Technical Report on the Development of a 360° Virtual Reality Application for Live-Show Visualization

Author: Felipe Marzanasco
Project Context: Sergio Botelho’s master's research project

Abstract

This technical report presents the design and development of an immersive 360° virtual reality application for Meta Quest standalone devices, created to support Sergio Botelho’s master's research project on immersive concert visualization and comparative sound analysis. The system was developed in Unity using a component-based architecture and was designed to allow users to experience a recorded musical performance from multiple selectable viewpoints while comparing spatial ambisonic audio with a conventional non-spatial audio mix. Rather than functioning as a conventional linear 360° video player, the application enables dynamic switching between synchronized perspectives while preserving temporal continuity, spatial audio coherence, interface legibility, and user comfort.

The application integrates Unity Engine, the Universal Render Pipeline, Meta SDK interaction systems, hand tracking, controller-based interaction, world-space user interfaces, head tracking, and ambisonic audio processing through the Meta Audio SDK. The project also addresses practical limitations associated with high-resolution 360° video playback on standalone VR hardware, including media file size, loading latency, APK packaging constraints, and the need for an external media loading strategy. To mitigate these constraints, the system adopts sideloaded media assets, dual video players, fade-to-black transitions, synchronized video/audio playback, and user-view recentering mechanisms.

A dedicated lobby environment was also implemented as a preparatory stage before the show experience. This space functions both as an initial interface and as a gradual sensory adaptation environment, allowing users to become familiar with spatialized sound, low-light conditions, VR interaction methods, and the application’s navigation logic before entering the main audiovisual experience.

Keywords: virtual reality; 360° video; Unity; Meta Quest; ambisonics; hand tracking; world-space interface; media synchronization; motion sickness; URP; immersive concert visualization.

---

1. Introduction

The development of interactive audiovisual systems for standalone virtual reality platforms requires the integration of real-time rendering, interaction design, media synchronization, spatial audio processing, and deployment strategies compatible with the limitations of the target hardware. In this context, the present application was conceived as a dedicated VR system for 360° live-show visualization on Meta Quest devices, allowing users to experience a musical performance from multiple points of view during continuous playback.

Unlike conventional 360° video playback systems, which usually position the viewer as a passive observer inside a fixed spherical recording, this application introduces an interactive layer in which users can actively select different viewpoints throughout the performance. These viewpoints correspond to predefined camera positions in the recorded event, such as the audience perspective and viewpoints associated with individual performers or stage locations.

This requirement introduced several technical challenges, including external media loading, continuity of playback time, synchronization between video and ambisonic audio, spatial interface readability, interaction comfort, transition design, and user orientation after perspective changes. The core objective was to create an experience in which users feel as if they are moving between different positions within the same live show without losing the temporal flow of the performance.

A central contribution of the project is therefore its interactive approach to immersive concert visualization. The user is not merely placed inside a spherical video environment, but is given agency to navigate between viewpoints, interact with interface elements embedded in the three-dimensional space, and decide how to experience the performance in real time. This interaction model increases the sense of presence and differentiates the solution from a traditional 360° video player.

---

2. Unity Engine, Rendering Pipeline, and Component-Based Architecture

The application was developed in Unity, a real-time engine widely used for games, simulations, visualization systems, and interactive applications. Unity was selected due to its scene-based workflow, multiplatform deployment support, component-oriented architecture, and mature ecosystem for VR development.

In Unity, each element placed in a scene is represented as a GameObject, and its behavior is defined by attached Components. These components may control rendering, audio playback, video playback, collision, interaction, animation, user interface behavior, or custom logic implemented through scripts. This architecture was particularly suitable for the project because the application required several independent but coordinated systems, including video playback control, viewpoint switching, interface management, spatial audio configuration, fade transitions, and user recentering.

Custom C# scripts were developed as reusable components responsible for controlling specific behaviors. This approach allowed the project logic to be divided into modular systems, improving organization, maintainability, debugging, and scalability. Each major behavior—such as media switching, viewpoint metadata, interface positioning, player recentering, and world-space UI orientation—was encapsulated into a dedicated component.

The selected rendering pipeline was Unity’s Universal Render Pipeline (URP). URP was chosen because it provides an efficient balance between visual quality and performance, which is essential for standalone VR hardware. Since Meta Quest devices have limited processing power compared with PC-based VR systems, the rendering strategy had to prioritize performance stability while maintaining sufficient visual clarity for the lobby environment, user interface elements, and 360° playback sphere.

---

3. Development Environment and Technology Stack

The application was implemented using C# as the primary programming language and Visual Studio as the development environment. Unity served as the main engine for real-time scene construction, interaction logic, media playback, and deployment.

Meta Quest integration was implemented through the Meta SDK, which provided the necessary framework for VR input abstraction, headset tracking, hand tracking, controller support, and interaction systems. The project was designed to support both hand tracking and traditional VR controllers, allowing the experience to remain usable across different interaction preferences and testing scenarios.

The user interface design process began in Figma, where the visual structure, layout hierarchy, button organization, and user flow were initially planned. These interface concepts were later translated into Unity as world-space UI elements, meaning that interactive buttons and visual indicators exist as objects within the virtual environment rather than as flat overlays fixed to the screen.

Spatial audio was implemented using ambisonic audio assets processed through the Meta Audio SDK. This enabled the perceived sound field to respond to the user's head rotation, contributing to a more coherent and immersive auditory experience. In the context of a live-show simulation, this feature is especially relevant because sound directionality and spatial perception are fundamental to the sensation of being present in a performance environment.

---

4. Interaction Model

The interaction model was designed to support both direct and distant interaction within a VR environment. Two complementary interaction paradigms were implemented: poke interaction and ray interaction.

Poke interaction allows the user to select nearby UI elements by physically reaching toward them with tracked hands or controllers. This interaction model supports direct manipulation and is useful when the user is close to interface elements. Ray interaction, on the other hand, allows users to select elements positioned at a greater distance by pointing toward them with a virtual ray. This is particularly important in a 360° video environment, where UI elements may be positioned in different areas of the spherical scene.

Head tracking is a fundamental component of the system. The 360° video is rendered as an equirectangular video mapped onto the internal surface of a sphere with inverted normals, with the user positioned at the center. As the user rotates their head, the visible frame of the show changes naturally according to the headset's orientation. This same head rotation also contributes to the perception of ambisonic audio, since the spatial sound field must remain consistent with the user's orientation.

The system was therefore designed around the relationship between visual perspective, spatial audio, and embodied interaction. The user's gaze, hand movement, controller pointing direction, and head rotation all contribute to how the experience is perceived and navigated.

---

5. Lobby Environment and User Preparation

Before entering the main show experience, the user is placed in a themed lobby environment. This lobby is not merely a menu screen; it functions as a transitional space for perceptual and interaction adaptation before the audiovisual experience begins.

The lobby was designed with visual and auditory elements related to live music environments, including light-emitting objects, speakers, and ambient components. These objects help introduce the user to the spatial behavior of sound before the main show starts. By allowing users to perceive how head and body orientation affect listening in a spatialized environment, the lobby reduces perceptual unfamiliarity and prepares the user for the use of ambisonic sound during the show.

The lobby also supports general adaptation to VR. Since the show environment is predominantly dark, the lobby uses a darker visual atmosphere to gradually acclimate the user's eyes to low-light conditions. This design choice creates a smoother perceptual transition between the application's initial state and the final concert visualization environment.

The three-dimensional models used in the lobby were created and optimized for the application. The scene uses lightmapped global illumination, in which lighting and shadow information are precomputed and stored in lightmaps. This strategy is advantageous for standalone VR because it allows visually richer lighting while minimizing real-time computational cost. Since lighting calculations are not performed dynamically during runtime, the headset can preserve performance while still presenting a more polished environment.

---

6. Visual Playback Architecture

The main show content is presented through equirectangular 2D video mapped onto the inside of an inverted sphere. The user is positioned at the sphere's center and can explore the recorded environment through natural head rotation. This approach is computationally efficient when compared with fully volumetric capture systems, while still providing a convincing immersive experience for standalone headsets.

The initial entry point into the show is the audience perspective. This perspective was chosen because it provides contextual orientation: it allows users to understand the general scale of the venue, the arrangement of the performers, and the atmosphere of the event before selecting more specific viewpoints.

A central architectural decision was the use of two Video Player components rather than a single playback component. This structure was implemented to reduce the perceived delay associated with loading large external 360° video files during viewpoint transitions. While the active video continues to play, the destination video can be assigned to a secondary player and prepared in the background. Once the new video is ready and the screen is fully faded to black, the system synchronizes the playback timestamp and switches the active player.

This dual-player strategy improves the smoothness of transitions and avoids abrupt interruptions. It also provides a technical window in which the system can prepare new media, synchronize time values, change audio sources, and prevent visual artifacts before returning visibility to the user.

---

7. External Media Strategy and Sideloading Workflow

One of the main technical limitations of the project was the handling of high-resolution 360° video files on Android-based standalone VR hardware. Large media files can significantly increase the size of the application package and may make it impractical to include all videos directly inside the APK. In addition, packaging and compression limitations during the build process can complicate deployment and testing when large media assets are embedded in the application.

To address this limitation, the application uses an external media loading strategy based on sideloading. Instead of packaging all video files directly into the build, the media assets are stored in a public or accessible directory on the Meta Quest device and loaded at runtime.

This approach provides several advantages. It reduces the build size, simplifies application deployment, allows media files to be replaced without recompiling the entire application, and separates executable logic from heavy audiovisual assets. From an engineering perspective, this separation improves iteration speed during testing and makes the system more flexible for future content updates.

---

8. Ambisonic Audio Configuration

Each selectable viewpoint is associated not only with a corresponding 360° video file, but also with an ambisonic audio asset. The audio system was configured using the Meta Audio SDK to support spatial processing compatible with head rotation.

This configuration allows the perceived sound field to change according to the user's orientation, reinforcing the realism of the live-show environment. Since the user may rotate their head freely while positioned inside the spherical video, the audio field must remain spatially coherent with the visual content.

The coupling between each visual perspective and its corresponding audio file allows the system to preserve contextual sound identity across viewpoint changes. This is particularly relevant in immersive performance visualization because different camera positions may imply different auditory emphases. For example, a viewpoint closer to a specific performer may require a different perceived sound balance than the audience perspective.

The application also adjusts audio orientation data during transitions to maintain a coherent relationship between the selected viewpoint and the perceived direction of the sound field. This prevents the user from experiencing a mismatch between visual position and auditory localization after changing perspectives.

---

9. World-Space Interface Design

The user interface was implemented as a world-space system, meaning that interface elements are positioned directly inside the virtual environment. Each selectable viewpoint is represented by an icon placed at a specific three-dimensional position related to the corresponding performer or camera location.

This spatial distribution makes the interface part of the immersive scene rather than a detached menu. Variations in scale and position were intentionally used to reinforce depth perception and to visually integrate the interface into the 360° environment.

Each button is configured with data related to its target viewpoint. This includes the video file to be loaded, the ambisonic audio asset to be played, the interface configuration associated with that perspective, and the audio orientation parameters required to maintain spatial coherence. Therefore, the buttons are not only visual UI elements; they also function as data containers that instruct the system how to transition to a new point of view.

To improve visibility and affordance, each icon performs a subtle vertical animation, moving gently up and down above the corresponding performer or spatial reference. This animation has both aesthetic and functional purposes: it draws the user's attention to available interaction points without overloading the scene with excessive motion.

---

10. Interface Orientation and Legibility

World-space interfaces in VR may become difficult to read when viewed from oblique angles. To address this problem, the application implements a continuous orientation routine that makes interface elements face the user while optionally restricting rotation to the vertical axis.

This behavior improves readability and interaction accuracy while avoiding the unnatural effect of full billboarding across all axes. By rotating mainly around the Y axis, the UI remains horizontally aligned and visually stable, preventing buttons from tilting upward or downward in a way that could feel artificial or uncomfortable.

This design decision is particularly important in a 360° video application, where users frequently rotate their heads to explore the environment. The interface must remain legible and usable regardless of the user's viewing direction, while still preserving a sense of spatial consistency within the scene.

---

11. Viewpoint Metadata System

The application includes a dedicated metadata component for viewpoint buttons. Each button contains the information required to define a transition target, including the destination video, corresponding audio source, interface state, and audio orientation settings.

This design allows each interactive element to represent a specific camera position or viewpoint in the show, such as audience, vocals, bass, drums, or guitar. When the user selects a button, the main control system reads this metadata and determines which media assets and interface configuration should be activated.

This structure makes the application easier to maintain and expand. Additional viewpoints can be introduced by configuring new buttons with the appropriate metadata, without requiring a complete restructuring of the central playback logic.

---

12. Perspective-Specific Interface Management

The interface layout changes according to the active viewpoint. For each position in the show, the system defines which UI elements should be visible, where they should appear, and what scale they should use.

This is necessary because a UI layout that works from one viewpoint may not remain spatially coherent from another. For example, a button that correctly indicates a performer from the audience perspective may need to move, resize, disappear, or be repositioned when the user changes to another camera location.

The interface management system therefore ensures that the world-space UI remains coherent with the user's current virtual position. This improves navigation clarity and prevents interface elements from appearing misaligned, confusing, or disconnected from the visual content.

---

13. User Recenter System

When switching to a new viewpoint, the user may physically be looking in any direction. Without correction, the new perspective could begin with the user facing an irrelevant or disorienting part of the scene. To address this issue, the application includes a recentering system that adjusts the horizontal orientation of the VR rig after a transition.

The purpose of this system is to make each new viewpoint begin with the user facing a meaningful default direction, usually toward the main point of interest in the scene. This adjustment is performed without altering the real headset height, preserving the user's physical sense of presence.

The recentering routine primarily corrects horizontal rotation while avoiding unnatural vertical tilt. This makes viewpoint transitions more controlled and reduces the risk of disorientation, especially when users rapidly explore different perspectives during the show.

---

14. Transition Algorithm Between Viewpoints

The transition system is one of the application's central technical components. Each viewpoint button stores the destination media data required for the transition. When the user selects a new perspective, the system begins a fade-to-black sequence designed to mask the visual process of loading and switching media.

During the fade interval, the destination video is assigned to the secondary Video Player and prepared in the background. Once the screen reaches full black and the new media is ready, the system reads the current playback time from the active video and applies that timestamp to the newly prepared video. The corresponding ambisonic audio is then aligned to the same temporal position.

After video and audio synchronization are completed, the secondary player becomes the active player, the appropriate audio source is activated, the interface layout is updated, the user's view is recentered, and the screen gradually fades back from black to the visible image.

The fade-to-black transition serves multiple purposes. Visually, it masks loading delays, frame mismatches, white flashes, or incorrect intermediate frames that may appear during media preparation. Perceptually, it softens the spatial discontinuity caused by changing from one viewpoint to another. Abrupt cuts in immersive environments can cause cognitive discomfort because the user's spatial reference changes instantly. The fade acts as a perceptual buffer that reduces disorientation and contributes to user comfort.

The system also prevents multiple transitions from occurring simultaneously. If the user selects another viewpoint while a transition is already in progress, the new input is ignored until the current transition finishes. This prevents playback conflicts, synchronization errors, and unstable interface states.

---

15. Temporal Synchronization and Continuity

Temporal synchronization is essential in this application because all viewpoints represent the same live performance. A perspective change should not restart the show from the beginning or cause the user to lose the current moment of the performance.

The adopted strategy is based on reading the playback timestamp of the currently active video, applying that same timestamp to the destination video, and aligning the corresponding ambisonic audio to the same position before completing the transition.

This procedure minimizes perceptible discontinuities and preserves the narrative and musical continuity of the experience. The user can therefore move between different viewpoints while remaining approximately at the same moment in the performance, creating the sensation of switching between synchronized cameras within a live audiovisual environment.

The fade-to-black interval also functions as an operational window for synchronization. It gives the system time to prepare the next media pair and only return visibility to the user once the required synchronization criteria have been met.

---

16. Application Flow

The user begins in the lobby environment. From this initial space, the user can start the show experience. When the experience begins, the screen fades to black, the initial 360° video and ambisonic audio are prepared, the playback environment is activated, and the show starts from the audience perspective.

During the show, the user can interact with spatial buttons placed inside the VR environment. Selecting a button initiates the transition sequence: the screen fades to black, the target media is prepared, the playback time is synchronized, the audio source and orientation are updated, the user view is recentered, and the interface is reorganized for the new perspective.

Once the first frame of the new video is ready and the system state has been updated, the 360° sphere becomes visible again and the screen fades back from black. From the user's perspective, the result is a smooth and comfortable transition between cameras while preserving the continuity of the performance.

When the show ends, or when the user chooses to leave the experience, the application fades to black, stops video and audio playback, clears the current media state, restores the initial configuration, and returns the user to the lobby. This final transition gives the experience a circular structure and provides a neutral space for reorientation after the immersive show.

---

17. Implemented System Components

17.1 Main Video Control System

The main video control system manages the transition between the lobby and the show environment, starts 360° playback, coordinates viewpoint changes, controls media synchronization, handles audio switching, and returns the application to the lobby when the show ends or when the user exits.

It is the central coordination layer of the experience. It ensures that a new viewpoint is not activated until the required transition steps have been completed, including fade, media preparation, timestamp transfer, audio synchronization, interface update, and view recentering.

17.2 Viewpoint Data System

The viewpoint data system allows each spatial button to store the information required to activate a specific perspective. This includes the destination video, associated audio, interface configuration, and audio orientation values.

This system separates viewpoint identity from the main playback logic, making the application more modular and easier to expand.

17.3 Perspective Interface System

The perspective interface system adapts the UI according to the active viewpoint. It controls which elements are displayed, where they are positioned, and how large they appear in each perspective.

This ensures that UI elements remain spatially meaningful and visually coherent across different camera positions.

17.4 VR View Recenter System

The recenter system adjusts the user’s horizontal orientation after each viewpoint change. Its purpose is to make the user begin each new perspective facing the intended visual direction while preserving natural headset height and physical presence.

17.5 User-Facing Interface Orientation System

The user-facing interface orientation system keeps world-space UI elements readable by rotating them toward the user. When required, rotation can be limited to the horizontal axis to preserve visual stability and avoid unnatural tilting.

---

18. Technical Limitations and Design Responses

The project required several technical decisions in response to limitations of standalone VR hardware.

First, high-resolution 360° video files are heavy and unsuitable for direct inclusion in the APK in large quantities. This limitation was addressed through external sideloading and runtime media loading.

Second, large video files may introduce loading delays during perspective changes. This was mitigated through a dual-player architecture and background preparation of the destination media.

Third, abrupt changes in viewpoint can cause discomfort in immersive environments. This was addressed through fade-to-black transitions, user recentering, and controlled interface reconfiguration.

Fourth, world-space UI elements may become difficult to read depending on viewing angle. This was mitigated through user-facing orientation behavior and perspective-specific UI placement.

Finally, visual and auditory coherence must be preserved during viewpoint changes. This was addressed through timestamp synchronization, corresponding ambisonic audio files for each perspective, and audio orientation adjustment.

---

19. Conclusion

The developed application demonstrates a technical approach to interactive 360° live-show visualization in standalone virtual reality. By combining synchronized 360° video playback, ambisonic audio, world-space interface design, hand and controller interaction, smooth viewpoint transitions, and user comfort mechanisms, the system expands the experience beyond passive video consumption.

The application allows users to actively navigate between multiple synchronized viewpoints of the same performance while preserving temporal continuity and spatial coherence. The use of fade-to-black transitions, dual video players, external media loading, viewpoint-specific interface logic, audio synchronization, and recentering contributes to a more stable and comfortable immersive experience.

In the context of Sergio Botelho’s master's research project, the system provides a practical platform for investigating immersive concert visualization, user interaction with multi-perspective audiovisual content in VR, and the perceptual comparison between spatial ambisonic audio and conventional non-spatial audio mixes. The final result is not only a 360° player, but a coordinated immersive system in which media playback, spatial audio, interaction design, and user comfort are integrated into a unified experience.

References

[1] Meta. Available at: https://www.meta.com/
[2] Unity Technologies. Unity Engine. Available at: https://unity.com/
[3] Meta Developers. Meta XR SDK for Unity. Available at: https://developers.meta.com/horizon/develop/unity/
[4] Unity Technologies. Universal Render Pipeline Documentation. Available at: https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal/

[5] Microsoft. Visual Studio. Available at: https://visualstudio.microsoft.com/
[6] Figma. Available at: https://www.figma.com/