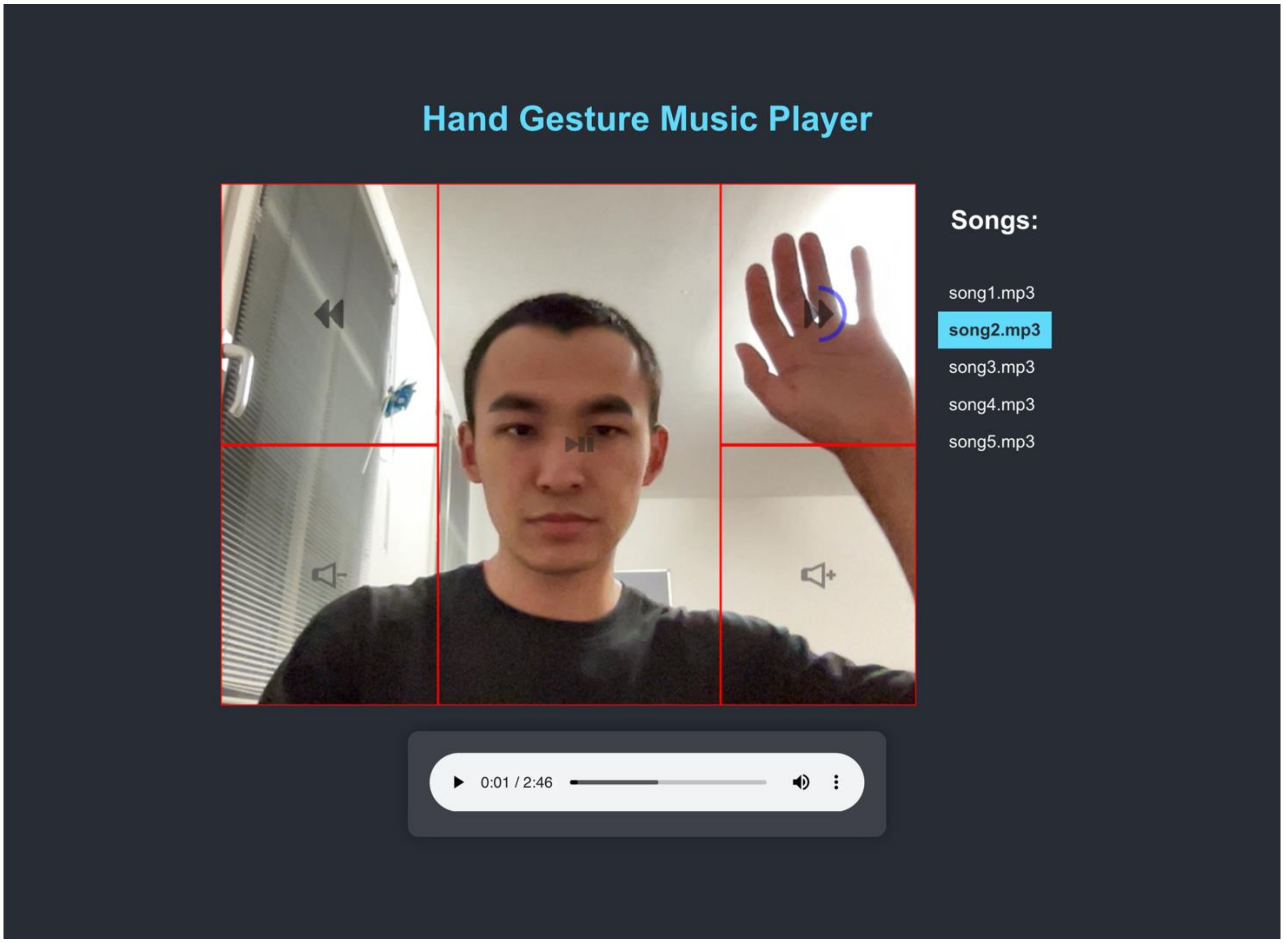


11/19/2024

Hand Gesture Music Player

Ikram Uygur, Cem Albal ,Chris Dittinger

Use Case Scenario

The App allows party guests to control music playback using simple hand gestures, providing a touch-free and intuitive way to manage the party playlist. The focus is on three essential controls: changing the song, adjusting the volume, and starting/stopping playback.

Primary Use Case: Hands-Free Music Control for Party Guests

Actors

- *Primary* Actor: Party guests.
- *System* PartyWave Lite app on a central device connected to the sound system.

Preconditions

1. The host sets up PartyWave Lite on a device (e.g., laptop, smart TV) with a camera
2. A playlist is preloaded

Steps

1. *Starting and Stopping the Music*
   - Guests want to pause the music to make an announcement.
   - One guest put a hand with into the middle of the camera. The music pauses.
   - To resume, they make the same gesture again.
2. *Changing the Song*
   - A guest dislikes the current song. They put their hand to the right top side to skip to the next track.
   - If they want to replay the previous song, they put their hand to the left top side.
3. *Adjusting the Volume*
   - The music is too quiet. A guest raises their hand to the right bottom side, and the volume increases incrementally.
   - To reduce the volume, they lower their hand to the left bottom side.

Postconditions

- Guests can seamlessly manage the music without touching the device.
- The music playback reflects the party's mood and preferences.

Key Benefits:

- Streamlines music management to just the essentials: play/pause, skip, and volume.
- Prevents the need for physical interaction with the music system.
- Progress Cycle for User feedback and avoid accidental actions

SW-Architecture

App:

Main React component with html code and setup of the songs and defines the gesture sections

GestureSection:

Represents interactive screen regions, tracks progress, and triggers actions based on hand detection.

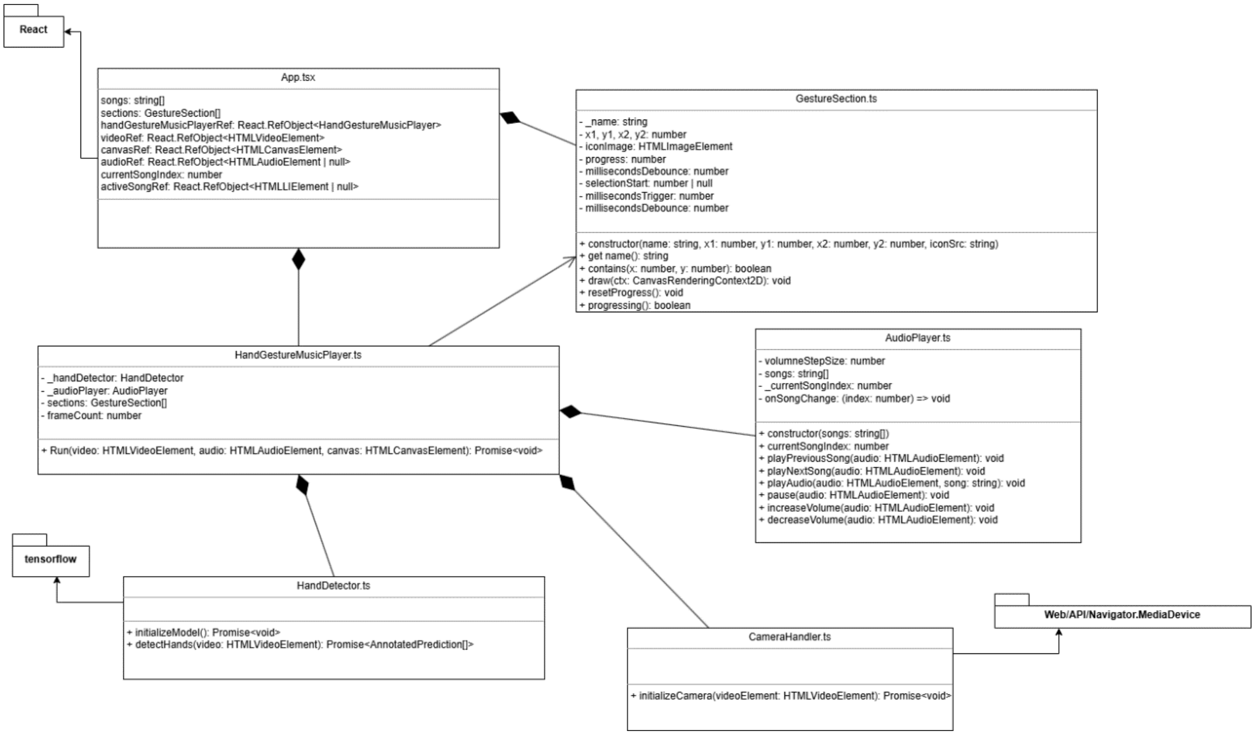
HandGestureMusicPlayer:

Manages gesture-based controls for playing, pausing, switching songs, and adjusting volume using detected hand positions.

AudioPlayer:

Controls audio playback, song switching, and volume adjustments with support for notifying changes in the current song.

CameraHandler: Handles initialization and streaming of the camera feed to a video element for hand gesture detection.


React
App.tsx
songs: string[]
sections: GestureSection[]
handGestureMusicPlayerRef: React.RefObject<HandGestureMusicPlayer>
videoRef: React.RefObject<HTMLVideoElement>
canvasRef: React.RefObject<HTMLCanvasElement>
audioRef: React.RefObject<HTMLAudioElement | null>
currentSongIndex: number
activeSongRef: React.RefObject<HTMLLIElement | null>
GestureSection.ts
- _name: string
- x1, y1, x2, y2: number
- iconImage: HTMLImageElement
- progress: number
- millisecondsDebounce: number
- selectionStart: number | null
- millisecondsTrigger: number
- millisecondsDebounce: number
+ constructor(name: string, x1: number, y1: number, x2: number, y2: number, iconSrc: string)
+ get name(): string
+ contains(x: number, y: number): boolean
+ draw(ctx: CanvasRenderingContext2D): void
+ resetProgress(): void
+ progressing(): boolean
HandGestureMusicPlayer.ts
- _handDetector: HandDetector
- _audioPlayer: AudioPlayer
- sections: GestureSection[]
- frameCount: number
+ Run(video: HTMLVideoElement, audio: HTMLAudioElement, canvas: HTMLCanvasElement): Promise<void>
AudioPlayer.ts
- volumneStepSize: number
- songs: string[]
- _currentSongIndex: number
- onSongChange: (index: number) => void
+ constructor(songs: string[])
+ currentSongIndex: number
+ playPreviousSong(audio: HTMLAudioElement): void
+ playNextSong(audio: HTMLAudioElement): void
+ playAudio(audio: HTMLAudioElement, song: string): void
+ pause(audio: HTMLAudioElement): void
+ increaseVolume(audio: HTMLAudioElement): void
+ decreaseVolume(audio: HTMLAudioElement): void
tensorflow
HandDetector.ts
+ initializeModel(): Promise<void>
+ detectHands(video: HTMLVideoElement): Promise<AnnotatedPrediction[]>
Web/API/Navigator.MediaDevice
CameraHandler.ts
+ initializeCamera(videoElement: HTMLVideoElement): Promise<void>

Used Technology

npm (Node Package Manager)

Npm is utilized in this project as the primary package manager for managing JavaScript dependencies. It facilitates the installation, versioning, and management of required libraries and modules, ensuring consistent and efficient development. Through npm, the project integrates various open-source packages from the npm registry, enabling streamlined collaboration and rapid implementation of features

Dependencies

React

React is a JavaScript library used in this project for building dynamic and interactive user interfaces. Developed and maintained by Meta (formerly Facebook), React allows for the creation of reusable UI components, enabling efficient development and consistent design patterns. Its declarative approach simplifies managing complex UI states, while its virtual DOM improves rendering performance by updating only the necessary parts of the interface.

TensorFlow

TensorFlow is used in this project to hand position on the screen. These models process input from a camera to identify specific gestures, which are then mapped to control the music player (e.g., play, pause, next track). TensorFlow's flexibility and high performance enable real-time gesture detection and seamless interaction with the application.

MediaDevices API

The MediaDevices API provides access to media input devices like cameras and microphones through the navigator.mediaDevices interface. This project uses the getUserMedia() method to request access to the device's camera, enabling real-time video streaming for tasks like gesture recognition. The API ensures secure and user-consented access, making it ideal for interactive web applications requiring media input.