MAGE Platform: A Website to Host Musical Autonomous Generated Environments

---

Project Vision

The MAGE Platform will be a full-stack web application for creating, storing, sharing, and broadcasting generative audio-reactive visual environments powered by the MAGE rendering engine.

An early demo of the MAGE Engine can be found here: https://bsiscoe.github.io/MAGE/

(I recommend turning down volume a bit and running on a strong GPU)

The goal is to build a scalable platform around a generative engine without coupling web infrastructure to the rendering internals. This system separates responsibilities into two layers:

MAGE Engine (my responsibility) - Client-side rendering runtime, consumes scene data, executes transitions and performance logic, partial working demo already complete

Web Platform (team's responsibility) - User accounts and persistence, preset management, sharing and discovery, distribution of preset state from database to client, loading and redirection of engine's audio stream

---

Core Features

Preset Storage System

Each preset contains:

- Metadata (name, author, tags, audio source)
- Engine scene data blob (opaque, generated by MAGE engine)
- Thumbnail image (also generated by MAGE engine)

Capabilities:

- User should be able to create, delete, or edit their own presets only

Engine Audio Control Interface

- Platform tells engine what audio to use
- Platform can play / pause
- Engine exposes playback state to be read and controlled by platform

Authentication & User Accounts

- Account creation and login
- User profile page
- Ownership of presets and sets

Advanced Features

Broadcast Rooms (Lightweight Live Mode)

- One host
- Multiple viewers
- Viewers receive current preset or set state
- No real-time simulation sync required

Preset Versioning

- Save revisions
- Fork presets of other users
- Track change history

Tagging and Discovery Improvements

- Tag filtering
- Featured presets
- User collections

Advanced Audio Control

- seek control
- waveform display
- audio metadata attachment
- performance timeline sync
- analysis signals exposed

Engine Integration Details

A helpful analogy is to compare the system to other streaming platforms. On YouTube for example:

- the platform stores videos, metadata, playlists, comments
- the player simply loads and plays a video stream (usually within a smaller window of the platform's page)

With the MAGE system:

- platform stores presets, metadata, sets, thumbnails
- engine loads and renders a scene state

The platform side never inspects sceneData input content, just loads it into the engine when requested. JSON generated by the engine is only ever handled by the engine itself.

The *renderer* object will provide these four functions to update visual state.

- `loadPreset(sceneData)`
  Replaces the current visual state with the provided scene configuration.
- `captureThumbnail(sceneData)`
  Renders a deterministic preview image of the provided scene state without requiring persistent engine state.
- `getCurrentPreset()`
  Returns the current scene configuration as an opaque JSON object.
- `dispose()`
  Destroys the runtime instance and releases resources.

The *audio* object will provide these five functions to control audio state.

- `setAudioSource(file: File | null, metadata?: {title, bpm, offset} | null)`

  Provides or clears the current audio source. The MAGE engine currently expects HTML5 File objects to be used as audio sources. Metadata is optional and may be used for synchronization and reproducibility but does not represent stored media.

- `play() and pause()`

  Controls simple audio playback within the engine.

- `getPlaybackState():`
  `{playing: boolean, currentTime: number, duration?: number}`

- `setPlaybackState({playing: boolean, currentTime: number, duration?: number})`

  Returns and sets playback status for UI synchronization. Intended for advanced audio control.

These functions are accessed through the MAGE engine instance

- Engine.audio.setAudioSource(audioFile)
- Engine.renderer.loadPreset(presetData)

Non-goals (Out of Scope)

To keep the project focused and achievable, the following are intentionally excluded:

- Server-side rendering of visuals
- Video streaming infrastructure
- Audio file hosting or direct storage of audio
- Modification of MAGE engine internals
- Real-time collaborative editing
- Frame-synchronized multi-user simulation
- Mobile-specific rendering optimization (i.e. probably too demanding to test on mobile devices without further engine improvments)