



Nama:

1. Muhammad Fauzan As Shabierin (122140074)
2. Desty Ananta Purba (122140076)
3. Ramon Riping (122140078)

Tugas : **Final Project**

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

Tanggal: 30 Mei 2025

1 Deskripsi Projek

Pengguna akan diberikan tiga soal matematika dasar secara acak penjumlahan, pengurangan, perkalian, atau pembagian yang ditampilkan satu per satu. Jawaban diberikan dengan menunjukkan jumlah jari tangan sesuai hasil dari soal yang muncul. Sistem akan mendeteksi gestur tangan dan audio akan mengatakan angka yang ada pada gestur tangan dan mencocokkannya dengan jawaban yang benar. Jika sesuai, akan terdengar suara “Anda Benar!”, dan jika salah, sistem akan memberi tahu “Anda Salah. Pengalaman ini dirancang untuk menggabungkan unsur belajar dan bermain secara interaktif dan menyenangkan, cocok untuk semua kalangan terutama anak-anak.e

2 Teknologi

Teknologi yang digunakan dalam mengerjakan Final Project ini, yaitu sebagai berikut:

- Bahasa Pemrograman Python

Digunakan sebagai bahasa utama dalam pembuatan aplikasi. Python dipilih karena kemudahannya dalam pengembangan aplikasi multimedia dan integrasi dengan pustaka pihak ketiga.

- MediaPipe

Digunakan untuk mendeteksi posisi dan jumlah jari tangan pengguna secara real-time. Digunakan modul Hand Tracking dari MediaPipe yang dapat mengidentifikasi landmark jari.

- OpenCV (CV2)

Digunakan untuk menangkap video langsung dari webcam dan mengirimkannya ke sistem MediaPipe untuk dianalisis.

- Numpy

Digunakan sebagai pustaka pendukung untuk operasi numerik dan pengolahan data array. Dalam proyek ini, NumPy membantu dalam pengolahan koordinat landmark tangan, perhitungan logika, dan efisiensi manipulasi data.

- Pygame

Pygame digunakan untuk membuat antarmuka grafis interaktif seperti tampilan soal matematika, teks jawaban, dan umpan balik audio/visual. Library ini cocok untuk aplikasi edukatif dan permainan sederhana.

3 Cara Kerja Sistem

Cara kerja pada proyek Pong With Hand Tracking adalah sebagai berikut:

1. **Soal Acak Ditampilkan**

Sistem akan menampilkan tiga soal matematika dasar secara acak satu per satu di layar.

2. Pengguna Menjawab dengan Tangan

Pengguna menjawab dengan menunjukkan jumlah jari tangan yang sesuai dengan jawaban soal.

3. Deteksi Jumlah Jari

Kamera webcam menangkap gerakan tangan, MediaPipe menghitung jumlah jari yang terbuka.

4. Pencocokan Jawaban

Sistem memverifikasi apakah jumlah jari yang ditampilkan sesuai dengan jawaban benar dari soal.

5. Umpan Balik Suara dan Visual

Sistem akan memberikan audio dan teks “Anda Benar!” jika jawaban benar, dan “Anda Salah.” jika salah. Soal kemudian dilanjutkan hingga tiga soal selesai.

4 Penjelasan Kode Program

Kode program pada aplikasi Interactive Maths terdiri dari empat bagian utama:

- Pusat Integrasi.
- Deteksi tangan dan penghitungan jari
- Logika soal matematika
- Antarmuka interaktif (grafis dan audio)

Kode program dibagi menjadi 7 bagian kode yaitu pada `math_problem_generator.py`, `game_state.py`, `hand_detector.py`, `face_detector.py`, `audio_manager.py`, `ui_renderer.py`, dan `main.py`. Berikut ini merupakan penjelasan dari masing-masing bagian code.

4.1 Pusat Integrasi

File `main.py` adalah file utama yang mengatur jalannya aplikasi dari awal hingga selesai. Termasuk:

- Membuka webcam
- Menangkap gestur tangan
- Menampilkan soal
- Menilai jawaban
- Memberi umpan balik suara dan visual

Berikut ini adalah implementasi :

```
1 import cv2
2 import os
3 import game_state
4 import hand_detector
5 import face_detector
6 import audio_manager
7 import ui_renderer
8
9 def main():
10     """
11     Fungsi utama untuk menjalankan permainan matematika interaktif.
12     """
```

```

13  Menginisialisasi aset, webcam, dan mengelola alur permainan.
14  """
15  # Memuat aset dari folder
16  asset_dir = os.path.join(os.getcwd(), 'project', 'asset')
17  logo_path = os.path.join(asset_dir, 'logo.png')
18  button_path = os.path.join(asset_dir, 'buttonstart.png')
19  tryagain_path = os.path.join(asset_dir, 'buttontryagain.png')
20  bgcard_path = os.path.join(asset_dir, 'backgroundcard.png')
21  bgm_path = os.path.join(asset_dir, 'sound_matematika.wav')
22  win_sound_path = os.path.join(asset_dir, 'sound_kematian.wav')
23  lose_sound_path = os.path.join(asset_dir, 'sound_kematian_kalah.wav')
24
25  # Memuat gambar
26  logo = cv2.imread(logo_path, cv2.IMREAD_UNCHANGED)
27  button = cv2.imread(button_path, cv2.IMREAD_UNCHANGED)
28  tryagain = cv2.imread(tryagain_path, cv2.IMREAD_UNCHANGED)
29  bgcard = cv2.imread(bgcard_path, cv2.IMREAD_UNCHANGED)
30
31  # Memeriksa keberhasilan memuat gambar
32  if any(img is None for img in [logo, button, tryagain, bgcard]):
33      print("Error: Gagal memuat file gambar!")
34      print(f"Logo path: {logo_path}")
35      print(f"Button path: {button_path}")
36      print(f"Try Again path: {tryagain_path}")
37      print(f"Background Card path: {bgcard_path}")
38      return
39
40  # Mengubah ukuran gambar
41  logo = cv2.resize(logo, (200, 200))
42  button = cv2.resize(button, (200, 60))
43  tryagain = cv2.resize(tryagain, (250, 180))
44  bgcard = cv2.resize(bgcard, (300, 150))
45
46  # Inisialisasi komponen permainan
47  game_state_instance = game_state.GameState()
48  hand_detector_instance = hand_detector.HandDetector()
49  face_detector_instance = face_detector.FaceDetector()
50  audio_manager_instance = audio_manager.AudioManager(bgm_path, win_sound_path, lose_sound_path)
51  ui_renderer_instance = ui_renderer.UIRenderer(logo, button, tryagain, bgcard)
52
53  # Inisialisasi webcam
54  cap = cv2.VideoCapture(0)
55  is_game_started = False
56  answer_display_time = 0
57  answer_feedback = ""
58  sound_delay = 30 # ~1 detik pada 30fps
59
60  try:
61      # Memulai loop utama permainan
62      while cap.isOpened():
63          success, frame = cap.read()
64          if not success:
65              print("Gagal mengambil frame dari webcam")
66              break
67
68          # Membalik frame secara horizontal
69          frame = cv2.flip(frame, 1)
70          height, width = frame.shape[:2]
71          rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
72
73          # Menangani menu utama
74          if not is_game_started:

```

```

75         button_x, button_y = ui_renderer_instance.draw_main_menu(frame, width, height)
76         results = hand_detector_instance.hands.process(rgb_frame)
77
78         if results.multi_hand_landmarks:
79             for hand_landmarks in results.multi_hand_landmarks:
80                 # Menggambar landmark tangan dengan garis putih di menu
81                 hand_detector_instance.mp_drawing.draw_landmarks(
82                     frame,
83                     hand_landmarks,
84                     hand_detector_instance.mp_hands.HAND_CONNECTIONS,
85                     hand_detector_instance.mp_drawing.DrawingSpec(color=(0, 0, 255),
thickness=2, circle_radius=4),
86                     hand_detector_instance.mp_drawing.DrawingSpec(color=(255, 255, 255),
thickness=2)
87                 )
88                 # Memeriksa interaksi dengan start_button
89                 if hand_detector_instance.check_button_interaction(hand_landmarks, (
button_x, button_y),
90                                                                                     (200, 60), width, height):
91                     is_game_started = True
92                     audio_manager_instance.play_background_music()
93
94             else:
95                 # Memproses tangan dan wajah
96                 left_count, right_count, total_fingers = hand_detector_instance.process_hands(frame
, rgb_frame)
97                 face_pos = face_detector_instance.get_face_position(rgb_frame, width, height)
98
99                 # Menangani layar akhir permainan
100                 if game_state_instance.problems_count >= game_state_instance.total_problems:
101                     if not game_state_instance.game_over_played:
102                         audio_manager_instance.play_game_over_sound(game_state_instance.score)
103                         game_state_instance.game_over_played = True
104
105                     # Menggambar layar akhir
106                     tryagain_x, tryagain_y = ui_renderer_instance.draw_game_over(frame,
game_state_instance, width, height)
107                     results = hand_detector_instance.hands.process(rgb_frame)
108
109                     if results.multi_hand_landmarks:
110                         for hand_landmarks in results.multi_hand_landmarks:
111                             hand_detector_instance.mp_drawing.draw_landmarks(
112                                 frame,
113                                 hand_landmarks,
114                                 hand_detector_instance.mp_hands.HAND_CONNECTIONS,
115                                 hand_detector_instance.mp_drawing_styles.
get_default_hand_landmarks_style(),
116                                 hand_detector_instance.mp_drawing_styles.
get_default_hand_connections_style()
117                             )
118                             # Memeriksa interaksi dengan tryagain_button
119                             if hand_detector_instance.check_button_interaction(hand_landmarks, (
tryagain_x, tryagain_y),
120                                                                                     (250, 180), width, height):
121
122                                 # Mengatur ulang permainan
123                                 game_state_instance = game_state.GameState()
124                                 is_game_started = True
125                                 answer_display_time = 0
126                                 answer_feedback = ""
127                                 audio_manager_instance.stop_all_sounds()
128                                 audio_manager_instance.play_background_music()
129                                 sound_delay = 30

```

```

129         continue
130
131     else:
132         # Memperbarui jawaban
133         if total_fingers is not None:
134             game_state_instance.current_answer_given = total_fingers
135
136         # Membuat soal baru jika diperlukan
137         if game_state_instance.current_problem is None:
138             game_state_instance.new_problem()
139             answer_display_time = 0
140             answer_feedback = ""
141
142         # Memperbarui status permainan
143         if game_state_instance.current_problem:
144             if not game_state_instance.update_timer():
145                 if not game_state_instance.show_feedback:
146                     # Memeriksa jawaban
147                     final_answer = game_state_instance.current_answer_given if
game_state_instance.current_answer_given is not None else 0
148                     correct = game_state_instance.check_answer(final_answer)
149                     game_state_instance.answer_history.append(correct)
150                     answer_feedback = f"Benar! +20" if correct else f"Salah! +0"
151                     audio_manager_instance.speak("Your answer is right!" if correct
else
152                         f"Your answer is wrong! The right answer is {int(
game_state_instance.current_answer)})")
153                     game_state_instance.show_feedback = True
154                     answer_display_time = game_state_instance.feedback_delay
155                     elif sound_delay > 0:
156                         sound_delay -= 1
157                     elif answer_display_time > 0:
158                         answer_display_time -= 1
159                     else:
160                         # Pindah ke soal berikutnya
161                         game_state_instance.problems_count += 1
162                         game_state_instance.current_problem = None
163                         game_state_instance.show_feedback = False
164                         sound_delay = 30
165
166         # Menggambar UI permainan
167         ui_renderer_instance.draw_game_ui(
168             frame, game_state_instance, width, height, face_pos, total_fingers,
169             answer_feedback, answer_display_time
170         )
171
172         # Menampilkan frame
173         cv2.imshow('Kamera', frame)
174         if cv2.waitKey(1) & 0xFF == ord('q'):
175             break
176
177     finally:
178         # Membersihkan sumber daya
179         cap.release()
180         hand_detector_instance.close()
181         face_detector_instance.close()
182         audio_manager_instance.quit()
183         cv2.destroyAllWindows()
184
185 if __name__ == "__main__":
186     main()

```

Kode 1: **main.py****4.1.1 Fungsi Utama****a. generate_question()**

- Mengacak satu soal matematika dasar dari operasi penjumlahan, pengurangan, perkalian, atau pembagian
- Mengembalikan soal dalam bentuk string dan hasilnya (integer).

b. count_fingers(hand_landmarks)

- Menggunakan MediaPipe untuk membaca posisi jari.
- Menghitung jumlah jari terbuka dari landmark tangan.
- Mengembalikan angka hasil gestur.

c. draw_interface()

Menggunakan Pygame untuk:

- Menampilkan soal di layar.
- Menampilkan hasil dari gestur jari yang terbaca.
- Menampilkan teks umpan balik (benar/salah) secara visual.

d. play_sound(status)

Memutar audio “Anda Benar” atau “Anda Salah” tergantung status jawaban.

4.1.2 Alur Program**Inisialisasi:**

- Menyiapkan kamera dan antarmuka Pygame.
- Mengatur warna, font, ukuran layar, dan variabel kontrol.

Loop Game (5 soal):

- Menampilkan soal.
- Menunggu input gestur dari pengguna.
- Mengonversi gestur ke angka.
- Mengecek apakah jawaban benar.
- Memberikan umpan balik suara dan visual.

Akhir permainan:

- Menampilkan total nilai.
- Menutup kamera dan Pygame.

4.2 Deteksi Tangan dan Wajah (Input Gerakan dan Validasi Pengguna)

File yang bertugas menangani input dari pengguna berupa gestur

4.2.1 hand_detector.py

Mengidentifikasi jumlah jari yang ditunjukkan pengguna sebagai input jawaban. MediaPipe Hands digunakan untuk mengenali landmark jari dari gambar webcam.

- Menginisialisasi detektor tangan.
- Mengambil frame kamera dan memprosesnya.

- Menghitung berapa jari yang terbuka berdasarkan posisi landmark.
- Mengembalikan jumlah jari sebagai angka (0–10).

Berikut ini adalah implementasi :

```

1
2 import cv2
3 import mediapipe as mp
4 import numpy as np
5 from typing import Tuple
6
7 class HandDetector:
8     """
9     Kelas untuk mendeteksi tangan dan menghitung jari yang diangkat menggunakan MediaPipe.
10
11     Attributes:
12         JEMPOL (List[int]): Indeks landmark untuk jempol.
13         TELUNJUK (List[int]): Indeks landmark untuk telunjuk.
14         JARI_TENGAH (List[int]): Indeks landmark untuk jari tengah.
15         JARI_MANIS (List[int]): Indeks landmark untuk jari manis.
16         JARI_KELINGKING (List[int]): Indeks landmark untuk kelingking.
17         hands (mp.solutions.hands.Hands): Objek MediaPipe untuk deteksi tangan.
18         mp_drawing (mp.solutions.drawing_utils): Utilitas untuk menggambar landmark.
19         mp_drawing_styles (mp.solutions.drawing_styles): Gaya visual untuk landmark.
20     """
21
22     JEMPOL = [1, 2, 3, 4]
23     TELUNJUK = [5, 6, 7, 8]
24     JARI_TENGAH = [9, 10, 11, 12]
25     JARI_MANIS = [13, 14, 15, 16]
26     JARI_KELINGKING = [17, 18, 19, 20]
27
28     def __init__(self):
29         """Inisialisasi detektor tangan dengan pengaturan MediaPipe."""
30         self.mp_hands = mp.solutions.hands
31         self.hands = self.mp_hands.Hands(max_num_hands=2, min_detection_confidence=0.8)
32         self.mp_drawing = mp.solutions.drawing_utils
33         self.mp_drawing_styles = mp.solutions.drawing_styles
34
35     def detect_finger_number(self, hand_landmarks, handedness: str) -> int:
36         """
37         Menghitung jumlah jari yang diangkat pada satu tangan.
38
39         Args:
40             hand_landmarks: Landmark tangan dari MediaPipe.
41             handedness (str): Sisi tangan ('Left' atau 'Right').
42
43         Returns:
44             int: Jumlah jari yang diangkat.
45         """
46         fingers = []
47
48         # Mendeteksi jempol
49         thumb_tip = hand_landmarks.landmark[self.JEMPOL[-1]]
50         thumb_ip = hand_landmarks.landmark[self.JEMPOL[2]]
51         fingers.append(thumb_tip.x < thumb_ip.x if handedness == 'Right' else thumb_tip.x >
52                        thumb_ip.x)
53
54         # Mendeteksi jari lainnya
55         for finger in [self.TELUNJUK, self.JARI_TENGAH, self.JARI_MANIS, self.JARI_KELINGKING]:
56             tip = hand_landmarks.landmark[finger[-1]]
57             pip = hand_landmarks.landmark[finger[-2]]
58             dip = hand_landmarks.landmark[finger[-3]]

```

```

58         fingers.append(tip.y < pip.y and tip.y < dip.y)
59
60     return sum(fingers)
61
62 def combine_hand_numbers(self, left_count: int, right_count: int) -> int:
63     """
64     Menggabungkan jumlah jari dari kedua tangan untuk menghasilkan angka 0-10.
65
66     Args:
67         left_count (int): Jumlah jari tangan kiri.
68         right_count (int): Jumlah jari tangan kanan.
69
70     Returns:
71         int: Total jari yang diangkat (maksimum 10).
72     """
73     if left_count == 0:
74         return right_count
75     if right_count == 0:
76         return left_count
77     if left_count == 5:
78         return min(10, 5 + right_count)
79     return min(10, left_count + right_count)
80
81 def process_hands(self, frame: np.ndarray, rgb_frame: np.ndarray) -> Tuple[int, int, int]:
82     """
83     Memproses landmark tangan dan mengembalikan jumlah jari yang diangkat.
84
85     Args:
86         frame (np.ndarray): Bingkai gambar untuk menggambar landmark.
87         rgb_frame (np.ndarray): Bingkai dalam format RGB untuk pemrosesan.
88
89     Returns:
90         Tuple[int, int, int]: Jumlah jari kiri, kanan, dan total.
91     """
92     left_count, right_count = 0, 0
93     total_fingers = 0
94     results = self.hands.process(rgb_frame)
95
96     if results.multi_hand_landmarks:
97         for idx, hand_landmarks in enumerate(results.multi_hand_landmarks):
98             handedness = results.multi_handedness[idx].classification[0].label
99             count = self.detect_finger_number(hand_landmarks, handedness)
100
101             # Menggambar landmark tangan dengan gaya berwarna
102             self.mp_drawing.draw_landmarks(
103                 frame,
104                 hand_landmarks,
105                 self.mp_hands.HAND_CONNECTIONS,
106                 self.mp_drawing_styles.get_default_hand_landmarks_style(),
107                 self.mp_drawing_styles.get_default_hand_connections_style()
108             )
109
110             # Memperbarui jumlah jari
111             if handedness == 'Left':
112                 left_count = count
113             else:
114                 right_count = count
115
116             # Menampilkan jumlah jari per tangan
117             cv2.putText(frame, f"{handedness}: {count}",
118                         (10, 50 + idx * 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
119                         (255, 255, 255), 2)

```



```

120         total_fingers = self.combine_hand_numbers(left_count, right_count)
121
122     return left_count, right_count, total_fingers
123
124 def check_button_interaction(self, hand_landmarks, button_pos: Tuple[int, int],
125                             button_size: Tuple[int, int], width: int, height: int) -> bool:
126     """
127     Memeriksa apakah tangan berinteraksi dengan tombol.
128
129     Args:
130         hand_landmarks: Landmark tangan dari MediaPipe.
131         button_pos (Tuple[int, int]): Posisi tombol (x, y).
132         button_size (Tuple[int, int]): Ukuran tombol (lebar, tinggi).
133         width (int): Lebar bingkai.
134         height (int): Tinggi bingkai.
135
136     Returns:
137         bool: True jika tangan berada di area tombol, False jika tidak.
138     """
139     if hand_landmarks:
140         index_finger = hand_landmarks.landmark[8]
141         x, y = int(index_finger.x * width), int(index_finger.y * height)
142         button_x, button_y = button_pos
143         button_w, button_h = button_size
144         return button_x < x < button_x + button_w and button_y < y < button_y + button_h
145     return False
146
147 def close(self):
148     """Menutup sumber daya MediaPipe untuk tangan."""
149     self.hands.close()
150
151

```

Kode 2: hand_detector

4.2.2 face_detector.py

Validasi kehadiran pengguna sebelum atau selama permainan berlangsung.

-Mengaktifkan kamera dan mendeteksi apakah ada wajah dalam frame.

-Menyediakan flag boolean yang bisa digunakan sebagai syarat "mulai permainan".

Berikut ini adalah implementasi :

```

1  import mediapipe as mp
2  import numpy as np
3  from typing import Optional, Tuple
4
5
6  class FaceDetector:
7      """
8      Kelas untuk mendeteksi wajah menggunakan MediaPipe untuk penempatan elemen UI.
9
10     Attributes:
11         face_detection (mp.solutions.face_detection.FaceDetection): Objek MediaPipe untuk deteksi
12         wajah.
13     """
14
15     def __init__(self):
16         """Inisialisasi detektor wajah dengan pengaturan MediaPipe."""
17         self.mp_face = mp.solutions.face_detection
18         self.face_detection = self.mp_face.FaceDetection(min_detection_confidence=0.5)

```

```

19 def get_face_position(self, rgb_frame: np.ndarray, width: int, height: int) -> Optional[Tuple[
20     int, int, int, int]]:
21     """
22     Mendeteksi wajah dan mengembalikan kotak pembatasnya dalam piksel.
23
24     Args:
25         rgb_frame (np.ndarray): Bingkai dalam format RGB.
26         width (int): Lebar bingkai.
27         height (int): Tinggi bingkai.
28
29     Returns:
30         Optional[Tuple[int, int, int, int]]: Kotak pembatas wajah (x, y, lebar, tinggi), None
31         jika tidak terdeteksi.
32     """
33     results = self.face_detection.process(rgb_frame)
34     if results.detections:
35         detection = results.detections[0]
36         bbox = detection.location_data.relative_bounding_box
37         return (
38             int(bbox.xmin * width),
39             int(bbox.ymin * height),
40             int(bbox.width * width),
41             int(bbox.height * height)
42         )
43     return None
44
45 def close(self):
46     """Menutup sumber daya MediaPipe untuk deteksi wajah."""
47     self.face_detection.close()

```

Kode 3: face_detector.py

4.3 Logika Soal Matematika dan Status Permainan

File yang berhubungan dengan pembuatan soal, pengecekan jawaban, dan status permainan

4.3.1 math_problem_generator.py

Membuat soal matematika dasar secara dinamis dan acak.

-Fungsi generate_problem() akan memilih secara acak satu operasi dari: +, -, *, /.

-Dua angka diacak sesuai kesulitan dan divalidasi (misal: hasil pembagian bulat).

Berikut ini adalah implementasi :

```

1 import random
2 from typing import Tuple
3
4 class MathProblemGenerator:
5     """Kelas untuk menghasilkan soal matematika acak dengan operator (+, -, *, /) dan hasil antara
6     0-10."""
7
8     @staticmethod
9     def generate_math_problem() -> Tuple[str, float]:
10         """
11         Menghasilkan soal matematika acak beserta jawabannya.
12
13         Returns:
14             Tuple[str, float]: Soal dalam bentuk string (misalnya, "2 + 3") dan jawaban dalam float
15             .
16         """
17         # Memilih operator secara acak

```

```

16 operators = ['+', '-', '*', '/']
17 operator = random.choice(operators)
18
19 # Membuat soal berdasarkan operator
20 if operator == '+':
21     result = random.randint(0, 10)
22     num2 = random.randint(0, result)
23     num1 = result - num2
24 elif operator == '-':
25     num1 = random.randint(0, 10)
26     num2 = random.randint(0, num1)
27 elif operator == '*':
28     result = random.randint(0, 10)
29     factors = [i for i in range(1, 11) if result % i == 0]
30     num1 = random.choice(factors)
31     num2 = result // num1
32 else: # Pembagian
33     num2 = random.randint(1, 10)
34     num1 = random.randint(0, 10) * num2
35
36 # Membentuk string soal dan menghitung jawaban
37 problem = f"{num1} {operator} {num2}"
38 answer = eval(problem)
39 return problem, float(answer)
40

```

Kode 4: `math_problem_generator.py`

4.3.2 `game_state.py`

Menyimpan status dan skor permainan selama sesi berlangsung.

- Menyimpan jumlah soal yang ditampilkan.
- Menghitung jumlah jawaban benar dan salah.
- Menyediakan metode untuk:
- Reset permainan.

Validasi akhir (misalnya: jika jawaban lebih dari 3 benar → lolos).

Berikut ini adalah implementasi :

```

1 import cv2
2 from typing import Optional, List
3 import math_problem_generator
4
5 class GameState:
6     """
7     Kelas untuk mengelola status permainan, seperti score, timer, dan kemajuan soal.
8
9     Attributes:
10         current_problem (Optional[str]): Soal matematika saat ini.
11         current_answer (Optional[float]): Jawaban benar untuk soal saat ini.
12         problems_count (int): Jumlah soal yang telah dikerjakan.
13         correct_answers (int): Jumlah jawaban yang benar.
14         total_problems (int): Total soal dalam satu permainan.
15         score (int): Skor pemain (20 poin per jawaban benar).
16         time_per_question (int): Waktu maksimum per soal (detik).
17         timer (float): Waktu tersisa untuk soal saat ini.
18         last_time (int): Waktu terakhir diukur (tick).
19         current_answer_given (Optional[int]): Jawaban yang diberikan pemain.
20         feedback_delay (int): Durasi tampilan umpan balik (frame).
21         show_feedback (bool): Status apakah umpan balik ditampilkan.

```

```

22     game_over_played (bool): Status apakah suara akhir sudah dimainkan.
23     answer_history (List[bool]): Daftar status jawaban (benar/salah).
24     """
25
26     def __init__(self, total_problems: int = 5, time_per_question: int = 7):
27         """Inisialisasi status permainan dengan nilai awal."""
28         self.current_problem: Optional[str] = None
29         self.current_answer: Optional[float] = None
30         self.problems_count: int = 0
31         self.correct_answers: int = 0
32         self.total_problems: int = total_problems
33         self.score: int = 0
34         self.time_per_question: int = time_per_question
35         self.timer: float = 0
36         self.last_time: int = 0
37         self.current_answer_given: Optional[int] = None
38         self.feedback_delay: int = 45 # ~1.5 detik pada 30fps
39         self.show_feedback: bool = False
40         self.game_over_played: bool = False
41         self.answer_history: List[bool] = []
42
43     def new_problem(self) -> bool:
44         """
45         Membuat soal matematika baru jika permainan belum selesai.
46
47         Returns:
48             bool: True jika soal baru dibuat, False jika permainan selesai.
49         """
50         if self.problems_count < self.total_problems:
51             # Membuat soal baru dan mengatur ulang timer
52             self.current_problem, self.current_answer = math_problem_generator.MathProblemGenerator
.generate_math_problem()
53             self.timer = self.time_per_question
54             self.last_time = cv2.getTickCount()
55             self.current_answer_given = None
56             return True
57         return False
58
59     def update_timer(self) -> bool:
60         """
61         Memperbarui timer dan memeriksa apakah waktu masih tersedia.
62
63         Returns:
64             bool: True jika waktu masih ada, False jika habis.
65         """
66         if self.timer > 0:
67             # Menghitung waktu yang telah berlalu
68             current_time = cv2.getTickCount()
69             elapsed = (current_time - self.last_time) / cv2.getTickFrequency()
70             self.timer = max(0, self.time_per_question - elapsed)
71             return self.timer > 0
72         return False
73
74     def check_answer(self, user_answer: int) -> bool:
75         """
76         Memeriksa apakah jawaban pemain benar dan memperbarui score.
77
78         Args:
79             user_answer (int): Jawaban yang diberikan pemain.
80
81         Returns:
82             bool: True jika jawaban benar, False jika salah.

```

```

83     """
84     if self.current_answer is not None:
85         # Membandingkan jawaban dengan toleransi untuk hasil float
86         if abs(user_answer - self.current_answer) < 0.01:
87             self.score += 20
88             self.correct_answers += 1
89             return True
90     return False
91

```

Kode 5: game_state.py

4.4 Antarmuka Interaktif (Visual dan Audio)

File yang bertanggung jawab untuk tampilan layar dan suara

4.4.1 ui_renderer.py

Menampilkan elemen grafis secara real-time di layar permainan.

- Menampilkan soal matematika secara menarik dan besar.
- Menampilkan angka dari hasil gestur jari yang terdeteksi.
- Memberikan feedback visual:

Warna merah/hijau untuk benar/salah.

Teks: "Anda Benar!" atau "Anda Salah, jawabannya: X".

- Pengaturan font, warna, layout layar agar ramah anak.

Berikut ini adalah implementasi :

```

1  import cv2
2  import numpy as np
3  from typing import Optional, Tuple, List
4  import game_state
5
6  class UIRenderer:
7      """
8      Kelas untuk merender elemen antarmuka pengguna seperti tombol, teks, dan indikator kemajuan.
9
10     Attributes:
11         logo (np.ndarray): Gambar logo.
12         start_button (np.ndarray): Gambar tombol mulai.
13         tryagain_button (np.ndarray): Gambar tombol coba lagi.
14         bgcard (np.ndarray): Gambar latar belakang untuk soal.
15         font (int): Jenis font untuk teks.
16     """
17
18     def __init__(self, logo: np.ndarray, start_button: np.ndarray, tryagain_button: np.ndarray,
19                 bgcard: np.ndarray):
20         """Inisialisasi perender UI dengan gambar aset."""
21         self.logo = logo
22         self.start_button = start_button
23         self.tryagain_button = tryagain_button
24         self.bgcard = bgcard
25         self.font = cv2.FONT_HERSHEY_SIMPLEX
26
27     @staticmethod
28     def overlay_image(background: np.ndarray, overlay: np.ndarray, position: Tuple[int, int]) -> np.ndarray:
29         """
30         Menumpuk gambar di atas latar belakang dengan blending alpha.
31
32         Args:
33

```

```

32         background (np.ndarray): Gambar latar belakang.
33         overlay (np.ndarray): Gambar yang akan ditumpuk.
34         position (Tuple[int, int]): Posisi tumpukan (y, x).
35
36     Returns:
37         np.ndarray: Latar belakang dengan gambar yang ditumpuk.
38     """
39     h, w = overlay.shape[:2]
40     y, x = position
41
42     if overlay.shape[2] == 4: # Menangani kanal alpha
43         overlay_rgb = overlay[:, :, :3]
44         alpha = overlay[:, :, 3] / 255.0
45         alpha_3d = np.stack([alpha, alpha, alpha], axis=2)
46         roi = background[y:y+h, x:x+w]
47         result = (overlay_rgb * alpha_3d + roi * (1 - alpha_3d)).astype(np.uint8)
48         background[y:y+h, x:x+w] = result
49     else:
50         background[y:y+h, x:x+w] = overlay
51     return background
52
53 def draw_main_menu(self, frame: np.ndarray, width: int, height: int) -> Tuple[int, int]:
54     """
55     Menggambar menu utama dengan logo dan start_button.
56
57     Args:
58         frame (np.ndarray): Bingkai untuk menggambar.
59         width (int): Lebar bingkai.
60         height (int): Tinggi bingkai.
61
62     Returns:
63         Tuple[int, int]: Posisi start_button (x, y).
64     """
65     # Menempatkan logo di tengah atas
66     logo_x = (width - self.logo.shape[1]) // 2
67     logo_y = 50
68     button_x = (width - self.start_button.shape[1]) // 2
69     button_y = logo_y + self.logo.shape[0] + 20
70
71     frame = self.overlay_image(frame, self.logo, (logo_y, logo_x))
72     frame = self.overlay_image(frame, self.start_button, (button_y, button_x))
73     return button_x, button_y
74
75 def draw_game_ui(self, frame: np.ndarray, game_state: game_state.GameState, width: int, height:
76     int,
77     face_pos: Optional[Tuple[int, int, int, int]], total_fingers: int,
78     answer_feedback: str, answer_display_time: int):
79     """
80     Menggambar antarmuka permainan, termasuk kartu soal, timer, score, dan answer_feedback.
81
82     Args:
83         frame (np.ndarray): Bingkai untuk menggambar.
84         game_state (GameState): Status permainan saat ini.
85         width (int): Lebar bingkai.
86         height (int): Tinggi bingkai.
87         face_pos (Optional[Tuple[int, int, int, int]]): Posisi wajah (x, y, lebar, tinggi).
88         total_fingers (int): Jumlah jari yang diangkat.
89         answer_feedback (str): Teks umpan balik (misalnya, "Benar! +20").
90         answer_display_time (int): Sisa waktu tampilan umpan balik.
91     """
92     # Menggambar bgcard di atas wajah
93     if face_pos:

```

```

93     face_x, face_y, face_w, face_h = face_pos
94     bgcard_x = face_x - (self.bgcard.shape[1] - face_w) // 2
95     bgcard_y = max(0, face_y - self.bgcard.shape[0] - 20)
96     bgcard_x = max(0, min(bgcard_x, width - self.bgcard.shape[1]))
97     bgcard_y = max(0, min(bgcard_y, height - self.bgcard.shape[0]))
98     frame = self.overlay_image(frame, self.bgcard, (bgcard_y, bgcard_x))
99
100     # Menggambar soal matematika
101     if game_state.current_problem:
102         text = game_state.current_problem
103         text_width, text_height = cv2.getTextSize(text, self.font, 1.0, 2)[0]
104         text_x = bgcard_x + (self.bgcard.shape[1] - text_width) // 2
105         text_y = bgcard_y + (self.bgcard.shape[0] + text_height) // 2
106         cv2.putText(frame, text, (text_x, text_y), self.font, 1.0, (255, 255, 255), 3)
107         cv2.putText(frame, text, (text_x, text_y), self.font, 1.0, (0, 0, 0), 2)
108
109     # Menggambar timer dan score
110     timer_text = f"Waktu: {int(game_state.timer)}s"
111     cv2.putText(frame, timer_text, (width - 150, 30), self.font, 0.7,
112                 (0, 0, 255) if game_state.timer < 2 else (255, 255, 255), 2)
113     score_text = f"Score: {game_state.score}"
114     cv2.putText(frame, score_text, (width - 150, 60), self.font, 0.7, (255, 255, 255), 2)
115
116     # Menggambar jawaban saat ini
117     if total_fingers is not None:
118         cv2.putText(frame, f"Jawaban: {total_fingers}", (10, 110), self.font, 1, (255, 255,
119         255), 2)
120
121     # Menggambar indikator kemajuan soal
122     self.draw_question_progress(frame, game_state.total_problems, game_state.problems_count,
123                               game_state.answer_history, width)
124
125     # Menggambar answer_feedback terakhir
126     if answer_display_time > 0 and answer_feedback:
127         feedback_color = (0, 255, 0) if "Benar" in answer_feedback else (0, 0, 255)
128         feedback_pos = (width // 2 - 60, height // 2)
129         cv2.putText(frame, answer_feedback, feedback_pos, self.font, 1, (0, 0, 0), 3)
130         cv2.putText(frame, answer_feedback, feedback_pos, self.font, 1, feedback_color, 2)
131
132     @staticmethod
133     def draw_question_progress(frame: np.ndarray, total_questions: int, current_question: int,
134                              answers: List[bool], width: int):
135         """
136         Menggambar indikator kemajuan soal berupa lingkaran dan garis penghubung.
137
138         Args:
139             frame (np.ndarray): Bingkai untuk menggambar.
140             total_questions (int): Jumlah total soal.
141             current_question (int): Nomor soal saat ini.
142             answers (List[bool]): Daftar answer_history (benar/salah).
143             width (int): Lebar bingkai.
144
145         """
146         CIRCLE_RADIUS = 8
147         LINE_LENGTH = 30
148         SPACING = 45
149         Y_POSITION = 30
150
151         total_width = SPACING * (total_questions - 1)
152         start_x = (width - total_width) // 2
153
154         # Menggambar garis penghubung
155         for i in range(total_questions - 1):

```

```

154     x1 = start_x + (i * SPACING) + CIRCLE_RADIUS
155     x2 = x1 + LINE_LENGTH
156     line_color = (150, 150, 150) if i >= len(answers) else (0, 255, 0) if answers[i] else
(0, 0, 255)
157     cv2.line(frame, (x1, Y_POSITION), (x2, Y_POSITION), line_color, 2)
158
159     # Menggambar lingkaran untuk setiap soal
160     for i in range(total_questions):
161         center_x = start_x + (i * SPACING)
162         center = (center_x, Y_POSITION)
163         if i == current_question:
164             cv2.circle(frame, center, CIRCLE_RADIUS + 2, (0, 255, 255), -1)
165             cv2.circle(frame, center, CIRCLE_RADIUS + 2, (255, 255, 255), 2)
166         elif i < len(answers):
167             color = (0, 255, 0) if answers[i] else (0, 0, 255)
168             cv2.circle(frame, center, CIRCLE_RADIUS, color, -1)
169         else:
170             cv2.circle(frame, center, CIRCLE_RADIUS, (150, 150, 150), 2)
171
172     def draw_game_over(self, frame: np.ndarray, game_state: game_state.GameState, width: int,
height: int) -> Tuple[int, int]:
173         """
174         Menggambar layar akhir permainan dengan score akhir dan tryagain_button.
175
176         Args:
177             frame (np.ndarray): Bingkai untuk menggambar.
178             game_state (GameState): Status permainan.
179             width (int): Lebar bingkai.
180             height (int): Tinggi bingkai.
181
182         Returns:
183             Tuple[int, int]: Posisi tryagain_button (x, y).
184         """
185         # Membuat lapisan overlay untuk menggelapkan latar
186         overlay = frame.copy()
187         cv2.rectangle(overlay, (0, 0), (width, height), (0, 0, 0), -1)
188         frame[:] = cv2.addWeighted(overlay, 0.7, frame, 0.3, 0)
189
190         # Mengatur posisi teks
191         y_start = height // 4
192         y_spacing = 50 # Jarak antar teks untuk pemisahan yang lebih baik
193         messages = [
194             ("Permainan Selesai!", 2.0, (0, 255, 0)),
195             (f"Score Akhir: {game_state.score}/100", 1.8, (0, 255, 0))
196         ]
197
198         # Menggambar teks pesan
199         for i, (msg, scale, color) in enumerate(messages):
200             text_size = cv2.getTextSize(msg, self.font, scale, 3)[0]
201             text_x = (width - text_size[0]) // 2
202             text_y = y_start + i * y_spacing
203             cv2.putText(frame, msg, (text_x, text_y), self.font, scale, (0, 0, 0), 4)
204             cv2.putText(frame, msg, (text_x, text_y), self.font, scale, color, 2)
205
206         # Menggambar tryagain_button
207         tryagain_x = (width - self.tryagain_button.shape[1]) // 2
208         tryagain_y = y_start + (len(messages) * y_spacing) + 20
209         frame = self.overlay_image(frame, self.tryagain_button, (tryagain_y, tryagain_x))
210
211         # Menggambar teks keluar
212         quit_text = "Tekan 'Q' untuk keluar"
213         text_size = cv2.getTextSize(quit_text, self.font, 1.0, 2)[0]

```



```

214     text_x = (width - text_size[0]) // 2
215     text_y = tryagain_y + self.tryagain_button.shape[0] + 20
216     text_y = min(text_y, height - text_size[1] - 10)
217     cv2.putText(frame, quit_text, (text_x, text_y), self.font, 1.0, (0, 0, 0), 3)
218     cv2.putText(frame, quit_text, (text_x, text_y), self.font, 1.0, (255, 255, 255), 2)
219
220     return tryagain_x, tryagain_y
221

```

Kode 6: ui_render.py

4.4.2 audio_manager.py

Memberikan umpan balik suara untuk mendukung gaya belajar audio.

- Fungsi play_correct() → memutar suara “Anda Benar!”
- Fungsi play_wrong(jawaban_benar) → memutar suara “Anda Salah, jawabannya adalah...”
- Fungsi tambahan (jika ada): play_number(n) untuk menyebut angka hasil jari.

Berikut ini adalah implementasi :

```

1     import os
2     import pygame
3     import pyttts3
4     import threading
5
6     class AudioManager:
7         """
8         Kelas untuk mengelola pemutaran suara, termasuk musik latar dan efek suara.
9
10        Attributes:
11            bgm_path (str): Lokasi file musik latar.
12            win_sound (pygame.mixer.Sound): Efek suara untuk kemenangan.
13            lose_sound (pygame.mixer.Sound): Efek suara untuk kekalahan.
14        """
15
16        def __init__(self, bgm_path: str, win_sound_path: str, lose_sound_path: str):
17            """Inisialisasi pengelola suara dengan file audio."""
18            pygame.mixer.init()
19            self.bgm_path = bgm_path
20            self.win_sound = pygame.mixer.Sound(win_sound_path) if os.path.exists(win_sound_path) else
21            None
22            self.lose_sound = pygame.mixer.Sound(lose_sound_path) if os.path.exists(lose_sound_path)
23            else None
24            if self.win_sound:
25                self.win_sound.set_volume(0.5)
26            if self.lose_sound:
27                self.lose_sound.set_volume(0.5)
28
29        def play_background_music(self):
30            """Memutar musik latar secara berulang."""
31            try:
32                pygame.mixer.music.load(self.bgm_path)
33                pygame.mixer.music.set_volume(0.5)
34                pygame.mixer.music.play(-1)
35            except pygame.error as e:
36                print(f"Error memutar musik latar: {e}")
37
38        def stop_background_music(self):
39            """Menghentikan musik latar."""
40            try:
41                pygame.mixer.music.stop()
42            except pygame.error as e:
43

```

```

41         print(f"Error menghentikan musik latar: {e}")
42
43     def play_game_over_sound(self, score: int):
44         """
45         Memutar suara akhir berdasarkan score.
46
47         Args:
48             score (int): Skor akhir pemain.
49         """
50         try:
51             self.stop_background_music()
52             if score >= 60 and self.win_sound:
53                 self.win_sound.play()
54             elif self.lose_sound:
55                 self.lose_sound.play()
56         except pygame.error as e:
57             print(f"Error memutar suara akhir: {e}")
58
59     def stop_all_sounds(self):
60         """Menghentikan semua suara yang sedang diputar."""
61         if self.win_sound:
62             self.win_sound.stop()
63         if self.lose_sound:
64             self.lose_sound.stop()
65         self.stop_background_music()
66
67     @staticmethod
68     def speak(text: str):
69         """
70         Memutar teks sebagai suara menggunakan text-to-speech di thread terpisah.
71
72         Args:
73             text (str): Teks yang akan diucapkan.
74         """
75         def _speak():
76             engine = pyttsx3.init()
77             engine.setProperty('rate', 150)
78             engine.say(text)
79             engine.runAndWait()
80
81         threading.Thread(target=_speak).start()
82
83     def quit(self):
84         """Membersihkan sumber daya pygame mixer."""
85         pygame.mixer.quit()
86

```

Kode 7: audio_manager.py

5 Hasil Analisis

5.1 Program Utama

```

1  if __name__ == "__main__":
2      while True:
3          game.run()
4

```

Kode 8: Program Utama

- if `__name__ == "__main__"` digunakan untuk memastikan program hanya dijalankan saat file dipanggil langsung.
- while True: menciptakan loop tanpa henti agar permainan bisa diulang.
- `game.run()` merupakan pemanggilan fungsi utama yang mengelola alur permainan dalam satu sesi.

5.2 Pengolahan Gestur

Deteksi jari dilakukan melalui file `hand_detector.py` menggunakan MediaPipe Hands. Prosesnya:

1. Mengambil gambar dari webcam menggunakan `cv2.VideoCapture()`.

2. MediaPipe mengenali landmark (titik-titik tangan) dari citra.

3. Fungsi `count_fingers()` menghitung berapa jari terbuka berdasarkan logika posisi: Ibu jari → orientasi horizontal,

- Jari telunjuk hingga kelingking → orientasi vertikal.
- Output: angka dari 0–10 sesuai jumlah jari.

Contoh logika sederhana:

```
1 if hand_landmarks[finger_tip].y < hand_landmarks[finger_dip].y:
2     fingers.append(1) # jari terbuka
3
```

5.3 Evaluasi Jawaban dan Audio Feedback

Modul `math_problem_generator.py` akan menghasilkan soal, misalnya:

("5 + 2 = ?", 7)

Lalu program akan membandingkan hasil deteksi jari dengan jawaban:

```
1 if jawaban_pengguna == hasil_soal:
2     play_correct()
3 else:
4     play_wrong(hasil_soal)
5
```

- Fungsi `play_correct()` → memutar suara "Anda Benar!"
- Fungsi `play_wrong(jawaban)` → memutar suara "Anda Salah, jawabannya adalah ..."

Pemutaran suara dikelola oleh `pygame.mixer` di `audio_manager.py`.

5.4 Tampilan Visual dan Responsif

Antarmuka dibangun dengan `pygame`, diatur melalui file `ui_renderer.py`, menampilkan:

- Soal secara besar dan jelas di bagian atas layar.
- Hasil deteksi jari dalam bentuk angka dan/atau gambar.
- Feedback visual berupa warna (hijau: benar, merah: salah) dan teks.

```
1 text = font.render("Anda Benar!", True, (0, 255, 0))
2 screen.blit(text, (100, 300))
3
```

5.5 Manajemen Permainan

Semua status permainan (jumlah soal, skor, hasil) dikelola oleh file `game_state.py`:

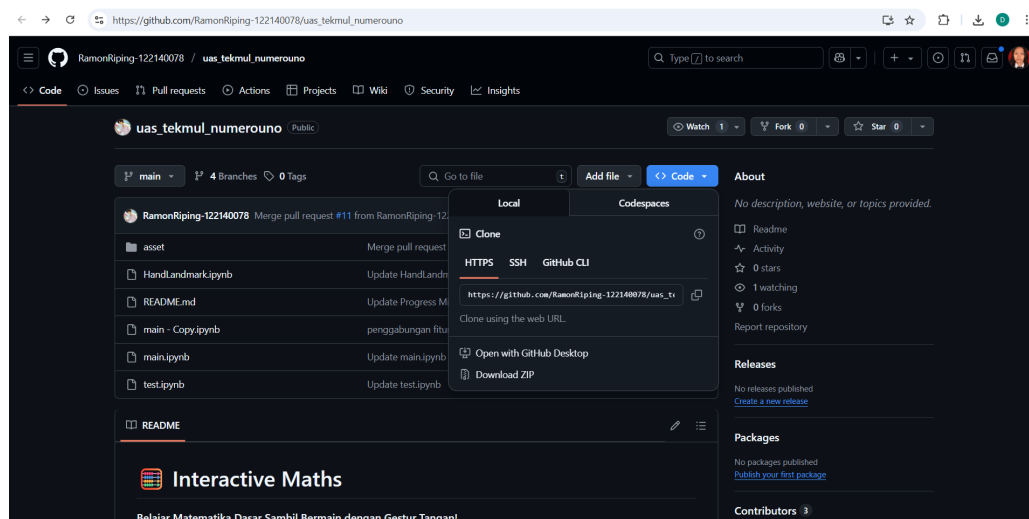
- Variabel `skor_benar` dan `skor_salah`
- Fungsi `reset_state()` untuk sesi baru
- Fungsi `is_game_over()` untuk menghentikan permainan setelah 3 soal

6 Implementasi Program

6.1 Langkah 1 Menyalin Link Repository

Langkah pertama dalam mengimplementasikan program ini adalah menyalin URL repository dari GitHub. Berikut adalah langkah-langkahnya:

1. Buka halaman repository GitHub melalui tautan berikut: ([Link Github Interactive Maths](https://github.com/RamonRiping-122140078/uas_tekmul_numerouno))
2. Klik tombol Code berwarna hijau di bagian atas repository.
3. Pilih opsi HTTPS dan salin URL yang ditampilkan

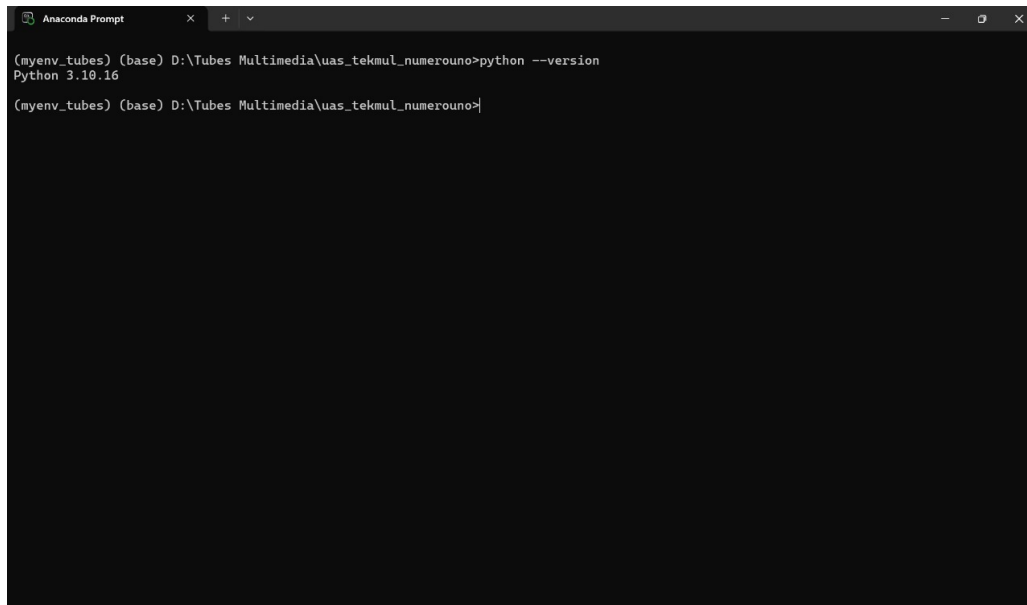


Gambar 1: Salin URL Repository

6.2 Langkah 2: Memeriksa Versi Python

Untuk memastikan program dapat dijalankan dengan benar, diperlukan Python **versi 3.10 hingga 3.12**. Berikut adalah langkah-langkah untuk mengecek versi Python:

1. Buka terminal atau command prompt.
2. Ketik perintah berikut: `python --version`
3. Pastikan versi Python yang ditampilkan berada dalam rentang **3.10 hingga 3.12**. Jika versi Python tidak sesuai, instal Python versi yang sesuai terlebih dahulu.



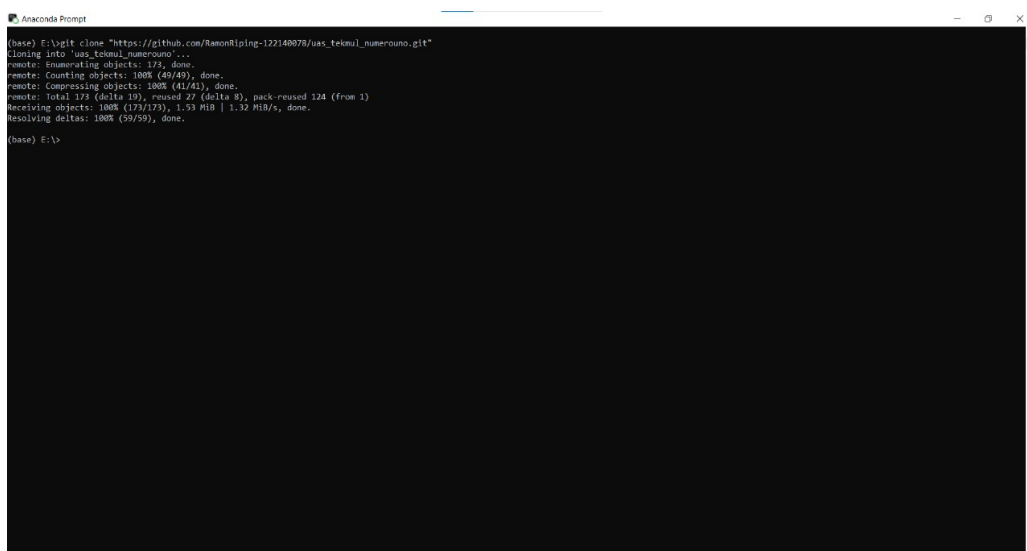
```
Anaconda Prompt
(myenv_tubes) (base) D:\Tubes Multimedia\uas_tekmul_numerouno>python --version
Python 3.10.16
(myenv_tubes) (base) D:\Tubes Multimedia\uas_tekmul_numerouno>
```

Gambar 2: Version Python Check

6.3 Langkah 3: Melakukan Clone Repository

Setelah memastikan versi Python, langkah berikutnya adalah melakukan clone repository. Langkah-langkahnya adalah sebagai berikut:

1. Buka terminal atau command prompt.
2. Pindah ke direktori tujuan untuk menyimpan repository dengan perintah:
cd /path/to/directory
3. Clone repository dengan perintah:
git clone https://github.com/RamonRiping122140078/uas-tekmul-numerouno.git
4. Setelah proses selesai, sebuah direktori baru dengan nama repository akan muncul di dalam direktori tujuan.

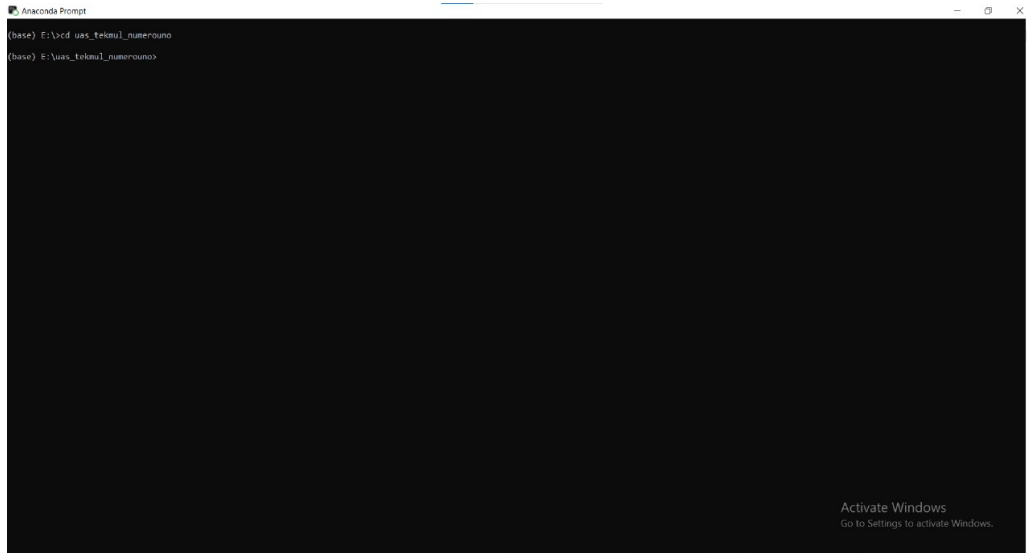


```
Anaconda Prompt
(base) E:\>git clone "https://github.com/RamonRiping-122140078/uas_tekmul_numerouno.git"
Cloning into 'uas_tekmul_numerouno'...
remote: Enumerating objects: 173, done.
remote: Counting objects: 100% (49/49), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 173 (delta 19), reused 27 (delta 8), pack-reused 124 (from 1)
Receiving objects: 100% (173/173), 1.33 MiB | 1.32 MiB/s, done.
Resolving deltas: 100% (39/39), done.
(base) E:\>
```

Gambar 3: Apply Clone Repository

6.4 Langkah 4: Pindah ke Direktori Repository

Untuk mengakses file repository yang telah di-clone, pindah ke direktori repository dengan perintah: `cd uas-tekmul-numerouno` (gunakan underscore)

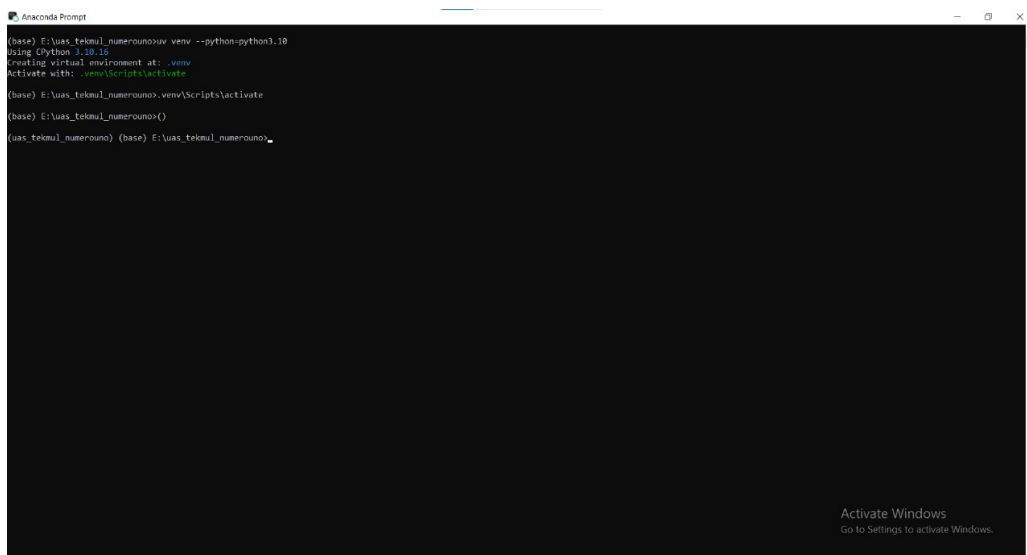


Gambar 4: Pindah ke Path Clone

6.5 Langkah 5: Menginstal Virtual Environment

Untuk

- Di Linux/macOS: `bashsource .venv/bin/activate`
- Di Windows: `bash`
`.venv/Scripts/activate`



Gambar 5: Menginstal Virtual Envionment

6.6 Langkah 6: Menginstal Dependensi

File requirements.txt berisi daftar library yang diperlukan untuk menjalankan program. Instal semua dependensi dengan perintah berikut:

pip install -r requirements.txt

Pastikan semua library terinstal tanpa error sebelum melanjutkan. Install Requirements
cd uas-tekmul-numerouno (gunakan underscore)

```

(uas_tekmul_numerouno) (base) E:\uas_tekmul_numerouno> pip install -U -r requirements.txt
Resolved 32 packages in 2.1s
Prepared 32 packages in 1m 02s
Installing wheels:
  0/32
Warning: Failed to hardlink files; falling back to full copy. This may lead to degraded performance.
If the cache and target directories are on different filesystems, hardlinking may not be supported.
If this is intentional, set 'export UV_LINK_MODE=copy' or use '--link-mode=copy' to suppress this warning.
Installed 32 packages in 14.98s
+ absl-py==2.3.0
+ attrs==25.3.0
+ cffi==1.17.1
+ comtypes==1.4.11
+ contourpy==1.3.2
+ cycle==0.12.1
+ flatbuffers==25.2.10
+ fonttools==4.58.1
+ jax==0.6.1
+ jaxlib==0.6.1
+ kiwisolver==1.4.8
+ matplotlib==3.10.3
+ mediapipe==0.10.21
+ ml-dtypes==0.5.1
+ numpy==1.26.4
+ opencv-contrib-python==4.11.0.86
+ opencv-python==4.11.0.86
+ opt-einsum==3.4.0
+ packaging==25.0
+ pillow==11.2.1
+ protobuf==4.25.8
+ pycparser==2.22
+ pygame==2.6.1
+ pygaming==1.2.3
+ pyproj==3.2.2
+ python-dateutil==2.9.0.post0
+ pytorch==2.9.0
+ pywin32==310
+ scipy==1.15.3
+ sentencepiece==0.2.0
+ six==1.17.0
+ sounddevice==0.5.2
(uas_tekmul_numerouno) (base) E:\uas_tekmul_numerouno>

```

Gambar 6: Menginstal Dependensi

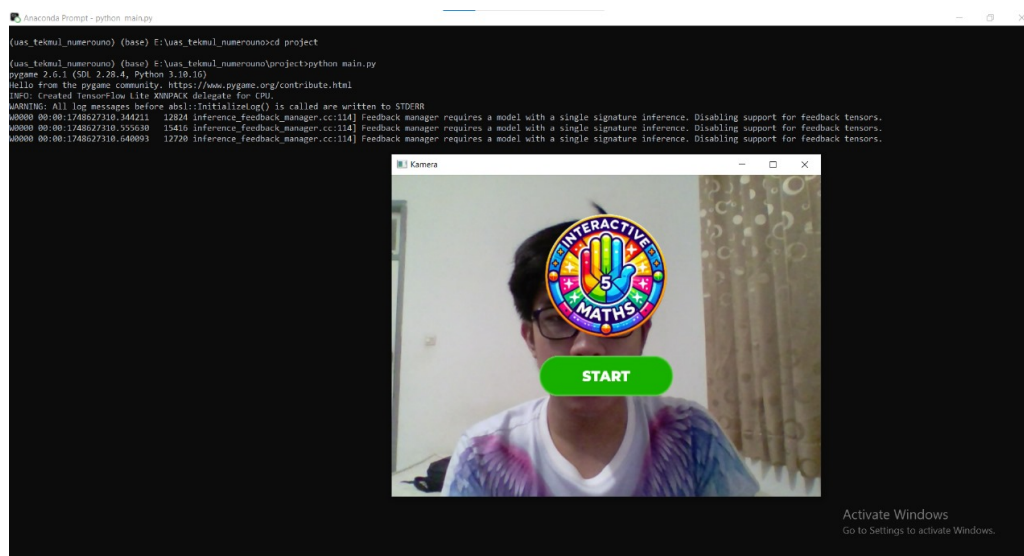
6.7 Langkah 7: Menjalankan Program

Setelah dependensi terinstal, jalankan program dengan perintah berikut:

cd project

folder main.py

Program akan mulai berjalan, dan antarmuka permainan akan ditampilkan.



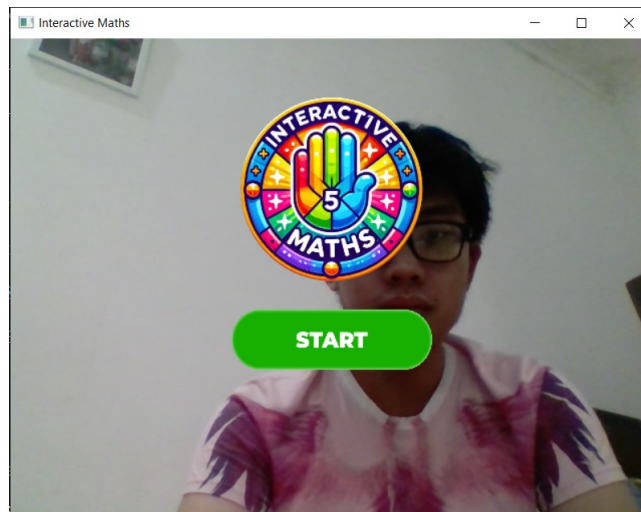
Gambar 7: Menjalankan Program

6.8 Langkah 8: Antarmuka Program

Antarmuka program terdiri dari beberapa bagian utama. Berikut adalah beberapa tangkapan layar dari antarmuka program:

1. Tampilan Mulai

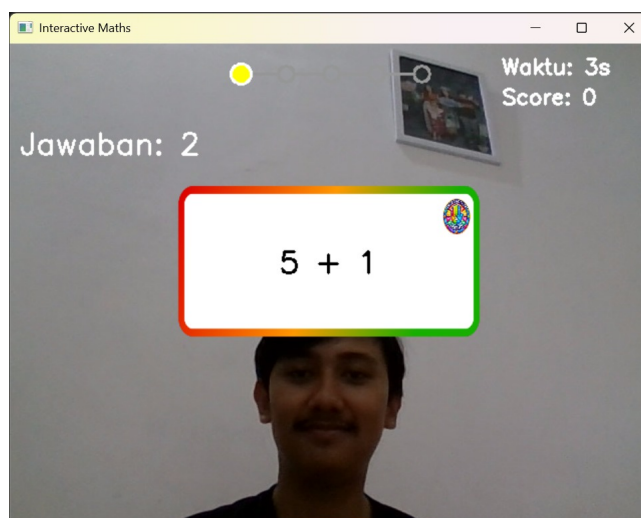
Menampilkan tombol start untuk memulai permainan



Gambar 8: Tampilan Mulai

2. Tampilan Soal Matematika

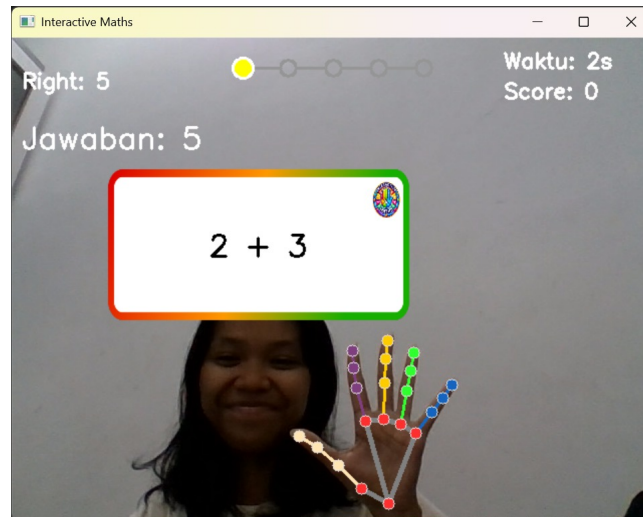
Menampilkan bagian utama dari antarmuka di mana soal matematika ditampilkan secara acak satu per satu. Memfokuskan perhatian pemain pada soal yang diberikan dan mendorong interaksi dengan menunjukkan jawaban menggunakan jari tangan ke kamera.



Gambar 9: Tampilan Soal Matematika

3. Gameplay

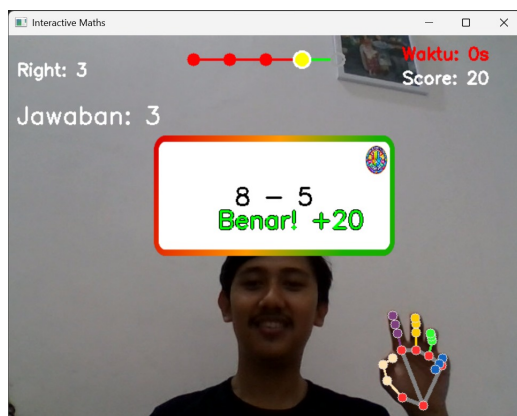
Menampilkan hasil tangkapan webcam secara real-time dengan overlay (tanda-tanda) dari deteksi tangan. Memberikan umpan balik visual kepada pemain bahwa tangan mereka berhasil dikenali, dan berapa jumlah jari yang terbaca oleh sistem.



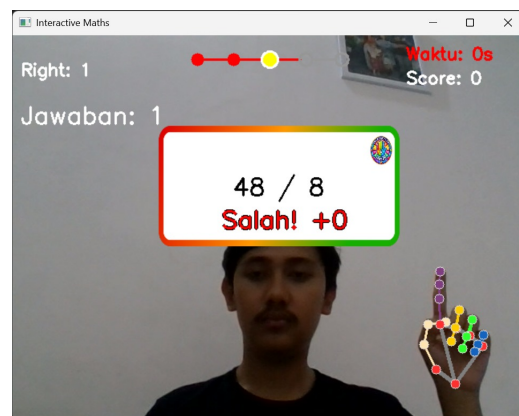
Gambar 10: Tampilan Gameplay

4. Tampilan Umpan Balik Jawaban

Menampilkan umpan balik segera setelah pengguna menunjukkan jari. Memberikan konfirmasi kepada pemain bahwa jawaban mereka telah diterima dan memberitahu apakah benar atau salah, dalam bentuk teks dan audio.



(a) Jawaban Benar



(b) Jawaban Salah

Gambar 11: Tampilan Umpan Balik Jawaban

5. Tampilan Score Akhir

Ketika salah satu pemain mencapai skor kemenangan, layar pemenang akan ditampilkan.



Gambar 12: Tampilan Score Akhir

7 Kredit/Sitasi/Refrensi

Berikut ini link refrensi project kami

(Copilot)

(Grok)

(GPT 1)

(GPT 2)