



Nama Anggota:

- Cindy Nadila Putri (122140002)
- M. Arief Rahman Hakim (122140083)
- Zidan Raihan (122140100)

Tugas: Tugas Besar

Mata Kuliah: Sistem teknologi Multimedia (IF25-40305) Tanggal: Rabu, 10 Desember 2025

---

## 1 Pendahuluan

### 1.1 Latar Belakang

Perkembangan teknologi multimedia dan *Computer Vision* dalam beberapa tahun terakhir telah mengubah cara manusia berinteraksi dengan perangkat *digital*. Jika sebelumnya interaksi lebih banyak mengandalkan perangkat input konvensional seperti keyboard dan mouse, kini muncul tren menuju *Natural User Interface* (NUI) yang memanfaatkan gestur tubuh sebagai bentuk komunikasi yang lebih alami. Kemajuan metode pose estimation memungkinkan komputer mengenali posisi dan pergerakan tubuh secara *real-time* hanya menggunakan kamera biasa, sehingga solusi interaksi non-invatif menjadi semakin mudah diakses dan tidak membutuhkan perangkat sensor khusus.

Kemunculan teknologi deteksi pose *modern* seperti BlazePose memberikan kontribusi besar terhadap perkembangan ini. Bazarevsky dkk. mengembangkan BlazePose sebagai model on-device real-time body tracking yang mampu mendekripsi lebih dari 30 landmark tubuh dengan akurasi tinggi dan latensi rendah bahkan pada perangkat CPU standar [1]. Kemampuan pemrosesan ringan ini menjadikan pose estimation dapat digunakan di berbagai skenario, termasuk aplikasi *Augmented Reality* (AR), analisis gerak, dan sistem interaktif berbasis kamera yang tidak membutuhkan perangkat mahal seperti Kinect.

Implementasi sistem multimedia interaktif berbasis visi komputer memerlukan arsitektur perangkat lunak yang mampu menjamin pemrosesan citra yang kompleks dengan visualisasi grafis yang responsif. Dalam konteks ini, penggunaan bahasa pemrograman Python dengan pustaka Pygame menawarkan solusi yang efisien untuk pengembangan purwarupa cepat (*rapid prototyping*). Sebuah studi oleh Parulekar et al. (2023) menunjukkan bahwa kombinasi OpenCV dan Pygame mampu mendukung pengembangan game *Augmented Reality* (AR) berbasis gestur yang responsif, di mana Pygame berperan sebagai modul visualisasi ringan yang dapat terintegrasi langsung dengan proses deteksi gerakan secara *real-time* [2].

Berdasarkan perkembangan dan tantangan tersebut, tugas besar ini mengimplementasikan sistem permainan interaktif “Cam-Fu” yang memanfaatkan webcam sebagai sumber input utama untuk membaca gerakan tubuh pemain. Sistem ini mengintegrasikan beberapa komponen teknis, mulai dari pengambilan *frame* kamera, pemrosesan koordinat pose menggunakan BlazePose/MediaPipe, hingga penyajian elemen grafis menggunakan Pygame. Penelitian ini juga membangun logika permainan seperti sistem skor, nyawa, *collision detection*, pengaturan objek musuh, serta mengorganisasi kode menggunakan pendekatan OOP agar lebih modular dan mudah dikembangkan.

Melalui proyek ini, didapatkan kesempatan untuk menerapkan berbagai konsep teknologi multimedia secara langsung, mulai dari akuisisi data visual, analisis gerakan menggunakan algoritma AI, hingga visualisasi interaktif berbasis game. Selain itu, mahasiswa dapat menganalisis performa sistem, seperti stabilitas FPS dan akurasi deteksi gerakan, untuk memahami tantangan nyata dalam membangun aplikasi multimedia berbasis *Computer Vision*. Harapannya, tugas besar ini dapat menjadi pengalaman komprehensif dalam memahami proses lengkap pengembangan sistem interaktif *modern*.

## 2 Alat dan Bahan

Sebelum memulai proyek ini, perlu dipersiapkan alat dan bahan yang nantinya akan digunakan untuk proses pembuatan filter game pada proyek ini. Berikut ini merupakan penjabaran alat dan bahan yang digunakan:

### 2.1 Bahasa Pemrograman

- **Python 3.11:** Dipilih karena memiliki dukungan pustaka multimedia dan pengolahan citra yang sangat luas serta sintaks yang mudah dipahami.

### 2.2 Pustaka (Library)

- **opencv-python:** Digunakan sebagai pustaka utama untuk manajemen input visual secara *real-time*. Dalam pengembangan game ini, OpenCV bertugas untuk mengakses perangkat kamera (webcam), menangkap aliran video *frame-by-frame*, serta melakukan pra-pemrosesan citra yang krusial, seperti membalik gambar (*flipping*) secara horizontal untuk menciptakan efek cermin yang intuitif bagi pemain, serta mengonversi ruang warna dari BGR ke RGB agar kompatibel dengan input yang dibutuhkan oleh MediaPipe.
- **mediapipe:** Digunakan untuk estimasi pose tubuh manusia (*Human Pose Estimation*) menggunakan solusi MediaPipe Pose [3]. Pustaka ini berfungsi untuk mendeteksi dan melacak 33 titik *landmark* tubuh 3D (seperti pergelangan tangan, siku, bahu, dan pinggul) dengan latensi rendah. Koordinat *landmark* yang dihasilkan kemudian digunakan untuk dua tujuan utama: pertama, untuk merender karakter *stickman* yang meniru gerakan pemain; dan kedua, sebagai acuan koordinat interaksi untuk mendeteksi pukulan terhadap target virtual, navigasi menu menggunakan kursor tangan, serta penghindaran rintangan.
- **pygame:** Digunakan sebagai kerangka kerja pengembangan permainan (*game development framework*) lintas platform yang berfungsi sebagai antarmuka utama sistem [2]. Dalam proyek ini, Pygame bertugas menjembatani data numerik dari MediaPipe menjadi visualisasi grafis yang interaktif. Perannya meliputi rendering karakter *stickman* dan objek permainan (target/rintangan) secara *real-time* di atas *layer* video, manajemen sistem audio untuk umpan balik suara (*sound effects*), serta pengelolaan logika perulangan utama (*game loop*) yang menjaga stabilitas performa aplikasi.
- **NumPy:** Digunakan sebagai pustaka fundamental untuk komputasi numerik dan manipulasi array multidimensi. Dalam arsitektur sistem ini, NumPy memiliki peran ganda yang krusial: pertama, sebagai struktur data dasar untuk pemrosesan citra, di mana setiap *frame* video yang ditangkap oleh OpenCV direpresentasikan sebagai matriks NumPy, memungkinkan manipulasi data piksel dilakukan dengan efisiensi tinggi [4]. Kedua, pustaka ini dimanfaatkan secara spesifik dalam modul audio untuk men-generate gelombang suara sintetis (seperti gelombang sinus) secara prosedural, yang berfungsi sebagai mekanisme penanganan kesalahan (*fallback mechanism*) yang tangguh apabila aset audio eksternal gagal dimuat oleh sistem.

### 2.3 Metode dan Algoritma

Dalam implementasi Cam-Fu digunakan tiga unsur pemrosesan multimedia yaitu video processing, image processing, dan audio processing. Tiap unsur memiliki peran spesifik untuk menghasilkan interaksi dalam filter permainan ini. Adapun metode yang digunakan sebagai berikut:

- **Pose Estimation:**

Pose estimation adalah teknik untuk mendeteksi posisi dan orientasi tubuh manusia berdasarkan

titik-titik kunci (landmark) yang merepresentasikan sendi tubuh. Model modern seperti BlazePose memanfaatkan *convolutional pose regression* yang memungkinkan estimasi pose secara *real-time* menggunakan kamera biasa tanpa sensor tambahan [1]. Dalam proyek ini, pose estimation digunakan untuk mengekstraksi koordinat landmark tubuh (terutama tangan dan lengan) secara *real-time*. Koordinat ini kemudian diterjemahkan menjadi posisi kursor atau pukulan pemain pada layar. Dengan demikian, pergerakan tubuh pemain menjadi sumber input utama dalam interaksi game, menggantikan perangkat input tradisional seperti mouse atau keyboard.

- **Collision Detection:**

Euclidean Distance merupakan metode pengukuran jarak antara dua titik dalam ruang dua atau tiga dimensi. Rumus ini secara luas digunakan dalam berbagai aplikasi multimedia dan grafika komputer, termasuk pendekripsi kedekatan objek, *tracking* objek, dan sistem interaksi berbasis koordinat [5]. Dalam proyek ini, collision detection diterapkan menggunakan pendekatan *Circle-Circle Intersection*, yaitu dengan membandingkan jarak Euclidean antara pusat lingkaran “tangan pemain” dengan pusat objek target. Jika jarak antara kedua pusat lebih kecil atau sama dengan jumlah radiusnya, maka sistem menganggap terjadi tabrakan. Mekanisme ini digunakan untuk mendeteksi pukulan pemain terhadap objek musuh secara akurat meskipun objek bergerak dinamis.

- **Finite State Machine (FSM):**

*Finite State Machine* adalah model komputasi yang merepresentasikan suatu sistem sebagai sekumpulan state dengan aturan transisi yang terstruktur. Model ini banyak digunakan dalam game dan aplikasi interaktif untuk mengatur alur logika dan respons sistem terhadap input pengguna [6]. Pada proyek ini, FSM digunakan untuk mengatur alur permainan mulai dari *Menu* → *Playing* → *Game Over*. Setiap state memiliki logika tampilan dan interaksi yang berbeda, sehingga game dapat berjalan lebih teratur dan mudah dikelola. FSM memastikan bahwa transisi antar state hanya terjadi ketika kondisi tertentu terpenuhi (misalnya: pemain memulai game, nyawa habis, atau tombol tertentu ditekan).

## 3 Penjelasan Implementasi

### 3.1 Persiapan Lingkungan (Environment)

Pada tahap ini dilakukan proses pembuatan lingkungan kerja terisolasi sebelum menjalankan proyek. Lingkungan virtual diperlukan agar seluruh dependensi tersimpan secara terpisah sehingga tidak mengganggu instalasi Python pada sistem utama. Berikut adalah dua metode yang dapat digunakan, yaitu menggunakan *Anaconda* (direkomendasikan) atau menggunakan *UV* sebagai *fast Python package manager*.

#### Metode A: Menggunakan Anaconda (Direkomendasikan)

```
1 # Buat environment baru dengan Python 3.10
2 conda create -n camfu python=3.10 -y
3
4 # Aktifkan environment
5 conda activate camfu
6
7 # Install dependencies
8 pip install -r requirements.txt
```

Kode 1: Membuat Environment Menggunakan Anaconda

### Metode B: Menggunakan UV (Fast Python Package Manager)

```
1 # Install uv menggunakan pip jika belum
2 pip install uv
3
4 # Buat virtual environment dengan uv
5 uv venv --python 3.10
6
7 # Aktifkan environment
8 # Windows PowerShell:
9 .venv\Scripts\Activate.ps1
10
11 # Windows CMD:
12 .venv\Scripts\activate.bat
13
14 # Linux/Mac:
15 source .venv/bin/activate
16
17 # Install dependencies dengan uv (lebih cepat)
18 uv pip install -r requirements.txt
```

Kode 2: Membuat Environment Menggunakan UV

## 3.2 Struktur File

Sistem dibangun dengan prinsip *Object-Oriented Programming* (OOP) yang modular. Berikut adalah deskripsi singkat file-file utama:

- **main.py**: *Entry point* program. Mengatur inisialisasi window, *game loop* utama, dan perpindahan *Game State* (Menu/Play).
- **game\_engine.py**: Mengelola logika inti permainan saat state *Playing*, termasuk update posisi objek dan pemanggilan deteksi tabrakan.
- **pose\_detector.py**: *Wrapper* untuk MediaPipe. Menerima frame gambar dan mengembalikan koordinat landmark tubuh.
- **collision\_detector.py**: Berisi fungsi matematika untuk menghitung jarak antar titik dan mendeteksi tabrakan.
- **renderer.py**: Menangani semua visualisasi, termasuk menggambar stickman bergaya "lidi" (*rounded*), UI skor, dan overlay objek game.
- **menu\_manager.py**: Mengelola tampilan menu awal, tombol, dan interaksi kursor tangan.

## 3.3 Logika Penting: Deteksi Interaksi - Analisis Detail

Salah satu aspek krusial dalam sistem game interaktif adalah bagaimana gerakan fisik pengguna diterjemahkan menjadi interaksi digital yang akurat. Berikut adalah analisis mendalam terhadap logika-logika penting dalam sistem deteksi interaksi:

### 3.3.1 Deteksi Landmark Tangan (Wrist Detection)

Logika pertama yang fundamental adalah proses deteksi posisi tangan pengguna menggunakan teknologi pose detection. Kode berikut menunjukkan implementasi ekstraksi posisi wrist dari MediaPipe Pose:

```
1 # Ambil posisi TANGAN (Wrist) dari Pose Detector
2 left_wrist = self.pose_detector.get_landmark_position(
3     landmarks, self.mp_pose.LEFT_WRIST.value, self.W, self.H
4 )
5 right_wrist = self.pose_detector.get_landmark_position(
6     landmarks, self.mp_pose.RIGHT_WRIST.value, self.W, self.H
7 )
```

Kode 3: Wirst Detection

**Penjelasan Teknis:** Sistem menggunakan MediaPipe Pose untuk mendeteksi 33 titik landmark tubuh manusia, dengan fokus khusus pada `LEFT_WRIST` dan `RIGHT_WRIST` sebagai indikator utama posisi tangan. Koordinat yang dihasilkan kemudian di-normalisasi berdasarkan dimensi layar (`self.W`, `self.H`) untuk memastikan konsistensi across different device resolutions.

### 3.3.2 Pengolahan Posisi Tangan (Hand Position Processing)

Setelah mendapatkan posisi kedua wrist, sistem melakukan proses pengumpulan dan filtrasi data yang optimal:

```
1 # Gabungkan tangan yang terdeteksi
2 hands = [p for p in [left_wrist, right_wrist] if p is not None]
```

Kode 4: Hand Position Processing

**Penjelasan Teknis:** Implementasi menggunakan list comprehension yang efisien untuk memfilter tangan yang berhasil dideteksi. Hanya tangan dengan status `not None` yang dimasukkan ke dalam list `hands`, memungkinkan fleksibilitas penggunaan satu atau dua tangan tergantung visibilitas dan kondisi pencahayaan.

### 3.3.3 Deteksi Tabrakan Euclidean (Euclidean Collision Detection)

Inti dari sistem interaksi adalah algoritma deteksi tabrakan menggunakan perhitungan jarak Euclidean:

```
1 # Menggunakan collision_detector untuk cek jarak Euclidean
2 hit_hand = self.collision_detector.check_hand_collision(
3     hands, [], target.get_position(), target.radius
4 )
```

Kode 5: Euclidian Collision Detection

**Penjelasan Teknis:** Algoritma menggunakan formula jarak Euclidean:  $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$  untuk menghitung jarak antara posisi tangan dan target. Parameter `hands` berisi daftar posisi tangan terdeteksi, `target.get_position()` mengembalikan koordinat target, dan `target.radius` menentukan threshold deteksi tabrakan. Sistem mengembalikan `True` jika terjadi tabrakan dalam radius yang ditentukan.

### 3.3.4 Manajemen Skor (Score Management)

Sistem reward dirancang untuk memberikan feedback progresif kepada pemain:

```
1 if hit_hand:
2     target.active = False
3     self.score_manager.add_score(target.points)
```

Kode 6: Score Management

**Penjelasan Teknis:** Ketika tabrakan terdeteksi, sistem menandai target sebagai tidak aktif (`target.active = False`) dan menambahkan skor sesuai dengan `target.points` yang telah ditetapkan. Setiap target dapat memiliki nilai berbeda, memungkinkan sistem untuk memberikan reward yang bervariasi berdasarkan difficulty atau tipe target.

### 3.3.5 Manajemen Audio (Sound Management)

Feedback audio merupakan komponen penting untuk meningkatkan user experience:

```
1 if hit_hand:  
2     # ... (kode sebelumnya)  
3     self.sound_manager.play_sound('hit')
```

Kode 7: Sound Management

**Penjelasan Teknis:** Sistem menggunakan sound manager untuk mengelola berbagai efek audio dalam game. Parameter string '`hit`' mengidentifikasi jenis suara yang akan dimainkan, memungkinkan expansi untuk berbagai event (miss, combo, bonus, dll.). Implementasi ini memberikan sensory feedback yang meningkatkan engagement dan usability.

### 3.3.6 Manajemen Status Target (Target State Management)

Sistem cleanup dan pergantian target memastikan game flow yang optimal:

```
1 for target in self.targets[:]:  
2     if not target.active: continue  
3  
4     # Cek tabrakan...  
5     if hit_hand:  
6         target.active = False  
7         # ... (kode lainnya)
```

Kode 8: Target State Management

**Penjelasan Teknis:** Menggunakan `self.targets[:]` untuk membuat copy list yang aman untuk iterasi. Sistem men-skip target yang sudah tidak aktif (`if not target.active: continue`) dan menandai target yang berhasil dihantam sebagai "mati". Sistem akan mengganti target yang tidak aktif dengan target baru di cycle berikutnya, memastikan konsistensi jumlah target dalam game. Alur logika deteksi interaksi dapat diringkas sebagai proses sequensial:

1. **Deteksi:** MediaPipe Pose mendeteksi dan mengekstrak posisi wrist
2. **Filtrasi:** Posisi tangan yang valid dikumpulkan dan diproses
3. **Perhitungan:** Jarak Euclidean dihitung untuk setiap target aktif
4. **Evaluasi:** Tabrakan dideteksi berdasarkan threshold radius
5. **Respons:** Skor ditambahkan, suara dimainkan, target dinonaktifkan
6. **Perbaruan:** Status game diperbarui untuk cycle selanjutnya

Setiap sub-sistem bekerja secara koordinatif untuk menciptakan pengalaman interaksi yang responsif, akurat, dan engaging antara gerakan fisik pengguna dan elemen digital dalam game environment.

## 4 Hasil

Berikut adalah tampilan antarmuka dan hasil uji coba permainan Cam-Fu.

### 4.1 Tampilan Menu Utama

Pada menu utama, kamera ditampilkan sebagai latar belakang (*AR Mode*) dan pemain berinteraksi menggunakan kursor tangan. Untuk memilih menu, pemain perlu memosisikan tangan di atas salah satu tombol seperti *START*, *GUIDE*, atau *CREDIT*. Setelah kursor tangan berada pada posisi tombol yang diinginkan, pemain dapat melakukan aksi klik dengan cara mengubah bentuk tangan dari posisi terbuka menjadi mengepal, sehingga sistem mengenali gerakan tersebut sebagai perintah memilih.



Gambar 1: Tampilan Main Menu dengan Kursor Tangan

#### 4.2 Tampilan Credit

Pada menu Credit, kamera tetap digunakan sebagai latar belakang (*AR Mode*) sehingga konsisten dengan nuansa antarmuka permainan. Bagian tengah layar menampilkan logo Cam-Fu serta daftar pengembang yang terdiri dari tiga anggota beserta nomor induk mahasiswanya. Di bagian bawah juga ditampilkan tautan menuju repositori GitHub sebagai sumber kode proyek. Selain informasi pengembang, terdapat pula tombol “Back to Menu” di sisi kiri bawah layar yang berfungsi untuk kembali ke halaman utama. Pemain dapat memilih tombol ini menggunakan kursor tangan dengan memosisikan tangan di atas tombol, kemudian melakukan gerakan klik dengan mengepalkan tangan agar sistem mengenali perintah pemilihan.



Gambar 2: Tampilan Credit

### 4.3 Tampilan Guide

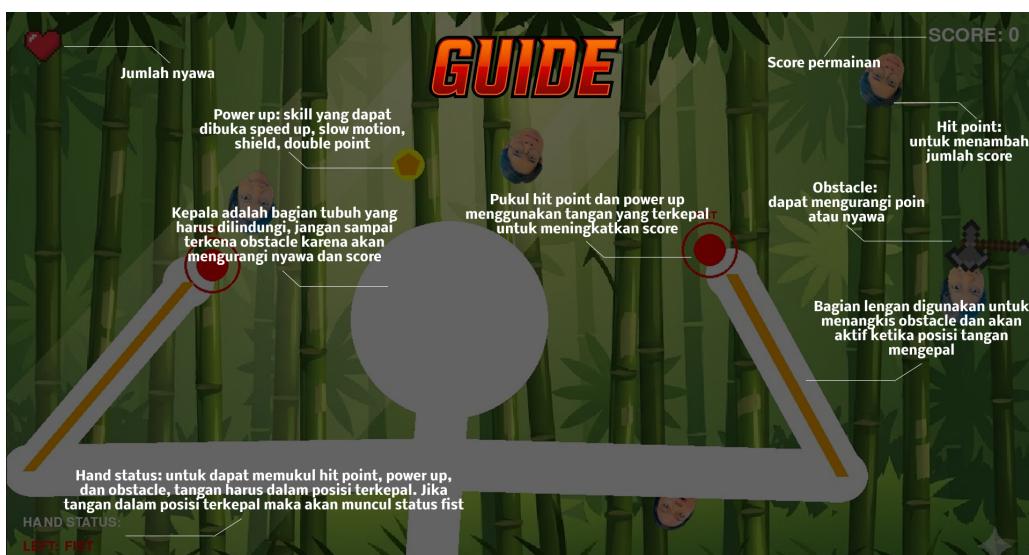
Pada menu *Guide*, pemain diberikan penjelasan mengenai fungsi setiap elemen yang muncul selama permainan. Bagian ini membantu pemain memahami mekanisme interaksi, cara bertahan, serta cara memperoleh skor. Tampilan guide menyoroti beberapa elemen penting sebagai berikut:

- **Jumlah Nyawa:** Ditampilkan di bagian kiri atas layar sebagai indikator sisa kesempatan pemain.
- **Score Permainan:** Terletak di kanan atas sebagai informasi skor yang terus bertambah ketika pemain berhasil memukul hit point atau power-up.
- **Hit Point (Target):** Objek yang harus dipukul menggunakan tangan mengepal untuk menambah skor.
- **Power-Up:** Objek berwarna kuning yang memberikan efek seperti *speed up*, *slow motion*, *shield*, atau *double point*.

Selain elemen tersebut, terdapat juga beberapa komponen yang harus diperhatikan selama permainan berlangsung:

- **Obstacle:** Rintangan yang dapat mengurangi nyawa atau skor jika mengenai kepala stickman. Obstacle juga dapat ditangkis menggunakan lengan ketika tangan berada pada posisi mengepal.
- **Kepala Stickman:** Area tubuh yang wajib dilindungi karena apabila terkena obstacle, pemain akan menerima penalti berupa pengurangan nyawa.
- **Lengan Stickman:** Mengikuti gerakan pemain secara *real-time* dan berfungsi untuk menangkis obstacle maupun memukul target.
- **Hand Status:** Menampilkan status tangan apakah terbuka atau mengepal. Status *fist* diperlukan untuk melakukan aksi menyerang maupun mengambil power-up.

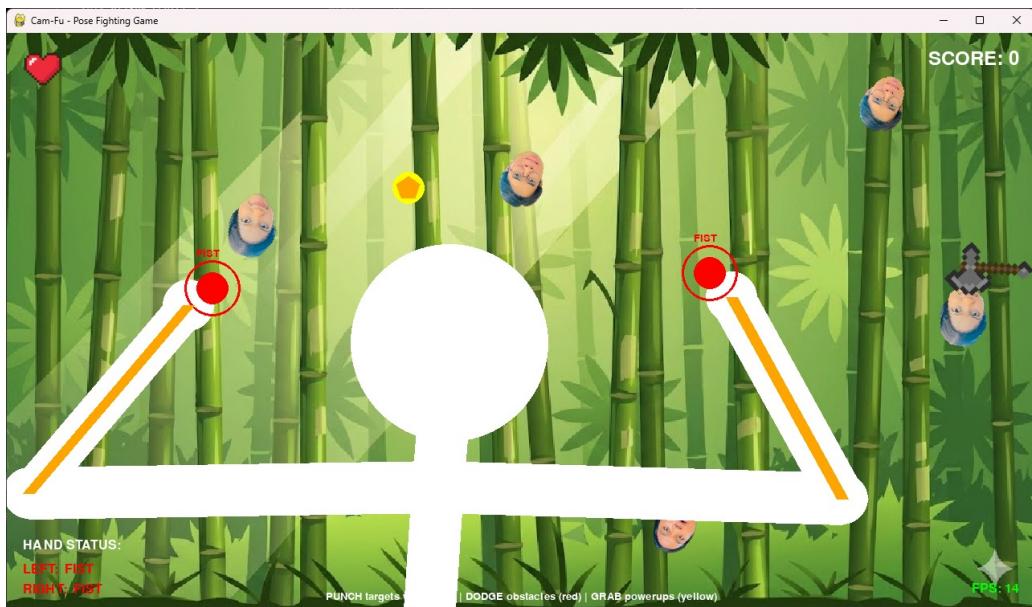
Dengan adanya penjelasan pada Gambar 3. Tampilan Guide ini, pemain dapat memahami mekanisme permainan secara menyeluruh sehingga dapat bermain lebih efektif dan strategis.



Gambar 3: Tampilan Guide

#### 4.4 Tampilan Gameplay

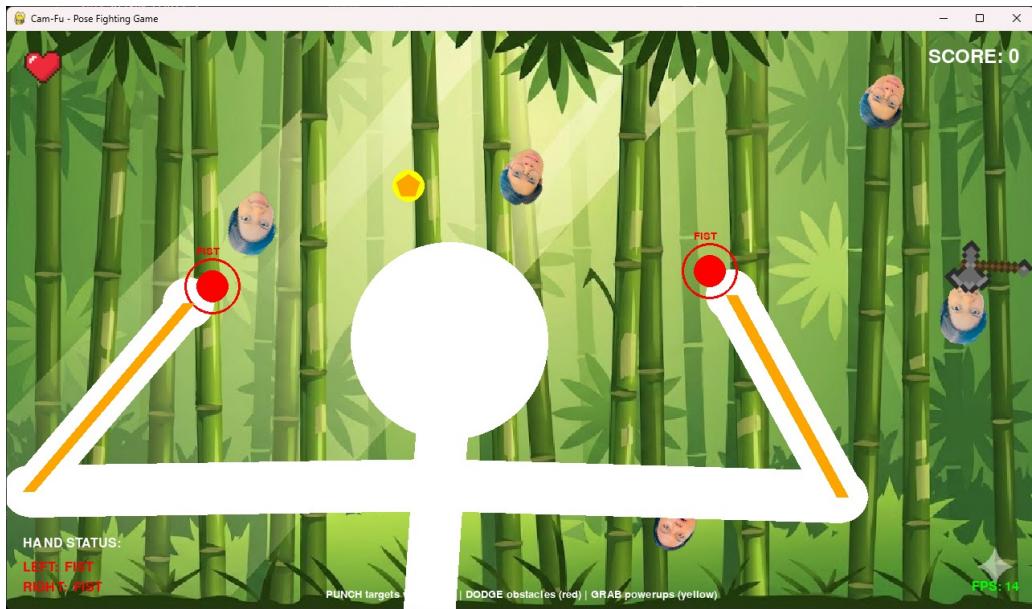
Pada mode gameplay, pemain mengendalikan karakter stickman yang digambar mengikuti pergerakan tubuh pemain secara *real-time*, termasuk posisi lengan dan perubahan gestur tangan. Berbagai objek akan muncul dari berbagai arah: target yang harus dipukul untuk menambah skor, rintangan berupa senjata yang harus dihindari karena dapat mengurangi nyawa, serta powerups berwarna kuning yang dapat diambil untuk memperoleh keuntungan tertentu. Pemain melakukan aksi memukul dengan mengepalkan tangan, sementara navigasi posisi dilakukan dengan menggerakkan lengan sesuai instruksi visual di layar.



Gambar 4: Tampilan Gameplay

#### 4.5 Tampilan Game Over

Pada layar Game Over, permainan menampilkan hasil akhir yang diperoleh pemain, berupa total skor dan durasi waktu bermain. Informasi Final Score ditampilkan di bagian tengah layar, diikuti dengan Time Played yang menunjukkan total waktu pemain bertahan selama permainan berlangsung. Selain itu, terdapat tombol “Back to Menu” di sisi kiri bawah layar yang dapat dipilih pemain dengan memosisikan tangan pada tombol dan melakukan gerakan mengepal untuk kembali ke halaman utama. Tampilan ini berfungsi sebagai ringkasan performa pemain sekaligus titik akhir sebelum memulai permainan kembali.



Gambar 5: Tampilan Game Over

## 5 Kesimpulan

Berdasarkan hasil implementasi dan pengujian sistem permainan interaktif Cam-Fu, dapat ditarik beberapa kesimpulan utama:

1. Integrasi antara pustaka OpenCV untuk akuisisi citra dan Pygame untuk visualisasi grafis berhasil menciptakan lingkungan permainan *Augmented Reality* yang responsif dan interaktif secara *real-time*.
2. Penerapan model *Deep Learning* MediaPipe BlazePose terbukti efektif dalam mendekripsi dan melacak *landmark* tubuh pemain dengan latensi rendah pada perangkat CPU standar, memungkinkan kontrol permainan yang akurat tanpa sensor tambahan.
3. Implementasi arsitektur perangkat lunak berbasis *Object-Oriented Programming* (OOP) dengan pemisahan modul yang jelas (Engine, Renderer, Detector) memfasilitasi pengelolaan logika permainan yang kompleks serta memudahkan proses *debugging* dan pengembangan fitur lanjutan.
4. Mekanisme interaksi berbasis gestur, seperti deteksi kepalan tangan (*fist detection*) untuk memukul dan posisi tangan untuk navigasi menu, berhasil memberikan pengalaman pengguna yang intuitif, imersif, dan mendorong aktivitas fisik pemain.

## 6 Saran

Pengembangan sistem Cam-Fu di masa mendatang disarankan untuk menambahkan umpan balik visual yang lebih kaya, seperti efek partikel saat target hancur, guna meningkatkan kepuasan visual pemain. Selain itu, variasi *gameplay* dapat diperluas dengan mendekripsi gestur tubuh yang lebih kompleks, misalnya tendangan atau gerakan menghindar spesifik, serta menerapkan sistem kesulitan adaptif yang menyesuaikan kecepatan atau jumlah rintangan berdasarkan performa pemain secara dinamis.

## References

- [1] V. Bazarevsky, I. Grishchenko, K. Raveendran, F. Zhu, M. Zhang, and M. Grundmann, “Blazepose: On-device real-time body pose tracking,” *arXiv preprint arXiv:2006.10204*, 2020.
- [2] M. Parulekar, A. Shukla, T. Sheka, C. Singhal, and P. Badgujar, “Interactive hand gesture control system for augmented reality-based games,” *International Journal of Scientific Research in Engineering and Management (IJSREM)*, vol. 7, no. 12, 2023.
- [3] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Jariwala, M. Galvez, G. Baxter, M. Gvili, M. Cuan, S. Raichura, F. Ringeval, B. Stauffer, and M. Grundmann, “Mediapipe: A framework for building perception pipelines,” 2019. [Online]. Available: <https://arxiv.org/abs/1906.08172>
- [4] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with numpy,” <https://www.nature.com/articles/s41586-020-2649-2>, 2020.
- [5] A. Sharma and R. Singh, “Computer vision techniques for object tracking: A survey,” *International Journal of Computer Applications*, 2019.
- [6] E. N. F. Dewi, A. N. Rachman, and R. H. Z. Hartadji, “Implementation of hierarchical finite state machine for controlling 2d character animation in action video game,” in *2023 Eighth International Conference on Informatics and Computing (ICIC)*. IEEE, 2023, pp. 1–6.

## 7 Lampiran

Bagian lampiran ini berisi tautan dan dokumentasi pendukung dari proses pengembangan permainan interaktif *Cam-Fu*. Lampiran mencakup repositori kode program, video demonstrasi permainan, serta video referensi yang digunakan dalam proses pengembangan.

### 1. GitHub: Cam-Fu Repository

Berisi kode program lengkap, asset permainan, serta dokumentasi pengembangan.

### 2. Video Demo Permainan

Tautan video demonstrasi permainan yang menampilkan cara bermain, mekanisme deteksi tangan, dan alur gameplay.

### 3. Referensi Project

Artikel referensi yang menjadi inspirasi dalam pembuatan game berbasis gesture recognition.

### 4. Sound Effect yang Digunakan dalam Permainan

Berikut sejumlah sumber audio yang digunakan sebagai efek suara selama gameplay:

- **In Battle** — digunakan sebagai background music saat pertarungan berlangsung.
- **Prologue** — digunakan untuk suasana pembuka atau menu awal.
- **Unlock Skill** — efek suara ketika pemain membuka kemampuan baru.
- **Punch** — efek suara ketika pemain melakukan pukulan.
- **Oof / Explosion Hit** — digunakan saat pemain terkena obstacle.
- **Game Over** — musik yang diputar ketika permainan berakhir.