Three World

Filip Turk

Faculty of Computer and Information Science, University of Ljubljana, Slovenia

Abstract. This project integrates a Flask back-end with a Three.js-based interactive virtual world to create an immersive experience driven by real-time facial and hand tracking. The Flask back-end processes input from the user's camera, detecting facial landmarks and hand positions, which are then transmitted to the Three.js front-end. The detected data enables dynamic interactions within a 3D environment, allowing users to manipulate their surroundings and navigate between two distinct, interactive scenes. Each scene offers unique elements to engage with, showcasing different forms of interactivity based on the user's movements and gestures.
The project combines computer vision, web technologies, and 3D rendering to explore the intersection of technology and interactivity. By blending precise tracking and artistic design, it highlights the possibilities of integrating human-computer interaction techniques into 3D virtual environments, paving the way for innovative applications in web design and virtual experiences.

Keywords: Three.js · Computer vision · Interactivity.

1 Introduction

1.1 Purpose and Objectives

The primary purpose of this project is to create an engaging and enjoyable interactive experience that allows users to explore and interact with virtual environments using hand gestures. By leveraging real-time face and hand tracking, the project bridges the gap between the physical and digital worlds, making virtual interactions more intuitive and immersive. Users can navigate between different virtual scenes, each offering unique interactions and mechanics that showcase the potential of combining computer vision and 3D rendering technologies.

2 Related Work

2.1 Technological Comparison

The integration of computer vision with real-time 3D rendering in a browser-based environment is a relatively unexplored domain. While there are existing applications that leverage computer vision for gesture recognition and Three.js

for creating 3D visualizations, these implementations are often limited to specific use cases, such as gaming, data visualization, or virtual reality simulations.

What sets this project apart is its unique artistic approach to merging these technologies. By focusing on creative and experimental interactions, this work offers a fresh perspective on how computer vision and Three.js can be used to create immersive and interactive art experiences.

3 System Architecture

3.1 Overview of the System

The system architecture is composed of two main components: the back-end and the front-end. The back-end is responsible for processing and serving the real-time data from computer vision models, while the front-end leverages this data to render an interactive 3D world in the browser. Communication between the back-end and front-end is handled using socket connections, enabling seamless real-time interactions within the virtual environment.

3.2 Backend Design

The back—end of the application is built using Python's Flask framework, which serves as the server for processing requests and handling client communication. Google's MediaPipe library plays a crucial role in the back-end by providing tools for detecting and tracking facial and hand landmarks in real-time. These landmarks are essential for capturing user gestures and facial movements.

Once the landmark data is processed by MediaPipe, it is forwarded to the front-end through a WebSocket connection. The back-end also handles gesture recognition using MediaPipe's hand tracking model, which interprets specific hand movements as interactions within the virtual world. All computational tasks related to landmark detection and gesture recognition are performed on the backend,

3.3 Frontend Design

The frontend is powered by the Three.js library, which is used to render the 3D world directly in the browser. Three.js enables the creation and rendering of virtual objects, camera manipulation, and interactive scenes in a seamless and visually appealing way. The landmark data received from the back-end is parsed and used to create virtual points in the 3D space, representing the user's face and hands. These points are rendered and tracked in real time, allowing users to interact with the environment through their gestures and facial movements.

The frontend design focuses on providing an immersive and intuitive experience, where the user's physical actions are translated into interactions with the virtual world. This is achieved by continuously updating the landmark coordinates to reflect the real-time position of the user's face and hands, enabling interaction with the objects and effects present in the virtual environment.

4 Scene Design and Features

Switching between scenes is accomplished through gestures with the left hand. The following table summarizes the gestures and their corresponding scene-switching actions:

Table 1. Left-Hand Gestures for Scene Switching

Gesture	**Action**
Victory (Two Fingers Up)	Switch to Scene One
ILoveYou (Hand Sign)	Switch to Scene Two

Scene interactions are controlled using the right hand.

4.1 Scene One: Light Control and Vectors

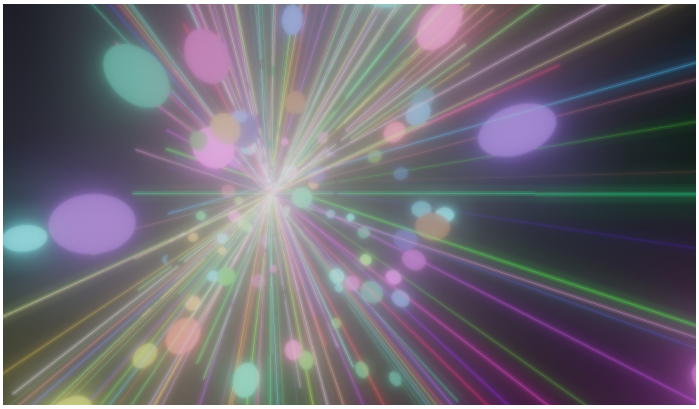

Fig. 1. Scene One

Concept In Scene One, you are placed at the center of a cosmic space where you can interact with glowing spheres using your hand gestures. These spheres can be turned on and off, creating a visually satisfying effect. The experience is further enriched by the ability to shoot light vectors into space, which interact with the glowing orbs, toggling the light on and off on contact. By pointing in different directions, you can manipulate the direction of the light rays in the scene. The camera constantly moves around, enhancing the feeling of being in motion and adding to the immersion.

Implementation The following hand gestures are mapped to specific interactions within Scene One.

Table 2. Right-Hand Gestures and Corresponding Actions in Scene One

Hand Gesture	**Action**
Open Palm	Turn on glowing spheres (planets)
Closed Fist	Turn off glowing spheres
Victory (Two Fingers Up)	Remove arrows
Pointing Up	Shoot light vectors in random directions
Thumb Up	Zoom in the scene
Thumb Down	Zoom out the scene

4.2 Scene Two: Drawing and Post-Processing Effects

Fig. 2. Scene Two

In scene two, you are immersed in a virtual environment surrounded by lines that give the impression of having been drawn in space. These lines subtly indicate the possibility of drawing in the scene by pointing the finger upward. However, due to the challenges encountered with the drawing feature implementation, this scene eventually evolved into one focused on manipulating post-processing effects.

Concept In this scene, the user interacts with the virtual space using hand gestures to control and manipulate various visual effects. Initially designed for

drawing, the scene shifted to focus on dynamic and real-time post-processing effects. The drawing functionality, though not as successful as anticipated, still serves as a gesture-based interaction method. The final implementation of the scene emphasizes creating engaging visual distortions, such as glitches, using simple hand movements. Users are given the ability to control the intensity of these effects with gestures, adding a layer of interactivity and artistic expression to the virtual environment.

Implementation The gestures in scene two are mapped to various actions within the environment. Below are the key hand gestures and their corresponding actions in this scene:

Table 3. Right-Hand Gestures and Corresponding Actions in Scene Two

Hand Gesture	**Action**
Pointing Up	Begin drawing in space
Closed Fist	Stop drawing
Thumb Up	Zoom in the scene
Thumb Down	Zoom out the scene
Open Palm	Resume camera movement
Closed Fist	Stop camera movement
Victory (Two Fingers Up)	Apply a wild glitch effect
ILoveYou (Hand Sign)	Apply a normal glitch effect

5 Technical Challenges and Solutions

5.1 Landmark Calculation and Camera Positioning

Initially, updating landmark and camera positions involved creating new objects for each frame, causing inefficiencies. This was optimized by retaining existing objects and updating their positions, reducing overhead. Additionally, a calibration step ensured accurate mapping of landmarks to 3D space, regardless of screen size.

5.2 Socket Data Transmission Speed

A noticeable lag between movements and their visual representation arose due to the high processing demand of MediaPipe. To address this, lower-quality frames and frame skipping were implemented, reducing processing time while maintaining accuracy. These optimizations improved responsiveness and user experience.

5.3 Scene Transitions

Smooth scene transitions were achieved by implementing asynchronous scene loading, ensuring assets were fully loaded before switching scenes. This minimized delays and interruptions during transitions.

6 User Experience and Evaluation

The user experience was influenced by system performance and accuracy. While promising, testing highlighted areas for improvement.

6.1 Point Transmission Speed

Notable lag during fast movements affected synchronization. Future improvements could include refining point calculations and using interpolation techniques to enhance responsiveness.

6.2 Drawing Experience in Scene Two

The moving camera and inconsistent gesture detection hindered the drawing experience. Introducing a timer-based approach to maintain drawing mode for a set duration after detecting the "finger-up" gesture could improve interaction predictability and reduce user frustration.

6.3 Improvements

To polish the project, several improvements can be made:

- **Optimizing Point Transmission**: By refining the point calculation and implementing interpolation algorithms, smoother and more responsive real-time tracking can be achieved.
- **Scene Refinements**: Further conceptual improvements to the interaction model in scene two, such as adding a more stable drawing surface or enhancing the drawing feedback, would significantly improve the user experience.
- **Expanding Interactivity**: Introducing more interactive elements using face gestures would add an innovative and engaging layer of interaction, allowing users to control specific features or trigger new visual effects through facial expressions.

7 Conclusion

This project aimed to create an interactive and immersive experience by combining computer vision, hand gesture recognition, and 3D virtual environments. The implementation explored the potential of using real-time hand gestures to

interact with virtual spaces, emphasizing intuitive and engaging user interactions.

There are numerous directions for future work that could significantly enhance the capabilities of this project. The interactive nature of this project holds potential applications in the creation of artistic web designs and gesture-based web navigation, opening pathways for innovative and creative digital experiences.

References

1. Three.js Examples. *Three.js Documentation and Examples*, Available at: https://threejs.org/examples/.