Lux SDK – Detailed User Documentation

---

1. Getting Started Guide

Welcome to Lux SDK! This toolkit will help you create **emotion-driven, immersive 5D experiences**. Below are the steps to get started with Lux SDK.

---

2. Installation Instructions

Web Integration

1. **Download the Lux SDK**:
    - Save the Lux SDK file (`lux-sdk-v2.1.js`) to your project directory. This file contains all the necessary functionality, including visual, sound, motion, and more.
2. **Include the SDK in your HTML file**:
    - Add the Lux SDK to your HTML page by linking the script tag. You will also need to include **Three.js** for 3D rendering and your audio files for interactive sound features.
3. Example HTML structure:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Lux SDK v2.1 – A 5D Storytelling Toolkit</title>
</head>
<body>
  <!-- Add a button or controls to trigger actions -->
  <button onclick="Lux.Payload.coreSequence()">Run Core Sequence</button>
  <button onclick="Lux.Payload.hopefulFuture()">Run Hopeful Future</button>

  <!-- Include Three.js for 3D rendering -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r134/three.min.js"></script>

  <!-- Include Lux SDK -->
```

```
  <script src="lux-sdk-v2.1.js"></script>

  <!-- Add your custom setup and functions -->
  <script>
    // Custom scene setup and Lux SDK initialization code
  </script>
</body>
</html>
```

Audio Files:

- Include the required audio files (e.g., `pulseSound.mp3`, `futureSound.mp3`) in the same directory as your HTML file.

Quantum Integration

1. **Prerequisites**:
    - Install a **Quantum computing environment**, such as IBM Qiskit, Microsoft Quantum SDK, or similar tools. Lux SDK integrates with quantum systems for dynamic interactions based on user feedback, including **quantum-based randomness**.
2. **Quantum SDK Setup**:
    - Ensure you have set up a compatible quantum computing environment.
    - Lux SDK's **Quantum Tier** will interface with quantum systems to perform emotion-driven dynamic calculations.
3. Example:

```
Lux.Quantum.runEmotionDrivenInteraction({ emotion: 'joy' });
```

3. How to Set Up Your First Project Using Lux SDK

1. **Initialize the Scene**: In your JavaScript file, initialize the scene, camera, renderer, and other key elements required for Lux SDK's functionality.

    Example Code:

```
let scene, camera, renderer;
function init() {
```

```
  scene = new THREE.Scene();
  camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1,
1000);
  renderer = new THREE.WebGLRenderer();
  renderer.setSize(window.innerWidth, window.innerHeight);
  document.body.appendChild(renderer.domElement);
}

function animate() {
  requestAnimationFrame(animate);
  renderer.render(scene, camera);
}

init();
animate();
```

Import Lux SDK: Link to the **Lux SDK** and initialize it by calling relevant functions from the Lux object.

Example:

```
Lux.Script.create('simpleAnimation', [
  { fn: () => Lux.Core.reset(), t: 0 },
  { fn: () => Lux.Visual.setColor(0.5, 0.8, 1), t: 1000 }, // Light Blue
  { fn: () => Lux.Motion.setSpeed(1), t: 2000 },
  { fn: () => Lux.Pulse.trigger(), t: 3000 },
  { fn: () => Lux.Sound.play('futureSound'), t: 4000 },
  { fn: () => Lux.Log.emit('Animation Complete!'), t: 5000 }
]);

Lux.Payload.simpleAnimation();
```

Interactive Color Change:

```
// Example of interactive storytelling where the scene color changes based on the player's input
document.querySelector('#changeColorButton').addEventListener('click', () => {
  const r = Math.random(), g = Math.random(), b = Math.random();
  Lux.Visual.setColor(r, g, b);
});
```

5. Instructions on Adding and Using Lux SDK in Different Platforms

Unity Integration

1. **Setup Unity Project**:
    - Import Lux SDK by downloading the latest version of **Lux SDK**.
    - In Unity, go to `Assets -> Import Package -> Custom Package`, then select the Lux SDK package.
2. **Create Lux Script**:
    - In Unity, you can create a script to control Lux SDK components (visuals, sound, etc.) based on user interaction.
3. Example Unity script:

```
using UnityEngine;
using LuxSDK;

public class LuxController : MonoBehaviour
{
    void Start()
    {
        Lux.Visual.setColor(1, 0, 0); // Red for starting color
        Lux.Sound.play("pulseSound");
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            Lux.Visual.setColor(0, 1, 0); // Change to green
        }
    }
}
```

Unreal Engine Integration

1. **Setup Unreal Project**:
    - Download and import Lux SDK into your **Unreal Engine** project.

- Ensure you have the **Three.js** engine running within a **UE WebGL/HTML5** container to interface with Lux SDK.

2. **Create Blueprint for Lux**:

   - Use **Blueprints** to create motion, visual, and sound effects based on the player's actions and interactions.

Web Integration:

As mentioned earlier, you can integrate Lux SDK into your HTML by including it via a `<script>` tag and initializing the SDK through simple JavaScript functions.

VR Integration (Oculus/HTC Vive)

1. **VR Setup**:

   - Import the Lux SDK into your VR project (Unity or Unreal recommended).

   - Enable **motion sensing** by utilizing input controllers and VR tracking for immersive motion effects.

   - Use **real-time emotional responses** based on user inputs (via sensors or voice commands).

2. **Example VR Interaction**:

```
// Integrating VR input with Lux SDK for a dynamic VR environment
function onVRInput(data) {
  if (data.emotion === 'happy') {
    Lux.Visual.setColor(0, 1, 0); // Change to green for happy
  } else if (data.emotion === 'sad') {
    Lux.Visual.setColor(0.5, 0, 0); // Change to dark red for sadness
  }
}
```