Exploring Audio Reactivity in glTF #252

humbletim started this conversation in **Ideas**

humbletim 6 hours ago Collaborator

I wanted to start a conversation around a concept that's becoming increasingly prominent in social virtual worlds like VRChat and Resonite: **audio reactivity**. We're seeing these incredible, immersive spaces where the world itself—the floors, the walls, the very air—pulses and vibes in sync with live music. 🎵

Right now, creating this content is a highly platform-specific and technical endeavor, often requiring deep knowledge of proprietary shader systems like AudioLink. This locks incredible creative potential inside particular gardens.

So, the food-for-thought question is: **What if we could embed the *intent* for audio reactivity directly into a glTF asset?**

The goal wouldn't be to standardize the *implementation*, but to create a common language for an asset to say, "When you hear a beat, I should flash," or "When the bass hits, I should glow." This would allow for:

- **Graceful Degradation:** The asset would look like a normal, beautiful static model in any standard glTF viewer.
- **Progressive Enhancement:** When loaded into a capable engine, the asset would "wake up" and come alive with the music.

A "Declarative" Approach: Describing What, Not How

Since we can't ship shaders or scripts in glTF, the approach would be purely declarative. An extension would simply provide metadata that **binds standardized audio signals** to an object's existing properties. A runtime could then interpret this metadata and hook it up to its native audio analysis system.

Here's a possible tiered approach, starting simple and building up in complexity.

Tier 1: Living Materials (`OMI_audio_reactive_material`) 💡

This is the foundation. It would allow artists to make a material's properties react to sound. Imagine an artist in Blender setting up a crystal material. They wouldn't write code; they'd just choose from a dropdown.

The Vision: A crystal that throbs with a soft, emissive glow in time with the music's bassline.

Under the hood, the glTF `material` definition could look something like this:

```
"extensions": {
  "OMI_audio_reactive_material": {
    "bindings": [
      {
        "signal": "BASS",
        "target": "emissiveFactor",
        "mode": "MULTIPLY",
        "factor": 5.0
      }
    ]
  }
}
```

- **`signal`**: A standard enum like `BASS`, `TREBLE`, `BEAT`, or `LOUDNESS`. This is the universal translator.
- **`target`**: A standard glTF property like `emissiveFactor`.
- **`mode`**: A simple instruction like `ADD` or `MULTIPLY`.

A VRChat/Resonite importer could see this and automatically wire it up to an AudioLink shader. Another engine could use its own audio analyzer. The artist's creative intent is preserved across platforms.

Tier 2: Moving the World (`OMI_audio_reactive_node`) 🛠️

The next step would be applying the same logic to a node's transform, allowing objects to move, scale, or rotate with the music.

The Vision: A series of pillars on a stage that gently "breathe" in and out, scaling up slightly on every beat.

The `node` definition could be extended like so:

```
"extensions": {
  "OMI_audio_reactive_node": {
    "bindings": [
      {
        "signal": "BEAT",
        "target": "scale",
        "mode": "ADD",
        "factor": [0.0, 0.1, 0.0]
      }
    ]
  }
}
```

In a non-reactive viewer, you just see pillars. In a reactive world, the stage feels alive and architectural elements become part of the performance.

Tier 3: Complex Choreography (`OMI_audio_reactive_animation`) ✨

For the ultimate level of creative control, we could allow an audio signal to drive the playback of a standard glTF animation. This lets artists author complex, nuanced reactions using the animation tools they already know.

The Vision: A complex mechanical flower whose petals open and close based on the overall loudness of a song, revealing a glowing core during the chorus.

The artist would create a normal 0-to-10-second animation of the flower opening. The extension would then "hijack" the animation's timeline.

```
"animations": [
  {
    "name": "Flower_Bloom_Animation",
    ...
    "extensions": {
      "OMI_audio_reactive_animation": {
        "signal": "LOUDNESS"
      }
    }
  }
]
```

Now, when the music is quiet (`LOUDNESS` = 0.0), the animation is at its 0-second mark (flower closed). When the music swells to its peak (`LOUDNESS` = 1.0), the animation scrubs to its end (flower fully open).

The Big Picture

This kind of extension could empower creators to build richer, more dynamic, and more valuable assets that work across the open metaverse. It respects the glTF philosophy of being a portable, declarative format while opening the door to the next generation of interactive content.

Would love to hear people's thoughts on this. Is this a viable direction? What are the potential pitfalls? Are there other signals or modes that would be essential for creators?

Let's discuss!

↑ 2 👍 1

1 comment

Oldest Newest Top

aaronfranke 5 hours ago Maintainer

`"target"` should probably be a glTF JSON pointer to allow it to reference any property, such as `/nodes/0/scale`.

↑ 1

0 replies

Category

💡 **Ideas**

Labels

None yet

2 participants