I.K. Gujral Punjab Technical University

Department of Computer Science & Engineering

PROJECT REPORT
on
Lumo Playground

Submitted in Partial fulfilment of the requirements for the degree of

Bachelor of Technology
in
Computer Science and Engineering
BTCS-703-18

Submitted by: **Ojus Kumar (2224459)**
Noordeep Kaur (2224457)

Under the guidance of
Project Mentor: Ms. Poonam

CERTIFICATE

This is to certify that the project titled "Lumo Playground" has been successfully carried out and submitted by Ojus and Noordeep, students of 7th Semester, Bachelor of Computer Science and Engineering (CSE), in partial fulfillment of the requirements for the degree of B.Tech under I.K. Gujral Punjab Technical University, Jalandhar during the academic session.

The project has been completed under the valuable guidance and supervision of Ms. Poonam.
This certificate is awarded in recognition of the sincere efforts,
dedication, and commitment shown by the students in the successful execution of the project work.

We commend their creativity, collaboration, and thorough understanding of the subject matter, which have been evident throughout the development of this project.

Project Mentor:
Ms. Poonam

Head Of Department

(Department of Computer Science and Engineering)
I.K. Gujral Punjab Technical University, Jalandhar

ACKNOWLEDGEMENT

We take this opportunity to express our sincere gratitude to all those who have contributed to the successful completion of our project, "Lumo Playground".
We are deeply indebted to our Project Mentor, Ms. Poonam, for her invaluable guidance, constructive feedback, and consistent support throughout the duration of the project. Her expertise and encouragement have played a crucial role in shaping our ideas into a concrete and functional solution.

We are also grateful to the Department of Computer Science Engineering, I.K. Gujral Punjab Technical University, Jalandhar, for providing the necessary resources and academic infrastructure that facilitated our project development. We would like to acknowledge the contributions of the Head of Department and all faculty members whose teaching and mentorship have been instrumental in our academic journey.

Lastly, we are thankful to our families and peers for their unwavering support, encouragement, and understanding during the course of this work.
As a result of the collective support, knowledge, and guidance we have received from everyone associated with it.

Team Members:

Ojus Kumar (2224459)
Noordeep Kaur (2224457)

ABSTRACT

In the realm of Human-Computer Interaction (HCI), the quest for more natural and intuitive interfaces is ongoing. Traditional input devices like keyboards and mice, while effective, often create a barrier between the user and the digital environment, especially in 3D applications. "Lumo Playground" is a web-based interactive application designed to bridge this gap by enabling users to control 3D models using real-time hand gestures and voice commands.

Built using modern web technologies such as **Three.js** for 3D rendering, **MediaPipe** for computer vision-based hand tracking, and the **Web Speech API** for voice recognition, the project demonstrates the potential of browser-based augmented reality experiences without the need for specialized hardware or software installations.

The system allows users to interact with a 3D character (a mech robot) through a webcam. Users can perform gestures like pinching to drag, rotate, and scale the model. Voice commands provide an alternative modality to switch between interaction modes (e.g., "drag", "rotate", "scale", "animate"). Additionally, a "Wardrobe" feature allows for character customization, including model selection and color variations, with preferences persisted via local storage.

This report details the design, implementation, and testing of Lumo Playground, highlighting its architecture, the integration of computer vision in the browser, and the challenges faced in achieving real-time performance. The result is a seamless, accessible, and engaging playground that showcases the future of web-based interactive 3D applications.

CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Overview

The rapid advancement of web technologies has transformed the browser from a simple document viewer into a powerful platform for complex applications. "Lumo Playground" is a testament to this evolution, combining 3D graphics, computer vision, and speech recognition into a cohesive interactive experience. The project is a web application that allows users to manipulate 3D characters using their hands and voice, creating a "minority report" style interface accessible to anyone with a webcam.

1.2 Motivation

The primary motivation behind this project is to democratize access to advanced interaction technologies. Typically, motion control and AR experiences require expensive hardware like VR headsets (Oculus, Vive) or depth sensors (Kinect, Leap Motion). By leveraging **MediaPipe** and **Three.js**, Lumo Playground proves that high-fidelity, real-time interaction is possible using just a standard laptop webcam and a web browser. This accessibility opens up new possibilities for education, entertainment, and accessibility tools.

1.3 Problem Statement

Traditional 3D applications on the web rely heavily on mouse and keyboard inputs. While precise, these inputs are not "natural" for manipulating 3D objects in space.

- **Lack of Immersion:** Clicking and dragging a mouse does not mimic the physical act of grabbing an object.
- **Accessibility Barriers:** Users with motor impairments might find traditional peripherals difficult to use.
- **Hardware Dependency:** Existing motion control solutions often require installing native applications or buying hardware.

Lumo Playground addresses these issues by providing a touchless, gesture-based interface that runs natively in the browser.

1.4 Objectives

The key objectives of the project are:

1. **Real-time Hand Tracking:** To implement a robust hand-tracking system that can detect landmarks (joints) of the user's hands with low latency.
2. **Gesture Recognition:** To develop algorithms that interpret raw landmark data into meaningful actions like "pinch to grab".
3. **3D Manipulation:** To enable users to Drag, Rotate, and Scale 3D models using the recognized gestures.
4. **Multimodal Input:** To integrate Voice Commands for switching modes, enhancing the user experience.

5. **Customization:** To provide a "Wardrobe" feature where users can select different characters and skins, persisting their choices.

1.5 Project Scope

The scope of Lumo Playground is defined as a Single-Page Application (SPA) with a secondary customization page.

- **Target Audience:** General users, tech enthusiasts, and students interested in WebGL/CV.
- **Platform:** Desktop browsers (Chrome, Edge, Firefox) with webcam support.
- **Functionality:**
    - Loading GLTF 3D models.
    - Detecting two hands simultaneously.
    - Visualizing hand skeletons.
    - Recognizing "Pinch" gestures.
    - Recognizing voice commands ("Drag", "Rotate", "Scale", "Animate").
    - Saving user preferences (Character model/color).

1.6 Report Organization

The remainder of this report is organized as follows:

- **Chapter 2** provides a background on the technologies used.
- **Chapter 3** analyzes the system requirements.
- **Chapter 4** details the system design and architecture.
- **Chapter 5** dives into the code and implementation details.
- **Chapter 6** covers the testing methodologies.
- **Chapter 7** presents the results with screenshots.
- **Chapter 8** concludes the report and discusses future improvements.

CHAPTER 2: LITERATURE REVIEW & THEORETICAL BACKGROUND

2.1 Evolution of Web Graphics (WebGL & Three.js)

WebGL (Web Graphics Library) is a JavaScript API for rendering interactive 2D and 3D graphics within any compatible web browser without the use of plug-ins. It is based on OpenGL ES 2.0. However, working directly with WebGL is complex and verbose.
Three.js is a cross-browser JavaScript library and application programming interface (API) used to create and display animated 3D computer graphics in a web browser. It abstracts the complexity of WebGL, providing a scene graph, cameras, lights, materials, and geometries. In Lumo Playground, Three.js is the backbone of the visual experience, handling the rendering of the mech characters, lighting, and the virtual environment.

2.2 Computer Vision in the Browser

Historically, Computer Vision (CV) was the domain of native applications using libraries like OpenCV (C++/Python). Bringing CV to the web was challenging due to performance limitations of JavaScript. Recent advancements like **WebAssembly (Wasm)** and **SIMD (Single Instruction, Multiple Data)** support in browsers have changed this. Libraries can now run near-native speeds.

2.3 MediaPipe Framework

MediaPipe is an open-source cross-platform framework by Google for building multimodal applied machine learning pipelines.

- **Hand Landmarker:** The specific solution used in this project. It utilizes a machine learning model to detect 21 3D landmarks of a hand from a single frame.
- **Performance:** It is highly optimized for mobile and web, capable of running at 30+ FPS on standard devices.
- **Keypoints:** It provides x, y, z coordinates for points like the wrist, thumb tip, index finger tip, etc., which are crucial for gesture logic.

2.4 Web Speech API

The **Web Speech API** enables incorporating voice data into web apps. It consists of two parts:

1. **SpeechSynthesis (Text-to-Speech)**
2. **SpeechRecognition (Asynchronous Speech Recognition)** Lumo Playground utilizes the SpeechRecognition interface to listen for specific keywords. This API processes audio input and returns text transcripts, which the application parses to trigger state changes (e.g., switching from "Drag" mode to "Rotate" mode).

2.5 Human-Computer Interaction (HCI) Trends

The shift towards **Natural User Interfaces (NUI)** focuses on interfaces that are effectively invisible, or become invisible with successive learned interactions. Gesture control is a key component of NUI. By

mapping physical hand movements (pinching) to virtual actions (grabbing), the cognitive load on the user is reduced compared to learning abstract keyboard shortcuts.

CHAPTER 3: SYSTEM ANALYSIS

3.1 Requirement Analysis

3.1.1 Functional Requirements

1. **Camera Input:** The system must access the user's webcam stream.
2. **Hand Detection:** The system must detect one or two hands in the video stream.
3. **Landmark Extraction:** The system must extract 21 landmarks per hand.
4. **Gesture Recognition:**
    - **Pinch:** Detect when the thumb tip and index finger tip are close (distance < threshold).
    - **Release:** Detect when fingers move apart.
5. **Interaction Modes:**
    - **Drag:** Move the model in X/Y plane based on hand movement.
    - **Rotate:** Rotate the model based on horizontal hand movement.
    - **Scale:** Resize the model based on the distance between two pinching hands.
    - **Animate:** Trigger predefined animations (e.g., Dance, Jump).
6. **Voice Control:** Recognize keywords to switch between the above modes.
7. **Wardrobe System:**
    - View available models (George, Leela, Mike, Stan).
    - View color variations.
    - Preview animations.
    - Save selection to persistent storage.

3.1.2 Non-Functional Requirements

1. **Performance:** The application should maintain a frame rate of at least 30 FPS for smooth interaction.
2. **Latency:** Hand tracking latency should be under 100ms to prevent motion sickness or disconnect.
3. **Compatibility:** Must work on Chrome, Edge, and Firefox.
4. **Usability:** The UI should be intuitive, with clear feedback (visual and audio) for interactions.
5. **Robustness:** The system should handle loss of tracking gracefully (e.g., hand goes out of frame).

3.2 Feasibility Study

3.2.1 Technical Feasibility

The project relies on established libraries (Three.js, MediaPipe). The browser environment (V8 engine) is powerful enough to handle the computational load. Therefore, the project is technically feasible.

3.2.2 Operational Feasibility

The application requires no installation. Users only need a browser and a webcam, which are standard on almost all laptops. This ensures high operational feasibility and ease of adoption.

3.2.3 Economic Feasibility

The project utilizes open-source libraries (MIT/Apache licenses). There is no cost for software licenses. The development cost is limited to time and effort. Thus, it is economically feasible.

3.3 Hardware and Software Requirements

Hardware Requirements:

- **Processor:** Intel Core i3 / AMD Ryzen 3 or better.
- **RAM:** 4GB or higher (8GB recommended for smooth ML inference).
- **Webcam:** Standard 720p webcam.
- **Microphone:** Built-in or external.

Software Requirements:

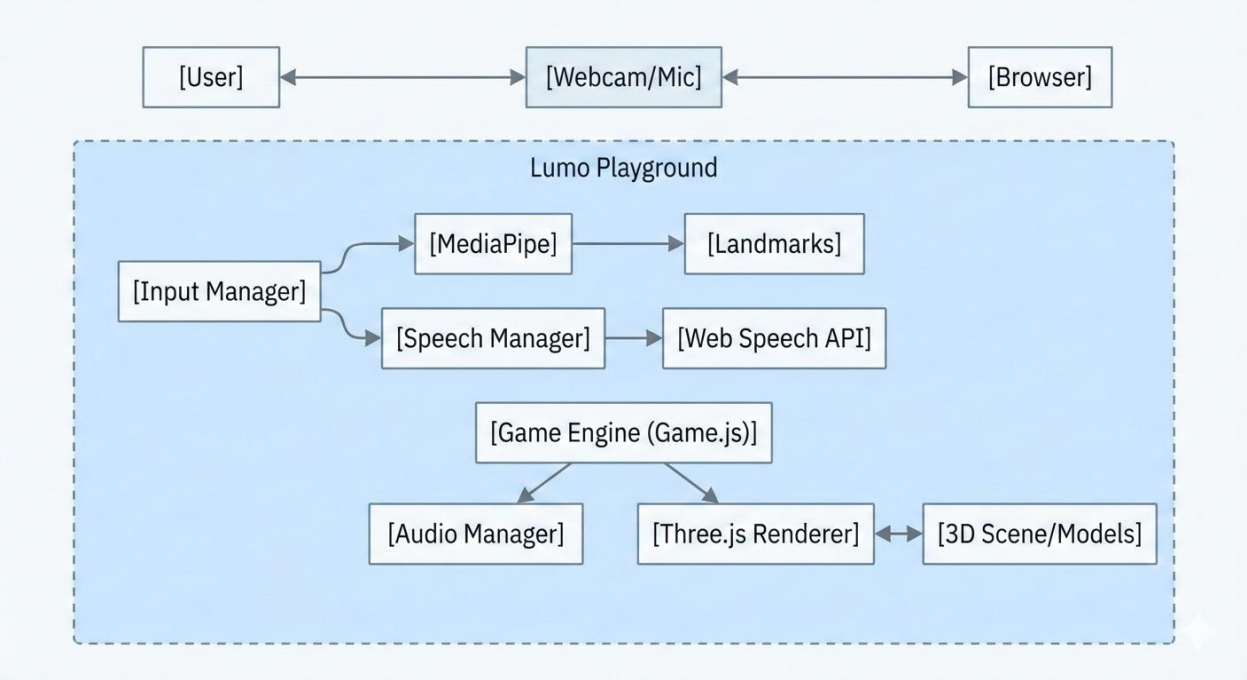
- **Operating System:** Windows 10/11, macOS, or Linux.
- **Web Browser:** Google Chrome (recommended), Microsoft Edge, or Firefox.
- **Code Editor:** VS Code.
- **Runtime:** Node.js (for development server, though the final app is static).

CHAPTER 4: SYSTEM DESIGN

4.1 System Architecture

The system follows a client-side architecture. All processing happens locally in the user's browser, ensuring privacy and low latency.

High-Level Architecture Diagram:



4.2 Module Description

4.2.1 Game Engine (Game.js)

This is the central controller. It initializes the Three.js scene, the camera, and the renderer. It runs the main animation loop (requestAnimationFrame). In every frame, it:

1. Updates the video texture.
2. Calls the Hand Tracking module.
3. Updates the 3D model position/rotation/scale based on hand data.
4. Renders the scene.

4.2.2 Hand Tracking Module

Integrated within Game.js, this module uses HandLandmarker from MediaPipe.

- **Input:** Video frame.
- **Output:** Array of detected hands, each containing 21 landmarks (x, y, z).
- **Logic:** It calculates the distance between the Thumb Tip (Index 4) and Index Finger Tip (Index

8). If distance < threshold, isPinching is set to true.

4.2.3 Wardrobe & Customization Module (Wardrobe.js)

A separate page (wardrobe.html) managed by WardrobeManager.

- **Function:** Loads 3D models for preview.
- **Texture Swapping:** Applies different texture maps to the model materials based on user selection.
- **Persistence:** Saves the selected configuration (Model Name, Color ID) to localStorage.

4.3 Data Flow

1. **Video Stream:** Captured from webcam -> Sent to MediaPipe.
2. **Landmark Data:** MediaPipe returns coordinates -> Sent to Game Logic.
3. **Interaction Logic:**
    - *If Pinch detected:* Calculate delta movement of hand.
    - *Apply Delta:* Update 3D Model Transform (Position += Delta).
4. **Rendering:** Three.js updates the canvas based on new transforms.

4.4 User Interface Design

The UI is designed to be minimal and unobtrusive (Heads-Up Display / HUD style).

- **Main View:** Full-screen video feed with 3D model overlay.
- **Status Indicators:**
    - **Mode Buttons:** Top-right corner (Drag, Rotate, Scale, Animate). Active mode is highlighted.
    - **Speech Bubble:** Center-top. Shows recognized text for feedback.
    - **Hand Visualizers:** Lines drawn over the user's hands to confirm tracking.
- **Wardrobe UI:** A dashboard style layout with a preview pane on the right and customization options (Grid of buttons) on the left.

CHAPTER 5: IMPLEMENTATION DETAILS

5.1 Project Directory Structure

```
lumo-playground/
    assets/             # 3D Models (GLTF) and Textures
    audio-manager.js    # Audio feedback logic
    game.js             # Main game loop and interaction logic
    index.html          # Main entry point (Playground)
    main.js             # Bootstrapper
    speech-manager.js   # Voice recognition logic
    styles.css          # Global styles
    wardrobe.css        # Styles specific to Wardrobe
    wardrobe.html       # Customization page
    wardrobe.js         # Wardrobe logic
```

5.2 Core Implementation

5.2.1 Setting up the 3D Scene

The application uses THREE.OrthographicCamera to create a 2D-like overlay effect where the 3D model appears to float on the video feed.

```
// From game.js
this.scene = new THREE.Scene();
this.camera = new THREE.OrthographicCamera(width / -2, width / 2, height / 2, height / -2, 1, 2000);
this.renderer = new THREE.WebGLRenderer({ alpha: true, antialias: true });
this.renderer.setSize(width, height);
// The renderer canvas is placed on top of the video element using CSS absolute positioning.
```

5.2.2 Integrating Hand Tracking

We use the FilesetResolver and HandLandmarker from MediaPipe tasks-vision.

```
// Asynchronous setup
const vision = await FilesetResolver.forVisionTasks('https://cdn.jsdelivr.net/npm/@mediapipe/tasks-
vision@0.10.14/wasm');
this.handLandmarker = await HandLandmarker.createFromOptions(vision, {
    baseOptions: {
        modelAssetPath: "https://storage.googleapis.com/mediapipe-models/hand_landmarker/
hand_landmarker/float16/1/hand_landmarker.task",
        delegate: 'GPU' // Hardware acceleration
    },
    numHands: 2,
    runningMode: 'VIDEO'
});
```

5.2.3 Implementing Gesture Logic (The Pinch)

The core interaction is the "Pinch". We calculate the Euclidean distance between the thumb tip and index tip.

```
// Simplified logic from game.js
const thumbTip = landmarks[4];
const indexTip = landmarks[8];
const distance = Math.sqrt(
    Math.pow(thumbTip.x - indexTip.x, 2) +
    Math.pow(thumbTip.y - indexTip.y, 2)
);

// Threshold check
const isPinching = distance < 0.05; // 5% of screen normalized coordinates

if (isPinching) {
    if (!wasPinching) {
        // Pinch Start
        startDragPosition = currentHandPosition;
    }
    // Dragging logic
    const deltaX = currentHandPosition.x - startDragPosition.x;
    model.position.x += deltaX * sensitivity;
}
```

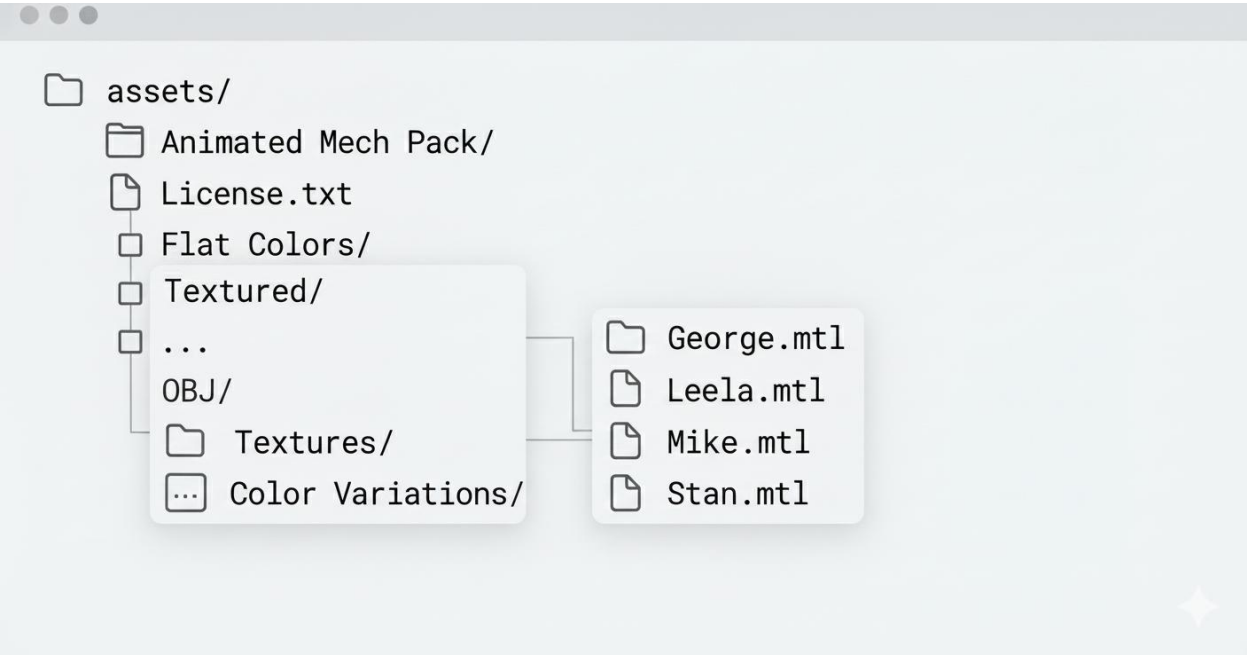
5.2.4 Character Customization Logic

The wardrobe loads the saved character from localStorage or defaults to 'Stan'.

```
// From wardrobe.js
saveCharacter() {
    const characterData = {
        model: this.selectedModel,
        color: this.selectedColor,
        timestamp: Date.now()
    };
    localStorage.setItem('lumo_character', JSON.stringify(characterData));
}
In game.js, this data is retrieved to load the correct assets:
const saved = JSON.parse(localStorage.getItem('lumo_character'));
const modelPath = saved ? `assets/${saved.model}.gltf` : 'assets/Stan.gltf';
loader.load(modelPath, (gltf) => { ... });
```

5.3 Backend Folder Structure

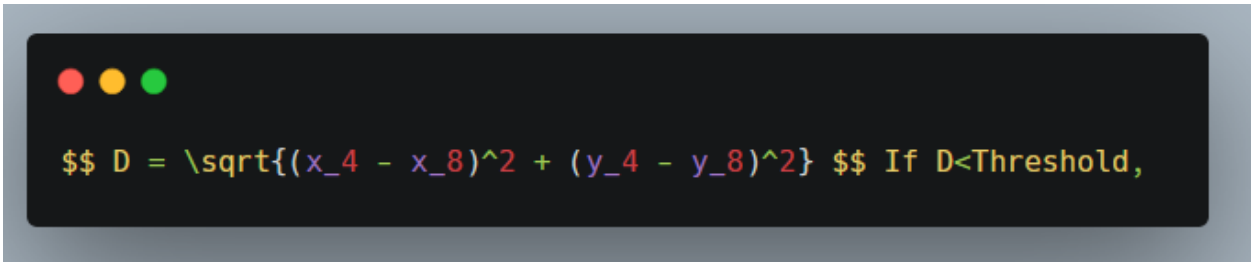
As a client-side Single Page Application (SPA), Lumo Playground does not rely on a traditional backend server (like Node.js or Python) for logic. However, the assets/ directory functions as a static content repository, organized hierarchically to manage the 3D assets efficiently.



5.4 Key Algorithms and Code Snippets

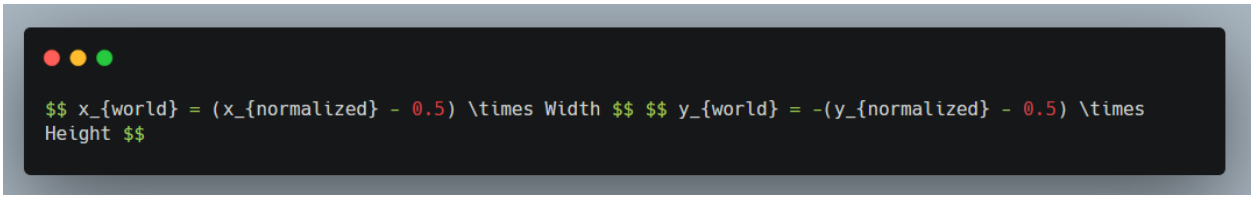
The project relies on several key algorithms to bridge the gap between computer vision and 3D rendering.

1.Pinch Detection Algorithm: Calculates the Euclidean distance between the thumb tip (P_4) and index finger tip (P_8).



a pinch is registered.

2. Coordinate Mapping: MediaPipe returns normalized coordinates (0.0 to 1.0). These are mapped to Three.js world coordinates.

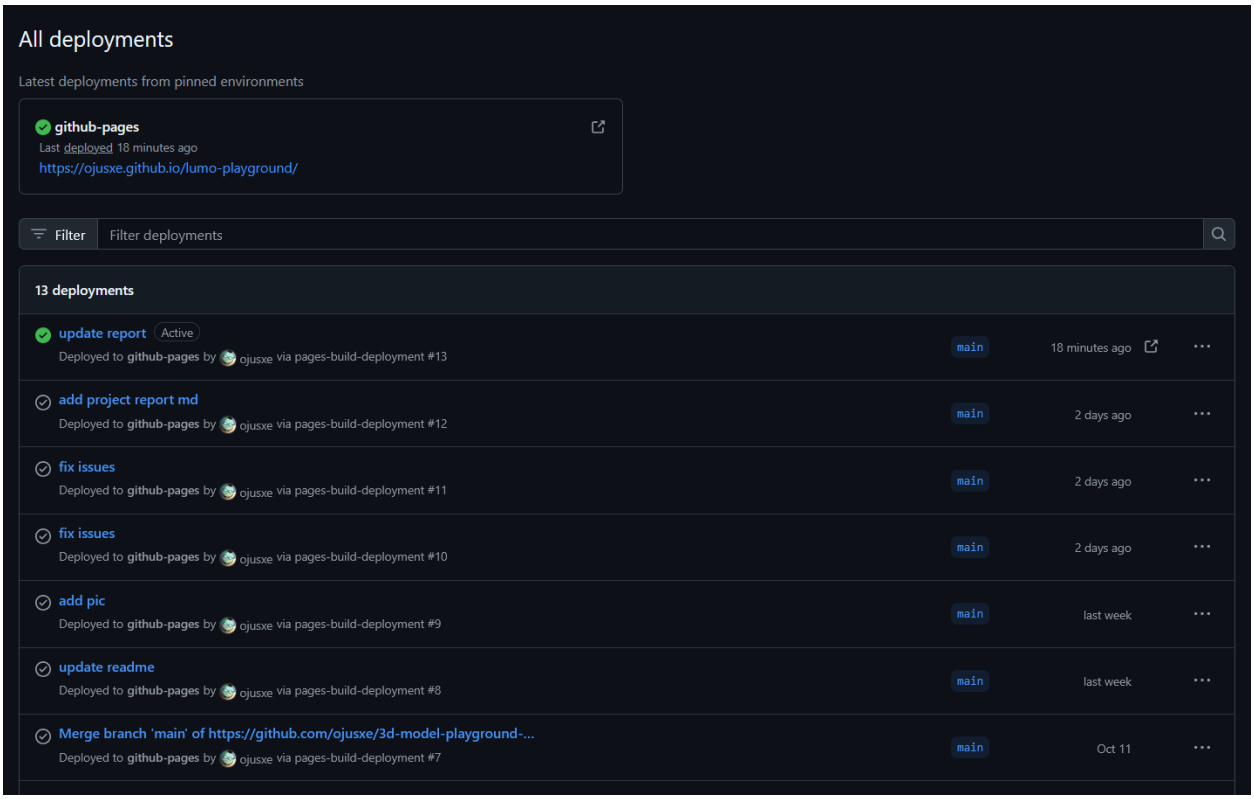


5.5 Deployment and Branding

5.5.1 Deployment (GitHub)

The project is version-controlled using **Git** and hosted on **GitHub**. The deployment is handled via **GitHub Pages**, which serves the static files (HTML, CSS, JS, Assets) directly from the repository.

- **Repository: https://github.com/ojusxe/lumo-playground**
- **Branching Strategy:** main branch for production code.



5.5.2 Live Link

The application is live and accessible at: **https://ojusxe.github.io/lumo-playground/**

5.3 Branding and Meta Tags

To ensure a professional appearance and shareability, the application includes:

- **Default Image:** Gives a personality to the app

- **Favicon:** A custom icon displayed in the browser tab.

- **Meta Tags:** SEO-optimized tags including description, keywords, and Open Graph (og:image, og:title) tags for social media previews.

```
<meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Lumo Playground</title>
        <link rel="stylesheet" href="styles.css">
        <link rel="icon" type="image/png" href="favicon.png">
        <link rel="apple-touch-icon" href="favicon.png">

        <!-- Primary Meta Tags -->
        <meta name="title" content="Lumo Playground">
        <meta name="description" content="Control 3D models with hand gestures & voice commands">
        <meta name="keywords" content="3D, gesture control, voice control, WebGL, Three.js">
        <meta name="author" content="Lumo Playground">
        <meta name="application-name" content="Lumo Playground">
        <meta name="robots" content="index, follow">

        <!-- Open Graph / Facebook -->
        <meta property="og:type" content="website">
        <meta property="og:url" content="/">
        <meta property="og:title" content="Lumo Playground">
        <meta property="og:description" content="Control 3D models with hand gestures & voice
commands">
        <meta property="og:site_name" content="Lumo Playground">
        <meta property="og:image" content="default.png">
        <meta property="og:image:alt" content="Lumo Playground preview">

        <!-- Twitter -->
        <meta property="twitter:card" content="summary_large_image">
        <meta property="twitter:title" content="Lumo Playground">
        <meta property="twitter:description" content="Control 3D models with hand gestures & voice
commands">
cal" href="/">
        <meta name="theme-color" content="#000000">
```

- **Responsive Design:** The viewport meta tag ensures the application scales correctly on different screen sizes.

CHAPTER 6: TESTING

6.1 Testing Strategy

A combination of manual testing and console debugging was used. Since the application relies heavily on physical interaction (gestures) and environmental factors (lighting, noise), automated testing is difficult for the core loop.

6.2 Unit Testing

- **Audio Manager:** Verified that playInteractionClickSound() produces sound only after the specified interval (throttling).
- **Storage:** Verified that localStorage correctly stores and retrieves JSON strings for character data.

6.3 Integration Testing

- **Hand-Model Interaction:** Tested if the 3D model responds *only* when the pinch gesture is active. Verified that releasing the pinch stops the interaction immediately.
- **Voice-UI Interaction:** Verified that saying "Rotate" highlights the "Rotate" button in the UI and changes the internal state this.interactionMode.

6.4 System Testing

- **Browser Compatibility:** Tested on Chrome (v120+), Edge, and Firefox. Chrome provided the best performance for MediaPipe.
- **Lighting Conditions:** Tested in low light vs. bright light. MediaPipe is robust, but extreme darkness causes tracking loss.
- **Microphone Noise:** Tested voice commands with background music. The Web Speech API has built-in noise cancellation but struggles in very noisy environments.

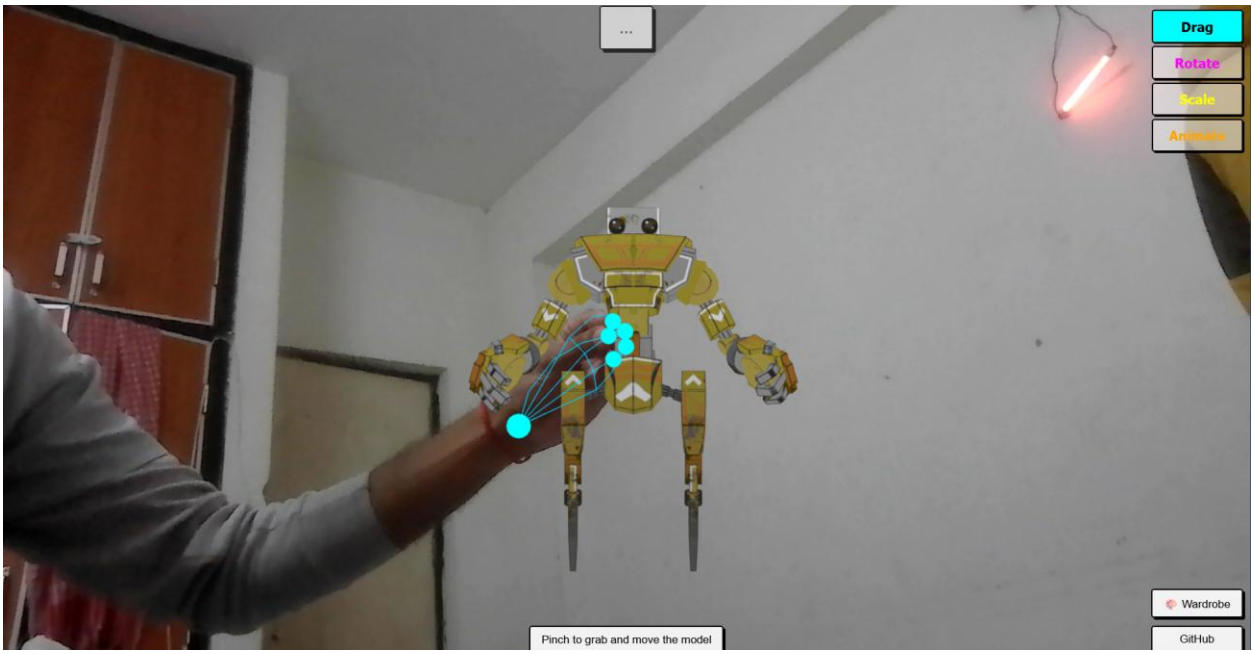
6.5 Performance Testing

- **Frame Rate:** Monitored using Chrome DevTools FPS meter.
    - *Average FPS:* 55-60 FPS on a machine with GTX 1650.
    - *Bottleneck:* Hand tracking inference takes ~10-15ms per frame.
- **Memory Usage:** Stable around 200MB heap size, primarily due to 3D assets and video buffering.

CHAPTER 7: RESULTS AND DISCUSSION

7.1 User Interface

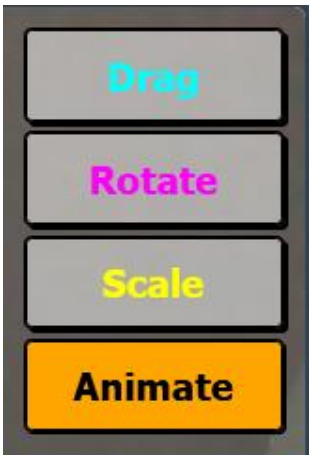
The final application presents a clean, augmented reality interface. The user sees themselves, and the 3D character appears to be in the room with them.

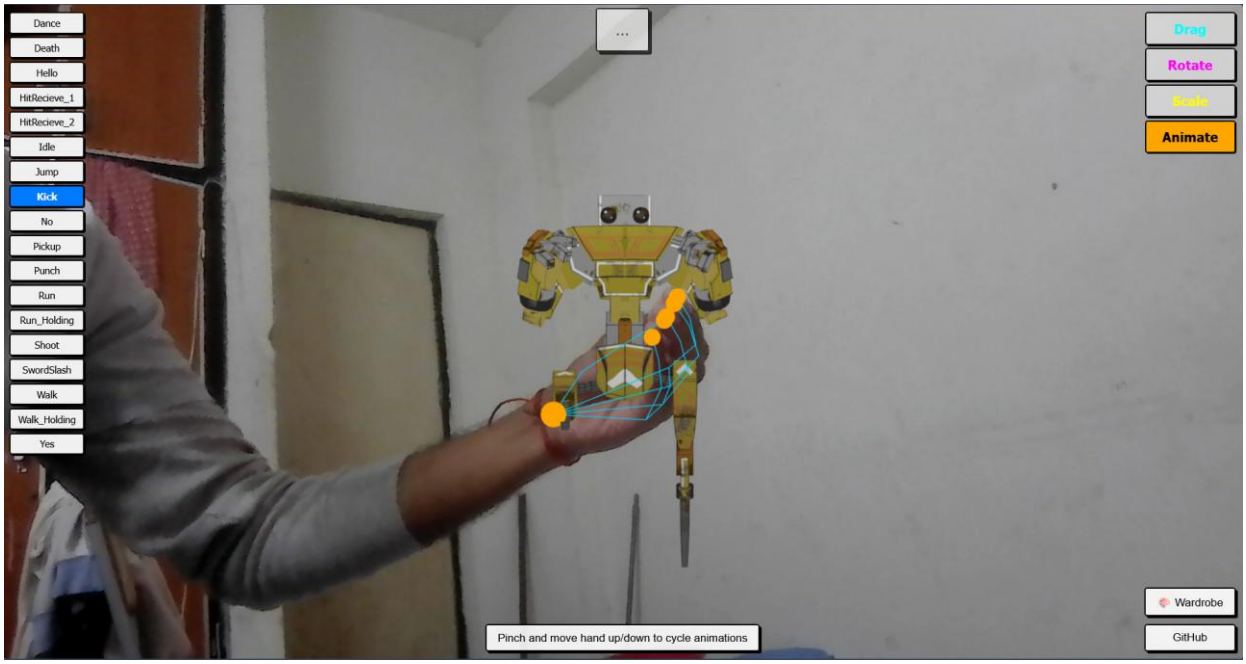


7.2 Gesture Interaction

The pinch gesture proved to be a reliable trigger.

- **Drag:** Smooth 1:1 movement.
- **Rotate:** Intuitive; moving hand left/right rotates the model around the Y-axis.
- **Scale:** Two-handed pinch zoom works effectively, mimicking multi-touch screens.

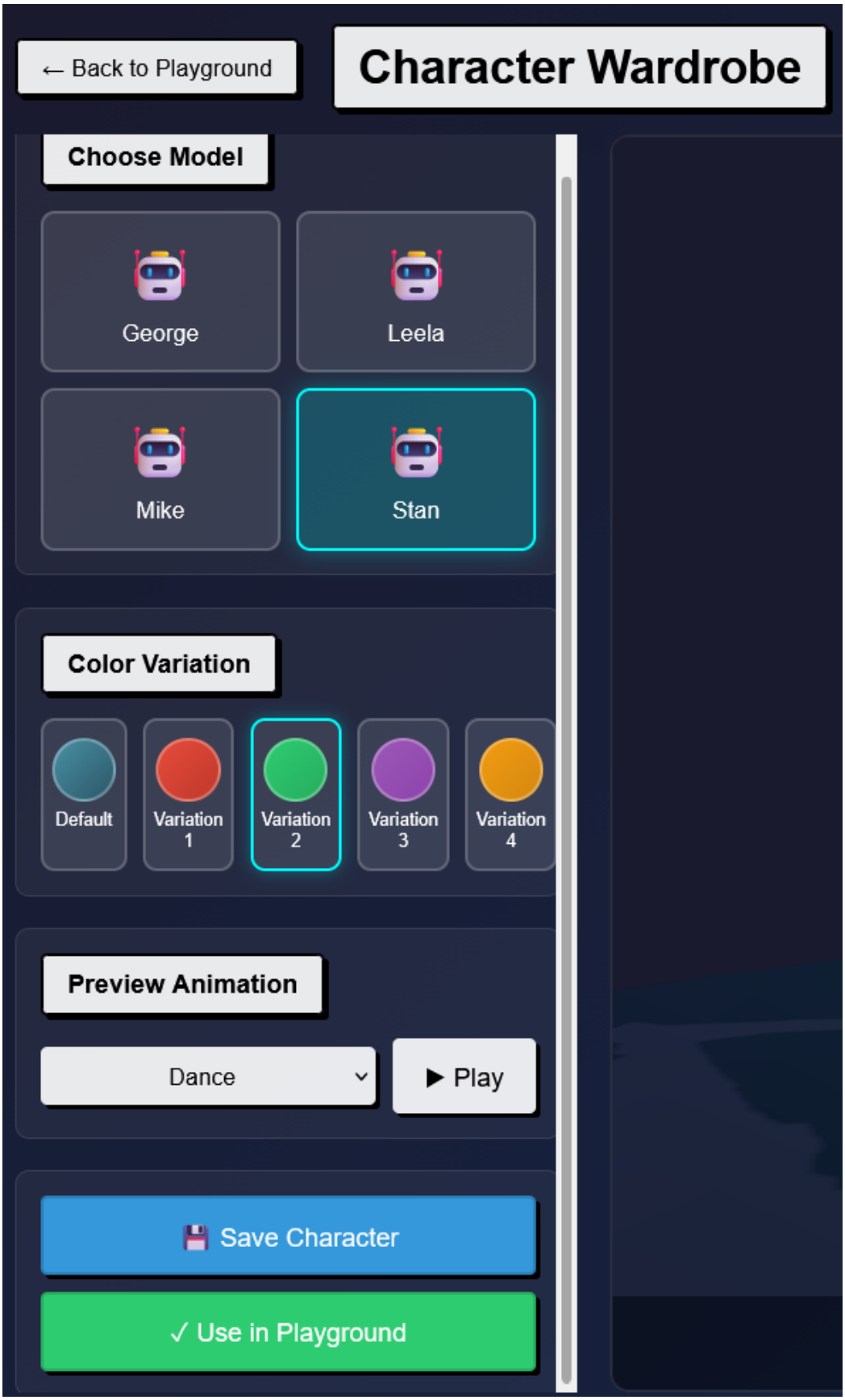




7.3 Customization Features

The Wardrobe page successfully allows users to personalize their experience.


← Back to Playground
Character Wardrobe
Choose Model
George
Leela
Mike
Stan
Color Variation
Default
Variation 1
Variation 2
Variation 3
Variation 4
Preview Animation
Dance
▶ Play
Save Character
✓ Use in Playground
Drag to rotate • Scroll to zoom


← Back to Playground
Character Wardrobe
Choose Model
George
Leela
Mike
Stan
Color Variation
Default
Variation 1
Variation 2
Variation 3
Variation 4
Preview Animation
Dance
▶ Play
Save Character
✓ Use in Playground

7.4 Performance Analysis

The application runs smoothly on mid-range hardware. The decision to use requestAnimationFrame ensures that the physics and rendering are synchronized with the display refresh rate.

CHAPTER 8: CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

Lumo Playground successfully demonstrates the viability of complex, multimodal interaction on the web. By combining **Three.js** for visuals, **MediaPipe** for perception, and **Web Speech API** for command, we created a system that feels futuristic yet is accessible today. The project met all its primary objectives: real-time tracking, gesture control, and customization. It serves as a strong foundation for future web-based AR applications.

8.2 Limitations

1. **Single Camera View:** The system lacks depth perception (Z-axis) for the hand relative to the screen, making "pushing" interactions difficult to implement accurately without a depth sensor.
2. **Occlusion:** If one hand blocks the other, tracking can fail.
3. **Lighting:** Poor lighting significantly degrades hand tracking quality.

8.3 Future Scope

1. **Multiplayer:** Implementing WebRTC to allow two users to interact with the same model in a shared virtual space.
2. **Physics Engine:** Integrating a physics engine (like Cannon.js) to allow the model to collide with virtual objects or "fall" on the user's hand.
3. **Custom Model Upload:** Fully implementing the drag-and-drop feature to allow users to upload their own .glb files.
4. **Mobile Support:** Optimizing the UI for touchscreens and mobile cameras to run on smartphones.

REFERENCES

1. **Three.js Documentation.** (n.d.). Retrieved from https://threejs.org/docs/

2. **MediaPipe Hands.** (n.d.). Google Developers. Retrieved from https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

3. **Web Speech API.** (n.d.). MDN Web Docs. Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

4. **Dirksen, J.** (2013). *Learning Three.js: The JavaScript 3D Library for WebGL*. Packt Publishing.

5. **Lugaresi, C., et al.** (2019). *MediaPipe: A Framework for Building Perception Pipelines*. arXiv preprint arXiv:1906.08172.