_in_sequence`) dan cooldown antar sekuens (`SEQUENCE_COOLDOWN`) sudah lewat:
 - Mulai sekuens serangan baru: set `self.is_in_sequence` ke `True`, reset `self.shots_fired_in_sequence` dan atur `self.last_shot_time` (dengan sedikit offset agar tembakan pertama bisa langsung terjadi).
 - * Jika sedang dalam sekuens serangan:
 - Jika jumlah tembakan dalam sekuens saat ini (`self.shots_fired_in_sequence`) masih kurang dari `SHOTS_PER_SEQUENCE` dan interval antar tembakan (`SHOT_INTERVAL`) sudah lewat: 0

- Buat rintangan baru (peluru musuh) pada posisi acak Y dari posisi X bos. Rintangan ini ditandai dengan 'type': 'musuh_bullet'.
- Tambah self.shots_fired_in_sequence dan perbarui self.last_shot_time.
- Jika jumlah tembakan dalam sekuens sudah mencapai batas:
- Akhiri sekuens serangan (self.is_in_sequence = False) dan catat self.last_sequence_time untuk cooldown sekuens berikutnya.

3.2.10 _update_obstacles(self)

- Penjelasan: Fungsi internal untuk memperbarui posisi semua rintangan (baik yang normal maupun peluru musuh) dan mendeteksi tabrakan dengan pemain.
- Tugas Utama:
 - Melakukan iterasi pada salinan daftar rintangan (self.rintangan_list[:]).
 - Menggerakkan setiap rintangan ke kiri. Peluru musuh ('type': 'musuh_bullet') bergerak lebih cepat (kecepatan 7) daripada rintangan normal (kecepatan 5).
 - Mengecek tabrakan antara pesawat pemain dan rintangan
 - * Jika bertabrakan dan pemain belum menang (not self.victory), maka set self.game_over ke True.
 - Jika rintangan keluar layar (melebihi batas kiri), rintangan tersebut dihapus dari daftar.

3.2.11 _draw_ui(self, frame)

- Penjelasan: Fungsi internal untuk menggambar semua elemen Antarmuka Pengguna (UI) ke frame game.
- Parameter:
 - frame (np.ndarray): Frame game (gambar) tempat UI akan digambar.
- Tugas Utama:
 - Progress Bar Menuju Bos:
 - * Menghitung waktu yang telah berlalu sejak game dimulai.
 - * Menghitung progres menuju munculnya bos (0.0 hingga 1.0).
 - * Menggambar persegi panjang sebagai latar belakang dan foreground progress bar. Warna foreground berubah jika bos sudah muncul.
 - * Menambahkan teks "Menuju Boss" atau "LAWAN BOSS!" di atas progress bar.
 - Skor: Menampilkan skor pemain saat ini.
 - Instruksi Menembak: Jika bos sudah muncul dan pemain belum menang, tampilkan teks "Buka Mulut untuk Menembak!".
 - Layar Game Over/Victory:

- * Jika self.game_over adalah True:
 - Menggambar lapisan semi-transparan hitam di tengah layar.
 - Menampilkan teks "VICTORY!" (hijau) atau "GAME OVER" (merah) berukuran besar.
 - Menampilkan skor akhir pemain.
 - Menampilkan instruksi "Tekan 'R' untuk Mulai Lagi".

3.2.12 _draw_boss_health_bar(self, frame)

- Penjelasan: Fungsi internal untuk menggambar bilah nyawa (health bar) milik bos.
- Parameter:
 - frame (np.ndarray): Frame game tempat bilah nyawa akan digambar.
- Tugas Utama:
 - Jika bos sudah muncul (self.musuh_muncul) dan nyawanya masih ada (self.boss_health > 0):
 - * Menentukan dimensi dan posisi bilah nyawa (di atas objek bos).
 - * Menggambar latar belakang bilah nyawa (merah).
 - * Menghitung lebar bilah nyawa saat ini berdasarkan persentase sisa nyawa bos.
 - * Menggambar bilah nyawa saat ini (hijau) di atas latar belakang.
 - * Menggambar bingkai putih di sekitar bilah nyawa.

3.2.13 run(self)

- Penjelasan: Ini adalah loop utama permainan yang berjalan terus menerus hingga pemain keluar.
- Tugas Utama:
 - Loop Utama (while self.cap.isOpened()):
 - * Membaca frame baru dari webcam. Jika gagal, loop berhenti.
 - * Membalik frame secara horizontal (cv2.flip(frame, 1)) agar seperti cermin.
 - * Memproses frame dengan self.face_mesh untuk mendeteksi landmark wajah.
 - * Jika Game Belum Berakhir (not self.game_over)
 - Memanggil _update_player(results) untuk memperbarui posisi pemain dan menembak.
 - Memanggil _update_player_projectiles() untuk memperbarui peluru pemain.
 - Mengecek apakah sudah waktunya bos muncul (elapsed_time > GAME_DURATION_UNTIL_BOS).
 - Jika ya:

- Set self.musuh_muncul = True.
 - Inisialisasi objek musuh (self.musuh_obj) di posisi awal.
 - Dapatkan target pergerakan awal musuh.
 - Hapus semua rintangan dan energi yang ada dari fase sebelum bos.
 - Jika bos sudah muncul, panggil _update_boss_phase().
 - Jika bos belum muncul, panggil _update_pre_boss_phase().
 - Memanggil _update_obstacles() untuk memperbarui rintangan.
- * Menggambar Semua Elemen:
- Menggambar semua peluru pemain (sebagai persegi panjang biru).
 - Menggambar semua item energi menggunakan _overlay_image().
 - Menggambar semua rintangan menggunakan _overlay_image().
 - Menggambar pesawat pemain menggunakan _overlay_image().
 - Jika bos sudah muncul dan ada objeknya:
 - Menggambar bos menggunakan _overlay_image(). Bos akan berkedip (tidak digambar setiap frame genap) jika sedang tak terkalahkan untuk memberikan efek visual.
 - Memanggil _draw_boss_health_bar() untuk menggambar nyawa bos.
 - Memanggil _draw_ui() untuk menggambar semua elemen UI.
- * Mengubah ukuran frame akhir ke ukuran tampilan yang diinginkan (DISPLAY_HEIGHT).
- * Menampilkan frame yang sudah diproses ke jendela "Game Filter Pesawat".
- * Menangani Input Keyboard:
- Menunggu input tombol selama 5 milidetik.
 - Jika tombol 'q' ditekan, keluar dari loop utama.
 - Jika game sudah berakhir (self.game_over) dan tombol 'r' ditekan, panggil self.reset() untuk memulai ulang permainan.
- Setelah Loop Selesai:
- * BARU: Menghentikan pemutaran musik (pygame.mixer.music.stop()).
 - * BARU: Menutup mixer Pygame (pygame.mixer.quit()).
 - * Melepaskan sumber webcam (self.cap.release()).
 - * Menutup semua jendela OpenCV (cv2.destroyAllWindows()).

3.2.14 `_overlay_image(self, background, overlay, x, y, w, h)`

- Penjelasan: Fungsi internal untuk menempelkan sebuah gambar (overlay) yang memiliki transparansi (alpha channel) ke atas gambar latar (background) pada posisi tertentu.
- Parameter:
 - background: Gambar latar (frame game).
 - overlay: Gambar yang akan ditempelkan (misalnya, pesawat, rintangan).
 - x, y: Koordinat pojok kiri atas tempat overlay akan ditempelkan.
 - w, h: Lebar dan tinggi yang diinginkan untuk overlay setelah di-resize.
- Tugas Utama:
 - Melakukan return jika lebar atau tinggi w atau h adalah nol atau negatif. Mengubah ukuran gambar overlay sesuai w dan h.
 - Melakukan return jika gambar overlay tidak memiliki alpha channel (kurang dari 4 channel).
 - Mengekstrak alpha channel dari `overlay_resized` dan menormalisasinya (nilai 0-1).
 - Menentukan Region of Interest (ROI) pada background tempat overlay akan ditempelkan, dengan memastikan tidak keluar batas background.
 - Memastikan ROI valid (tinggi dan lebar > 0).
 - Mengambil bagian RGB dari `overlay_resized` yang sesuai dengan ROI.
 - Menggunakan alpha channel untuk menggabungkan (blend) `background_roi` dengan `overlay_rgb_roi`. Formula blending: $\text{background} = (1 - \text{alpha}) * \text{background} + \text{alpha} * \text{overlay}$. Ini dilakukan untuk setiap channel warna (R, G, B).

3.2.15 `_create_rintangan(self)`

- Penjelasan: Fungsi internal untuk membuat objek rintangan baru.
- Tugas Utama:
 - Menentukan batas atas (`min_y`) dan bawah (`max_y`) untuk posisi Y rintangan agar tidak terlalu ke pinggir layar.
 - Menghasilkan posisi Y acak (`y_pos`) di antara `min_y` dan `max_y`. Jika `min_y` tidak kurang dari `max_y` (misalnya karena layar terlalu kecil), posisi Y diatur ke tengah layar.
 - Menambahkan dictionary baru yang merepresentasikan rintangan ke `self.rintangan_list`. Rintangan ini dimulai dari sisi kanan layar (`self.original_width`) dengan lebar (`RINTANGAN_W`), tinggi (`RINTANGAN_H`), dan tipe 'normal'.

3.2.16 _create_energi(self)

- Penjelasan: Fungsi internal untuk membuat objek energi baru.
- Tugas Utama:
 - Sama seperti _create_rintangan, menentukan batas atas dan bawah untuk posisi Y item energi.
 - Menghasilkan posisi Y acak (y_pos).
 - Menambahkan dictionary baru yang merepresentasikan item energi ke self.energi_list. Item energi ini juga dimulai dari sisi kanan layar dengan lebar (ENERGI_W) dan tinggi (ENERGI_H).

3.2.17 _get_new_musuh_target(self)

- Penjelasan: Fungsi internal untuk menghasilkan posisi target acak baru bagi pergerakan bos.
- Tugas Utama:
 - Memilih koordinat X acak (target_x) dalam batas horizontal pergerakan musuh (self.musu_boundaries['min_x'] dan self.musu_boundaries['max_x']).
 - Memilih koordinat Y acak (target_y) dalam batas vertikal pergerakan musuh (self.musu_boundaries['min_y'] dan self.musu_boundaries['max_y']).
- Mengembalikan: Dictionary yang berisi koordinat x dan y dari target baru.

3.3 Desain Tampilan Object

Desain visual objek dalam game menggunakan asset gambar PNG dengan latar belakang transparan untuk integrasi yang mulus ke dalam frame video. Ukuran dan path asset dikonfigurasi dalam config.py

3.3.1 Desain Pesawat

Objek pesawat.png adalah asset visual yang digunakan untuk menggambarkan entitas utama yang dikendalikan oleh pemain, yaitu pesawat luar angkasa, dalam game King-Astro: Game Filter Pesawat Luar Angkasa. Ukuran pesawat ini, yakni lebar dan tinggi, telah ditentukan melalui file konfigurasi config.py dengan variabel PESAWAT_W dan PESAWAT_H. Tampilan visual dari pesawat.png seperti bentuk, warna, dan gaya desain ditentukan oleh isi gambar itu sendiri dan berfungsi sebagai representasi grafis dari pesawat yang digunakan pemain selama permainan.



Gambar 1: Tampilan Desain Pesawat

3.3.2 Desain Meteor (Rintangan)

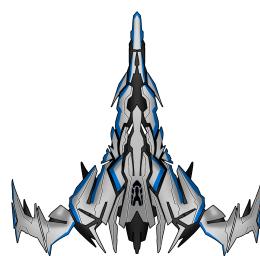
Objek `rintangan.png` digunakan sebagai representasi grafis meteor dalam game King-Astro. Gambar ini memiliki dua fungsi utama, yaitu sebagai rintangan biasa yang muncul secara berkala selama permainan dan sebagai peluru yang ditembakkan oleh karakter bos. Ukuran dari rintangan ini dapat diatur melalui file konfigurasi `config.py`, dengan ukuran berbeda untuk rintangan reguler (`RINTANGAN_W`, `RINTANGAN_H`) dan rintangan dari musuh (`RINTANGAN_MUSUH_W`, `RINTANGAN_MUSUH_H`). Karena digambarkan sebagai meteor, visual dari `rintangan.png` kemungkinan menampilkan objek seperti batu luar angkasa dengan tekstur kasar atau kawah. detail seperti warna, bentuk, dan efek visual lainnya seperti pada Gambar 2 di bawah ini.



Gambar 2: Tampilan Desain rintangan

3.3.3 Desain Boss

Objek `musuh.png` digunakan untuk merepresentasikan karakter antagonis utama atau bos akhir yang akan dihadapi pemain di penghujung permainan King-Astro. Kemunculan bos ini diatur berdasarkan durasi permainan yang telah ditentukan melalui variabel `GAME_DURATION_UNTIL_BOSS` pada file konfigurasi. Gambar diambil dari direktori `assets/musuh.png` dan memiliki ukuran lebar dan tinggi yang sudah ditentukan dalam file `config.py` dengan variabel `MUSUH_W` dan `MUSUH_H`. Desain visual dari `musuh.png` seperti bentuk karakter, pilihan warna, detail ornamen, dan gaya artistik secara keseluruhan bergantung pada isi gambar aslinya. Gambar inilah yang akan ditampilkan sebagai tantangan utama bagi pemain di akhir permainan.



Gambar 3: Tampilan Desain Boss

3.3.4 Desain Energi

Objek `energi.png` digunakan untuk menampilkan objek kolektibel dalam game King-Astro yang berfungsi sebagai item penambah skor bagi pemain. Saat pesawat pemain menyentuh objek energi ini, skor akan bertambah sesuai nilai yang telah ditentukan melalui variabel `SCORE_PER_ENERGI` dalam file konfigurasi. Gambar ini berada di direktori `assets/energi.png`, dengan ukuran lebar dan tinggi

yang telah diatur melalui variabel ENERGI_W dan ENERGI_H pada config.py. Desain visual dari energi.png—seperti bentuk kristal, simbol bercahaya, atau elemen visual lainnya—serta warna dan detail artistik yang menggambarkan fungsinya sebagai penambah skor, sepenuhnya bergantung pada isi gambar tersebut.



Gambar 4: Tampilan Desain Energi

3.4 Dokumentasi dan Penjelasan Hasil Running Program

3.4.1 Awal Mulai



Gambar 5: Tampilan Awal Permainan

Gambar 5 menampilkan tampilan antarmuka pengguna (UI) dan elemen-elemen utama dari

mekanisme permainan King-Astro pada saat permainan sedang berlangsung secara real-time. Adapun penjelasan dari elemen-elemen yang terdapat dalam gambar tersebut adalah sebagai berikut:

- Interaksi Pemain dan Representasi Avatar:

Pada bagian tengah layar, terlihat wajah pengguna yang berhasil terdeteksi oleh sistem. Hal ini menunjukkan bahwa fitur pelacakan wajah (face tracking) dari pustaka MediaPipe sedang aktif dan digunakan sebagai metode input utama dalam permainan. Sebuah gambar pesawat (pesawat.png) ditampilkan tepat pada area hidung pengguna. Gambar ini berperan sebagai avatar pemain di dalam game, yang pergerakannya mengikuti posisi hidung pengguna yang terdeteksi secara real-time.

- Objek-Objek Interaktif dalam Game:

Rintangan (rintangan.png) Tampak dua objek berbentuk seperti asteroid berwarna abu-abu yang berada di sisi kiri dan kanan kepala pengguna. Kedua objek ini berfungsi sebagai rintangan yang harus dihindari oleh pemain agar permainan dapat terus berlangsung. Energi (energi.png) Terlihat pula dua objek berwarna oranye yang menyerupai baterai atau sumber daya. Objek ini diletakkan di sisi kiri dan kanan layar, sedikit lebih tinggi dari posisi rintangan. Fungsi utamanya adalah sebagai item yang bisa dikumpulkan oleh pemain untuk menambah skor.

- Elemen Antarmuka Pengguna (GUI):

Progress Bar Permainan Di bagian atas layar, terdapat bar horizontal berwarna hijau dengan teks "Menuju Boss" di bagian tengahnya. Bar ini digunakan untuk menunjukkan progres permainan, khususnya sejauh mana pemain telah mendekati fase pertempuran dengan bos. Skor Pemain Tepat di bawah progress bar, pada bagian kiri atas layar, ditampilkan skor pemain dengan format teks "SCORE: 0" berwarna kuning. Pada Gambar 5, skor masih menunjukkan angka nol yang berarti permainan baru saja dimulai.

3.4.2 Terkena Rintangan



Gambar 6: Tampilan Game Over

Gambar 6 menampilkan tampilan layar "Game Over" dari permainan King-Astro. Tampilan ini menunjukkan bahwa sesi permainan telah berakhir, yang disebabkan oleh tabrakan antara pesawat pemain dengan salah satu objek rintangan.

- Tampilan Status Game Over:

Bagian paling mencolok adalah teks "GAME OVER" berwarna merah yang ditampilkan besar di tengah layar. Tulisan ini secara langsung memberi tahu pemain bahwa permainan telah dihentikan. Selain itu, terdapat lapisan gelap semi-transparan yang menutupi sebagian besar area permainan. Efek visual ini membantu mengarahkan perhatian pemain pada status "Game Over" dan membedakan antara kondisi permainan yang aktif dan permainan yang telah selesai. Tampilan ini ditampilkan oleh sistem saat variabel self.game_over bernilai True, yang diproses melalui metode _draw_ui.

- Informasi Skor dan Instruksi Ulang:

Tepat di bawah tulisan "GAME OVER", terdapat informasi skor akhir berupa teks "Final Score: 10" yang berwarna putih. Angka ini menunjukkan skor total yang dikumpulkan pemain sebelum permainan berakhir. Dalam konteks ini, skor 10 berarti pemain berhasil mengumpulkan satu item energi (karena nilai SCORE_PER_ENERGI adalah 10). Kemudian, di bagian bawah layar muncul pesan instruksi "Tekan 'R' untuk Mulai Lagi", yang memberi tahu pemain cara untuk memulai ulang permainan.

- Indikasi Penyebab Kalah (Tabrakan):

Walaupun sebagian besar area tertutup oleh tampilan "Game Over", masih terlihat objek rintangan (asteroid) yang cukup dekat dengan posisi pesawat pemain, yang ditampilkan di sekitar wajah. Secara logika, dalam metode _update_obstacles, status self.game_over akan berubah menjadi True ketika sistem mendeteksi adanya tabrakan antara pesawat dan rintangan.

3.4.3 Melawan Musuh/Boss



Gambar 7: Tampilan Melawan Boss

Gambar 7 menunjukkan antarmuka pengguna (User Interface) dan elemen gameplay utama yang muncul saat fase pertarungan melawan bos dalam permainan King-Astro. Fase ini menjadi bagian paling menantang dalam permainan dan menguji keterampilan pemain secara maksimal.

- Representasi Pemain dan Mekanisme Serangan:

Terlihat wajah pemain dengan kondisi mulut terbuka, yang menunjukkan bahwa pemain sedang mengaktifkan mekanisme menembak. Hal ini sesuai dengan petunjuk di layar, yaitu "Buka Mulut untuk Menembak!". Avatar pesawat (menggunakan asset pesawat.png) tetap berada di area wajah,

khususnya sekitar hidung, sebagai penanda posisi pemain dalam permainan. Sebuah proyektil berwarna biru terlihat meluncur dari arah pesawat menuju ke arah musuh, yang menjadi bukti bahwa aksi menembak sedang berlangsung.

- Karakter Bos dan Atributnya:

Di sisi kanan layar, terlihat sebuah wahana luar angkasa besar dengan desain yang kompleks. Ini merupakan representasi visual dari karakter bos (musuh.png), yang muncul setelah permainan mencapai waktu tertentu. Di atas karakter bos terdapat bar darah (health bar), yang menampilkan sisa HP bos dalam bentuk segmen hijau. Sisa bar (jika berwarna merah atau kosong) menunjukkan bagian kesehatan yang sudah berkurang. Bar ini divisualisasikan melalui metode `_draw_boss_health_bar`. Tak jauh dari pesawat pemain, terdapat sebuah objek kecil mirip asteroid merupakan proyektil musuh (`musuh_bullet`) yang harus dihindari oleh pemain selama fase ini[4].

- Elemen UI Khusus Fase Bos:

Bagian atas layar menampilkan teks besar "LAWAN BOSS!" dengan latar warna berbeda, sebagai penanda bahwa permainan telah masuk ke dalam fase pertarungan bos. Perubahan ini dikendalikan oleh metode `_draw_ui` ketika variabel `self.musuh_muncul` bernilai aktif. Skor pemain saat ini, yaitu "SCORE: 140", tetap terlihat di pojok kiri atas. Instruksi penting "Buka Mulut untuk Menembak!" juga ditampilkan di bawah skor sebagai pengingat cara menyerang bos.

3.4.4 Kemenangan / Akhir Game



Gambar 8: Tampilan Menang

Gambar 8 memperlihatkan tampilan antarmuka pengguna (User Interface) dari permainan King-Astro ketika pemain berhasil mencapai kemenangan. Keberhasilan ini terjadi setelah pemain sukses mengalahkan entitas bos dalam fase pertarungan sebelumnya.

- Tampilan Visual Status Kemenangan (Victory):

Elemen utama yang menarik perhatian pada layar ini adalah tulisan "VICTORY!" yang ditampilkan dengan ukuran besar dan warna hijau mencolok. Teks ini secara langsung menandakan bahwa pemain telah memenangkan permainan. Warna hijau tersebut diatur melalui konstanta

COLOR_GREEN yang didefinisikan dalam berkas config.py, dan ditampilkan melalui pemanggilan metode _draw_ui ketika variabel self.victory bernilai True. Sebagaimana halnya layar Game Over, tampilan ini juga dilengkapi dengan lapisan semi-transparan berwarna gelap yang menutupi latar belakang area permainan aktif. Tujuannya adalah mengarahkan fokus pemain pada informasi utama berupa pesan kemenangan dan data akhir permainan.

- Informasi Skor Akhir dan Opsi Permainan Selanjutnya:

Di bawah teks kemenangan, ditampilkan informasi "Final Score: 180" dengan warna putih. Angka tersebut menunjukkan akumulasi skor total yang dicapai pemain sepanjang sesi permainan, termasuk saat menghadapi bos. Pada bagian bawah layar, terdapat arahan kepada pemain berupa teks "Tekan 'R' untuk Mulai Lagi", yang juga ditampilkan dalam warna putih. Petunjuk ini memberi informasi kepada pemain tentang cara memulai ulang permainan setelah mencapai kemenangan.

- Indikator Logis dan Visual atas Kemenangan:

Berbeda dari tampilan pada saat pertarungan melawan bos, karakter bos (musuh.png) dan bar kesehatannya tidak lagi muncul di layar. Ketiadaan elemen-elemen ini memberikan petunjuk visual bahwa bos telah dikalahkan. Dari sisi logika permainan, kondisi ini terjadi karena saat nilai self.boss_health mencapai nol atau kurang, metode _update_player_projectiles akan mengatur self.victory dan self.game_over menjadi True. Selain itu, penggambaran karakter bos hanya dilakukan jika self.boss_health > 0, sehingga ketika bos kalah, ia tidak akan ditampilkan lagi. Sementara itu, gambar pesawat pemain (pesawat.png) masih tetap berada di area hidung pengguna sebagai bagian dari antarmuka yang konsisten.

4 Kesimpulan

Proyek King Astro berhasil diimplementasikan sebagai permainan interaktif yang memanfaatkan teknologi face detection dari MediaPipe sebagai mekanisme input utama. Pemain dapat mengendalikan pesawat menggunakan gerakan kepala serta menembakkan peluru melalui aksi membuka mulut, menghadirkan pengalaman bermain yang imersif dan inovatif. Pengembangan aplikasi ini mencakup berbagai komponen penting dalam desain dan implementasi permainan, mulai dari logika dasar pergerakan objek dan sistem deteksi tabrakan, hingga fitur kompleks seperti pertarungan melawan bos yang dilengkapi dengan sistem nyawa, pola serangan tertentu, dan mekanisme kekebalan. Integrasi elemen antarmuka pengguna seperti tampilan skor, bilah nyawa, instruksi bermain, serta latar musik turut berkontribusi dalam memperkaya kualitas pengalaman bermain. Dari sisi pengelolaan kode, refaktorisasi struktur program ke dalam bentuk multi-file (terdiri atas main.py, game_manager.py, dan config.py) serta pemanfaatan folder assets secara signifikan meningkatkan modularitas, keterbacaan, dan kemudahan pemeliharaan sistem. Penambahan dokumentasi internal dalam bentuk docstring pada setiap fungsi dan kelas turut mempermudah proses pemahaman dan pengembangan lebih lanjut. Secara keseluruhan, proyek ini tidak hanya berhasil memenuhi fungsi utamanya sebagai permainan berbasis deteksi wajah, tetapi juga dapat diposisikan sebagai studi kasus penerapan computer vision dalam konteks interaksi manusia dan komputer di bidang hiburan digital. Ke depannya, terdapat ruang untuk pengembangan lebih lanjut, seperti penambahan variasi musuh, fitur power-up, maupun tingkat kesulitan yang adaptif. Namun, fondasi yang telah dibangun dalam versi ini sudah cukup kokoh dan menunjukkan keberhasilan dalam mengintegrasikan teknologi deteksi wajah ke dalam sebuah aplikasi permainan secara efektif.

5 Lampiran

5.1 Code Program

Berikut merupakan Link Code Program dan Asset : ([Github Code Program](#)).

5.2 Link Video Demonstrasi Permainan

Berikut merupakan Link Video Demonstrasi pada proyek ini : ([Video Demonstrasi Penggunaan Filter](#)).

References

- [1] . Martin Clinton Tosima Manullang, “Multimedia technology module (itera),” 5 2025, accessed: 10-05-2025.
- [2] W. Shariden, “Video tiktok sebagai gambaran pembuatan proyek,” <https://vt.tiktok.com/ZSrvPcXbe/>, 5 2025, accessed: 28-05-2025.
- [3] GEMINI-PRO, “menggunakan bantuan ai gemini pro dalam membantu pembuatan code program,” https://drive.google.com/drive/folders/1ZsqDDU4LoLa0oFU5NvyjSRGgqkzMF6rW?usp=drive_link, 5 2025, accessed: 28-05-2025.
- [4] CHATGPT, “Kumpulan dokumentasi referensi menggunakan chat gpt dalam membantu pembuatan code program,” <https://drive.google.com/drive/folders/1u27Fr5ARIuSwv6j-0SZWkOmvlAYCLkMC?usp=sharing>, 5 2025, accessed: 28-05-2025.